



# Development of hybrid models for genome-scale metabolic networks

Léon Faure

## ► To cite this version:

Léon Faure. Development of hybrid models for genome-scale metabolic networks. Quantitative Methods [q-bio.QM]. Université Paris-Saclay, 2023. English. NNT : 2023UPASL103 . tel-04562281

**HAL Id: tel-04562281**

**<https://theses.hal.science/tel-04562281>**

Submitted on 29 Apr 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Development of hybrid models for genome-scale metabolic networks

## *Développement de modèles hybrides pour les réseaux métaboliques à l'échelle du génome*

**Thèse de doctorat de l'université Paris-Saclay**

École doctorale n° 577, Structure et Dynamique des Systèmes Vivants (SDSV)  
Spécialité de doctorat: Biotechnologies  
Graduate School : Life Sciences and Health (LSaH). Référent : Faculté des sciences  
d'Orsay

Thèse préparée dans l'unité de recherche **Micalis Institute** (Université Paris-Saclay,  
INRAE, AgroParisTech), sous la direction de **Jean-Loup FAULON**, directeur de recherche,  
et la co-direction de **Wolfram LIEBERMEISTER**, directeur de recherche.

**Thèse soutenue à Paris-Saclay, le 08 Décembre 2023, par**

**Léon FAURE**

### **Composition du jury**

Membres du jury avec voix délibérative

**Antoine CORNUÉJOLS**  
Professeur des universités, Université Paris-Saclay  
**Sabine PERES**  
Professeure des universités, Université Lyon I  
**Stéphanie HEUX**  
Professeure des universités, Université Toulouse III  
**Martin LERCHER**  
Professeur des universités, Université Heinrich Heine

Président  
Rapporteuse & Examinatrice  
Rapporteuse & Examinatrice  
Examineur



**Titre:** Développement de modèles hybrides pour les réseaux métaboliques à l'échelle du génome

**Mots clés:** Modélisation métabolique, Apprentissage automatique, Modèles hybrides

**Résumé:** Au cours des deux dernières décennies, la communauté de la biologie des systèmes a consacré des efforts considérables à la construction de modèles métaboliques à l'échelle du génome (GEM), qui offrent des représentations détaillées de l'ensemble du métabolisme d'un organisme. Les GEM présentent le métabolisme comme un réseau, reliant les réactions métaboliques et les métabolites. Malgré la richesse des informations qu'ils contiennent, les GEM présentent des limites notables. Ils tentent d'englober tous les phénotypes métaboliques potentiels, ce qui conduit à un vaste espace de solutions qu'il peut être difficile d'explorer efficacement. L'approche prédominante pour l'exploitation des GEM, l'analyse de l'équilibre des flux (FBA), repose sur des simplifications et n'a pas la capacité de se généraliser à diverses conditions. En revanche, les techniques d'apprentissage automatique ont gagné en intérêt pour la modélisation métabolique, notamment en exploitant les données -omiques à grande échelle pour prédire les comportements biologiques dans divers environnements. Bien que de nombreuses approches combinent les GEM et l'apprentissage automatique, elles séparent toujours les parties modélisation métabolique

et apprentissage automatique, ce qui limite leur adaptabilité et leur réutilisation. Dans le cadre de cette thèse de doctorat, j'introduis une approche innovante qui s'attaque à cette limitation : un modèle hybride neuronal-mécaniste pour les GEM, appelé Réseau Métabolique Artificiel (AMN). Cela implique le développement de méthodes de substitution au FBA compatibles avec la rétropropagation du gradient et la création d'une fonction de perte mécaniste pour aligner les prédictions AMN avec les contraintes des GEMs. Cette thèse se penche sur les phénomènes biologiques abordés par les AMN et passe en revue les méthodes d'utilisation des GEM les plus récentes. Elle démontre ensuite que les AMN sont plus performants que le FBA pour prédire les taux de croissance d'*E. coli* dans divers milieux et conditions génétiques, sans qu'il soit nécessaire d'obtenir des données expérimentales supplémentaires. Les capacités et les limites des AMN sont ensuite examinées en détail. Enfin, je résume les résultats et propose des pistes pour poursuivre le développement de modèles hybrides pour les GEM, qui peuvent aider à construire des modèles de cellules entières performants et informatifs - un objectif ambitieux dans le domaine de la biologie des systèmes.

**Title:** Development of hybrid models for genome-scale metabolic networks

**Keywords:** Metabolic modeling, Machine learning, Hybrid models

**Abstract:** Over the past two decades, the systems biology community has dedicated substantial efforts to constructing genome-scale metabolic models (GEMs), which offer detailed representations of an organism's entire metabolism. GEMs present metabolism as a network, linking metabolic reactions and metabolites. Despite their wealth of information, GEMs come with notable limitations. They attempt to encompass all potential metabolic phenotypes, leading to an extensive solution space that can be challenging to explore efficiently. The predominant approach for exploiting GEMs, Flux Balance Analysis (FBA), relies on simplifications and lacks the ability to generalize across diverse conditions. In contrast, Machine Learning (ML) techniques, have gained interest for metabolic modeling, notably by harnessing large-scale -omics data to predict biological behaviors in various environments. While many approaches combine GEMs and ML together, they still separate the metabolic modeling and ML parts, limiting their adaptability and reusability. Within this Ph.D. thesis, I introduce an innovative approach that tackles this limitation: a hybrid neural-mechanistic model for GEMs, termed Artificial Metabolic Network (AMN). This entails the development of FBA surrogate methods compatible with gradient backpropagation and the creation of a mechanistic loss function to align AMN predictions with GEMs' constraints. This dissertation delves into the biological phenomena addressed by AMNs and surveys the state-of-the-art GEM utilization methods. Then, it demonstrates how AMNs outperform FBA in predicting *E. coli* growth rates across diverse media and genetic conditions, without requiring additional experimental data. The capabilities and limitations of AMNs are then thoroughly examined. Finally, I summarize the findings and offer insights into ways to pursue the development of hybrid models for GEMs, that may help in building high-performance, insightful whole-cell models—an ambitious goal in the realm of systems biology.



# Résumé étendu

Avant de lire ce résumé d'une dizaine de pages, je conseille au lecteur de commencer par celui en début de manuscrit (page 2), afin d'avoir rapidement une idée globale du projet. Ici, vous trouverez un résumé en français avec plus de détails sur le contenu des différents chapitres du manuscrit, accompagnés de quelques figures importantes. N'hésitez pas à vous référer au manuscrit complet si le résumé manque de contexte ou de références. Bonne lecture !

## Chapitre 1 : Introduction

Dans son ensemble, l'introduction propose une mise en contexte approfondie du champ de recherche dans lequel cette thèse s'inscrit, en soulignant les différentes potentialités et limites des connaissances et méthodes existantes. En particulier, je souligne le défi immense de la modélisation des systèmes métaboliques – des systèmes complexes qui nécessitent le développement d'approches innovantes pour les modéliser, et *in fine* mieux les comprendre. J'y expose également les objectifs de la thèse, notamment par la formulation de deux questions scientifiques concises.

Dans le préambule de l'introduction, je définis des notions clés et le cadre conceptuel pour la thèse. C'est une étape importante avant d'aborder les descriptions détaillées des concepts et de l'état de l'art de la recherche en modélisation métabolique. Une attention particulière est accordée à la définition des termes clés comme "métabolisme", "régulation", "phénotype métabolique"; ainsi que le "champ biologique" dans lequel le travail de thèse s'inscrit. Ces définitions fournissent le cadre conceptuel nécessaire pour naviguer dans la complexité des sujets traités par la suite.

Ensuite, la première partie de l'introduction examine les phénomènes biologiques connus qui expliquent l'apparition de divers phénotypes métaboliques en réponse à l'environnement et au matériel génétique des organismes. J'y souligne la complexité inhérente à la régulation métabolique, discutant des divers niveaux auxquels cette régulation opère, depuis les interactions moléculaires "simples" comme l'allostérie, jusqu'aux réponses cellulaires globales, par exemple via l'intervention des facteurs sigma, ou même les régulations à l'échelle des populations. Je donne par la suite une explication et une critique détaillée de l'approche réductionniste pour la compréhension du métabolisme. L'argument principal contre cette approche réside dans la simplification à outrance des processus biologiques complexes ; notamment du fait des limitations conceptuelles et expérimentales inhérentes au domaine. J'y discute les implications de cette approche différente pour la recherche en biologie et propose des perspectives alternatives, spécifiquement dans le contexte de la biologie des systèmes. L'approche holistique est introduite comme un contrepoids nécessaire à l'approche réductionniste. J'explore comment cette approche permet une meilleure compréhension des systèmes biologiques en considérant les interactions et les interdépendances entre les différentes parties de ces systèmes. Dans le cadre de la biologie des systèmes, dans lequel cette thèse s'inscrit, ces considérations sont primordiales puisque l'objectif de ce domaine est de comprendre la vie dans son ensemble et non

pas ses sous-parties comme l'approche réductionniste traditionnelle le propose. Par la suite, je me penche sur le rôle crucial du métabolisme dans la biologie des systèmes, en expliquant les concepts centraux de modélisation métabolique. La figure 1.6 ci-dessous est une figure conceptuelle importante, qui montre comment un réseau de réactions métaboliques est 'traduit' en objet mathématique de modélisation, la matrice de stoechiométrie  $S$ .

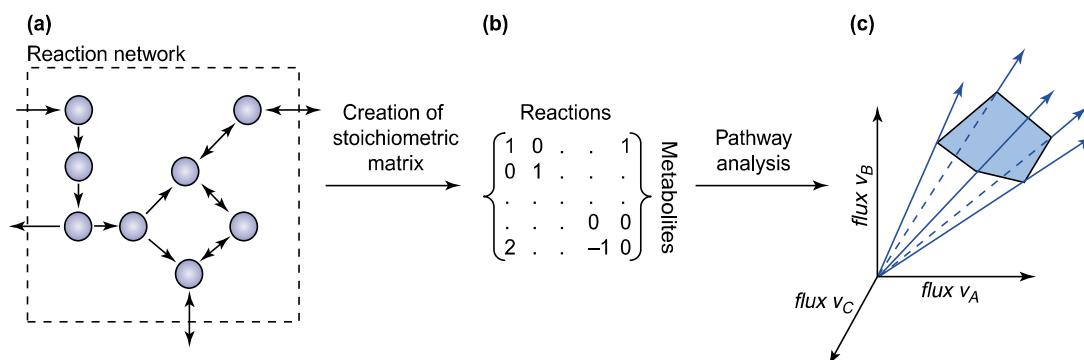


Figure 1.6: Formulation mathématique du réseau de réactions: la matrice stœchiométrique. À partir de (a) la stœchiométrie du réseau réactionnel, on dérive (b) une matrice stœchiométrique, qui peut à son tour être utilisée pour (c) une analyse mathématique des voies, par exemple en définissant l'espace des solutions stables possibles ; ici représenté pour seulement 3 flux imaginaires afin d'être facilement visualisé. (Auteurs : Papin *et al.*[1] ; réutilisé avec l'autorisation de l'auteur correspondant ; licence obtenue sur la plateforme Rightslink©).

Enfin je conclus la première partie de l'introduction en décrivant les nombreuses implications du métabolisme dans divers processus biologiques et applications industrielles. J'évoque également un champ d'application innovant: utiliser le métabolisme comme unité de traitement de signal pour la biocomputation. Je décris enfin les possibilités de ce champ d'application ainsi que les défis à relever pour une mise en œuvre réelle, qui sont encore nombreux.

Dans la seconde partie de l'introduction, je me focalise sur la description des modèles métaboliques à l'échelle du génome (GEMs). Ces modèles sont au cœur de mon travail de thèse, puisqu'ils ont constitué la base des modèles mécanistiques sur lesquels s'appuyer pour développer les modèles hybrides innovants présentés ici. Je commence par une présentation détaillée de ces réseaux métaboliques, de leur construction et de leurs différentes formulations et utilisations pour prédire et comprendre différents comportements métaboliques dans différents contextes. J'utilise ensuite l'exemple du GEM le plus récent pour *E. coli* à ce jour, iML1515, pour donner plus de détails concrets sur le contenu et la formulation mathématique du modèle. Ainsi, je donne une analyse de son contenu, de ses compartiments, et des flux métaboliques qu'il modélise. J'aborde aussi des limites inhérentes au modèle, liées à sa méthode de construction, comme les réactions de biomasse et de maintenance d'ATP, qui reposent sur des expérimentations peu nombreuses et difficilement généralisables à de nombreuses conditions.

Dans la partie finale de l'introduction, je propose une comparaison des deux approches phares de modélisation : la modélisation mécanistique et la modélisation par apprentissage automatique. Je présente chacune des approches dans le contexte de la modélisation pour la biologie, en particulier pour la biologie des systèmes, avec un accent sur les avantages et inconvénients de chaque approche. Je souligne la complémentarité des deux approches et je cite de nombreux travaux de couplage des deux approches pour la modélisation métabolique. Je prends ensuite le temps de détailler le fonctionnement des réseaux de neurones, évoquant les réseaux récurrents et les réservoirs,

qui sont d'importants concepts pour le développement des méthodes innovantes présentées dans cette thèse. Enfin, je discute de l'émergence des modèles hybrides qui combinent les approches mécanistiques et d'apprentissage automatique, en allant plus loin qu'un simple couplage : ils émergent d'une fusion entre plusieurs méthodes et approches, ce qui produit un modèle unique et pas un assemblage de plusieurs modèles. Un exemple parlant et commun est le 'Physics-Informed Neural Network' ('PINN'), un modèle hybride où des réseaux de neurones vont prédire des solutions qui concordent avec un système physique, grâce à une fonction de perte qui prend en compte le système physique. Ce genre de modèle hybride n'ayant pas encore été formulé pour les GEMs, je présente cette lacune comme la motivation centrale de mon travail de thèse. Je souligne l'importance de cette fusion pour la modélisation métabolique, ouvrant de nouvelles perspectives pour la compréhension et la manipulation des systèmes métaboliques complexes.

On appellera ces modèles Réseaux Métaboliques Artificiels (en anglais, 'AMNs'). Ils reposent sur un principe simple, décrit par la figure 1.15 ci-dessous: depuis divers types de donnée en entrée (par exemple la description d'un environnement ou de matériel génétique), un algorithme d'apprentissage automatique va prédire des paramètres pour un modèle mécanistique qui va à son tour prédire un phénotype métabolique, la sortie du modèle hybride.

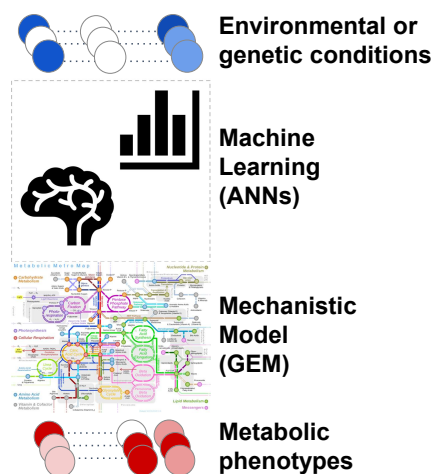


Figure 1.15: Schéma simple de l'approche de modélisation hybride sélectionnée. À partir de n'importe quel type de données d'entrée décrivant des conditions environnementales ou génétiques, un modèle d'apprentissage automatique prédit les paramètres d'un modèle mécaniste (GEM) afin de prédire les phénotypes métaboliques correspondants. Il convient de noter que la formulation de ce modèle est un modèle unique, qui ne sépare pas les parties d'apprentissage automatique et mécanistique, et peut donc être considéré comme un modèle hybride. Les détails sur la construction, l'entraînement et l'exploitation du modèle sont donnés dans les chapitres 2 et 3.

Il est très important de comprendre que l'apprentissage automatique et le modèle mécanistique doivent fonctionner ensemble comme un seul modèle, pour être appelé modèle 'hybride' comme un AMN. Dans le travail de recherche présenté ici, on utilisera des réseaux de neurones comme algorithme d'apprentissage automatique et on va appeler couche mécanistique la partie du modèle qui contient des éléments tirés d'un modèle purement mécanistique. Cette couche mécanistique consiste, pour l'AMN, à substituer les méthodes classiques d'optimisation sous contrainte comme l'analyse d'équilibre des flux (FBA). Ajouté à cela, en s'inspirant des PINNs, les AMNs vont se servir d'une fonction de perte mécanistique, qui a pour but de prédire des solutions concordantes avec les contraintes des GEMs.

Enfin, je conclus l'introduction par 2 questions scientifiques auxquelles mon travail de thèse répondra :

- Pouvons-nous développer des modèles hybrides de GEMs qui augmentent leur capacité prédictive tout en respectant leurs contraintes ?
- Quelles sont les meilleures voies d'améliorations et champs d'application possibles pour les modèles hybrides de GEMs ?

## **Chapitre 2 : Une Approche Hybride Neuronale-Mécanistique Améliorant la Puissance Prédictive des Modèles Métaboliques à l'Échelle du Génome**

Ce chapitre contient l'article publié durant la thèse dans le journal Nature Communications. Cet article présente les bases conceptuelles et les fondements mathématiques des modèles hybrides développés, les AMNs.

Dans ce chapitre, je commence par rappeler les motivations qui nous poussent à développer ces modèles hybrides, avec une introduction concise rappelant l'état de l'art en modélisation métabolique et les concepts innovants amenés par les modèles hybrides. Ensuite, les résultats principaux de l'article sont présentés, divisés en différentes parties. D'abord, une vue globale des différentes formulations et des capacités de chacune est donnée. La figure 2.1 ci-dessous résume le cadre conceptuel dans lequel s'inscrivent les AMNs. On peut y voir la différence entre le FBA classique et les couches mécanistiques ('mechanistic layers'), développés pour résoudre le même problème que le FBA avec des calculs matriciels itératifs, permettant d'intégrer la méthode dans un modèle neuronal. On voit ensuite comment ces couches mécanistiques sont intégrées dans l'AMN, et comment une approche inspirée du calcul par réservoir peut nous aider à trouver les meilleures entrées pour le FBA. Cette figure donne une vue d'ensemble pour les concepts innovants développés durant la thèse, ce qui permet de faciliter leur compréhension et de saisir rapidement le sens des différentes notations qui seront utilisées dans la suite du chapitre.

La section suivante se focalise sur le processus de développement des couches mécanistiques, apportant plus de détails mathématiques sur le fonctionnement de chacune, ainsi que l'inspiration trouvée dans la littérature pour leur développement. Ensuite, une section décrit les premiers résultats obtenus avec les AMNs, qui ont été entraînés soit avec des jeux de données simulés en figure 2.2 (par FBA performé avec des flux d'entrées tirés aléatoirement) ou des jeux de données expérimentaux en figure 2.3 (par acquisition de taux de croissance en laboratoire, avec des milieux de croissance tirés aléatoirement). On démontre ici la capacité des AMNs à apprendre depuis ces deux types de jeux de données, en fournissant des performances satisfaisantes. En d'autres termes, nous montrons que les AMNs sont capables de prédire des distributions de flux complètes à partir de données de flux partielles, tout en respectant les contraintes du GEM utilisé, via la couche mécanistique et la fonction de perte mécanistique. Dans cette section, nous soulignons également la capacité des AMNs à intégrer différents types de données d'entrée, au-delà des entrées possibles avec les GEMs. Nous démontrons cela avec un exemple tiré d'un jeu de données externe, trouvé dans la base de données ASAP, qui regroupe 17 400 taux de croissance de 145 souches cultivées dans 120 milieux différents. L'AMN est capable ici de prédire les taux de croissance avec une grande fidélité, obtenant un  $R^2$  bien supérieur au FBA, la méthode classique pour exploiter un GEM dans cette situation. Les résultats de cette approche sont visibles sur la figure 2.4 ci-dessous.

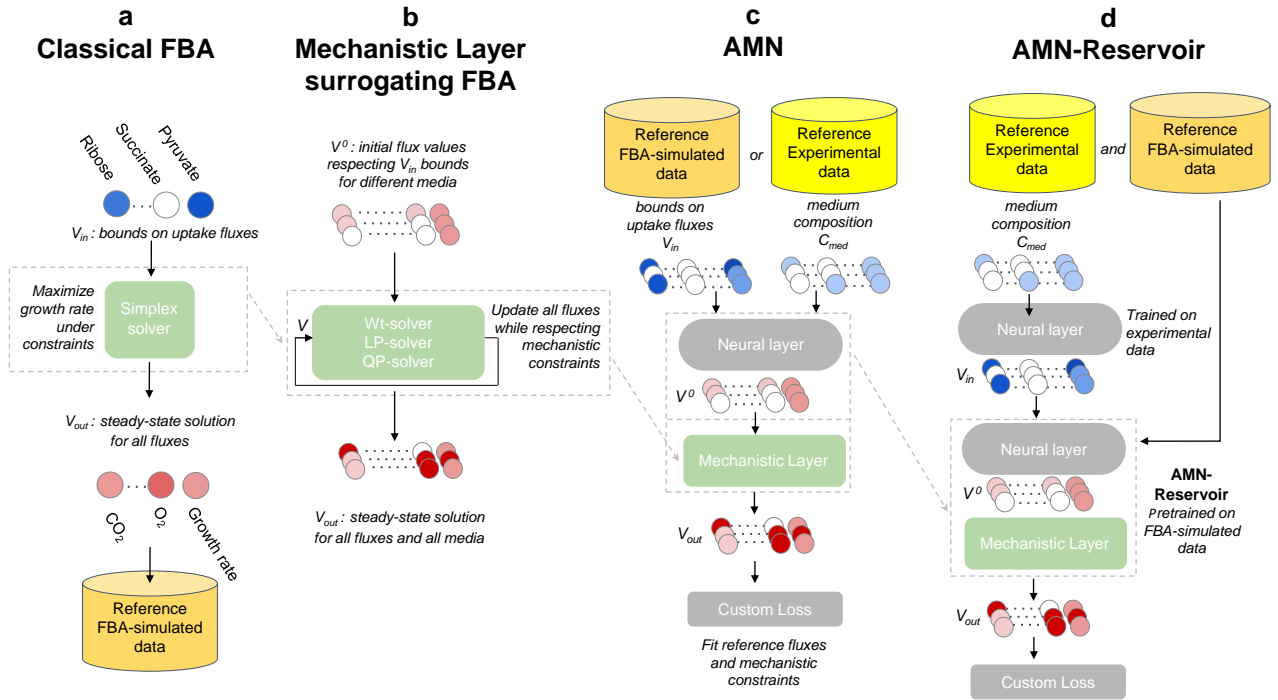


Figure 2.1: Cadres de calcul et d'apprentissage pour le FBA, les modèles mécanistiques alternatifs, l'AMN et l'AMN-Reservoir. **a.** Cadre de calcul pour le FBA classique. Le processus est répété pour chaque milieu, calculant les flux d'états stationnaires correspondants. Les cercles bleus représentent différentes limites sur les flux d'entrée des métabolites et chaque cercle rouge représente une valeur de flux à l'état stationnaire. **b.** Cadre de calcul pour les méthodes mécanistiques substituant le FBA. Les méthodes peuvent gérer plusieurs milieux de croissance à la fois. Pour tous les solveurs (Wt, LP et QP), la couche mécanistique prend en entrée un vecteur de flux initial arbitraire,  $V^0$ , respectant les limites des flux d'entrée pour différents milieux, et calcule toutes les valeurs de flux à l'état stationnaire ( $V_{out}$ ) par un processus itératif. **c.** Cadre d'apprentissage pour les modèles hybrides AMN. L'entrée (pour plusieurs milieux de croissance) peut être soit un ensemble de limites sur les flux d'entrée ( $V_{in}$ ), lors de l'utilisation de données de simulation (générées comme dans le panneau a), soit un ensemble de compositions de milieux,  $C_{med}$ , lors de l'utilisation de données expérimentales. L'entrée est ensuite passée à une couche neuronale entraînable, prédisant un vecteur initial,  $V^0$ , pour la couche mécanistique (une méthode du panneau b). À son tour, la couche mécanistique calcule la sortie finale du modèle,  $V_{out}$ . L'apprentissage est basé sur une fonction de perte personnalisée (cf. Méthodes, 2.5) assurant que les flux de référence soient ajustés (i.e., afin que  $V_{out}$  corresponde aux flux simulés ou mesurés) et que les contraintes mécanistiques (sur les limites des flux et la stœchiométrie) soient respectées. **d.** Cadre d'apprentissage pour un "AMN-Reservoir". La première étape consiste à entraîner un AMN sur des données simulées de FBA (comme dans le panneau c), après quoi les paramètres de cet AMN sont figés. Ce modèle AMN, dont le but est de substituer le FBA, est nommé "AMN-Reservoir", il est non-entraînable. À la deuxième étape, une couche neuronale est ajoutée avant  $V_{in}$  prenant en entrée les compositions de milieux,  $C_{med}$ , et apprenant la relation entre les compositions et les limites sur les flux d'entrée.

La section suivante se focalise sur les résultats obtenus avec l'approche de calcul par réservoir développé pour les AMNs. Ici, un AMN est d'abord pré-entraîné sur des simulations, afin de reproduire exactement le comportement du FBA. Ensuite, on ré-entraîne cet AMN, qu'on appelle AMN-Reservoir et qui a ses paramètres immobilisés, cette fois avec des données expérimentales. Ainsi, les entrées de l'AMN-Reservoir prédites grâce au réentraînement permettent de trouver les meilleurs flux d'entrées pour le FBA. Nous démontrons ces capacités en figure 2.5 pour deux jeux de données : un pour la régression des taux de croissance d'*E. coli*, l'autre pour la classification de la pousse de *P. putida*. Dans les deux cas, on trouve des flux d'entrées pour le FBA qui améliore ses capacités prédictives, ce qui montre l'avantage de l'approche hybride.



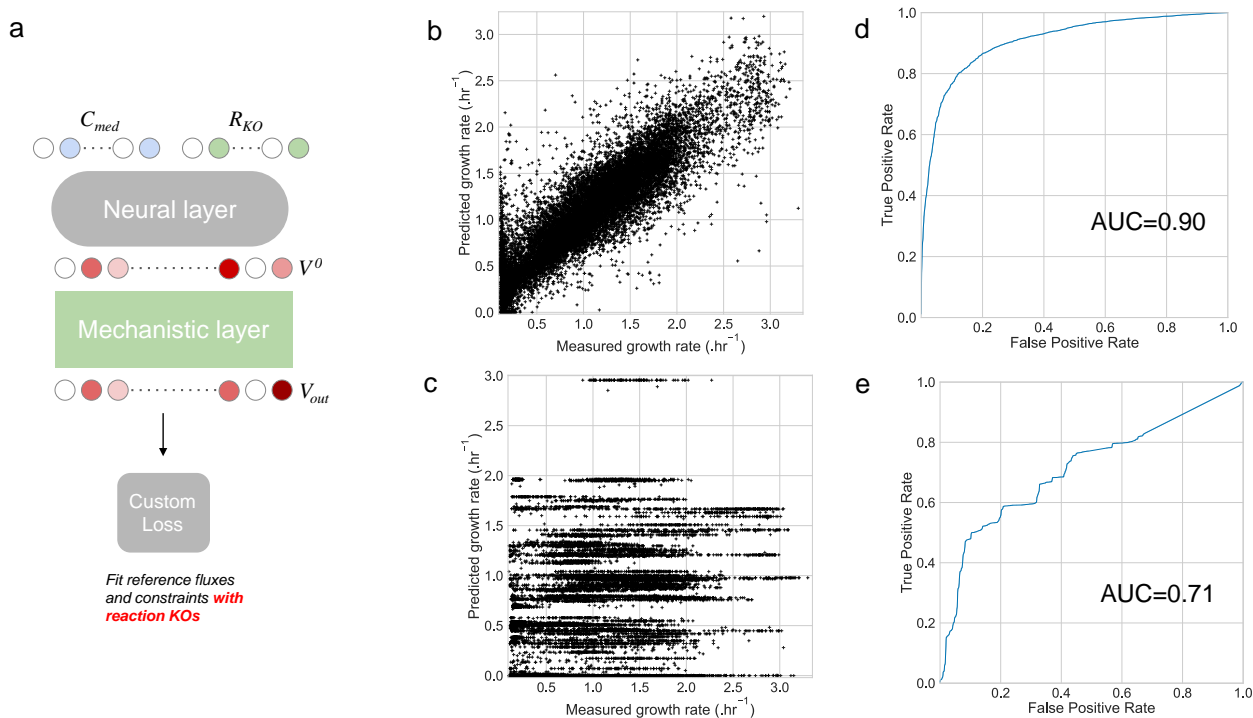


Figure 2.4: Prédictions de taux de croissance par AMNs pour les mutants de gènes KO d'*E. coli*. Un modèle AMN a été entraîné sur un ensemble de 17 400 taux de croissance d'*E. coli* cultivés dans 120 compositions de milieux uniques et 145 KO de gènes métaboliques différents. **a**. Architecture AMN intégrant les KO de gènes métaboliques. Cette architecture est similaire à celle de la Figure 2.1c, à l'exception d'une entrée secondaire ( $R_{KO}$ ) pour la couche neuronale, aux côtés de la composition du milieu  $C_{med}$ . L'entrée  $R_{KO}$  est un vecteur binaire décrivant quelles réactions sont KO. La fonction de perte personnalisée garantit que les flux de référence (*i.e.*, les taux de croissance mesurés des mutants d'*E. coli*) et les contraintes mécanistiques soient respectées et que les réactions KO expérimentalement ont une valeur de flux nulle dans  $V_{out}$ . La partie neuronale a une couche cachée de taille 500 et le modèle a été entraîné pendant 200 époques avec un dropout = 0,25, une taille de batch = 5, et l'optimiseur "Adam" avec un taux d'apprentissage de  $10^{-3}$  a été utilisé. **b**. Performance de régression de l'AMN sur les prédictions de taux de croissance agrégées à partir d'une validation croisée à 10-fold. La couche mécanistique utilisée pour cette architecture était le solveur QP. **c**. Performance de régression du FBA classique avec des bornes supérieures mises à l'échelle pour les composés présents dans le milieu et en fixant la borne supérieure et la borne inférieure à zéro pour les réactions qui sont KO (ayant une valeur de 0 dans  $R_{KO}$ ). **d**. Courbe ROC des résultats AMN. Nous avons seuillé les taux de croissance mesurés (valeurs continues) afin de les transformer en mesures binaires de croissance vs. non-croissance. **e**. Courbe ROC des résultats du FBA classique. Le même seuillage que pour le panneau d a été appliqué. Les données sources sont fournies dans un fichier de données sources (*cf.* 'Data availability').

Enfin, l'article se conclut par une discussion. D'abord, on résume les innovations apportées par l'approche hybride présentée ici : l'intégration de méthodes neuronales avec les méthodes de modélisation métabolique classiques ont permis l'émergence d'un nouveau type de modèle hybride, qui montre des performances prometteuses pour le domaine. En effet, les performances de régression et classification ont été systématiquement meilleures avec l'AMN qu'avec l'approche basique du FBA, tout en respectant les contraintes mécanistiques du modèle métabolique. Ensuite, nous revenons sur le contexte biologique afin de prendre du recul sur les tâches de modélisation que nous pouvons considérer avec les AMNs, en insistant sur le fait que nous utilisons des boîtes noires pour toutes les régulations. On rappelle ensuite l'augmentation claire de performance prédictive amenée par les AMNs, sans besoin de données expérimentales supplémentaires, contrairement aux méthodes alternatives au FBA comme le satFBA. Nous ouvrons ensuite une discussion sur les applications possibles

des AMNs, par exemple pour la construction automatisée de modèles métaboliques, notamment en trouvant de meilleurs coefficients pour la réaction de biomasse et de maintenance de l'ATP. Nous mentionnons également la capacité à réduire le nombre de mesures nécessaires en utilisant des approches hybrides comme l'AMN, comparé aux méthodes d'apprentissage automatiques classiques, ce qui permettrait éventuellement de s'attaquer à la malédiction de la dimensionnalité ("curse of dimensionality"). Enfin, nous expliquons brièvement comment de nombreux projets de bio-ingénierie, avec des portées biomédicales, alimentaires ou environnementales, pourraient bénéficier d'approches hybrides comme les AMNs. En effet, leurs meilleures capacités prédictives pourraient être utiles à tous les projets d'optimisation, tout en apportant des pistes de compréhension et d'amélioration par les connaissances intégrées aux modèles.

La suite de l'article présente de nombreux détails sur les méthodologies employées. Le fonctionnement des couches mécanistiques est précisé grâce à des descriptions mathématiques plus formelles des algorithmes utilisés. Le processus d'entraînement des AMNs y est également décrit en détail, de la génération de jeux de données expérimentales ou simulés, jusqu'à la conception de l'architecture des réseaux de neurones, et les approches utilisées pour intégrer ces systèmes avec les réseaux métaboliques existants. Un aspect méthodologique important du projet est le développement de fonctions de pertes personnalisées, qui permettent à l'AMN de respecter les contraintes du réseau métabolique avec des réseaux des neurones. Ainsi, cette partie explique comment la fusion des approches mécanistiques et des techniques d'apprentissage automatique a été rendue possible. La méthodologie pour que les AMNs fonctionnent dans un cadre de calcul en réservoir est également décrite. Enfin, on y décrit aussi l'acquisition et le contenu des différents jeux de données expérimentaux utilisés dans l'article. Par exemple, le détail de chaque condition expérimentale est donné, ainsi que les lois de probabilité qui ont permis le tirage aléatoire des conditions. Le chapitre se conclut en donnant un accès ouvert aux données et au code utilisés dans l'article, pour assurer la transparence, la collaboration et l'évaluation du travail de recherche présenté ici.

## **Chapitre 3 : Évaluation et Amélioration des Modèles Hybrides pour Exploiter les GEMs**

Ce chapitre a pour but d'approfondir l'évaluation des AMNs, leurs capacités et limites, ainsi que d'apporter des améliorations ou des pistes d'amélioration pour une utilisation plus large et plus performante de l'approche hybride présentée dans le chapitre précédent. Chaque section se concentre sur un problème spécifique identifié comme significatif et important à évaluer. Pour chacun de ces problèmes, je donne une introduction qui rappelle les éléments contextuels importants, puis la méthodologie pour évaluer (et éventuellement corriger) ce problème, avant de présenter les résultats obtenus et une discussion autour de ceux-ci.

Avant de présenter la première limite et son amélioration, je présente dans un préambule les différents jeux de données utilisés dans ce chapitre, comment ils ont été générés (table 3.1) ainsi que les mesures utilisées spécifiquement pour évaluer les performances de l'AMN au-delà de la capacité prédictive du modèle. La mesure la plus intéressante utilisée dans ce chapitre est la norme de SV (le produit matriciel du vecteur de flux métaboliques  $V$  avec la matrice de stoechiométrie  $S$ ), qui indique à quel point une distribution de flux respecte les contraintes du modèle métabolique. En effet, dans le chapitre précédent, cette norme était mentionnée mais elle n'a pas été interprétée et comparée

dans de nombreuses situations. Dans le préambule, je présente cette mesure et indique sa haute sensibilité aux variations de flux sur une distribution de flux à l'équilibre (figure 3.1).

La première limitation s'intéresse au solveur QP (une des couches mécanistiques utilisée dans le chapitre précédent). Il a été identifié que certaines contraintes ne sont pas respectées par ce solveur (figure 3.3), ce qui nous motive à reformuler la méthode. Dans cette partie, la reformulation est décrite et comparée à la formulation originale, puis on montre comment la reformulation corrige le respect des contraintes (figure 3.3), et comment elle amène une nouvelle fonction de perte mécanistique. Cette nouvelle fonction de perte mécanistique, plus fiable pour le respect des contraintes, est utilisée pour formuler un nouvel AMN-QP (figure 3.4), qui sera utilisé dans la suite du chapitre.

La deuxième partie explore comment les couches mécanistiques influent sur les prédictions des AMNs. En testant différents nombres d'itérations avec les différentes couches mécanistiques, on observe des effets très différents sur les solutions prédites par les AMNs. Globalement, les résultats montrent un large avantage pour la couche mécanistique LP qui permet de réduire grandement la norme de SV, et donc d'améliorer le respect des contraintes dans les prédictions, comparé aux couches QP et Wt (figure 3.5). Malheureusement ces observations n'ont pas pu être réalisées avec de grands réseaux métaboliques comme iML1515 ; elles sont pour l'instant limitées aux petits réseaux. Une autre limite observée pour les couches mécanistiques est le temps de calcul nécessaire pour entraîner un modèle qui contient beaucoup d'itérations sur sa couche mécanistique. En effet, cela va augmenter exponentiellement le temps d'entraînement et potentiellement rendre impossible la tâche d'apprentissage.

La troisième partie de ce chapitre se penche sur l'optimisation des hyperparamètres utilisés pour les AMNs. Ces nombreux hyperparamètres peuvent concerner la partie neuronale du modèle : le nombre de couches neuronales utilisées, leurs tailles, ainsi que leurs fonctions d'activation et leur taux de 'dropout'. Mais ils peuvent aussi concerner la partie mécanistique, en appliquant des poids pour chaque terme de la fonction de perte (table 3.3). Ainsi, l'importance de chaque terme dans la valeur finale de fonction de perte peut être modulée. En considérant l'optimisation des hyperparamètres neuronaux et mécanistiques séparément, je décris ici plusieurs processus d'optimisation des hyperparamètres, qui permettent d'améliorer substantiellement les performances des AMNs (figure 3.6-7). Enfin, je discute des résultats en comparant les normes de SV obtenues après optimisation des hyperparamètres avec des résultats obtenus par la méthode mécanistique classique du FBA. On peut observer un écart significatif entre les normes de SV obtenues, ce qui ouvre une discussion quant à l'interprétabilité et la fiabilité des résultats prédits par l'AMN. Dans cette discussion, il est important de rappeler la sensibilité de la norme de SV aux perturbations, comme montré dans le préambule (figure 3.1). La figure 3.10 ci-dessous illustre bien cette limite des AMNs, en mettant en avant la performance prédictive très basse du FBA sous un respect strict des contraintes du GEM utilisé, ainsi que la bonne performance prédictive de l'AMN, qui ne respecte pas strictement les contraintes du GEM utilisé.

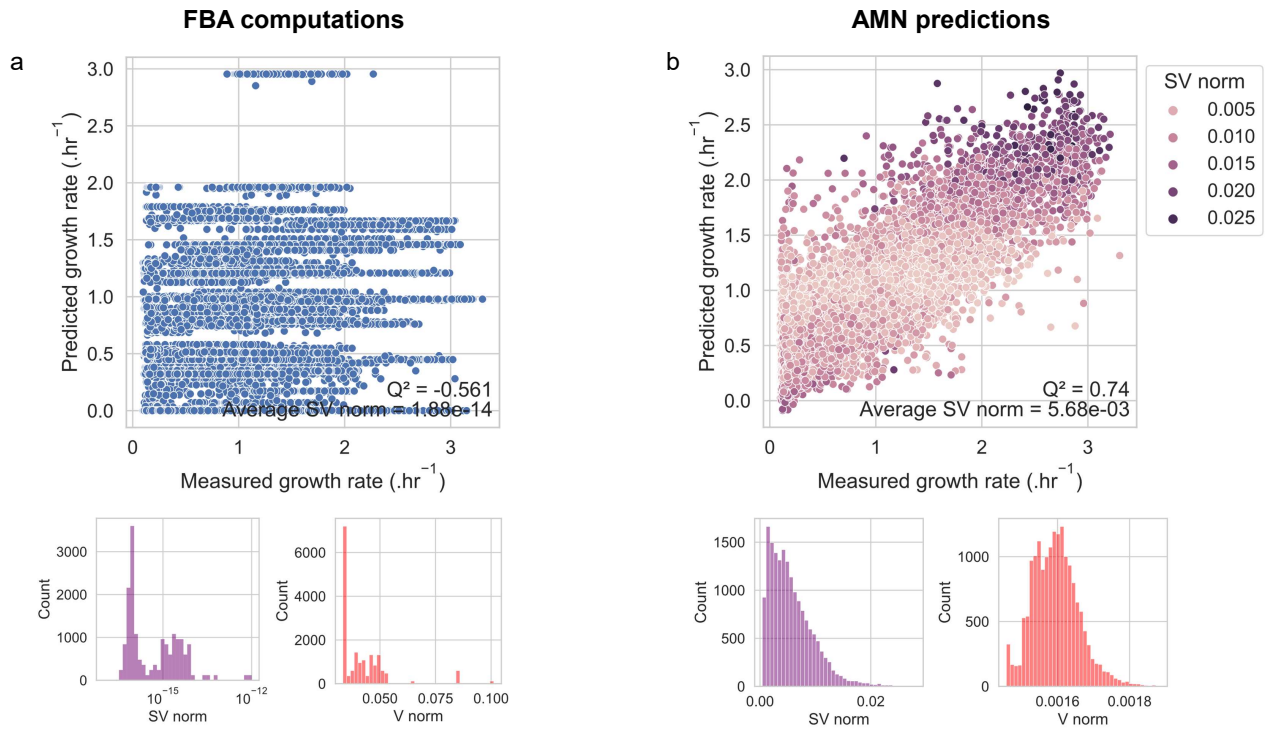


Figure 3.10: L'AMN surpasse le pouvoir prédictif du FBA avec les données d'apprentissage "Cov-BiologKO", mais ne respecte pas strictement les contraintes de stœchiométrie. (a) FBA seul avec des limites arbitraires sur les flux d'entrée (comme dans la Figure 2.4c) (b) AMN (approche classique, comme dans la Figure 2.4b). Les deux panneaux affichent, de haut en bas et de gauche à droite : un graphique des prédictions du taux de croissance (axe Y) par rapport aux taux de croissance mesurés expérimentalement (axe X), avec les couleurs des points correspondant à la valeur de la norme de SV calculée sur la prédiction ; un histogramme de la norme de SV pour toutes les prédictions et un histogramme de la norme de V pour toutes les prédictions. Pour les calculs de FBA, la norme V moyenne est de 0,042 et pour les prédictions de l'AMN, la norme V moyenne est de 0,0016. L'AMN surpasse le FBA en termes de  $Q^2$  mais les contraintes stœchiométriques ne sont pas bien respectées, avec une norme SV moyenne de  $5,68 \times 10^{-3}$ , contre une norme SV moyenne de  $1,88 \times 10^{-14}$  pour le FBA.

La partie suivante du chapitre 3 explore les capacités de méthodes d'apprentissage automatique alternatives aux réseaux de neurones, dans la tâche de construire un modèle de substitution au FBA. Deux méthodes vont être comparées aux réseaux de neurones: XGBoost ('extreme gradient-boosted trees') et MTEN ('Multitask Elastic Net regression'). Ces deux méthodes sont décrites en détail, et leurs sources sont citées. La comparaison avec les réseaux de neurones se fait par plusieurs mesures : la capacité prédictive sur le growth rate, la norme de SV et la norme de V. Pour une discussion plus riche, ces méthodes alternatives vont être testées avec des données d'apprentissage plus ou moins complexes. Globalement, on peut conclure, au vu des résultats présentés en figure 3.8, que la méthode MTEN a un fort potentiel pour devenir un substitut au FBA, par sa capacité à atteindre des valeurs de norme de SV bien plus basses. Cependant, comme montré dans la figure 3.9, cette méthode est sensible à la complexité des données d'apprentissage utilisées. En résumé, cette partie souligne la diversité des approches possibles pour obtenir des substituts de modèles métaboliques mécanistiques, et leurs différents potentiels pour améliorer la fiabilité des prédictions effectuées.

Dans la cinquième partie du chapitre 3, j'ai exploré la dépendance des prédictions des AMNs aux données d'apprentissage. Ce problème étant une limite commune à tous les modèles d'apprentissage automatique, il est important d'explorer son effet ici. En particulier, j'ai comparé la performance des AMNs lorsque le 'training' et le 'validation set' étaient choisis aléatoirement, avec un cas où le

'validation set' contenait des conditions nouvelles, absentes du 'training set'. Concrètement, dans ce dernier cas, la tâche de l'AMN est de prédire l'effet de conditions de cultures ou de modifications génétiques sur les flux métaboliques, sans avoir pu apprendre de ces conditions au préalable. Comme attendu, la performance de l'AMN est significativement dégradée dans ces cas (figures 3.10-12), ce qui constitue une des principales limites de l'AMN si il est comparé aux méthodes mécanistiques. Cette évaluation est cruciale pour comprendre la portée et les limites des AMNs dans des applications réelles et expérimentales.

Pour conclure, je propose une synthèse des observations et conclusions faites dans ce chapitre, puis une brève description des perspectives qui découlent de ces conclusions. En bref, les points les plus importants du chapitre sont les suivants : (i) des solveurs mécanistiques alternatifs peuvent être formulés, ainsi que des fonctions de pertes mécanistiques personnalisées alternatives pour les AMN, qui modifient le comportement général du modèle, (ii) les couches mécanistiques ont un fort potentiel pour améliorer les prédictions des AMN mais nécessitent encore des améliorations, (iii) l'optimisation des hyperparamètres peut améliorer les performances d'un AMN, mais les AMNs prédisent encore des distributions de flux loin de respecter les contraintes des GEMs comme le FBA, (iv) le modèle MTEN fonctionne très bien pour remplacer le FBA, notamment en termes de contraintes stœchiométriques, et (v) un AMN peut prédire l'effet de nouvelles compositions de médias ou perturbations génétiques, mais cela dépend grandement des motifs statistiques trouvés dans les données d'apprentissage. Pour finir, je décris le potentiel de l'architecture AMN-Reservoir pour résoudre les défis associés aux AMNs (figure 3.13). Cette conclusion ouvre des perspectives sur de futures directions de recherche et d'application des modèles hybrides dans la biologie des systèmes, ce qui sera plus longuement discuté dans le chapitre suivant.

## Chapitre 4 : Discussion Générale et Perspectives

Ce chapitre est divisé en deux parties. Il présente d'abord une vue globale des capacités et limites des AMNs. Je commence par un rappel de leur fonctionnement innovant qui permet d'augmenter le pouvoir prédictif des GEMs, puis je mentionne les investigations nécessaires pour aboutir à un modèle plus fiable et réutilisable par la communauté de biologie des systèmes. Ensuite, je discute des voies possibles d'amélioration pour dépasser ces limites, et des champs d'application les plus adaptés aux AMNs. Pour conclure, je propose un résumé global de la thèse, et la réponse aux deux questions scientifiques énoncées au chapitre 1.

La première section commence par récapituler les différentes contributions significatives apportées par l'utilisation des AMNs pour l'exploitation des GEMs. De ce fait, je souligne le passage d'un principe d'optimalité à un principe d'apprentissage dans la modélisation métabolique, ce qui constitue un changement d'approche radical, qu'on pourrait éventuellement qualifier de changement de paradigme. Ensuite, on souligne l'efficacité des AMNs à prédire des phénotypes métaboliques complexes à partir d'ensemble de conditions et de mesures de flux partielles. On rappelle également que les AMNs peuvent accepter une bien plus grande diversité de données d'entrée et de sortie que les GEMs, ce qui en fait des modèles plus versatiles et plus largement utilisables. Leurs architectures sont également personnalisables selon l'usage, ce qui renforce encore plus le caractère réutilisable du modèle.

Ensuite, je résume les limites principales des AMNs. En effet, l'apprentissage avec contraintes est un défi de modélisation, dont les limites doivent être soigneusement énoncées. D'abord, je rappelle

que la fonction de perte mécanistique repose sur un équilibre entre les termes de ‘fitting’ et de contraintes. J’y mets en évidence la complexité du compromis nécessaire pour équilibrer les termes de perte personnalisée, pour respecter les contraintes des GEMs. Ensuite je rappelle qu’en l’état actuel, l’AMN ne respecte pas les contraintes mécanistiques comme les méthodes classiques d’optimisation sous contrainte, comme le FBA. Puis je souligne les limites des couches mécanistiques, ainsi que la dépendance des performances de l’AMN au contenu des données d’apprentissage utilisées. Je conclus cette première section du chapitre 4 par la description du potentiel de l’AMN-Reservoir pour résoudre les différentes limites mentionnées, mais je signale aussi que cette approche n’est pas ‘hybride’ à proprement parler, mais plutôt un couplage d’approche neuronale et mécanistique. En ce sens, cette approche est moins innovante que les AMNs ‘classiques’.

La deuxième section se focalise sur les perspectives d’amélioration et d’application possibles pour les AMNs. Je commence par lister les améliorations souhaitables dans les approches et formulations développées pour cette thèse. En premier lieu, une acquisition plus standardisée des données d’apprentissage, par des approches plus rationnelles (en utilisant plus de méthodes de design expérimental), automatisées (par robotisation) et globales (en élargissant les espaces combinatoires). Ensuite, j’explique comment et par quelles inspirations il est possible d’améliorer les performances des fonctions de perte et des couches mécanistiques. Par la suite, je décris les voies d’amélioration par le remplacement des réseaux de neurones par des méthodes alternatives, et par l’utilisation d’une version améliorée de l’architecture AMN-Reservoir ou des alternatives à celle-ci, comme l’apprentissage actif ou les algorithmes génétiques (figures 4.2-3). La dernière sous-section de la thèse s’intéresse aux applications les plus adaptées à l’AMN. On y liste notamment l’intégration de données omiques, le développement automatisé de modèles métaboliques, l’intégration de séries temporelles pour modéliser les processus dynamiques du métabolisme, le guidage d’optimisation de bioproduction et la biocomputation. Enfin, j’ouvre la question des directions à prendre pour la modélisation en biologie des systèmes, en discutant du potentiel des modèles hybrides pour construire des modèles de cellules entières. Voici mon point de vue personnel sur la direction à prendre pour la biologie des systèmes dans les années suivantes : développer des modèles hybrides de cellules entières, intégrant divers jeux de données omiques au moyen de méthodes de ML, qui permettent la simulation mécanistique précise de processus biologiques de plus en plus détaillés à petite échelle, ainsi que des traits phénotypiques à plus grande échelle. Pour ce faire, les questions de standardisation de données sont centrales, afin de garantir la réutilisabilité et la pertinence des modèles ; et une compétition internationale similaire à CASP pourrait stimuler l’émergence de nouvelles approches.

Je conclus la thèse par les réponses aux questions scientifiques énoncées au chapitre 1:

- Pouvons-nous développer des modèles hybrides de GEMs qui augmentent leur capacité prédictive tout en respectant leurs contraintes ?

Même si le pouvoir prédictif des AMN est impressionnant par rapport au FBA seul, il doit encore être approfondi pour respecter de manière plus fiable les contraintes des GEMs de grande taille et être éventuellement utilisé par la communauté de modélisation métabolique comme une méthode standard.

- Quelles sont les meilleures voies d’améliorations et champs d’application possibles pour les modèles hybrides de GEMs ?

De nombreuses améliorations peuvent être apportées dans cette direction, et je pense donc que les modèles hybrides de GEM ont un grand potentiel dans une vaste gamme de projets, de la construc-

tion de modèles métaboliques à l'optimisation de la bioproduction, et même pour le développement de modèles de cellules entières plus performants, le Graal de la biologie des systèmes.

Enfin, je donne quelques mots de conclusion, afin de prendre du recul sur le travail de thèse réalisé et mieux l'inscrire dans son domaine de recherche. En voici les derniers mots: "L'avenir de la recherche en biologie sera guidée par ordinateur, et la puissance de l'intelligence artificielle pour des tâches de plus en plus complexes appelle à poursuivre l'intégration de l'IA dans les modèles biologiques. Cette thèse est une étape de cette voie, qui mènera éventuellement à une meilleure compréhension et exploitation des organismes. J'espère que cela pourra accélérer les percées biotechnologiques et médicales, pour un avenir plus sûr et plus durable."

*J'aimerais dédier cette thèse à ma famille et à mes amis.  
Je vous aime fort, merci pour tout.*





# Remerciements

First of all, I would like to thank all jury members for accepting to review and examine my Ph.D. dissertation, and attend the defense. I sincerely hope that your interest will be picked, and that you will enjoy reading through this dissertation.

Next, I would like to thank my supervisors, Jean-Loup and Wolfram; who gave me the Ph.D. opportunity, then supported and mentored me throughout the 3 years. Jean-Loup, thank you for having guided the project, that helped me so much for publishing in time, and thank you for all your machine learning codes and advices. Wolfram, thank you for your unshakable support and kindness, and for your many crash courses on metabolic modeling. Many thanks for your guidance, I hope that our paths cross again in the future. Even though the 3 years were not a long steady river, I will keep a good memory of it, and I feel like I greatly learned from both of you. To conclude, here is a pun I was recently told: during this Ph.D., I was scientifically raised by two wolves... make of it what you will!

I would like to thank the MICALIS department of INRAe and the metaprogram DIGIT-BIO for financing my Ph.D. and proposing stimulating scientific environments for these 3 years. I am proud to have been a part of these structures, which helped me develop my scientific expertise and curiosity.

Finishing this Ph.D. would have not been possible without the amazing support I had from the BioRetroSynth team. Bastien, Manish, Mahnaz and Thomas, you were amazing colleagues to work with and I learned a lot from you, I deeply thank you. Bastien, I don't think I could ever thank you enough, and I am not sure I would be writing these lines without you. Starting as an intern working with me and you are now taking the lead as a Ph.D. student, it's super nice. I am very lucky to have worked with you, you brought so much to the project, it tremendously helped me. I wish you all the best for your Ph.D. and the next adventures, I am sure you will greatly succeed!

I also want to thank BRS team members and MICALIS mates with whom we shared good moments. I will keep a good memory of my Ph.D. time thanks to you! Abhinav, Paul (Soudier and Ahavi!), Angelo (Cardoso and Banares!), Yorgo, Alexandra, Alicia, Laetitia, Anne, Nolwenn, Joan, Ioana, Nacer, Hadi, Inès and Pierre: thank you so much, it was great being with you. I will all miss you a lot.

Finally, I would like to thank all the ADAS football association members. It was a real pleasure to re-discover my love of playing football with you, only good times! It really helped me keep some physical activity during these 3 years, which was very valuable. Thank you all, I will miss the Thursday noon with you a lot!

I would also like to deeply thank the two interns I supervised: Maria and Tom. More than helping me doing the experimental work, you pushed me reflect on the project and we shared good times together. Having you working with me was a great pleasure and I am sure you will both be very successful in your career. I wish you all the best, and I hope to cross your path someday!

Je passe en français pour mes proches!

Pour commencer, j'aimerais avoir un mot pour Mamie qui nous a quittés avant qu'elle me voie docteur. Elle s'en faisait déjà une fierté, pleine de confiance en ma réussite. Je t'aime fort Mamie, tu nous manques tous beaucoup.

Maman et Gilles, vous avez été des vraies bouées de sauvetage pour moi durant ces 3 ans et particulièrement ces derniers mois pour écrire le manuscrit. Vous avez été d'un soutien et d'une écoute sans faille... Je suis tellement reconnaissant, j'ai énormément de chance de vous avoir, je ne pourrai pas assez vous remercier ici. Si j'écris ces lignes c'est en grande partie grâce à vous, merci du fond du cœur.

Papa, tu as toujours été là pour moi, ton soutien moral et psychologique m'a énormément aidé. En particulier pour me reconforter après ma rupture, et pendant mes moments de doutes sur la thèse, t'avoir comme soutien a été une aide précieuse, merci pour tout.

Mes frères et ma soeur, un énorme merci pour votre soutien. Émile, ton parcours a été un exemple à suivre, tu m'as tellement aidé pour aller au bout de cette thèse, je t'en serai toujours reconnaissant. Et les concerts qu'on a vécus ensemble (que tu sois sur scène ou dans la fosse) m'ont bien aidé à décompresser ! Olive, on a partagé des super moments et tu m'as soutenu pendant ces 3 ans, merci beaucoup. Mathilde, tu m'as ramassé à la petite cuillère après ma rupture et tu as su me rassurer, et même me redonner espoir ; j'oublierai jamais, merci. Camille, je suis super content que tu aies trouvé ta voie dans la musique, hâte que tu perces. Paul, je suis super fier de toi et de comment tu grandis, c'est impressionnant ; j'ai hâte de la suite même si ça va un peu trop vite ! Jeanne et Sélim, merci de m'avoir toujours encouragé et soutenu pendant ces 3 ans, je vous souhaite que le meilleur pour la suite !

J'aimerais maintenant remercier ma famille montpelliéraine: Mivette, Philippe, Tatine, Tonton, Leda, Stéphane, Lucas, Marek, Violette. Ça a toujours été un plaisir de vous rendre visite dans le Sud ! Merci pour votre soutien et votre accueil chaleureux pendant ces 3 ans. J'aimerais te remercier particulièrement Mivette, tu as toujours été un soutien sans faille pour ma réussite, et une oreille attentive à tous les problèmes que j'ai pu avoir, merci beaucoup.

Pour finir, j'aimerais remercier mes amis, qui ont toujours été là pour me soutenir, pour m'aider à m'évader, et pour me redonner confiance. Arthur, François, Pablo, Bizo, Nico, Max, Ben, Daniel, Elliot, Julie, Nina, Coco, Niko, Zélie, Adèle, Prisc, Diane et tous ceux qui m'ont soutenu: un immense merci. J'en serais pas là sans vous. Hâte qu'on se marre et qu'on vive plus d'aventures ensemble. J'aimerais remercier spécialement Tanguy, qui m'a tellement boosté pour en arriver ici. C'est grâce à toi que j'écris ces lignes mon reuf, je t'en dois une belle.

# Contents

<b>1</b>	<b>Introduction</b>	<b>31</b>
1.1	Preamble: defining important terms and scopes . . . . .	31
1.1.1	Metabolism . . . . .	31
1.1.2	Regulation . . . . .	32
1.1.3	Metabolic phenotype . . . . .	32
1.1.4	Biological scope . . . . .	32
1.2	Metabolism: more than breaking down nutrients . . . . .	33
1.2.1	From environmental cues to metabolic phenotypes: a concert of regulatory mechanisms . . . . .	33
1.2.1.1	A myriad of well-described regulation mechanisms: the reductionist approach . . . . .	33
1.2.1.1.1	Metabolism regulation is more complex than controlling individual enzymes activities . . . . .	33
1.2.1.1.2	Enzyme activity regulation by post-translational modifications . . . . .	34
1.2.1.1.3	Gene expression regulation: the <i>lac</i> operon and carbon catabolite repression examples . . . . .	35
1.2.1.1.4	Gene expression regulation: review of the main mechanisms . . . . .	37
1.2.1.1.5	Limitations of the reductionist approach . . . . .	38
1.2.1.2	Whole-cell regulation: the holistic approach . . . . .	39
1.2.1.2.1	Gene Regulatory Networks . . . . .	39
1.2.1.2.2	Limitations of GRNs . . . . .	40
1.2.2	Modern research on metabolism: embracing complexity . . . . .	41
1.2.2.1	Shaping phenotypes with a purpose: a complex endeavor . . . . .	41
1.2.2.1.1	Synthetic biology and metabolic engineering: industrially relevant fields . . . . .	41
1.2.2.1.2	Systems biology: understanding life as systems . . . . .	42
1.2.2.2	Metabolism as a signal-processing unit in biocomputing . . . . .	43
1.2.2.2.1	Definition of signal-processing and considered scope of metabolism . . . . .	43
1.2.2.2.2	Open discussion on signal-processing abilities of life . . . . .	43
1.3	Genome-scale metabolic networks: state-of-the-art metabolic modeling . . . . .	44
1.3.1	Overview . . . . .	44
1.3.1.1	Metabolic models in the postgenomic era . . . . .	44
1.3.1.2	Steady-state or dynamic formulations to exploit GEMs . . . . .	46
1.3.1.3	A variety of ways to exploit GEMs, with different biological scopes . . . . .	47
1.3.2	Detailed content of the state-of-the-art <i>E. coli</i> GEM: iML1515 . . . . .	48
1.3.2.1	General content . . . . .	48

1.3.2.2	Compartments . . . . .	49
1.3.2.3	Uptake fluxes . . . . .	49
1.3.2.4	ATP maintenance . . . . .	49
1.3.2.5	Biomass reaction . . . . .	49
1.3.3	Measuring and predicting metabolic phenotypes with GEMs . . . . .	50
1.3.3.1	GEMs rely on two main constraints . . . . .	50
1.3.3.2	Metabolic Fluxes Analysis: predicting metabolic phenotypes from flux measures . . . . .	51
1.3.3.3	Flux Balance Analysis: predicting metabolic phenotypes from an optimality principle . . . . .	51
1.3.3.3.1	Computation framework overview . . . . .	51
1.3.3.3.2	Linear Programming for FBA . . . . .	52
1.3.3.3.3	FBA variations: different computation frameworks and biological scopes . . . . .	53
1.3.3.3.4	Optimality principle in FBA: a questionable approach . . . . .	54
1.4	Hybrid models: reconciling mechanistic and machine learning models . . . . .	55
1.4.1	Machine learning and mechanistic models: two seemingly opposed approaches . . . . .	56
1.4.1.1	Mechanistic modeling for biology . . . . .	56
1.4.1.1.1	Overview . . . . .	56
1.4.1.1.2	Categories of MM . . . . .	56
1.4.1.1.3	Compelling MM achievements in biology . . . . .	56
1.4.1.1.4	MMs limitations . . . . .	57
1.4.1.2	Machine learning for biology . . . . .	57
1.4.1.2.1	Overview . . . . .	57
1.4.1.2.2	Categories of ML models . . . . .	58
1.4.1.2.3	Compelling ML models achievements in biology . . . . .	58
1.4.1.2.4	ML models limitations . . . . .	58
1.4.1.3	Artificial Neural Networks . . . . .	59
1.4.1.3.1	Biological inspiration and mathematical foundations . . . . .	59
1.4.1.3.2	Gradient backpropagation for learning abilities . . . . .	62
1.4.1.3.3	Recurrent Neural Networks . . . . .	63
1.4.1.3.4	Reservoir computing . . . . .	63
1.4.2	Mechanistic and Machine Learning models as complementary approaches for metabolic modeling . . . . .	64
1.4.2.1	Combining machine learning and mechanistic approaches . . . . .	64
1.4.2.1.1	State-of-the-art approaches . . . . .	64
1.4.2.1.2	A gap between MM and ML in combined approaches . . . . .	65
1.4.2.2	Hybrid models: towards a fusion of mechanistic and machine learning approaches . . . . .	66
1.4.2.2.1	Overview . . . . .	66
1.4.2.2.2	Historical use of hybrid models . . . . .	67
1.4.2.2.3	Hybrid models for systems biology . . . . .	67
1.4.2.2.4	Hybrid model of GEMs: a research gap . . . . .	67
1.5	Research questions and content of the dissertation . . . . .	69

<b>2</b>	<b>A hybrid neural-mechanistic approach improving the predictive power of genome-scale metabolic models</b>	<b>71</b>
2.1	Abstract . . . . .	71
2.2	Introduction . . . . .	72
2.3	Results . . . . .	74
2.3.1	Overview of AMN hybrid models . . . . .	74
2.3.2	Alternative mechanistic models to surrogate FBA . . . . .	75
2.3.3	AMNs: metabolic & neural hybrid models for predictive power with mechanistic insights . . . . .	77
2.3.4	AMNs can be trained on experimental datasets with good predictive power . . .	78
2.3.5	AMNs can be used in a reservoir computing framework to enhance the predictive power of traditional FBA solvers . . . . .	82
2.4	Discussion . . . . .	83
2.5	Methods . . . . .	85
2.5.1	Making metabolic networks suitable for neural computations . . . . .	85
2.5.2	Generation of training sets with FBA . . . . .	86
2.5.3	Derivation of loss functions . . . . .	87
2.5.4	Wt-solver . . . . .	88
2.5.5	LP-solver . . . . .	88
2.5.6	QP-solver . . . . .	89
2.5.7	ANN architecture . . . . .	90
2.5.8	AMN architectures . . . . .	90
2.5.9	ANN and AMN training parameters . . . . .	90
2.5.10	Searching uptake fluxes upper bounds in FBA . . . . .	91
2.5.11	Generation of an experimental training set . . . . .	91
2.5.12	Culture conditions . . . . .	91
2.5.13	Growth rates determination . . . . .	92
2.5.14	External training sets acquisition . . . . .	92
2.6	Statistics & reproducibility . . . . .	93
2.7	Data availability . . . . .	94
2.8	Code availability . . . . .	94
2.9	Acknowledgements . . . . .	94
2.10	Author contributions . . . . .	94
2.11	Competing interests . . . . .	95
<b>3</b>	<b>Further assessment and improvements of hybrid models to exploit GEMs</b>	<b>97</b>
3.1	Preamble: training sets and metrics presentation . . . . .	98
3.1.1	Training sets . . . . .	98
3.1.2	$R^2$ and $Q^2$ . . . . .	98
3.1.3	SV norm, V norm, and Euclidean distance . . . . .	99
3.1.4	Metrics interpretability . . . . .	100
3.2	An improved QP-solver formulation to better respect GEMs' constraints . . . . .	101
3.2.1	Introduction . . . . .	101
3.2.2	Methods . . . . .	101
3.2.2.1	QP-solver reformulation . . . . .	101
3.2.2.2	Optimization of solvers' terms weighting . . . . .	102

3.2.2.3	Comparison of QP-solver and QP-bnds-solver for respecting GEMs' constraints . . . . .	103
3.2.3	Results and discussion . . . . .	103
3.2.3.1	Tradeoff between the fitting and SV loss terms . . . . .	103
3.2.3.2	No significant improvement of the MM solver by loss terms weights optimization . . . . .	104
3.2.3.3	The QP-bnds-solver enhances the respect of GEMs' constraints . . . . .	104
3.2.3.4	AMN-QP-bnds formulation better respects GEMs constraints and is more versatile . . . . .	106
3.3	How much do mechanistic layers improve AMN predictions? . . . . .	107
3.3.1	Introduction . . . . .	107
3.3.2	Methods . . . . .	108
3.3.2.1	Training set and models architectures . . . . .	108
3.3.2.2	Varying mechanistic layers iterations . . . . .	108
3.3.3	Results and discussion . . . . .	108
3.3.3.1	The mechanistic layer improves predictions of AMNs, with a large advantage for AMN-LP . . . . .	108
3.3.3.2	Current implementations have strong limits . . . . .	109
3.4	Fine-tuning AMNs by hyperparameter optimization . . . . .	110
3.4.1	Introduction . . . . .	110
3.4.2	Methods . . . . .	110
3.4.2.1	Training sets and models architectures . . . . .	110
3.4.2.2	Hyperparameters space and optimization framework . . . . .	111
3.4.3	Results and discussion . . . . .	111
3.4.3.1	Optimization of neural layer hyperparameters and weights for custom loss terms can lead to better AMN performance . . . . .	111
3.4.3.2	Hyperparameter optimization depends on the training set . . . . .	113
3.4.3.3	AMNs predictions are not respecting constraints like FBA computations . . . . .	113
3.5	Exploration of alternative ML methods to better surrogate FBA . . . . .	114
3.5.1	Introduction . . . . .	114
3.5.2	Methods . . . . .	115
3.5.2.1	Training sets and models architectures . . . . .	115
3.5.3	Results and discussion . . . . .	116
3.5.3.1	MTEN and XGB are promising ML models to surrogate FBA . . . . .	116
3.5.3.2	MTEN surrogates FBA with different performances depending on the training set . . . . .	116
3.6	AMNs' ability to predict the effect of media and genetic conditions on metabolic fluxes depends on training set contents . . . . .	118
3.6.1	Introduction . . . . .	118
3.6.2	Methods . . . . .	118
3.6.2.1	Training sets and models architectures . . . . .	118
3.6.3	Results and discussion . . . . .	120
3.6.3.1	AMNs better predict the effect of reaction KOs and media conditions that appeared in the training data . . . . .	120

3.6.3.2	AMNs can predict the effect of a regulator gene KO only if statistical patterns are found in the reference fluxes . . . . .	121
3.7	Closing remarks . . . . .	123
3.7.1	Chapter summary . . . . .	123
3.7.2	Reservoir computing, a promising approach to tackle AMN issues? . . . . .	124
<b>4</b>	<b>General discussion and perspectives</b>	<b>127</b>
4.1	A novel and challenging approach to exploit GEMs . . . . .	127
4.1.1	Improving GEMs predictive power with AMNs . . . . .	127
4.1.1.1	From an optimality to a learning principle . . . . .	127
4.1.1.2	AMNs can predict full metabolic phenotypes with partial flux data . . .	128
4.1.1.3	Mechanistic layers can improve predictions quality . . . . .	128
4.1.1.4	A wider range of input or output data type . . . . .	128
4.1.1.5	A wide range of architectures . . . . .	128
4.1.2	Learning with constraints: a challenging approach . . . . .	129
4.1.2.1	The custom mechanistic loss relies on tradeoffs between the terms . .	129
4.1.2.2	The SV loss term is not sufficient to satisfy GEMs constraints in AMNs predictions . . . . .	129
4.1.2.3	Mechanistic layers have limitations . . . . .	129
4.1.2.4	The AMN performance depends heavily on the training set . . . . .	129
4.1.2.5	The AMN-Reservoir approach is reliable but suffers from limitations . .	130
4.2	Perspectives on hybrid models for GEMs: improvements and applications . . . . .	131
4.2.1	Improved approaches and formulations . . . . .	131
4.2.1.1	Standardized training set acquisition . . . . .	131
4.2.1.1.1	<i>in silico</i> training sets . . . . .	131
4.2.1.1.2	<i>in vivo</i> training sets . . . . .	131
4.2.1.1.3	Automated experimentation for large training sets acquisition	132
4.2.1.1.4	Data preprocessing . . . . .	132
4.2.1.2	Improved custom mechanistic loss formulations . . . . .	132
4.2.1.3	Improved mechanistic layers formulations . . . . .	133
4.2.1.4	Replacing ANNs by other ML methods . . . . .	133
4.2.1.5	Improved AMN-Reservoir workflow and alternatives . . . . .	133
4.2.1.6	Alternative hybrid modeling of GEMs: comparison of AMNs with existing approaches . . . . .	135
4.2.2	Areas of application . . . . .	136
4.2.2.1	Omics data integration . . . . .	136
4.2.2.2	Data-driven metabolic models curation . . . . .	136
4.2.2.3	Integration of time-series datasets . . . . .	137
4.2.2.4	Guiding bioproduction efforts with hybrid models . . . . .	137
4.2.2.5	Biocomputing with hybrid models . . . . .	137
4.2.2.6	Towards a whole-cell hybrid model? . . . . .	137
4.3	Concluding remarks . . . . .	138
4.3.1	Chapters summaries . . . . .	138
4.3.2	Answers to scientific questions of Chapter 1 section 1.5 . . . . .	139
4.3.3	Final words . . . . .	139



<b>A</b>	<b>Supplementary Information for Chapter 1</b>	<b>141</b>
A.1	Historical breakthroughs in metabolism research: from single enzymes to complete metabolic maps . . . . .	141
A.1.1	Discovery of enzymes . . . . .	141
A.1.2	Discovery of metabolic pathways . . . . .	143
A.1.3	Metabolism at the organism scale . . . . .	144
A.1.4	Questioning our knowledge of metabolism . . . . .	145
A.2	GEMs reconstruction process . . . . .	146
A.2.1	Automated or manual curations depend on the GEM's scope . . . . .	146
A.2.2	Refining the previous GEM: enhancing the stoichiometric matrix and metadata links . . . . .	147
A.3	Pathway analysis: deriving topological insights from metabolic networks . . . . .	147
A.4	Open discussion on MM and ML capabilities . . . . .	148
A.4.1	Two seemingly opposed approaches . . . . .	148
A.4.2	MM and ML for systems biology . . . . .	149
<b>B</b>	<b>Supplementary Information for Chapter 2</b>	<b>151</b>
<b>C</b>	<b>Supplementary Information for Chapter 3</b>	<b>181</b>
C.1	Training sets presentation . . . . .	181
C.2	Code and Data Availability . . . . .	184
C.3	QP-solver reformulation and optimization . . . . .	185
C.4	Mechanistic layers performance assessment . . . . .	187
C.5	AMNs hyperparameter optimization . . . . .	191
C.6	Alternative ML models to better surrogate FBA . . . . .	192
C.7	Extended results for 'Rijs-RegKO' . . . . .	193
<b>D</b>	<b>Co-authored publication n°1: <i>In silico, in vitro, and in vivo</i> machine learning in synthetic biology and metabolic engineering</b>	<b>197</b>
<b>E</b>	<b>Co-authored publication n°2: Cell-Free Biosensors and AI integration</b>	<b>207</b>
<b>F</b>	<b>Co-authored publication n°3: A versatile active learning workflow for optimization of genetic and metabolic networks</b>	<b>229</b>

# List of Figures

1.1	Allostery, a widespread enzyme regulation mechanism . . . . .	35
1.2	An enzyme-coding gene expression regulation mechanism controls the <i>lac</i> operon . .	36
1.3	Overview of central metabolic pathways regulations in wild-type <i>E. coli</i> . . . . .	38
1.4	Overview of Gene Regulatory Networks representations . . . . .	40
1.5	Evolution of network-based pathways . . . . .	45
1.6	Mathematical formulation of the reaction network: the stoichiometric matrix . . . . .	46
1.7	Venn diagram of various GEMs computing frameworks . . . . .	48
1.8	Methodology for flux balance analysis . . . . .	52
1.9	Biological and computational neurons . . . . .	60
1.10	Multi-Layer Perceptron architecture example . . . . .	61
1.11	Generic feedforward Multi-Layer Perceptron computation schematic . . . . .	61
1.12	Reservoir Computing frameworks . . . . .	64
1.13	Overview of possible machine learning interventions on the constraint-based modeling pipeline . . . . .	65
1.14	Integration example of experimental data with machine learning and constraint-based models . . . . .	66
1.15	Simple schematic of the selected hybrid modeling approach for GEMs . . . . .	69
2.1	Computing and learning frameworks for FBA, alternative Mechanistic Models, AMN, and AMN-Reservoir . . . . .	75
2.2	Benchmarking AMNs with different training sets and mechanistic layers . . . . .	78
2.3	Benchmarking growth rate predictions by AMNs with experimental measurements . .	80
2.4	AMNs growth rate predictions for <i>E. coli</i> gene KOs mutants . . . . .	81
2.5	Reservoir computing for improving the predictive power of FBA modeling . . . . .	83
3.1	SV norm or V norm metrics response to different flux increases applied on a steady-state flux distribution . . . . .	100
3.2	Pareto plots of QP-solver and QP-bnds-solver . . . . .	104
3.3	QP-bnds-solver better respects GEMs constraints . . . . .	105
3.4	Improved workflow of AMN-QP-bnds compared to AMN-QP . . . . .	106
3.5	Increasing number of iterations in the mechanistic layers of AMNs reduces the SV norm but increases computational time . . . . .	109
3.6	Hyperparameter optimization of the neural layer and custom loss terms weighting of AMNs . . . . .	112
3.7	Comparison of AMN performance before and after hyperparameter optimization . . .	113
3.8	Alternative ML models to surrogate FBA . . . . .	116
3.9	MTEN performance to surrogate FBA with six different training sets . . . . .	117

3.10	AMN-QP-bnds outperforms FBA with the 'Cov-BiologKO' training set but do not respect stoichiometry constraints . . . . .	120
3.11	AMN-QP-bnds performance with unseen combinations of reaction KOs and unseen media compositions . . . . .	121
3.12	AMNs can predict the effect of a regulator gene KO only when statistical patterns are found in the training data . . . . .	122
3.13	Performances of AMN-Reservoir using predicted $V_{in}$ as input to FBA . . . . .	124
4.1	Reservoir Computing workflow to generate Figure 2.5 . . . . .	130
4.2	Proposition of an improved Reservoir Computing workflow . . . . .	134
4.3	Proposition of a learning loop workflow, with FBA and ML separated . . . . .	135
A.1	Timeline of historical breakthrough in metabolism research . . . . .	141
A.2	Glucosidase 3-dimensional structure . . . . .	142
A.3	Glycolysis pathway . . . . .	143
A.4	Metabolic metro map . . . . .	145
A.5	New central dogma of molecular biology proposed by Tan and Anderson . . . . .	146
C.1	<i>in silico</i> training sets generated by FBA on <i>E. coli</i> core or iML1515 . . . . .	182
C.2	<i>in vivo</i> training sets generated from experimental data . . . . .	183
C.3	Pareto plots of QP-solver and QP-bnds-solver for lower bounds and upper bounds constraints . . . . .	185
C.4	QP-bnds-solver reformulation improves the respect of GEM constraints . . . . .	186
C.5	Performance of AMN-QP-bnds, AMN-LP and AMN-Wt for increasing number of iterations of the mechanistic layer . . . . .	187
C.6	Increasing the iteration number of the mechanistic layers of AMNs reduces the SV norm but increases computational time . . . . .	187
C.7	Increasing the iteration number of the mechanistic layer has limited effect on AMN-QP-bnds prediction performance . . . . .	188
C.8	Increasing the iteration number of the mechanistic layer has a strong effect on AMN-LP prediction performance . . . . .	189
C.9	Increasing the iteration number of the mechanistic layer has a mitigated effect on AMN-Wt prediction performance . . . . .	190
C.10	Comparison of AMN-QP-bnds performance before and after hyperparameter optimization . . . . .	191
C.11	MTEN performance with 'iML-glucose', 'iML-random', 'iML-extended' training sets . . . .	192
C.12	AMN-QP-bnds performance with 'Rijs-RegKO' . . . . .	193
C.13	AMN-QP-bnds performance with 'Rijs-RegKO-glucose' . . . . .	194
C.14	AMN-QP-bnds performance with 'Rijs-RegKO-galactose' . . . . .	195
C.15	AMN-QP-bnds performance with 'Rijs-RegKO-GR-only' . . . . .	196

# List of Tables

3.1	Summary of the training sets used in Chapter 3 . . . . .	98
3.2	QP-solver and QP-bnds-solver loss terms, purpose, weights name and range for optimization . . . . .	102
3.3	Hyperparameters for AMN optimization . . . . .	111
3.4	Comparison of SV norm and V norm metrics found in FBA computations and AMN predictions . . . . .	114
3.5	Summary of the alternative ways to use 'Cov-BiologKO' and 'Rijs-RegKO' training sets .	119
C.1	Summary of the training sets used in Chapter 3 . . . . .	184



# Chapter 1

## Introduction

This Ph.D. dissertation aims to propose a novel approach to state-of-the-art metabolic modeling, by integrating mechanistic modeling and machine learning methods together. This approach was motivated by the lack of predictive power of mechanistic modeling methods, and the lack of mechanistic insights brought by machine learning methods, when used separately. In particular, I developed hybrid models for genome-scale metabolic models (GEMs) that use neural networks as a machine learning basis to guide flux predictions by GEMs. Before diving into the biological and methodological concepts that shall be described in order to understand the motivation for such an approach and its construction, it is imperative to define important terms and set the biological scope that will be considered. Indeed, some widely used terms' precise meaning can differ among studies, as it will be discussed in this preamble. Also, setting the particular biological scope that I limited myself to in this dissertation is important, as considering all organisms and all known processes across the tree of life would be too exhaustive.

### 1.1 Preamble: defining important terms and scopes

#### 1.1.1 Metabolism

First of all, let us question the meaning of metabolism. As underlined by Lazar and Birnbaum[2], the most common definition of metabolism, *i.e.* "the chemical processes that occur within a living organism in order to maintain life", overlaps with the common definition of biochemistry, *i.e.* "the branch of science concerned with the chemical and physico-chemical processes and substances which occur within living organisms". The precise scope of metabolism is unclear: we need to draw a line on the particular subset of biochemical reactions that constitute metabolism. In this dissertation, I personally chose to use a common delimitation, considering metabolism as the set of all possible enzyme-catalyzed biochemical reactions that concern small molecules (*i.e.*, metabolites), happening in an organism, or in a population of organisms. Importantly, some other delimitations of metabolism (i) exclude non-essential reactions (and term 'secondary metabolism' the set of non-essential reactions), (ii) include all chemical reactions (enzymatic and non-enzymatic), (iii) include macromolecule synthesis, such as DNA replication and protein synthesis, or (iv) include biochemical reactions acting as regulation processes, such as the activity of chaperones, transcription factors or kinases. Importantly, I don't want to make the distinction between primary and secondary metabolism, as it does not seem well defined for most species, and condition-dependent (in some conditions, the 'secondary' metabolism might become 'primary'). Also, the metabolic models that will be used in this dissertation

do not include non-enzymatic reactions, nor those related to macromolecule synthesis and regulation processes; which motivates the proposed delimitation.

### 1.1.2 Regulation

Next, let us attempt to define regulation in general terms. A regulation can be defined as a management mechanism (defined by a set of rules) of a complex system (*e.g.* a living cell). For biological systems in particular, it is unclear what processes are considered as regulatory or not. Bich *et al.* explored this question and define “regulation as the capacity of a biological organism to mediate the effect of a perturbation, modifying its own internal behavior by means of a specialized subsystem that modulates the action of diverse control mechanisms and selects between various available and viable dynamic regimes”[3]. In the context of this dissertation (and in simpler terms) we will consider regulation as the set of mechanisms that directly depends on abiotic or biotic cues, in order for the organism to adapt to changing internal and external conditions.

### 1.1.3 Metabolic phenotype

Another important term I will use throughout this dissertation is the ‘metabolic phenotype’. This generally refers to the metabolic activity of an organism, whose precise scope is rather unclear. Indeed, we could consider the metabolic activity of single cells or cell populations (of the same species or multi-species communities), time dynamics (on different time-scales) or a single snapshot, with metabolites, enzymes, reactions, or all of these entities, that each can be expressed in different units. Moreover, as stated in the first section of the Preamble (1.1.1), the scope of metabolism itself relies on a subjective choice. Importantly, the notion of metabolic *phenotype* underlines the fact that it results from regulation processes, with environmental cues driving the phenotypic responses based on genotype-phenotype maps[4]. Here, let me define the ‘metabolic phenotype’ of an organism as a snapshot of all metabolic reaction speeds (also called metabolic fluxes), and all metabolites production rates. In that sense, a metabolic phenotype performs biochemical reactions that are adapted to its environment, at a given time point. Importantly, we will consider in this dissertation the reaction fluxes and metabolite production rates to be expressed in  $\text{mmol.gDW}^{-1}.\text{hr}^{-1}$ , namely in millimol of chemical species per gram of dry weight of the organism per hour. As for the delimitation I set for metabolism, the proposed definition of a metabolic phenotype is motivated by the scope of the metabolic models that I will use throughout this dissertation, which are based on the same conception.

### 1.1.4 Biological scope

Importantly, I will limit my scope to *Escherichia coli* (*E. coli*) unless specified otherwise, in order to describe metabolic processes and regulation mechanisms. This is motivated by the fact that I almost only used that model organism for the work presented in this dissertation. This is true for experiments that have been conducted, metabolic models that have been exploited, and external datasets that have been acquired from literature (except one *Pseudomonas putida* metabolic model and external dataset). In particular, I used models and experiments from the substrain MG1655, that was isolated and treated with UV light and acridine orange, from the W1485 strain. The latter was isolated from a stab-culture of the original K-12 strain, which was itself obtained from a stool sample of a diphtheria patient in 1922[5]. Notably, compared to the original W1485 substrain, MG1655 mildly starves for pyrimidine which can be compensated by adding uracil to the growth medium[6]. Note that such

nutrient dependence is often called ‘auxotrophy’ (MG1655 is an uracil partial auxotroph). I also used for some results the DH5- $\alpha$  strain of *E. coli*, engineered to lack methylation-dependent restriction systems and consequently enhance cloning efficiency[7]. In section 1.2.2.1.1, I will give details on *E. coli* as a model organism and chassis for synthetic biology. Obviously, some of the metabolic processes and regulation mechanisms further described are common to many species and not only *E. coli*. However, by limiting the scope to *E. coli*, I will ignore all eukaryote-specific regulation mechanisms, which can be radically different[8], with exclusive processes such as alternative splicing.

## 1.2 Metabolism: more than breaking down nutrients

This section will present the main biological processes that we aim to model in this dissertation. It will mostly cover the known mechanisms that control the metabolic response of an organism from environmental cues.

To understand the state-of-the-art metabolism research, it is helpful to look back at historical breakthroughs, that span from unknowingly exploiting metabolic processes *circa* 7,000 BC, to modern modeling approaches involving more and more complete knowledge of metabolism. In the appendix section A.1, such a historical overview is given, describing how the first enzymes, then metabolic pathways, and complete metabolic maps were discovered. I also mention the modern questionings around our conception of metabolism, mentioning underground metabolism and enzyme promiscuity. In appendix section A.1, I also introduce basic metabolic key concepts, such as enzymes and their thermodynamic constants ( $k_{cat}$ ,  $K_m$ ,  $V_{max}$ ), metabolites, pathways, cofactors or ribozymes. Finally, I mention useful resources for metabolism research, such as the Enzyme Commission numbers classification, databases such as KEGG and BiGG, and a graphical view of a renewed central dogma of molecular biology (Figure A.5).

### 1.2.1 From environmental cues to metabolic phenotypes: a concert of regulatory mechanisms

In biological systems, regulatory and metabolic processes are intricate and intertwined: there is an intensive interplay between the regulations shaping metabolic activities and the change of regulation processes from different metabolic activities. However, for modeling purposes, such interplay does not seem achievable as of today. Most approaches consider regulation *or* metabolism, and have separate submodels for regulation and metabolism when combining them (see Section 1.2.1.2.1 for examples). It is the case for hybrid models presented in this Ph.D. dissertation: we chose to model all regulations through ‘agnostic’ machine learning methods, and use a mechanistic modeling approach for metabolism, with genome-scale metabolic models (GEMs). The following section 1.2.1.1 will describe many regulatory processes found in bacteria (specifically in *E. coli*), and mechanistic modeling approaches to simulate such processes. In the hybrid models presented in Chapters 2 and 3, we attempt to model those processes by machine learning.

#### 1.2.1.1 A myriad of well-described regulation mechanisms: the reductionist approach

##### 1.2.1.1.1 Metabolism regulation is more complex than controlling individual enzymes activities

Intuitively, and especially in scope of the ‘old’ central dogma of molecular biology (see Figure A.5 for a renewed version), one might guess that metabolic phenotypes are only controlled by regulat-



ing the expression of enzyme-coding genes. However, there was no clear and universal relationship found between the expression level of genes (*i.e.* the abundance or concentration of mRNA) and the amounts of corresponding enzymes found in an organism[9]. Nonetheless, a correlation can be found in many cases, and many attempts to build models predicting metabolic phenotypes from transcriptomic data have been conducted[10], unfortunately showing limited generalization abilities across environmental conditions. Moreover, the amount of enzyme involved in a given flux is not necessarily correlated with the actual measured flux[11].

For a time, the scientific community considered a metabolic pathway to be only regulated by a 'rate-limiting step', *i.e.* an enzymatic reaction that is the bottleneck of the pathway (limiting the overall pathway flux). Therefore, the approach to manipulate a metabolic pathway's product flux was to attempt a change in this particular enzyme activity (*e.g.* by increasing its expression level), which proved unsuccessful in most cases. Note that an enzyme's 'activity' is considered here as the catalyzed biochemical reaction flux enabled by the enzyme. The fields of Metabolic Control Analysis (MCA) and Biochemical Systems Theory (BST) tackled this issue by proposing a novel conception of the link between an enzyme activity and its metabolic pathway activity. Instead of identifying a single 'rate-limiting step' assumed to control the pathway, this field explores the quantitative notion of 'control' exerted by an enzyme activity on the pathway, *i.e.* how much the change in activity of this enzyme affects the overall pathway's activity. This quantitative notion of 'control' is formulated through mathematical models, with 'control coefficients' estimated from experimental data. After compelling achievements, showing better abilities to control pathways fluxes, the 'metabolic control' concept introduced by MCA and BST fields replaced the previous intuition of 'rate-limiting steps', to explain the influence of individual enzyme activities on overall metabolic pathways activities[12].

We mentioned above that a single rate-limiting step is not sufficient to explain an overall metabolic pathway regulation. These steps (also called 'bottlenecks', 'key enzymes', 'pacemaker enzymes' or 'regulatory enzymes') were historically identified through a variety of experimental or theoretical means [12]. For example, one can determine the lowest  $V_{\max}$  enzyme of a pathway to find a rate-limiting step (kinetic approach); but one can also interpret the metabolic pathway architecture (*e.g.* assuming first steps to be rate-limiting, in a teleological approach). In most cases, experimentally determining a rate-limiting step does not require unraveling *how* this step is rate-limiting. It can sometimes be explained without regulation mechanisms. For example, the thermodynamic properties of enzymes ( $k_{\text{cat}}$ ,  $K_m$ ,  $V_{\max}$ ) and the cofactors, substrates, products and enzyme availability can modulate metabolic fluxes that may induce such rate-limiting steps. Also, competitive inhibition, *i.e.* the competition between different substrates of the same enzyme, can play a significant role in rate-limiting steps. However, rate-limiting steps often involve more sophisticated enzyme regulation mechanisms, that we shall further describe. I will start by describing regulation mechanisms affecting the enzyme activities (often referred to as 'fast regulation'), and then proceed with gene expression regulation related to metabolism (often referred to as 'slow regulation').

#### **1.2.1.1.2 Enzyme activity regulation by post-translational modifications**

Enzyme activity can be influenced by a widespread phenomenon: allostery (also called allosteric regulation or non-covalent post-translational modifications). It consists in a change of conformation of an enzyme triggered by the binding of a molecule called an effector, to a different site than the active catalytic site of the enzyme. The effector binding can change the conformation of an enzyme, to inhibit or activate its activity. The term was first used in 1965 [13], describing the first allostery model, known as the Monod, Wyman and Changeux (MWC) 'concerted' model. Nowadays, allostery is de-

scribed with other models, such as the Koshland, Nemethy, and Filmer (KNF) 'sequential' model[14]. The KNF model differs from the MWC in the way allostery affects an enzyme's subunits. In the 1980s, allostery was re-conceived as a more continuous and global phenomenon, notably by considering the physical environment of the enzyme, such as the pH and temperature, as allosteric effectors. In the 2000s, more complex allosteric mechanisms were uncovered: a model of allostery without conformational change, and a continuous conception of allostery (as a landscape of allosteric modulation instead of binary activation), as well as 'allostery networks' were proposed[9]. Figure 1.1 shows a general schematic of allosteric regulation. Another main enzyme activity regulation mechanism is covalent Post-Translational Modification (PTM), with a diversity of effects found in bacteria[15]. The best known and most widespread is the phosphorylation of serine, threonine or tyrosine residues in proteins, performed by kinases and removed by phosphatases; inducing changes for longer periods than allostery in the proteins activity. This phenomenon was long thought to be insignificant in *E. coli* and other prokaryotes, but recent studies have shown the widespread and clear functional effects of phosphorylation on *E. coli* phenotypes[16].

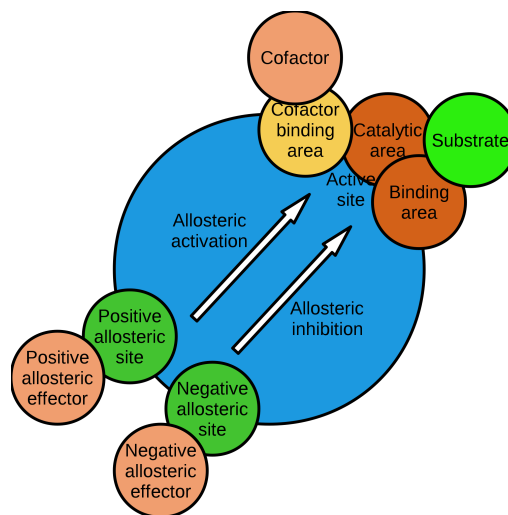


Figure 1.1: Allostery, a widespread enzyme regulation mechanism. Either a positive (activator) or negative (inhibitor) allosteric effector binds to an enzyme allosteric site (bottom-left), to trigger allosteric activation or inhibition (white arrows) of the enzyme active site. The active site might be dependent on a cofactor (which is *not* an allosteric effector). (Author: Tenthkrige, License: CC BY 1.0)

#### 1.2.1.1.3 Gene expression regulation: the *lac* operon and carbon catabolite repression examples

Now, I will describe enzyme-coding gene expression regulation mechanisms (the 'slow' regulation mechanisms). The first thorough description of such a mechanism was the groundbreaking work of Jacques Monod and François Jacob in 1961[17]. Their investigation started from a simple observation: *E. coli* grown on a mixed carbon source culture medium, of glucose and lactose, would first consume all glucose and then start consuming lactose. This phenomenon, termed diauxie, is the cause of a phenomenon later described and called 'carbon catabolite repression' (CCR), and refers to the mechanisms explaining an organism's modulation of its metabolic routes based on a hierarchy of carbon sources utilization.

Naturally willing to explain this observation, Monod and Jacob pursued their investigation. This led them to define the 'operon' as a transcriptional unit of *E. coli*, *i.e.* a physically clustered set of genes that are transcribed in concert. In particular, they investigated the *lactose (lac)* operon organization and regulation mechanism, of which a schematic is displayed on Figure 1.2. The operon consists of 5 contiguous DNA elements: a promoter sequence (where the RNA polymerase can bind), an operator

sequence (where a repressor protein, *lacI*, can bind), and 3 'structural' genes *lacZ*, *lacY*, *lacA* that encode for enzymes of the lactose transport and catabolism pathway, and are transcribed into a single polycistronic mRNA (*i.e.* coding for several enzymes). Note that *lacI* is constitutively expressed (*i.e.*, its expression is not depending on a specific condition). The production of the operon enzymes is controlled by a relatively simple regulation mechanism. The 'default' behavior of the operon is to *not* transcribe the enzyme-coding genes, in the absence of lactose or other *lacI*-binding molecules. If a *lacI*-binding molecule is present in the cell (such as lactose, or allolactose, a product of *lacZ* creating a positive feedback loop), it will allosterically change the conformation of *lacI*, making it unbind the DNA operator and enabling the transcription of the *lac* operon. Therefore, this regulation mechanism can be viewed as a boolean logic expression system: if the lactose substrate is present, the operon is transcribed, otherwise it is not. However, we can expect to find much more complex logics in other expression systems found in nature.

Another mechanism explaining glucose and lactose catabolite repression was found in 1984 and called 'CRP-cAMP *lac* operon activation'[18]. cAMP stands for cyclic Adenosine MonoPhosphate and CRP stands for cAMP Receptor Protein. In the presence of glucose, cAMP levels are low because of the inhibition of adenylate cyclase (producing cAMP) when glucose is transported inside the cell. CRP-cAMP complexes bind many sequences, notably to an upstream sequence of the *lac* operon promoter, increasing its overall transcription activity in absence of glucose.

The above-described mechanisms explain how the *lac* operon activity is responding to the presence of lactose and absence of glucose, however it is not explaining how the presence of glucose inhibits the *lac* operon activity. It was more recently found that the glucose transport system, known as the Phosphotransferase system (PTS), triggers a de-phosphorylation of a PTS protein, which in this form binds and inhibits the lactose transport permease enzyme *lacY*[19].

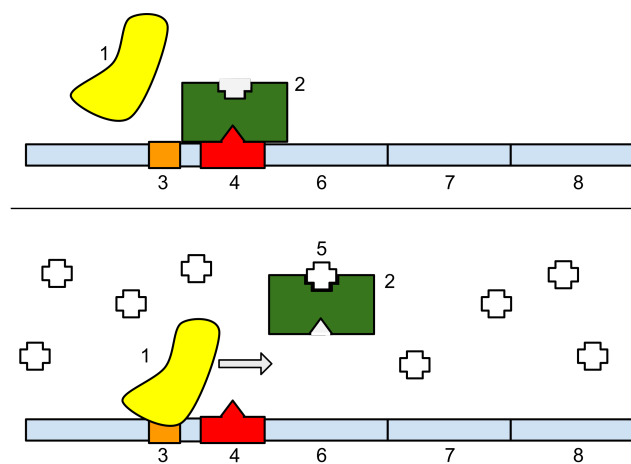


Figure 1.2: An enzyme-coding gene expression regulation mechanism controls the *lac* operon. 1: RNA polymerase, 2: Repressor, 3: Promoter, 4: Operator, 5: Lactose, 6: *lacZ*, 7: *lacY*, 8: *lacA*. In the top panel, no lactose (white crosses) bind to the repressor (green), which binds to the operator sequence, blocking the RNA polymerase activity. In the bottom panel, lactose binds to the repressor which changes its conformation and suppresses its ability to bind the operator, thus allowing the RNA polymerase activity. This leads to the production of lactase by the operon. (Author: T.A. Raju; license: CC BY 3.0)

More operons were discovered over the years, with a total of 2584 operons in *E. coli*[5]. One can assume each of these operons are regulated in a different way, and CCR or CCR-like mechanisms can also regulate how the operons are 'competing' with each other.

#### 1.2.1.1.4 Gene expression regulation: review of the main mechanisms

A more general family of regulatory proteins in which the *lacI* repressor falls into, is the transcription factors (TFs) family. Generally, TFs are defined as DNA-binding proteins that influence positively or negatively the transcription of one or several genes. A TF binds to an operator sequence. The activation or inhibition of transcription by a TF can be dependent on co-activator or co-repressor molecules (such as lactose for *lacI*). Another family of regulatory proteins are the sigma factors, which are commonly known as more global regulators than TFs: they bind to RNA polymerase to transcribe large sets of genes. For example, *E. coli* has 7 sigma factors and roughly 300 TFs. Importantly, most sigma factors' expression is regulated by other regulation mechanisms, and the competition between sigma factors influences what genes will be expressed. In general, one gene is expressed with the help of a single sigma factor. However in some rare cases, more than one sigma factor can support the transcription of a gene[20].

Two-component regulatory systems are another important family of regulation mechanisms that can explain how bacteria react to external stimuli, in which TFs play a central role[21]. Note these two-component systems are ubiquitous across life domains but mostly found in bacteria so far. These two-component systems transduce the signal of the external stimuli up to effective regulation, mostly by modulating gene expression. The basic functioning of two-components systems is as follows: the first component, a membrane-bound histidine kinase, senses the external stimuli which triggers the autophosphorylation of a histidine residue; then the second component, called response regulator protein, has its receiver domain phosphorylated by the histidine kinase (transferring its histidine phosphate group on an aspartate residue of the receiver domain), which in turn activates the effector domain of the response regulator, in most cases by allostery. The response regulator with its effector domain activated will perform the actual regulation, in most cases by directly modulating gene expression (thus they are considered as TFs). Note that in some cases the response regulator does not possess a receiver domain, and a phosphorelay transduces the signal from the histidine kinase to the response regulator. Moreover, in some cases, the effector domain of the response regulator is interacting with another protein to trigger the regulation response, and does not directly modulate gene expression. A major example of two-component system regulation is the expression of porin genes in *E. coli*[21].

All regulations described so far were focusing on the cellular scale. However, a population-level regulation happens in bacteria, called 'quorum sensing'. It is defined as the gene expression regulation triggered from fluctuations in cell population density. Quorum sensing is implicated in a wide range of traits such as motility, virulence or biofilm formation[22]. Another mechanism, 'quorum quenching' can disrupt quorum sensing communications and regulations. Chemotaxis is another interesting regulation mechanism worth citing, enabling bacteria to direct their movement according to chemical gradients found in their environment. Very briefly, other types of regulations can be performed in *E. coli* by: (i) histone-like nucleoid structuring proteins (H-NS), (ii) small non-coding RNAs (sRNAs), (iii) proteases and chaperones, (iv) signaling molecules (such as the above mentioned cAMP), and (v) potentially other mechanisms that have not yet been discovered.

An extensive description of how *E. coli* metabolism is regulated has been proposed by Kayuzuki Shimizu[23]. In this thorough review, Shimizu describes how different environmental stimuli such as catabolite repression recruit global regulators such as the abovementioned cAMP-CRP system and sigma factors, in order to regulate metabolic pathways. Figure 1.3 taken from this review shows an overview of the description made by Shimizu. Importantly, note that this figure focuses here on global (multiple targets) regulators of gene expression, and does not give a complete metabolic regulation

description.

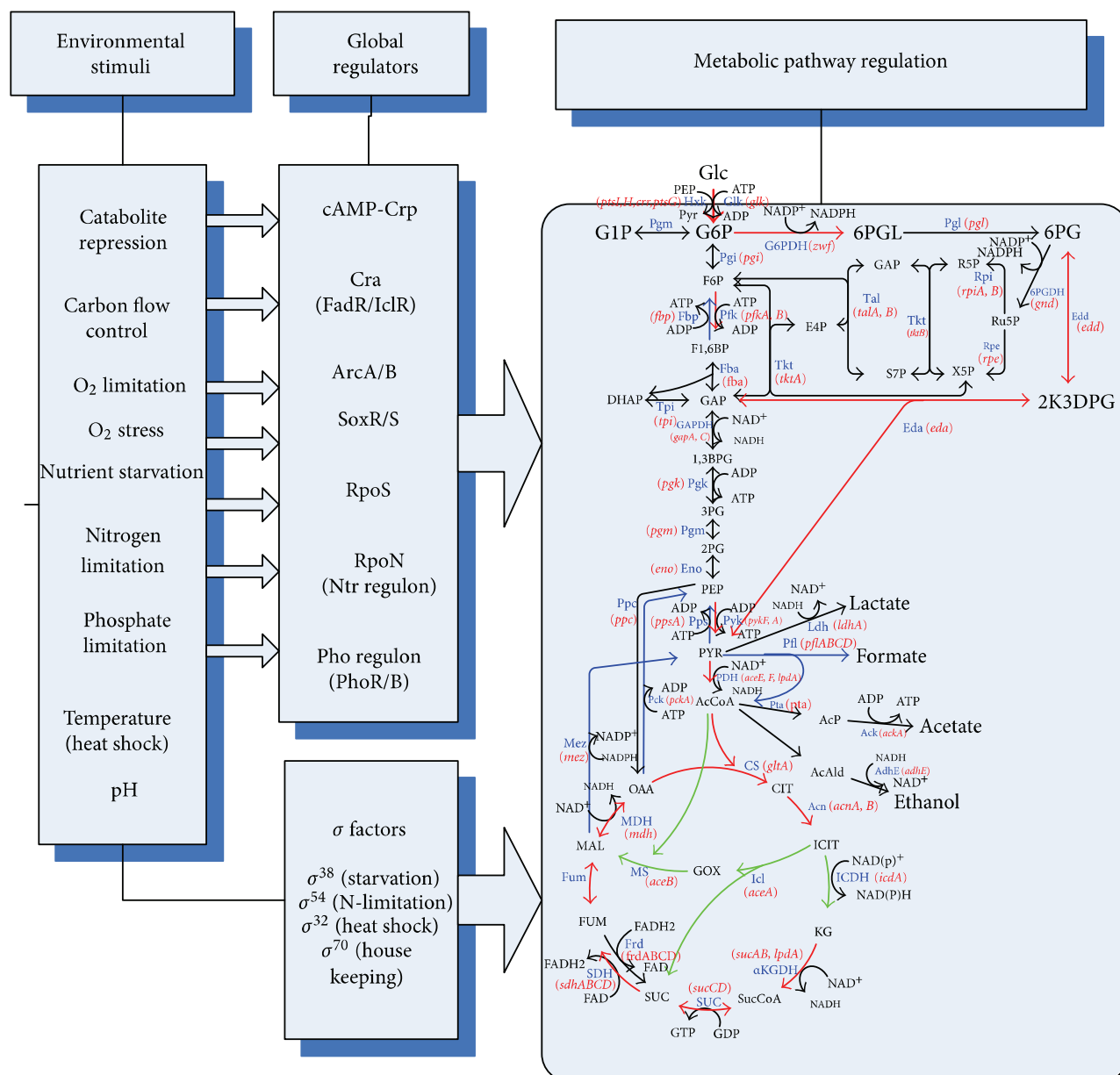


Figure 1.3: Overview of central metabolic pathways regulations in wild-type *E. coli*. A variety of environmental stimuli can trigger changes in global regulators and sigma factors activities, in turn regulating the central metabolism of the cell. (Author: Kazuyuki Shimizu [23]; license: CC BY 3.0)

#### 1.2.1.1.5 Limitations of the reductionist approach

All regulation mechanisms listed so far are affecting enzyme activity either by gene expression or PTMs. We previously mentioned that one can observe the ‘end-point’ effects of such mechanisms on metabolic phenotypes, notably with changes in rate-limiting steps of a metabolic pathway. Another way for regulation mechanisms to tune metabolic phenotypes are changes in branched pathways activities (also called ‘branch points’). In metabolic networks, it is often observed that a product metabolite can be subsequently used by more than one enzyme. The metabolic phenotypes are therefore extremely dependent on such branch points, which are defining which route are the metabolites taking, *i.e.* the topology of the metabolic phenotype. It has been shown that branch points fluxes are

critical to determine metabolic phenotypes, such as the acetyl-CoA branch point between fermentative and respiratory metabolism, that is controlled by transcriptional regulators[24].

In this section (1.2.1.1), we have described many metabolism regulation mechanisms: allostery and covalent PTMs modulating enzyme activity; regulation of operons in concert with CCR; two-component systems with sensing proteins and TFs to regulate gene expression; sigma factors... And these are only the tip of the iceberg of the myriad of regulation mechanisms, that are described in-depth in literature. As quickly mentioned previously, the accumulation of many precise descriptions of such mechanisms is a part of the reductionist approach. Reductionism - the scientific approach breaking down a complex system into simpler subparts - has been the main approach to identify regulation mechanisms, because of experimental constraints and scientific traditions[25]. Indeed, breaking down the tremendous complexity of life into intelligible, interpretable subparts, was a necessary step on the path to accumulate knowledge. As Shimizu attempted to summarize in his review[23], one may envision the possibility to assemble all these descriptions and encapsulate into a single model, a single formulation that simulates the cell-level behavior. This would be a step towards achieving the holy grail of systems biology, as stated by Shimizu and Matsuoka: "the ultimate goal of systems biology is to develop *in silico* models of a whole cell or cellular processes that can predict cellular phenotypes in response to culture environment and/or genetic perturbation"[26]. In that sense, we would switch our approach from reductionism to holism, *i.e.* considering the whole system instead of individual subparts.

Even though we have seen a lot of regulation mechanisms extensively described in the past, "the metabolic regulation in response to the change in culture environment is itself not well understood", as stated by Shimizu and Matsuoka in 2011[26]. Indeed, describing a subset of all existing regulation mechanisms is not sufficient to model the whole-cell regulation mechanism (assuming that we do not know all regulation mechanisms). Even if we did characterize all existing individual regulation mechanisms, emerging properties of the combination of these mechanisms would bring another level of complexity, making it even more challenging to precisely characterize regulations at this scale. Moreover, most regulation mechanisms cited in the present section 1.2.1.1 are qualitative descriptions: their effect might be mitigated depending on quantitative changes of conditions. Consequently, it seems very challenging to model whole-cell regulation in a holistic approach. In the next section 1.2.1.2, I will present some attempts in that direction.

### **1.2.1.2 Whole-cell regulation: the holistic approach**

As stated in the introductory paragraph of the present section 1.2.1, the hybrid models I developed for this Ph.D. attempt to model all regulations through machine learning methods, without requiring knowledge about the underlying mechanisms. However, radically different approaches attempt to model all regulations through detailed, mechanism-based models. These approaches shall be described next, with an emphasis on their limitations.

#### **1.2.1.2.1 Gene Regulatory Networks**

RegulonDB[20] is the most complete and comprehensive database of transcriptional regulation mechanisms until today. It compiles all known and putative mechanisms found in *E. coli*. From this large quantity of information, whole-cell models of regulation were attempted. Gene Regulatory Networks (GRNs) are the general formulation to attempt a whole-cell mechanistic model of regulation[27]. They are based on a network structure, where nodes are regulators and vertices represent the interaction



of a regulator with others. Input signals are generally the presence or the quantity of metabolites to which the organism is exposed, and the output is generally the gene expression levels. Regulators can be DNA, RNA, protein, or any complex of those. The interactions can be of many kinds depending on the GRN representation. GRNs have a variety of representations, which can be delimited to 3 categories as proposed by Karlebach and Shamir: logical, continuous or single-molecule level GRNs. Figure 1.4 shows the 3 categories of GRNs identified, with examples of models and their specificities.

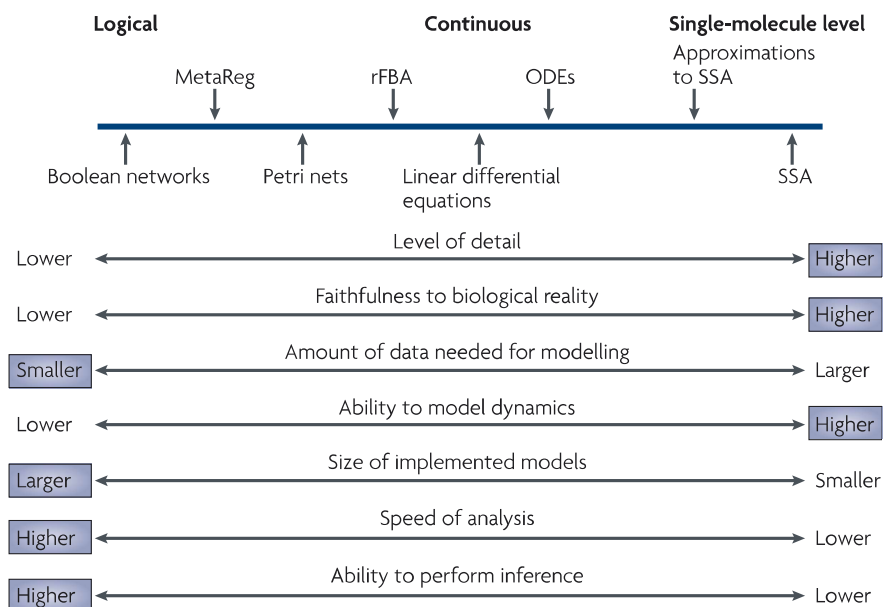


Figure 1.4: Overview of GRN representations. 3 main categories can be identified: Logical, Continuous, and Single-molecule level models. Examples of GRNs are displayed on the thick blue line on top. Specificities of the different representations are displayed on 7 imaginary qualitative axes, with the most 'appealing' pole of each specificity in a blue box. In short, GRNs formulations range from simple to complex, with different levels of detail and accuracy. The choice depends on the specific application and the available data. (Authors: Karlebach & Shamir[27]; reused with authorization of the authors and editors).

A GRN formulation that integrates a wide range of biological objects is regulated Flux Balance Analysis (rFBA), since it is a quantitative formulation in which the metabolic network is 'merged' with the GRN[28]. This opens the door for many interactions that are ignored in other formulations, and provides a quantitative whole-cell model of both regulation and metabolism. This particular formulation is important in this dissertation, since most of the work has been conducted with similar metabolic network models as those used in rFBA. Similar merged formulations of GRNs and metabolic networks have been proposed, such as SR-FBA[29] and iFBA[30].

#### 1.2.1.2.2 Limitations of GRNs

As stated in the previous section 1.2.1.1, gene expression levels alone cannot explain the metabolic phenotypes of an organism. Therefore, most GRNs face a major limitation: modeled mechanisms are limited to transcriptional regulation, ignoring the 'fast' regulation mentioned above, *i.e.* allosteric and covalent PTMs regulation of enzyme activity. Added to that, their output is most often limited to gene expression levels[27]. Moreover, in my opinion, the major drawback of GRNs is even more important, and generally related to an issue common to all mechanistic models. When using such models, if we aim to accurately model regulation mechanisms, we must assume that (i) we are modeling all interactions that exist in the organism, or at least all that are significant in the particular conditions considered, and (ii) emerging properties from the concert of interactions working together are either

known and modeled, or insignificant. These two assumptions are strong oversimplifications, that at least limit the array of conditions accurately simulated by GRNs. In section 1.4.1.1.4 of the introduction, we will come back in more depth to the capabilities and limits of mechanistic models.

As seen in the previous section 1.2.1.1, the complexity of whole-cell regulation processes is tremendous, even for a ‘simple’ and very well characterized model organism such as *E. coli*. We have listed in the present section 1.2.1.2 some attempts of whole-cell regulation models, but capabilities of those seem still limited because of oversimplifications, due to (i) unknown knowledge gaps and lack of quantitative models for some regulation mechanisms, and (ii) emerging properties of the complex systems we are trying to model. Other holistic formulations than GRNs may suffer from the same issues. Modern research on metabolism also attempts to grasp and decipher complexity with more and more standardized, rational, and quantitative approaches, which will be described in the next section 1.2.2. Such holistic approaches of metabolism are ambitious but show great industrial potential. We will also briefly cover the signal-processing abilities of metabolism itself.

## **1.2.2 Modern research on metabolism: embracing complexity**

### **1.2.2.1 Shaping phenotypes with a purpose: a complex endeavor**

#### **1.2.2.1.1 Synthetic biology and metabolic engineering: industrially relevant fields**

One of the major drives of metabolism research, except from the natural curiosity and human endeavor to understand life, is the medical and industrial potential of metabolic processes. Obviously, it began with fermented beverages and foods, with for example Louis Pasteur’s research on fermentation funded by the french wine industries in the early 19th century[31]. More recently, the breakthrough of Cohen & Boyer in 1973[32] (showing the ability of easily and massively cultivable bacterial strains to host and express heterologous genes), drew the attention of the biomedical industries. But, even though the ability of *E. coli* to massively produce a complex molecule such as insulin by simply overexpressing a gene was considered a revolution for the biomedical industries, it was strongly tempered for chemical industries applications, when facing the complexity of biological regulation and metabolism, even for producing a simple molecule such as ethanol[33]. Indeed, it required more than 20 years of research after Cohen & Boyer’s breakthrough to see the emergence of the metabolic engineering field. During these 20 years, an accumulation of knowledge on metabolic networks unlocked a deeper and deeper understanding of metabolism. Moreover, as stated in the previous section 1.2.1.1, regulation mechanisms were gradually deciphered, especially in *E. coli* which was (and is still) the most widely used host for heterologous gene expression, thus the main candidate for biotechnological applications. Consequently, in 1998, the first book of the field was published defining metabolic engineering as the “directed improvement of product formation or cellular properties through the modification of specific biochemical reactions or the introduction of new genes with the use of recombinant DNA technology”[34], with the hope of developing biotechnological interests beyond biomedicine. As stated in the previous section 1.2.1.1, it was for a long time believed that individual enzyme activities could explain overall metabolic pathways activities and modify phenotypes significantly. This assumption underlied many approaches of genetic engineering efforts, before the emergence of metabolic engineering: instead of a single-gene scope, the engineering would use a global and systemic scope of both regulation and metabolism, in order to finely control organisms’ phenotypes. In that way, metabolic engineering was the precursor of systems biology, which will be described hereafter. Importantly, the synthetic biology field should be briefly mentioned, because it



stems from metabolic engineering as they are both fields attempting to produce industrially interesting molecules from biological materials. But, instead of focusing on organism-wide engineering of metabolic phenotypes by fine-tuning existing regulation mechanisms and metabolic reactions, it rather focuses on the *de novo* design of novel biosynthetic pathways and controlling mechanisms. Moreover, synthetic biology develops systems that have more diverse applications than metabolic engineering (*i.e.*, more than the production of proteins or metabolites), such as biosensors, drug delivery, and cell-free systems[35]. This delimitation is still debated and the clear distinction between synthetic biology and metabolic engineering is open for discussion.

#### **1.2.2.1.2 Systems biology: understanding life as systems**

As underlined just above, modern metabolic engineering and synthetic biology fields often rely on holistic (systemic, global) views of regulation and metabolism. This approach is at the very core of systems biology, *i.e.* the modeling of life as systems, with the great ambition of understanding life as a whole. After decades of reductionism identifying many intelligible components of metabolism and regulation, the challenge of postgenomic era's biology was identified to be the assembly of these components in more global formulations, to attempt and exploit global complexity instead of reducing it. Interestingly, the need for such an approach was formulated as early as 1961[36]. However, it was much underappreciated, as it was an overambitious endeavor at that time. Indeed, we can understand such a holistic approach was found unrealistic knowing both experimental and computational constraints of this era, as well as the simple lack of detailed knowledge about regulatory and metabolic processes. Systems biology naturally re-emerged with great interest of the biology community in the early 21st century[37], with the advent of high-throughput experimental technologies, namely the -omics methods. These encompasses measures at many levels such as genomics, transcriptomics, metabolomics and phenomics (including fluxomics and interactomics). Leveraging tremendous amounts of data at many scales, research was pushed to change the reductionist representations of knowledge. The most widespread representation that enables the integration of such large and diverse datasets are networks. A whole field of research is dedicated to the creation and analysis of biological networks, namely Networks Biology[38]. These can be of many kinds, such as the abovementioned GRNs, metabolic networks, or protein-protein interaction (PPI) networks. Systems biology, obviously, is a much related field since most of this field's models rely on network representations. Many challenges are still on the road for the successful integration of these large amounts of data into systems biology models. This dissertation will present one effort towards completing this grand challenge. In the following sections 1.3 ('Genome-scale metabolic networks: state-of-the-art metabolic modeling') and 1.4 ('Hybrid models: reconciling mechanistic and machine learning models'), I will come back in much more detail on examples of metabolic model formulations proposed in the systems biology field.

Added to the new holistic representations brought by systems biology, -omics data and especially metabolomics have unveiled high complexity of metabolism, which questions the nature of metabolism itself. In the next section 1.2.2.2, I will quickly describe unconventional use cases of metabolism, notably as a form of signal-processing.

### 1.2.2.2 Metabolism as a signal-processing unit in biocomputing

#### 1.2.2.2.1 Definition of signal-processing and considered scope of metabolism

The definition of intelligence being highly controversial, I will avoid its use and prefer the less problematic signal-processing term. We can define a signal-processing system as a system able to take a variety of decisions depending on the input signals it is exposed to. In other words, any computing framework is a signal-processing system to which we feed an input signal and measure an output signal.

Most often, complex relationships are found between an organism's environment and the metabolic phenotype it displays. In that sense, metabolism is involved in complex decisions based on input signals (the cell's environment), and can be definitely considered as signal-processing. But since highly complex mechanisms regulate metabolism, as described earlier, is it fair to consider metabolism alone as having signal-processing abilities? In the end, this question is highly dependent on the delimitation we set for metabolism. Considering the same delimitation as in the Preamble of this chapter, therefore excluding all regulation mechanisms, its signal-processing abilities consist only in the spatio-temporal thermodynamics of the enzymes and metabolites interplay. We shall discuss hereafter to what extent such signal-processing is performed, under that delimitation.

#### 1.2.2.2.2 Open discussion on signal-processing abilities of life

Life was shaped by evolution to adapt to changing environments. As stated above, such adaptation abilities, based on complex regulation mechanisms, can definitely be considered as signal-processing. In that sense, evolution has shaped the signal-processing abilities of organisms, by selection pressures depending on intricate interactions between organisms and their environment. Interestingly, this iterative process refining the signal-processing abilities of cells through numerous generations, can be itself considered as a signal-processing ability.

Importantly, evolution has not only led organisms to gain complex regulation mechanisms enabling such adaptation, but also complex metabolic interactions. In fact, metabolites and enzymes have co-evolved in a very intricate manner[39]. Competitive inhibition, for example, is a form of concurrency that can lead to signal-processing abilities by metabolism alone. Also, in a thermodynamic view, the cell is a probabilistic space of encounters of metabolites and enzymes, which also can lead to 'noisy' signal-processing abilities[25]. This is even more striking when looking at time dynamics of metabolism which can display relatively complex behaviors over time, depending on properties of enzymes and their relation with metabolites. Added to that, promiscuity, as cited in the beginning of this chapter, is an ubiquitous phenomenon that adds another possible layer of complexity to competitive inhibition, therefore increasing the signal-processing abilities of metabolism itself.

As discussed by Grozinger *et al.*[40], signal-processing abilities of cells, including metabolism, could empower cellular computing frameworks surpassing traditional ones. Even though we are far from achieving these frameworks concretely, the potential signal-processing abilities seem very promising. A metabolic computation framework was built, called the 'metabolic perceptron'[41], which was based on multiplexed sensing of 3 molecules and a perceptron-like architecture of differentially expressed enzymes eventually leading to a fluorescence response. Even though traditional computers were used for the biosensor design, in order to set the expression levels of enzymes to be used as the metabolic perceptron's weights, it clearly shows the signal-processing abilities of metabolism. At a larger organism scale, the famous *Physarum polycephalum* species, commonly known as 'blob', was used to compute many problems, with so-called Physarum machines[42] (echoing the more tra-

ditional Turing machines); and another kind of perceptron was also built with plants[43].

To conclude this first section 1.2, let me summarize the key points mentioned so far. We have seen that, in the past, myriads of regulation and metabolic processes were described in fine biochemical detail, through the usual reductionist approach of biology. But today, large amounts of -omics data are integrated in more and more complete and complex models, with the systems biology community efforts to gather in rational, quantitative and intelligible ways the so-far acquired knowledge. Finally we have reviewed how this holistic approach, rather ambitious, shows enormous potentials for controlling organisms' phenotypes, for applications in industrial biotechnology but also biocomputing.

The next section 1.3 will put the focus on the specific kind of systems biology model I used in my Ph.D., *i.e.* genome-scale metabolic models (GEMs). These models describe all possible metabolic pathways of an organism, and some associated metabolic capabilities, leading to complex representations, and a variety of methods to exploit them.

## 1.3 Genome-scale metabolic networks: state-of-the-art metabolic modeling

The hybrid models developed for this Ph.D. exploit genome-scale metabolic models (GEMs) to obtain mechanistic insights on metabolic phenotypes. Such mechanistic models have a variety of formulations and mathematical means to exploit them. In this section 1.3, I will start by giving a general overview of such models. Then, I will detail the content of a typical genome-scale model (iML1515[44]), and describe the available mathematical methods for GEMs. Doing so, I will put an emphasis on contents and methods that are of prime importance for the development of hybrid models in the dissertation remainder, such as the uptake fluxes, the biomass reaction, FBA and Linear Programming.

### 1.3.1 Overview

#### 1.3.1.1 Metabolic models in the postgenomic era

One endeavor of systems biology is to represent metabolism at the genome-scale (*i.e.* at the entire organism scale). Indeed, in the so-called 'postgenomic' era, *i.e.* the period after full sequencing of genomes was enabled and then gradually made cheaper, there was a shift in the scale of metabolic pathways modeling. As described in the appendix section A.1, in the pre-genomic era, the community accumulated individual metabolic reactions descriptions with biochemical experimental analysis, which led to the precise knowledge of stoichiometry for many metabolic reactions[1]. After this first step, individual metabolic reactions were naturally grouped into pathways, according to a common function performed by these reactions together, or simply by scientific intuition. The next stage of metabolic modeling was to surpass the grouping of reactions in pathways, and go to the genome-scale. Indeed, the interconnection of pathways was sometimes missing and single formulations of metabolic networks at the organism scale were not possible. The annotation of sequenced genomes (the identification of DNA sequences' functions) opened the door to this stage. In short, algorithms and models based on DNA and protein sequence similarities enabled the inference of yet unknown functions for DNA sequences. It is important to keep in mind that such genome annotations may not be perfectly reliable, especially for less well-characterized organisms than *E. coli*, for

which more uncertain inferences have to be made, since we have less detailed literature knowledge of their metabolism. The three building steps from individual reactions to metabolic pathways then to genome-scale metabolic networks, are schematized in Figure 1.5.

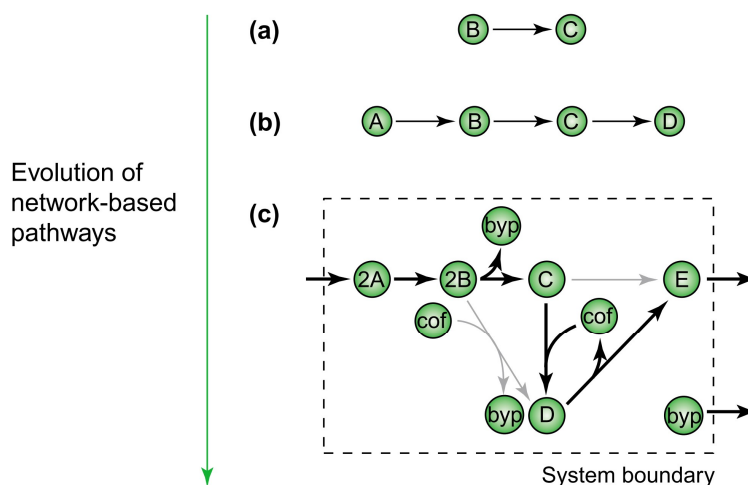


Figure 1.5: Evolution of network-based pathways. From (a) identifying single-step enzymatic reactions with biochemical details of reaction stoichiometry and enzyme thermodynamics, to (b) grouping such individual reactions in pathways, to (c) considering the whole-cell metabolism in a single model, that change the paradigm of metabolic modeling by defining the system's boundary and compiling all possible metabolic routes, that are either active (black arrows) or inactive (gray arrows)(Authors: Papin *et al.*[1]; reused with authorization of the corresponding author; license obtained on the Rightslink® platform).

Acquiring genome-scale metabolic models (GSMMs or GEMs) was made possible from both literature knowledge and automated genome annotations. Indeed, the most important piece of knowledge contained in GEMs is the enzymatic reactions network stoichiometry, even if we will see in section 1.3.2 that more diverse pieces of knowledge are integrated in modern GEMs. The fact that such models are genome-scale brings up more challenges and questions on how to use them, compared to traditional metabolic pathways that were limited to roughly 20 reactions. The main difference between traditional metabolic pathways and genome-scale models is the availability, in the latter, of all *possible* metabolic routes the organism can display; whereas in traditional metabolic pathways, biochemists were investigating the activity of the overall pathway without considering the remaining metabolism. This is visually depicted on panel c of Figure 1.5. Importantly, one must also note that thermodynamic properties of enzymes that were possibly identified when studying individual reactions are not directly usable in the genome-scale network context[45].

The stoichiometric matrix is the usual mathematical representation of the metabolic network in GEMs, with each row representing a metabolite and each column a reaction. Each element of the **S** matrix denoted  $s_{ij}$  is the stoichiometric coefficient of the metabolite  $m_i$  in the reaction  $r_j$ . The mathematical formulation of the metabolic network as a stoichiometric matrix and a glimpse of how it can be used is depicted in Figure 1.6. In section 1.3.3 of the present chapter, I will describe in much more detail the different mathematical means of making meaningful and useful inferences with such a large space of possible metabolic routes.

To ease the large-scale representation efforts of systems biology, more rationalized ways of cataloging biological objects were proposed, with community standards such as the Systems Biology Markup Language (SBML)[46] or the Systems Biology Ontology (SBO)[47]. These provide common vocabularies and semantics to the systems biology models, enabling interoperability and better understanding. For example, an SBML markup can tag an entity as a reaction or chemical species in a

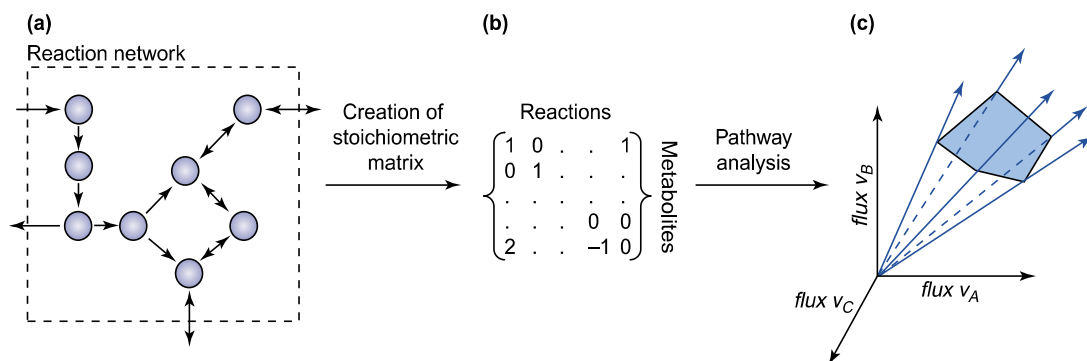


Figure 1.6: Mathematical formulation of the reaction network: the stoichiometric matrix. From (a) the reaction network stoichiometry, one derives (b) a stoichiometric matrix, which in turn can be used for (c) mathematical pathway analysis, for example defining the space of possible steady-state solutions; here represented for only 3 imaginary fluxes in order to be easily visualized. (Authors: Papin *et al.* [1]; reused with authorization of the corresponding author; license obtained on the Rightslink® platform).

GEM, and an SBO term may designate a Michaelis-Menten kinetic parameter in a dynamic model[48]. In most GEMs, the reaction fluxes and metabolite production rates are expressed as  $\text{mmol.gDW}^{-1}.\text{hr}^{-1}$ . Importantly, the biomass production rate (or simply termed ‘growth rate’) is an exception, expressed in  $.\text{hr}^{-1}$ . This particular reaction is further described in section 1.3.2.

Importantly, GEMs are sometimes termed ‘whole-cell’ models, as they aim to represent the metabolic behavior of the entire cell. As previously stated, it is the most ambitious endeavor of systems biology to build such a model, and it is naturally *E. coli* which seems the most adapted for this quest, given our extensive knowledge of it. Obviously, metabolism is not the only process happening in cells: transcription, translation, transport or complexation, for example, can play important roles in the overall cell behavior. Consequently, some approaches such as the ‘*E. coli* Whole-Cell Modelling Project’ aims to integrate GEMs and other modeling processes in rather intricate workflows of model combinations, to propose an actual whole-cell model, simulating more than metabolism at the organism scale[49].

### 1.3.1.2 Steady-state or dynamic formulations to exploit GEMs

Two different ways of exploiting GEMs need to be distinguished; that are the steady-state and kinetic models. Both of these formulations rely on the stoichiometry of metabolic reactions, termed stoichiometric matrix and denoted  $S$  or  $N$  in many works. This matrix describes how metabolites are consumed or produced by each metabolic reaction of the network; it is the most critical knowledge that a GEM contains. The main difference between steady-state and kinetic formulations is that the latter is considering the dynamics of metabolism, with the evolution in time of enzymes and metabolites concentrations. In that sense, steady-state metabolic models assume a mass-balanced state of metabolism. Experimentally, such steady-states are assumed to be reached in the mid-log phase of batch-culture bacterial growth. Therefore, steady-state models consider the metabolic phenotype as a ‘snapshot’ of the metabolic reactions at mass-balance equilibrium (fitting the definition given in the Preamble, section 1.1). On the contrary, kinetic models exhibit the temporal metabolism dynamics which leads to a drastically more complex formulation, since each enzyme in the network needs thermodynamic parameters ( $k_{\text{cat}}$ ,  $K_m$ ) to be estimated, either by experimental work or by sampling possible values when making predictions without experimental determinations. State-of-the-art methods to simulate genome-scale kinetic models rely on deep learning to accelerate the sampling of realistic enzyme kinetic parameters[50]. Even with such state-of-the-art methods, the computational cost of simulations is significantly larger for kinetic models than for steady-state models.

Kinetic models describe in very fine detail the dynamics of metabolism. In that sense, they are closer to what happens in reality, and they provide more insights into the metabolic processes. For example, when a steady-state metabolic model predicts a low flux for a metabolic reaction in a given condition, there is no further interpretation that can be made. In contrast, kinetic models describe underlying mechanisms that could explain this lower flux, such as enzymes and metabolites levels, or regulatory and thermodynamic processes affecting enzyme activities. However, kinetic models show limited predictive capacities as *in vivo* biochemical data are too sparse. In addition, diverse -omics data integration with these models is more challenging than with steady-state formulations[45]. Indeed, steady-state models can output the possible solutions that kinetic models can reach, when built with the same stoichiometric matrices. Therefore, the advantage of kinetic models over steady-state, in most cases, is limited to the study of metabolites concentration dynamics over time before reaching steady-state, which is often disregarded, given the complexity increase to obtain such dynamics. For example, in industrial settings for bioproduction, cells are at metabolic steady-state and the dynamics before reaching that steady-state are rather useless. In short, kinetic models are closer to reality but steady-state models bring simplifications that make them more usable. These two approaches may seem opposed, however, many computing frameworks attempt to combine kinetic and steady-state models into single frameworks, in order to better integrate -omics data[45], or simply use the insights of each method to help interpret the other, in a synergistic fashion[51, 52].

As depicted in Figure 1.7, kinetic models can be distinguished from constraint-based models (CBMs). Even though CBMs can explicitly take time into account, and therefore describe dynamics, they differ in their mathematical foundation. Indeed, kinetics models intrinsically describe dynamical behaviors, with suited methods such as Ordinary Differential Equations (ODEs) or Hybrid Cybernetic Modelling (HCM). In contrast, CBMs can only take time into account with 'tweaks' to their methods, which are originally designed to describe steady-states. For example, Flux Balance Analysis (FBA), a CBM that finds single steady-state solutions based on optimality (see section 1.3.3.3), can be tweaked to integrate time, with dynamic FBA (dFBA) simulating a series of steady-states to compute the time dynamics. Importantly, in the remainder of the dissertation, I will use the term CBM to designate the *steady-state* CBM formulations to exploit GEMs.

### 1.3.1.3 A variety of ways to exploit GEMs, with different biological scopes

As stated in section 1.2.1.2.1, some GEMs formulation include regulatory mechanisms, most notably by merging GRNs with metabolic networks. In fact, another diversity level extends the array of ways to exploit GEMs. Indeed, as shown in the Venn diagram of Figure 1.7, many GEMs computing frameworks exist. According to this classification from Moulin *et al.*[51], the different frameworks encompass the integration of regulation, resource cost, and explicit time; with both Constraint-Based Models (CBM) and kinetics models. Note this figure is of course not exhaustive of all formulations. Recently, community-level models, where GEMs of several organisms are merged into a single formulation[53], emerge as promising tools to understand bacterial communities. In the remainder of the introduction, I will focus on single-organism GEMs, exploited with steady-state CBMs formulations, such as FBA that I extensively use in Chapters 2 and 3.

I just gave an overview of the origin of GEMs, their many different formulations, and a glimpse of how they can be used. The following section 1.3.2 will focus on the detailed content of GEMs, under the scope of steady-state CBM formulations, namely Flux Balance Analysis (FBA) and related approaches.



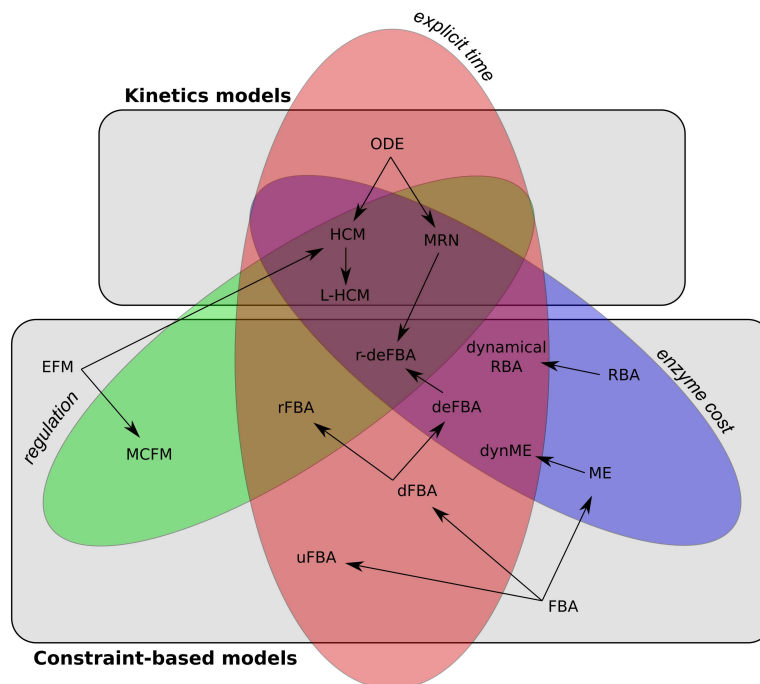


Figure 1.7: Venn diagram of various GEMs computing frameworks, along with their interconnected relationships. Each framework is sorted into categories based on their nature (either constraint-based or kinetic) and the attributes they offer, such as explicit time, regulatory mechanisms, or enzyme cost. An arrow from one framework to another indicates it is a further development or expansion. For the specific meaning of each framework's acronym, please refer to the original study (Authors: Moulin *et al.*[51]; reused with authorization of the corresponding author; license: CC BY 4.0)

### 1.3.2 Detailed content of the state-of-the-art *E. coli* GEM: iML1515

In the appendix section A.2, I give a succinct description of how GEMs are derived from manual curations (based on expert knowledge) and genome annotations (based on uncertain inferences). In particular, these methods compile metabolic reactions that, when assembled into a single network, constitute the most basic GEM content: the stoichiometric matrix. What this matrix is including actually depends on the modeler's choice of processes to include (*i.e.* the scope of the model). iML1515[44] only includes small-molecule enzyme-catalyzed metabolic reactions, and is the state-of-the-art *E. coli* GEM in that scope. Its detailed content, beyond the stoichiometric matrix, will be further described.

#### 1.3.2.1 General content

iML1515[44] contains 1877 metabolites, 2712 reactions and 1516 genes; all formulated with SBML terms, that link genes to reactions (with metadata links) and reactions to metabolites (with the stoichiometric matrix). It represents the metabolic network of the K-12 MG1655 strain of *E. coli*, and has been constructed from the genome NC\_000913.3 (NCBI Reference Sequence). Released in 2017, most of its content comes from the previous formulation, iJO1366, which was itself based on older formulations (from recent to old): iAF1260, iJR904, iJE660 (the very first *E. coli* GEM released in 2003)[54]. In the appendix section A.2, I briefly describe how such iterative refinements are performed. The following sections will focus on the additional iML1515[44] content that has not been described yet.

### 1.3.2.2 Compartments

Added to the metabolic network stoichiometric matrix, an important part of the knowledge contained in iML1515[44] lies in the compartmentalization of reactions. Indeed, each reaction is annotated according to its localization in the organism (with SBML terms). There are three compartments in iML1515[44]: the external (extracellular), periplasmic, and cytosol (intracellular) compartments. Note that some GEMs may include more organelles compartments. Different types of reactions can be identified according to their compartments localization. Exchange reactions (also called uptake fluxes, see appendix section B, 'Terminology') are very important as they simulate the available external compounds for the organism. In other words, they simulate the composition of the external compartment. Transport reactions are also critical as they are linking the external, periplasmic, and cytosol compartments. Internal reactions are happening in the cytosol compartment[55].

### 1.3.2.3 Uptake fluxes

Uptake fluxes (also called exchange reactions, see appendix section B, 'Terminology'), are the most critical fluxes to parametrize for a GEM. Setting a non-zero upper bound on such an uptake flux simulates a maximal speed of inflow for a given substrate that is available for the organism. In practice, these uptake fluxes are often used to simulate the influence of media compositions on the metabolic activity of the organism. However, when organisms are grown in complex media, finding suited upper bounds for uptake fluxes is very challenging. Therefore, measuring uptake fluxes is necessary in many metabolic modeling projects involving GEMs. Further assessment of uptake fluxes limitations will be done in the remainder of this section, when describing the biomass reaction (section 1.3.2.5) and the optimality principle (section 1.3.3.3) underlying FBA, the usual method to exploit GEMs.

### 1.3.2.4 ATP maintenance

iML1515[44] contains the ATP maintenance (ATPM) reaction that simulates the basal consumption of ATP by *E. coli* for survival (housekeeping). This is mathematically formulated by a lower bound on an imaginary ATP maintenance reaction, *i.e.* a minimum ATPM value that all iML1515[44] solutions should verify. Other default bounds on the remaining reactions are set according to the reversibility of reactions: if a reaction is irreversible, its lower bound is null, but if it is reversible, its lower bound is negative and the flux can take either positive or negative values (depending on the reaction direction). The ATPM is determined through experimental measures that may vary according to GEMs reconstructions. For iML1515[44], *E. coli* was grown on different substrates and the ATP fluxes were measured as well as the growth rate. Fitting a regression line relating the growth rate (x) and the ATP fluxes (y), Monk *et al.* could determine the non-growth associated ATPM (NGAM) to  $6.86 \text{ mmol.gDW}^{-1}.\text{hr}^{-1}$ , as the regression line y-intercept. This value represents the above mentioned basal ATP consumption of *E. coli* for its survival. Importantly, they also determined the growth-associated ATPM (GAM) from the regression line slope, at  $75.55 \text{ mmol.gDW}^{-1}.\text{hr}^{-1}$ , which is used as a stoichiometric coefficient of ATPM consumption for the biomass reaction described below.

### 1.3.2.5 Biomass reaction

Finally and most importantly, the biomass reaction is a critical part in reconstructing a GEM (added to the stoichiometric matrix, of course). This reaction is basically an imaginary reaction that simulates the biomass production rate, *i.e.* the growth rate of an organism. The reactants are all the



identified biomass precursors, with non-integer stoichiometric coefficients that are found to comply with experimental measures. For iML1515[44] and most GEMs, this relies on literature reviews, and a variety of measures such as fluxomics to relate the growth rate to different metabolite fluxes, or macromolecular fractions to estimate the cell's composition and consequently its needs to produce biomass. These measures are not performed for a vast array of conditions, so its confidence and extrapolation capacity to other conditions are quite limited. For example, the above mentioned GAM value may be varying according to the number of tested substrates and their concentrations. Consequently, studies proposing refined biomass reaction formulations from wider experimental data integration, such as for the Chinese Hamster Ovary (CHO) model by Zanghelinni *et al.*[56]. Ambitious approaches also propose to change the biomass reaction estimation process, such as the BOFdat approach relying on a genetic algorithm searching for the best biomass formulation from diverse experimental datasets[57].

In general, the realistic nature of growth rate predictions through the biomass reaction of GEMs heavily relies on uptake fluxes. Indeed, such uptake fluxes will determine the value obtained for the biomass reaction, through the optimality principle (section 1.3.3.3). In fact, the biomass reaction in GEMs can only predict the growth *yield* and not the growth *rate*, when uptake fluxes are left unmeasured. The growth yield describes the amount of biomass produced per amount of substrate consumed, an information that is directly described by the biomass reaction and metabolic network topology.

In this section 1.3.1 we have seen that GEMs reconstructions may vary according to their scope. I took as an example iML1515[44], the state-of-the-art *E. coli* GEM, to describe its content beyond the stoichiometric matrix reconstruction. In the next section 1.3.3, I will cover the different mathematical formulations for exploiting GEMs, which bring a diversity of applications.

### 1.3.3 Measuring and predicting metabolic phenotypes with GEMs

Exploiting GEMs rely on a variety of mathematical formulations and analysis methods. These can be roughly divided into (i) pathway analysis, which entirely relies on the metabolic network topology and the mathematical domain of convex analysis, (ii) Metabolic Fluxes Analysis (MFA), which relies on the integration of metabolic fluxes datasets and algorithms estimating the propagation of metabolites in the network, and (iii) Flux Balance Analysis (FBA) and related approaches, that rely on an optimality principle to predict possible solutions of GEMs. The first approach, pathway analysis, is described in the appendix section A.3; the second approach, MFA, is briefly described, and the third approach, FBA, is extensively described hereafter.

#### 1.3.3.1 GEMs rely on two main constraints

These approaches are different in many aspects, but they all rely on the two most central constraints that GEMs should satisfy in CBMs formulations. Namely, these are the mass-balance and flux boundaries constraints. Mass-balance states that any flux vector  $\mathbf{V}$  (*i.e.* a distribution of all fluxes values) and the stoichiometric matrix  $\mathbf{S}$  should always verify, at steady-state, the following equality constraint, that guarantees all reactions are balanced and equilibrated:

$$\mathbf{S} \cdot \mathbf{V} = 0 \quad (1.1)$$

The other constraint that metabolic networks should verify are inequalities, *i.e.* boundaries set

for each flux value. This ensures any metabolic flux ( $v_i$ ) value of the flux vector  $\mathbf{V}$  to be lower than its corresponding upper bound ( $UB_i$ ) and higher than its lower bound ( $LB_i$ ). This is simply formulated, for each flux  $v_i$  of  $\mathbf{V}$  by the following inequality:

$$LB_i \leq v_i \leq UB_i \quad (1.2)$$

Tuning the lower and upper bounds can be used to simulate metabolic states. Indeed, setting minimal or maximal values constraints for particular fluxes can be used to simulate reaction KOs (in that case the lower and upper bounds are null), observe metabolic phenotypes that respect precise flux values (in that case the lower and upper bounds are equal to a flux measure or flux value to test), or simulate minimal (e.g. for ATPM) or maximal (e.g. for substrate uptakes) flux activities.

### 1.3.3.2 Metabolic Fluxes Analysis: predicting metabolic phenotypes from flux measures

Metabolic Fluxes Analysis (MFA), in a nutshell, aims to integrate flux data in metabolic pathways or GEMs, to predict remaining fluxes that have not been measured. It is a widely used technique in metabolic engineering projects. The most basic approach is stoichiometric MFA[58], which consists in minimizing the error between uptake and secretion fluxes computed by a GEM under the constraint of equation 1.1, and the actual measurements. Added to uptake fluxes measurements, one can measure internal metabolites isotopomers, most often with  $^{13}\text{C}$ -labeling methods. This requires a formulation called  $^{13}\text{C}$ -MFA which adds another minimization of the error between measures and GEM computations, this time on measured isotopomers production rates. Finally, more sophisticated approaches were also developed, such as  $^{13}\text{C}$ -NMFA to account for non-steady-state MFA solutions, or COMPLETE-MFA for the integration of multiple isotopomer labelings.

Flux data to be used in MFA frameworks can be acquired through many experimental ways. External (uptake or secretion) fluxes can be obtained through exo-metabolomics or simpler quantification techniques of extracellular metabolites (which can be solutes or gas). Internal fluxes can be measured with isotopic labeling experiments. It consists in replacing nutrient sources with their isotopomers to enable, after spectrometry and chromatography measures, the precise quantification of individual metabolites production rates. But these rates can also be indirectly estimated from metabolomics[59] or more recently from transcriptomics[60] datasets.

In many cases, MFA can be used to decipher the metabolic activity of an organism in different conditions. For example, it has been used to characterize *E. coli* DH5- $\alpha$  metabolic phenotype when grown in complex rich media[61]. Also, it has been used to discriminate between the transcriptional control mechanisms of respiration and fermentation metabolism[24].

### 1.3.3.3 Flux Balance Analysis: predicting metabolic phenotypes from an optimality principle

#### 1.3.3.3.1 Computation framework overview

The main ways to use and exploit GEMs are FBA and related approaches. Instead of minimizing the errors between measurements and GEMs computations as in MFA, these rely on an optimality principle to find solutions. Thus, they can be used without any flux measurement. A reaction or combination of reactions is set as an objective to maximize (or minimize, in rare cases). Consequently, the mathematical framework for such approaches is a Linear Program (LP). Subjected to constraints of equations

1.1 and 1.2, this framework maximizes the following term:

$$Max(c^T \cdot V) \quad (1.3)$$

With  $\mathbf{c}$  the vector indicating which reaction(s) to maximize. Most often, this objective-defining vector is full of zeros with a single value of 1 for the reaction to maximize. This reaction is most often the previously described biomass reaction. Constraints from equations 1.1 and 1.2 defining the convex solution cone, the maximization then searches for the highest possible value of 1.3 inside this cone. Note that there is no guarantee of a unique flux distribution maximizing 1.3. A visual depiction taken from Kaufmann *et al.*[62] of the FBA methodology is given on Figure 1.8.

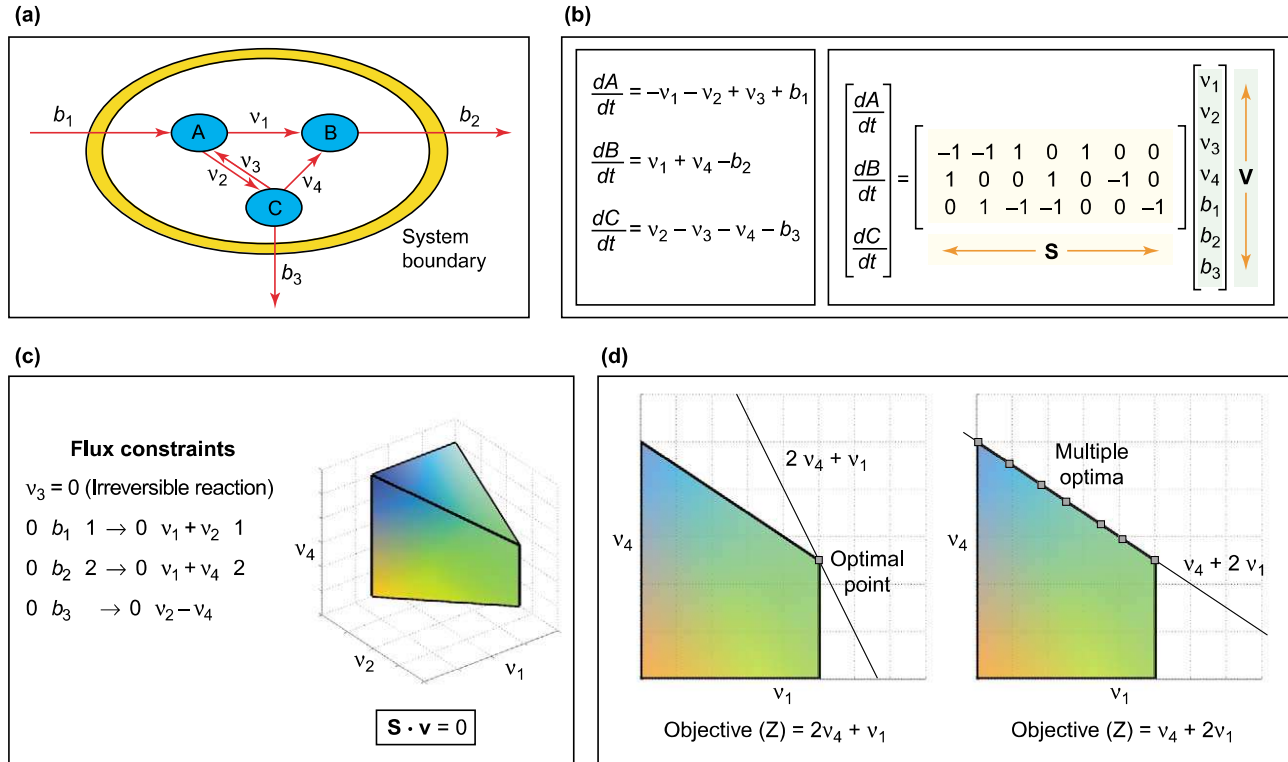


Figure 1.8: Methodology for flux balance analysis. (a) A model system comprising three metabolites (A, B and C) with three reactions (internal fluxes,  $v_i$ , including one reversible reaction) and three exchange fluxes ( $b_i$ ). (b) Mass balance equations accounting for all reactions and transport mechanisms are written for each species. These equations are then rewritten in matrix form. At steady state, this reduces to  $S \cdot V = 0$  (equation 1.1). (c) The fluxes of the system are constrained on the basis of thermodynamics and experimental insights. This creates a flux cone corresponding to the metabolic capacity of the organism. (d) Optimization of the system with different objective functions ( $Z$ ). Case I gives a single optimal point, whereas case II gives multiple optimal points lying along an edge. (Authors: Kauffman *et al.*[62]; license obtained on the Rightslink® platform).

### 1.3.3.3.2 Linear Programming for FBA

A variety of LP solvers exist to perform such FBA computations. Two main families will be cited here, that are interior-point and edge-following algorithms. As briefly explained above, LPs have a solution space that is a convex cone (polytope, polyhedron). In simple terms, edge-following algorithms such as the simplex algorithm[63] find the optimal solution relying on a procedure that explores the edges of the solution cone, whereas interior-point methods' procedures explore the cone from the inside[64]. Both edge-following and interior-point methods are solvable in polynomial time through iterative procedures, falling in the complexity class P, which makes these algorithms relatively effi-

cient and computationally cheap. Interestingly, enumerating possible solutions would be intractable compared to finding a single optimal solution. In the FBA context, this is the case with the enumeration of Elementary Flux Modes (EFMs) being computationally intensive. Also, note that in the FBA context, edges of the cone are Extreme Pathways (EPs, see appendix A.3 for explanations on EFMs and EPs). For intelligible mathematical details of both edge-following and interior-points methods, I recommend the review of Chandru & Rao[65]. These methods are still widely used today, with different formulations, but there is also a recent emergence of more innovative ways to approach LP solving, notably through fuzzy[66] and machine learning[67] approaches.

Computational toolboxes have been developed for FBA and more generally Constraint-Based Modeling (CBM) methods for GEMs. Worth citing here are the Constraint-Based Reconstruction and Analysis (COBRA) packages in python and Matlab programming languages[68]. They provide an array of functions, standardized objects, and mathematical methods to ease the Linear Programming manipulations with GEMs. That includes standard LP solvers, *e.g.* the GNU Linear Programming Kit (GLPK, an open-source simplex-based method).

#### 1.3.3.3 FBA variations: different computation frameworks and biological scopes

To overcome the shortcomings of FBA, variants of the method have been proposed. Some approaches imply a subsequent LP solving after regular FBA is performed: the first FBA solution value for the reaction to optimize is used as an exact ( $LB=UB=solution$ ) or loose (*e.g.*  $LB=0.9*solution$  and  $UB=1.1*solution$ ) constraint for that reaction, then another LP is performed. In parsimonious FBA[69] (pFBA), the second LP consists in minimizing the sum of all fluxes, which is biologically meaningful for the cell's resources. In Flux Variability Analysis[70] (FVA), the second LP computes the minimal and maximal flux values for a list of reactions, showing different possible solutions instead of a single FBA solution. Flux sampling[71] does not rely on two subsequent LPs, but on a variety of possible sampling algorithms that explore the solution space and return sets of solutions compliant with the constraints of the problem. Therefore flux sampling methods do not rely on the LP optimization of an objective function.

Other variants of FBA focus on extending the biological scope of the framework. As mentioned previously, many formulations exist, whether they integrate regulation mechanisms, resource cost, or time (Figure 1.7). A few are worth citing here given the scope of this dissertation. Regulatory FBA (rFBA) is linking GRNs with GEMs thereby integrating regulation and metabolism together[28], dynamic FBA (dFBA) simulate time-series as discrete steady-states of FBA[72], Resource Balance Analysis (RBA) integrates macromolecular synthesis processes as a part of GEMs[73]. Note that these formulations can also be analyzed with pFBA, FVA, or flux sampling frameworks, as they are just modifying the biological context and scope compared to FBA, but all rely on similar mathematical formulations (*i.e.* constrained linear optimizations).

A specific task to make FBA predictions quantitatively accurate consists in constraining the uptake fluxes, as stated in section 1.3.2.5. Without measures of such fluxes, the FBA computation relies on a choice of exchange reactions bounds values, to simulate a particular environment for the cell. To ease this task, a variety of formulations can be used. For example, Hoppe *et al.* have used additional thermodynamic constraints[74]; Marmiesse *et al.* have used regulatory network steady states to estimate uptake fluxes[75]; saturation FBA (satFBA) simulates the uptake fluxes of metabolites by transporter saturation kinetics[76]; CoRegFlux integrates dynamic transcriptomics to predict dynamic uptake fluxes[77].

#### 1.3.3.3.4 Optimality principle in FBA: a questionable approach

A crucial criticism one can have on FBA is the need for a reaction to optimize in order to make predictions. Indeed, without such a Biological Objective Function (BOF) to maximize, constraints imposed by GEMs only define a large set of possible solutions. This space, as stated in the previous section 1.3.3.3.3, can be efficiently sampled with flux sampling algorithms. But these possible solutions may vary a lot according to the sampling algorithm and the actual solution space (*i.e.* the constraints imposed on the system, the shape of the polytope), and it might be hard to interpret. Therefore, there is a clear need for a BOF to optimize, to ease and standardize the exploitation of GEMs. The BOF is most often the biomass production reaction, previously described and criticized for its lack of generalizing ability. The optimality principle is a core assumption of the FBA framework to reach supposedly realistic solutions: under FBA assumptions, an organism would always maximize its growth, in any condition. This is obviously questionable, as many examples are found in nature of organisms that adapt to changing environment by actually *reducing* their growth rate, *e.g.* with dormant states of persistence under stress conditions[78]; or with niche construction and other community level interactions (*e.g.* biofilm formation)[79]. More generally, we can assume that evolution has led organisms and their metabolic networks to adapt to changing environments, with survival and reproduction as objectives, rather than the maximization of their growth rates. Such criticism of the BOF in FBA and more generally for the study of metabolic networks is an important and complex discussion that is further explored by Berkhout *et al.*[80]. Two of the proposed solutions to overcome the BOF limitation is to use multi-objective formulations of metabolic networks[81], and integrate experimental data in more standardized and automated ways[57]. To temper the above criticism over the BOF and optimality principle, it is important to note that in ideal laboratory conditions, metabolic networks do show such growth rate maximization objectives. Indeed, Artificial Laboratory Evolution (ALE) experiments have shown that, if an organism is left for hundreds of generations to grow and adapt to the same substrates, it can reach the metabolic network theoretical optimality of growth found by FBA[69].

As stated in section 1.3.2.5, determining uptake fluxes are critical for the BOF to accurately predict growth rates, otherwise able to predict growth yields only. When parametrizing a GEM, upper bounds on uptake fluxes are thus critical. However, the relationship between the uptake fluxes and the BOF is clearly oversimplified: in most cases, the optimality principle tends to maximize uptake fluxes, in order to maximize the BOF. The reality is much more complex: from physico-chemical conditions and the cell's internal state, the organism will have a certain uptake flux. There is a complex relationship between the availability of a substrate and its uptake, which is strongly oversimplified with GEMs, by the optimality principle.

Even though there are some drawbacks to FBA and its variants, it is still a powerful approach to make large-scale metabolism predictions with relatively simple frameworks. Using such models and frameworks is very widespread in metabolic engineering, drug discovery, and other domains of application[82], with compelling achievements such as the design of antibiotics[83], or the engineering of industrially relevant strains[84]. In general, such applications rely on large-scale *in silico* screening of candidate metabolic modifications to perform, such as reactions to add or remove (by knock-ins or knock-outs of enzyme-coding genes), or medium composition optimization. On a more theoretical ground, FBA frameworks can be used for regulation mechanism discoveries, with 'gap-filling' approaches similar to those used for GEM reconstruction[55], as described in the appendix section A.2. For example, screening a large number of phenotypes with Biolog phenotyping arrays[85] enables the identification of novel possible substrates for the growth of an organism, then refining the model

to include such substrates' uptake fluxes.

In this section 1.3, I have reviewed what is the general GEM content and how they are reconstructed, as well as some of their mathematical frameworks to be exploited. I have also mentioned the capabilities and limits of such GEMs, and cited some alternative formulations. So far, we have not mentioned the formulations including Machine Learning (ML) processes, which are more and more frequently used to increase the capabilities of GEMs. In the next section 1.4, I will recall the differences between mechanistic and ML approaches, then cover approaches to exploit GEMs with ML processes, and the emerging hybrid modeling field that aims to build models with both machine learning and mechanistic abilities.

## 1.4 Hybrid models: reconciling mechanistic and machine learning models

Mathematical models are abstract representations of ideas, concepts, observations and generally all kinds of natural phenomena. They are necessarily approximations of the real-world. The famous aphorism of Georges Box, "all models are wrong, but some are useful"[86] underlines such inherent imperfection of models. Importantly, one must keep in mind the indirect relationships between the 'truth' (the real-world phenomenon), the observations we make, and the models that are based on such observations. Observations of the 'truth' are inherently imperfect, as they are made through a 'window' limited by instrumentation capabilities or individual bias. This is especially true when observing complex and molecular scale phenomena such as metabolism. Integrating observations in models is also imperfect, because mathematical models can have inherent limits to what behavior they can simulate or not. Therefore, a mathematical model is far from an equation reproducing the truth - in that sense, 'all models are wrong'. For a model to be 'useful', two main objectives seem to be facing in a trade-off: it should provide insights on the 'truth', deciphering causality with mechanisms, and built in the most simple way possible; but it should also be also approximating the 'truth' in the most accurate and generalizing way possible. Also, a useful model should be able to predict new behaviors that have not been observed during the model construction.

These two objectives relate to the two main families of mathematical models: Mechanistic Models (MMs), based on precisely predefined (*a priori*) components that model phenomena by causality encoded in mechanisms[87, 88]; and the Machine Learning (ML) models which are based on statistical learning methods that model phenomena by analyzing the patterns found (*a posteriori*) in observational data[88-90]. In simpler terms, MMs relate the input (independent variables) to the output (dependent variables) with mechanisms derived from knowledge; whereas ML relates the input to the output from their observed statistical relationships.

The hybrid models developed in this Ph.D. incorporate concepts and methods from both ML and MM, in particular using Artificial Neural Networks (ANNs) and GEMs. Consequently, in the following part of the introduction, I will review the basics of both MM and ML, with more details given for ANNs, then compare the two approaches capabilities and limits. Next, I will present approaches that combine MM and ML for exploiting GEMs. Finally, I will present the hybrid modeling approach which aims to formulate single models with both ML and MM capabilities, instead of combining both approaches with separate models. Therefore, I will introduce the research gap that is targeted by this Ph.D. dissertation, *i.e.* the development of hybrid models for GEMs.



## 1.4.1 Machine learning and mechanistic models: two seemingly opposed approaches

### 1.4.1.1 Mechanistic modeling for biology

#### 1.4.1.1.1 Overview

As briefly stated just above, Mechanistic Models (MMs) can be defined as the family of mathematical models whose computations rely on precisely predefined mathematical components (variables, equations, parameters...) that represent causal mechanisms, which all have a meaning and purpose based on our knowledge of the system we are modeling, or new hypotheses we want to test. They are sometimes referred to as “white-box”, “knowledge bases” or “knowledge-driven models”, to emphasize the fact that every part of a MM is defined based on knowledge.

In most cases, MMs focus on their parsimonious nature, *i.e.* they are models built as simply as possible, with as few components as possible, in order to have better explanatory abilities. A simple, useful and extremely parsimonious example of MM, is the ideal gas law ( $PV = nRT$ ) which is relating pressure and volume of an hypothetical ideal gas to the quantity of matter, temperature, and the ideal gas constant. This trivial MM example, even if it does not generalize to all kinds of gasses in all conditions, is very useful for its ability to make a fair approximation of many gas phenomena, with just one constant derived from experimentation.

#### 1.4.1.1.2 Categories of MM

Many categories of MMs have been delimited. They can take time into account, and fall in the category of dynamic MMs, unlike static MMs that focus on steady-states. This is the case for different methods to exploit GEMs, as discussed in section 1.3.1.2. Briefly, other categories discriminate between models that are stochastic or deterministic, linear or nonlinear, discrete or continuous. For example, FBA applied on a GEM is a static, deterministic, linear and continuous MM approach. According to the modeling goals, one will choose the most suited category of model to use or develop. Importantly, MMs can rely on different mathematical methods, to relate the input to the output. These mathematical methods are sometimes referred to as ‘governing equations’ of the model. These are most often differential equations such as Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs) in dynamic MMs. Other types of governing equations can be found in logical models (*e.g.* Boolean Networks), Petri Nets (discrete events dynamic MM), or agent-based models.

#### 1.4.1.1.3 Compelling MM achievements in biology

In biology, MMs have been ubiquitous to understand underlying phenomena, with relatively old models such as the Hodgkin-Huxley’s model of neuron action potential[91], or the Michaelis-Menten’s model of enzyme kinetics[92]. Both of these models have proven to be very useful: from parsimonious formulations that rely on identified, biologically meaningful components, they are able to approximate the behavior of many systems in many situations. More complex and more recent useful MMs have also been developed. For example, one can compute the annealing temperature of DNA from its sequence[93] or *E. coli* traits from its metabolic network, with iML1515[44] as discussed in section 1.3.2. In systems biology, MMs are widely used to confront complexity as well discussed by Robert Phair[87]: “Modeling is quantitative hypothesis testing; it is classical scientific method combined with computation to help us to manage the enormous complexity of cell biology”. A wide variety of MMs

for biology are stored under the database BioModels[94], which provides valuable resources for modelers that aim to exploit MMs in such quantitative hypothesis testing approaches.

#### **1.4.1.1.4 MMs limitations**

Observed data is usually integrated in MMs in two main ways: either to tune the model's parameters (for example tuning the kinetic parameters of enzymes in a Michaelis-Menten model); or to curate the model itself based on prediction-measure disagreements (like in the gap-filling approach of GEMs, see appendix A.2). The first way relies on statistical methods pushing the MM to reproduce observed data more faithfully, while the second way relies on quantitative hypothesis testing, *i.e.* the uncovering of novel components of a MM based on its agreement with observed data. Beyond these two approaches, there are not many ways of data integration with MMs. When facing very large and diverse datasets, MMs may have difficulties at integrating all data in efficient and scalable manners. This issue is related to a major limitation of MMs, in their ability to accurately predict behaviors of complex systems. Indeed, since they are built with the constraint of precisely defining the mechanisms underlying a given phenomenon, their predictive power performance will be necessarily limited by the completeness of the knowledge used to model the system. When facing extremely complex systems, such as protein folding or natural language processing, MMs cannot make reliable predictions as we are missing too much knowledge on the actual mechanisms underlying such phenomena. In such cases, modelers tend to use another class of models: Machine Learning (ML).

I just mentioned that statistical methods enable a tuning of MMs' parameters, based on observations, making them 'statistical models' to some extent. But another class of models place such statistical methods at their very core: the ML models. They are purely statistical models, since they entirely rely on tuning large sets of 'meaningless' parameters, without any constraints from knowledge, assumptions, or hypothesis to test. The following section 1.4.1.2 will cover how such purely statistical models are built, and what are their achievements in biology.

### **1.4.1.2 Machine learning for biology**

#### **1.4.1.2.1 Overview**

Machine Learning (ML) is a part of the wider Artificial Intelligence (AI) field, which is not a precisely delimited field, but that can be defined with its goal: the scientific endeavor of building computing systems that have human-like problem-solving abilities, so they can be considered as intelligent. ML, a subfield of AI, is often defined as the set of algorithms that enable computers to 'learn' by themselves. In more practical terms, ML is the set of mathematical methods and models purely relying on statistical observations made on datasets[90]. Unsurprisingly, ML is sometimes called 'statistical learning', and ML models are sometimes called 'statistical models'. The basic assumption underlying ML is that observing a system's behavior yields enough information to reliably predict the general behavior of the system in unseen conditions. For example, feeding the observation of the sun rising for 10,000 days to an ML model, it will predict that it will rise again the next day, without an understanding of how the sun rises. Indeed, statistical ML models aim to 'mirror' what is observed, agnostically of the underlying mechanisms. In other words, ML models aim to capture a data-generating phenomenon without an understanding of it.

The three main ML frameworks can be roughly delimited as the following: (i) supervised learning, which consists in feeding example of inputs-outputs pairs to the model so that it is able to predict



an output from a new input, (ii) unsupervised learning, which is closer to statistical analysis in the way it searches for data patterns without input/output segmentation of a dataset, (iii) reinforcement learning, which is based on an iterative exploration of an agent response, constructing the data and ML model with a balance between the exploration of the agent response space and the optimization of a desired behavior of the agent. These 3 frameworks can rely on a variety of mathematical methods. Note that in the remainder of the Ph.D. dissertation we only use supervised learning.

#### **1.4.1.2.2 Categories of ML models**

Many mathematical formulations have been proposed for ML models, such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Gaussian Processes (GPs), Random Forests (RF) or one of the most basic form of ML, Linear Regression (LR)[89]. Each of these methods have a varying degree of flexibility and tunability, brought by their architecture (*i.e.* set of hyperparameters), which can be related to the maximum number of iterations or parameters in the model, the optimization methods that are used, etc... ANNs are the most flexible type of ML models, as they can be built with varying number and size of different layers types (convolutional, Long Short-Term Memory, Recurrent Neural Networks, transformers, auto-encoders...), each with different activation functions such as the Regularized Linear Unit (ReLU) or hyperbolic tangent (tanh) functions. Moreover ML models and especially ANNs may have many hyperparameters to tune, such as dropout rates, L1/L2 regularizers weights, training rates, solvers and more. This overall increases the diversity of ML models, making them adaptable to various datasets. In section 1.4.1.3, I will explain in much more detail the basic functioning of ANNs, their core mathematical principles, as well as the reservoir computing approach.

#### **1.4.1.2.3 Compelling ML models achievements in biology**

Since the postgenomic era and the advent of high-throughput -omics data, the use of ML has gained increasing popularity in biology, notably with a variety of models integrating multi-omics data[95]. These performant models push the emergence of new fields such as precision medicine. Recently, another grand breakthrough in biology that was brought by ML advances is the accurate protein structure prediction performed by AlphaFold[96]. Importantly, many fields outside biology benefit from ML capabilities, such as natural language processing and image recognition. By late 2022, specific ML technologies, namely generative AI, have impressed with performant tools such as ChatGPT or MidJourney that can reach creativity and precision levels comparable to human works[97], which questions the implication of ML to replace human workforces in societies[98].

#### **1.4.1.2.4 ML models limitations**

We mentioned in section 1.4.1.2.2 that ML models can have a diversity of hyperparameters, which brings flexibility to the models. But this can also be a drawback as they can be hard to fine-tune. This common issue of ML models is especially true for ANNs. Another issue that is especially striking with ANNs, is the difficulty to interpret learned parameters beyond statistical relationships between variables. The deeper an ANN is, the harder it will be to interpret the learned parameters to identify mechanisms of causality; even though some approaches attempt to tackle this issue[90]. Importantly, each ML method (and each architecture by extension) has different capabilities depending on the datasets and problems that we aim to solve. In other words, ML is highly context-dependent, in terms of training sets (the data used to build them), model's hyperparameters and formulated tasks to solve. Finally, one flaw of ML is to 'overfit', which is a scenario where ML is mirroring the observed

data ‘too well’, up to the point that it is fitting the intrinsic noise of the observations, instead of the general pattern. In such cases, even if the ML model might be perfectly reproducing observed data, it will be unable to generalize for unseen data; therefore failing to model the actual data-generating mechanism. A widespread method to detect such behavior is the cross-validation method, where a part of the training set is reserved for testing the ML model without having it trained on (in a case where the training performance is far superior to the testing performance, the model is overfitting). Another flaw of ML models is the curse of dimensionality, *i.e.* the need for exponentially larger datasets when increasing the number of parameters in the model[99], which is necessary when modeling increasingly complex data-generating phenomena.

To summarize, in contrast to MMs, the fact that ML models are solely based on statistical analysis of observed data brings strong limitations to (i) the interpretability of learned parameters, (ii) the generalization abilities of the model beyond the dataset used to build the model, (iii) the ability to understand mechanisms through hypothesis-driven reasoning, and (iv) the need for large datasets. In one hand, ML models show more predictive power than MMs, in the sense they are able to reproduce more faithfully the observations, at least in the context of a dataset used to train an ML model. In the other hand, ML models clearly lack explanatory abilities of MMs, in order to perform quantitative hypothesis testing and better understand the systems we are modeling. However, MMs lack predictive power when used to model complex and partially known systems, and their goal of being built with parsimony and meaningful mathematical components may be seen as too ambitious when facing the huge complexity of certain modeling tasks. These differences are further discussed in the appendix section A.4, in general and in the particular scope of metabolic modeling.

In section 1.4.2, we will introduce methods that tackle such limitations of ML and MM, for metabolic modeling. But first, I will describe in more detail how ANNs function, in the next section 1.4.1.3, since it is imperative to understand the basic functioning of ANNs for the remainder of the dissertation.

### **1.4.1.3 Artificial Neural Networks**

The first neural networks appeared *circa* 1800, in their simplest form: linear regression. Indeed, linear networks proposed by Legendre and Gauss were built with inputs directly connected to the output, by a weighted sum of inputs to compute the output[100]. With input vectors and desired target values (also called labels), the method could adjust the weights linking each input to each output, based on the minimization of the sum of squared errors between the outputs and the corresponding targets. This method was at the time called ‘least squares’ or ‘linear regression’ but already showed the very core functioning of ANNs. An analogy with biological neurons was made, and further improvements brought more diverse concepts to the mathematical foundations of ANNs, such as activation functions, hidden layers and gradient backpropagation.

#### **1.4.1.3.1 Biological inspiration and mathematical foundations**

In ANNs, we call the nodes of the network ‘neurons’. These can take any real number value during the computation. An ANN has at least two layers of neurons: an input layer (which does not perform any computation and simply receives a given input vector) and an output layer. In between, it can also have hidden layers that are used for modeling nonlinear problems. Neurons of the hidden and output layers, in most architectures, compute their output as a ‘computational neuron’, the basic unit of ANNs. This computational neuron stems from a biological analogy that inspired the perceptron, which will be briefly described next.

Historically, the perceptron has set the mathematical foundations of ANNs, by introducing the ‘computational neuron’. In a nutshell, the perceptron is inspired from biological neurons, to perform a linear classification. It is a single-output neural network without hidden layers. The perceptron makes a weighted sum of inputs, adds a bias (any real number), and feeds the result to a step function, which outputs a binary response (0 or 1). The analogy between a neuron and such a perceptron is quite straightforward: (i) the input signals are bioelectric signals from several synaptic terminals in the neuron, and several arbitrary input values in the perceptron, (ii) these signals travel through the axon in the neuron, and through the weighted sum in the perceptron, (iii) the neuron transmits or blocks the bioelectric signal with an action potential function, and the perceptron produces its final output through a step function, the activation function of the computational neuron. A schematic of a biological neuron and a computational neuron are given in Figure 1.9.

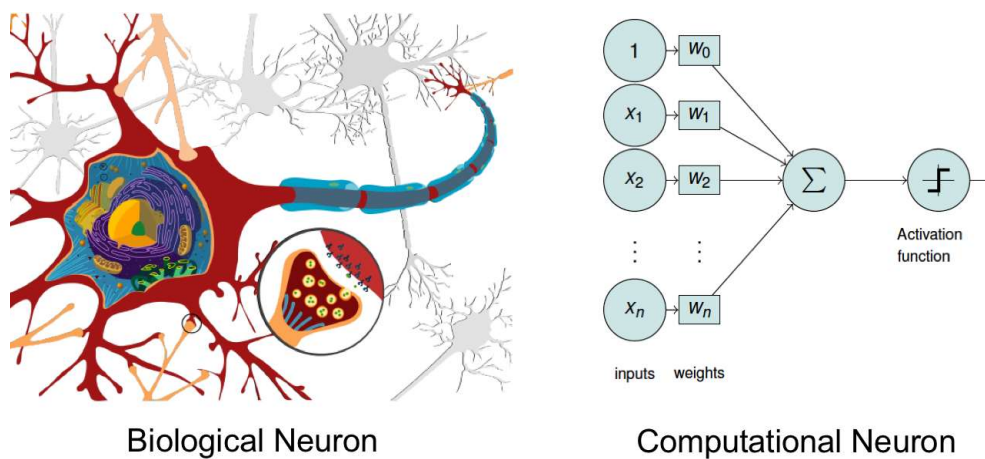


Figure 1.9: Biological and computational neurons. The biological neuron schematic (left) displays the cell body on the left, with its axon in blue on the right, ending with synaptic terminals. The computational neuron schematic (right) displays  $n$  inputs, from  $x_1$  to  $x_n$ , that are summed after being multiplied by their corresponding weights  $w_1$  to  $w_n$ , as well as a bias  $w_0$  to add prior to an activation (here a step) function for the final response. (Authors: Maier *et al.*[101]; license: CC BY 4.0)

Mathematically, the computational neuron shown in Figure 1.9 follows this formula, with  $f$  the output of the neuron, and *step* the Heaviside step function (which outputs 0 if the input is below 0, and 1 if the input is above 0):

$$f(x, w) = \text{step} \left( \sum_{i=1}^n x_i w_i + w_0 \right) \quad (1.4)$$

Importantly, a multi-output perceptron contains as many computational neurons as outputs. Also, in a multi-layer perceptron (with one or more hidden layers), each hidden layer and output layer node’s output will be computed with such a computational neuron. More generally, most ANNs have an architecture based on the interconnection of many computational neurons. Importantly, note that the activation functions are not limited to a step function in other ANNs than the perceptron, and depend on the task to solve (multi-class classification, regression, etc...). Moreover, such step functions were later replaced by a sigmoid activation function, since its differentiability enables gradient backpropagation through hidden layers (the step function is non-differentiable for zero, and its derivatives are zero elsewhere, which blocks gradient descent, the optimization method for ANNs that is explained below).

An example of Multi-Layer Perceptron (MLP), with one hidden layer, is schematized in Figure 1.10.

Here, the input layer does not perform any computation, whereas the hidden and output layers have computational neurons as their basis.

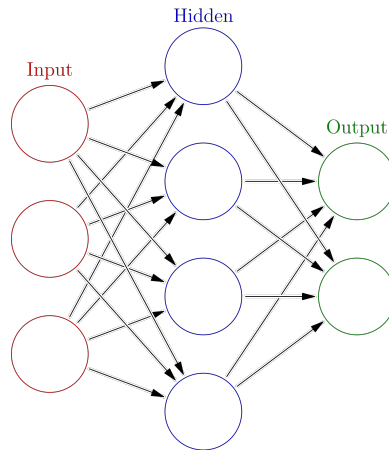


Figure 1.10: Multi-Layer Perceptron architecture example. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another. The input layer (red) receives vectors of 3 values, which are transformed to produce vectors of 4 values in the hidden layer (blue), which are in turn transformed to produce vectors of 2 values, in the output layer (green) of the perceptron. (Authors: Glosser.ca; license: CC BY 3.0)

To give more mathematical detail into the MLP functioning, the next Figure 1.11 shows how the MLP performs a computation, with each computational neuron displayed.

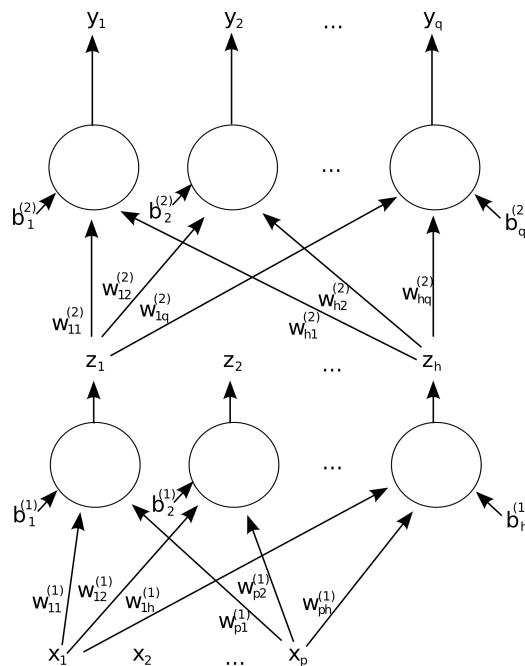


Figure 1.11: Generic feedforward MLP computation schematic, with weights and biases terms labeled. Arrows originating from  $x_2$  and  $z_2$  are omitted for clarity. This network has  $p$  inputs,  $h$  nodes in its hidden layer, and  $q$  outputs (in the previous Figure 1.10,  $p=3$ ,  $h=4$ ,  $q=2$ ). (Author: Mcstrother; license: CC BY 3.0)

Figure 1.11 shows a general example of a feedforward MLP computation. The chained computation is displayed, with each input ( $x_1$  to  $x_p$ ) connected to all hidden layer nodes ( $z_1$  to  $z_h$ ), and each of those nodes connected to all output layer nodes ( $y_1$  to  $y_q$ ). Two weights matrices are used for the computation,  $w^{(1)}$  and  $w^{(2)}$ , respectively to compute  $z$  from  $x$ , and  $y$  from  $z$ .  $w^{(1)}$  is of size  $(p, h)$ , with  $w_{ij}^{(1)}$  the weight to apply on  $x_i$  to produce  $z_j$ ; and  $w^{(2)}$  is of size  $(h, q)$  with  $w_{ij}^{(2)}$  the weight to apply on

$z_i$  to produce  $y_j$ . There are also two bias vectors that are used, namely  $b^{(1)}$  to compute  $z$  and  $b^{(2)}$  to compute  $y$ . These are simply added before the computation of hidden ( $z$ ) or output ( $y$ ) nodes, and they are the same size as their corresponding vectors,  $z$  and  $y$ . The hidden ( $z$ ) and output ( $y$ ) nodes are using the same computational neurons as shown in Figure 1.9, to compute their output. Note that here, the bias is written  $b$  instead of  $w_0$  (unlike in equation 1.4).

Now that we have seen how a classical ANN architecture makes a computation from a given input, we shall see how an ANN learns, *i.e.* how it adapts its weights and biases to best fit the data.

#### 1.4.1.3.2 Gradient backpropagation for learning abilities

In this part, we will focus on the main method to train ANNs in supervised learning: gradient backpropagation. ANNs learn in two steps: (i) compute an output from an input, as explained previously (1.4.1.3.1), called ‘feedforward’ or ‘forward pass’ step, then (ii) from the output, compute a loss function and its gradient, and backpropagate that gradient to update the ANNs parameters (*i.e.*, the weights and biases), called ‘backward pass’.

Training an ANN is an optimization problem. In particular, by minimizing a loss function, which is usually designed to be lower when the ANNs computation matches the target data better, the ANNs learn from data. Here, we will consider an imaginary loss function to be minimized, noted  $L$ . In short, the minimization aims to find the best set of parameters (weights and biases), noted  $\theta$ , in order to minimize  $L$ . To do so, one computes the gradient of the loss function, *i.e.* the set of partial derivatives of that loss function for each element of  $\theta$ . The vector chain rule is used to make such computation, since the loss function is not directly expressed with  $\theta$  as variables, but with the ANNs computations.

The gradient of a function can be thought of as a map of the direction and speed of change of that function’s value. This ‘map’ can be used to minimize a given function in efficient manners, with different gradient descent algorithms. Computing a gradient of the loss function is easy for the final layer’s weights, but hardly doable for other layers. The solution is to backpropagate the gradient: first computed only on the final layer’s weights, the derivatives are then computed layer by layer, toward the beginning of the ANNs. Once all partial derivatives are computed, they are used to perform gradient descent: the ANN updates corresponding weights, with changes equivalent to the negative partial derivative value multiplied by the learning rate (which is a learning hyperparameter to set) in the classical gradient descent algorithm. This has been tweaked in more recent gradient descent algorithms such as Stochastic Gradient Descent (SGD) algorithms, *e.g.* ‘adam’[102]. Importantly, the batch size is a hyperparameter for SGD algorithms, that I mention in the following chapters. It controls how many training data entries are used to perform one gradient descent step where parameters are updated (*i.e.* one training phase, a forward and a backward pass). Another parameter that I will use is the dropout rate, which controls a fraction of a layer’s neurons to be deactivated for a training phase (both forward and backward pass). Using dropout is popular to increase generalizing and robustness abilities of ANNs[100].

Until now, we considered classical ANNs to explain their mathematical foundations and learning abilities. Those ANNs were always performing their computation in one way, from the inputs to the outputs, a category termed Feedforward Neural Networks (FNNs). But in other ANN formulations, such as in Recurrent Neural Networks (RNNs), the computation does not rely on such a straightforward path, which drastically changes their computation abilities.

### 1.4.1.3.3 Recurrent Neural Networks

RNNs are based on similar mathematical foundations as FNNs, with computational neurons as the basic processing unit, and gradient backpropagation most often used as the learning mechanism. However, computational neurons in RNNs can have feedback loops, where the output of a node (or a subsequent node) is used as its own input. This plays a crucial role in establishing a memory of previous events when processing a sequence, *i.e.* a series of inputs.

In fact, the first RNNs (1920s) were not showing learning abilities: they were simply iterative procedures, designed to reach an equilibrium from certain input conditions. In that sense they were mechanistic models that would not learn from data but solve a given problem under certain conditions[100]. In the following chapters, we use such non-learning RNNs to surrogate FBA, inspired from projection networks[103], and in particular Hopfield networks[104].

More widely used are the RNNs with learning abilities, such as the Long Short-Term Memory (LSTM) unit and variations being one of the most popular ANN architectures[100]. When connecting outputs to inputs of a neuron, some shortcomings had to be overcome. One of those shortcomings was the issue of vanishing or exploding gradients. This issue is also observed with deep FNNs. By backpropagating the gradient through many layers (in RNNs, each feedback loop is considered a layer), one can observe the gradient values to either vanish (extremely low values) or explode (extremely high values). In both cases, no convergence can be achieved for the loss minimization, the learning fails. Some mathematical tools were developed to avoid this issue, such as gradient clipping, batch normalization and the LSTM unit for RNNs[100].

RNNs can display nonlinear dynamical behaviors. Such nonlinear dynamics are used in an ANN-related approach called ‘reservoir computing’, that attempts to ‘store’ computational power in so-called ‘reservoirs’ (either non-trainable *in silico* networks, or physical phenomena).

### 1.4.1.3.4 Reservoir computing

Reservoir Computing (RC) is an approach originally designed to process sequential data (time-series). The overall goal of RC is to reduce computational burden for modeling complex dynamical behaviors by exploiting the computational power of a ‘reservoir’, which can be an *in silico* process that is not trainable or a physical phenomenon that is perturbed and has its responses measured. A reservoir computing system comprises a reservoir for mapping inputs into a high-dimensional space and a readout for analyzing patterns within these high-dimensional states[105]. The readout is trainable but should be kept simple, with linear regression for example. Figure 1.12 shows a schematic of conventional RC (using *in silico* non-trainable RNNs that simulate dynamical behaviors) and physical RC (using real-world phenomena of which we measure dynamical behaviors) frameworks.

Usually, in conventional RC, RNN-based reservoirs are generated by randomly sampling weight matrices, with constraints on their spectral radii; but novel approaches also propose more performant ways to generate *in silico* reservoirs[106].

In physical RC, a bucket of water has been used as a reservoir for complex sequential data processing. One might wonder if biological processes could be used as well for the same purpose as the bucket of water. For that, we would need to be able to precisely measure some responses to some perturbations, which seems possible with -omics technologies. With particular ‘killer applications’, biological reservoirs might lead to efficient biocomputing[40, 105].

Importantly, we use the term ‘Reservoir Computing’ in the remainder of the dissertation to designate a modeling approach loosely inspired from the regular RC methods described here. This is further explained in Chapter 2, section 2.3.5.



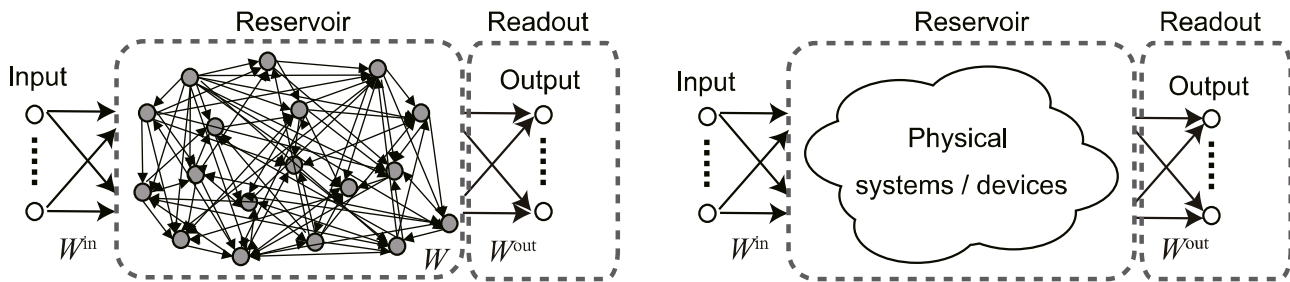


Figure 1.12: Reservoir Computing frameworks, where the reservoir is fixed and only the readout weights ( $W^{out}$ ) are trained. Importantly  $W^{in}$  and  $W$  are not trained. (a) A conventional RC system with a RNN-based reservoir. (b) A physical RC system in which the reservoir is a physical system or device. (Authors: Tanaka *et al.*[105]; license: CC BY 4.0)

In the following section 1.4.2, I will describe how MM and ML are complementary approaches for metabolic modeling. Then, I will introduce the emerging hybrid modeling field which aims to formulate innovative models incorporating elements from both MM and ML modeling principles.

## 1.4.2 Mechanistic and Machine Learning models as complementary approaches for metabolic modeling

### 1.4.2.1 Combining machine learning and mechanistic approaches

#### 1.4.2.1.1 State-of-the-art approaches

MM and ML are seemingly opposed approaches. As discussed in the appendix section A.4, their design and goals are radically different, including for systems biology models. Instead of opposing the two approaches, many projects attempt to combine them for metabolic modeling, as I will review in this section.

As previously stated, MMs are less flexible than ML models due to *a priori* knowledge used to build them. This is true for GEMs, bringing several limitations, common to MMs as described in section 1.4.1.1.4. First, the diversity of datasets that can be directly integrated with GEMs through FBA is narrow: they must be either growth rates, or metabolic fluxes. Secondly, the FBA method to exploit GEMs finds single solutions for each condition, and does not generalize over sets of conditions. Finally, GEMs are criticized for the low reliability of the biomass reaction (BOF) and the optimality principle they rely on. Moreover, the advent of -omics data brings up the challenge of integrating heterogeneous data types with GEMs, which is not possible in a straightforward manner[45, 107, 108]. However, using purely ML models to replace MMs would suppress any mechanistic insights on the metabolic activity of the modeled organism; and it would bring all other ML limitations previously mentioned in section 1.4.1.2.4, such as the lack of predictive power outside the data used to train the model, or the need for larger datasets.

Due to such limitations, a lot of studies propose synergisms between ML models and GEMs, combining the knowledge contained in GEMs with the predictive power and flexibility of ML.

These combined approaches have been extensively listed and thoroughly described in many reviews[52, 107, 109–112]. A few examples of studies are worth mentioning here: Zhang *et al.* combined MMs to detect potential gene perturbation targets and ML to perform combinatorial optimization of such targets, in order to maximize tryptophan production[113]; Heckmann *et al.* extracted features

from CBM simulations and experimental data, to feed to an ML process that predicts kinetic parameters for a dynamic GEM parametrization[114]; and Wu *et al.* developed a ML workflow to predict  $^{13}\text{C}$  flux data from culture conditions, using CBM to discard biologically irrelevant predictions[115]. Another approach was termed ‘white-box machine learning’[116], which uncovered antibiotics mechanisms of actions on metabolism through the use of an ANN that was trained on GEM simulations, according to metabolite supplementations when bacteria were exposed to antibiotics. Therefore, their white-box ML system enabled the discovery of antibiotics effects on pathway mechanisms.

The reviews listing such CBM-ML synergic approaches focus on the exploitation of GEMs with ML, mainly for bioprocess optimization or discovery, but some studies show another kind of synergy. Indeed, ML can be also used as a way to curate and reconstruct GEMs, for example in yeast to decipher gene interactions[117]. More generally, as shown in Figure 1.13 taken from Rana *et al.*, ML can be combined with GEMs at many different steps of the modeling pipeline.

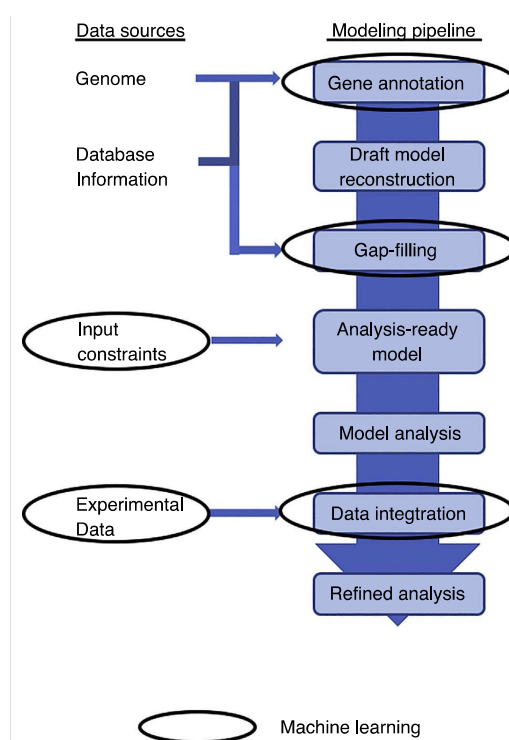


Figure 1.13: Overview of possible machine learning interventions on the constraint-based modeling pipeline, highlighting areas where machine learning has been applied (oval shapes). It additionally depicts four categories of data sources and seven broad steps in the CBM modeling pipeline. (Authors: Rana *et al.*[111]; license: CC BY 4.0)

#### 1.4.2.1.2 A gap between MM and ML in combined approaches

As reviewed, a lot of approaches attempt to mix ML and GEMs together to overcome the shortcomings of traditional CBM formulations of GEMs, that are purely MMs. These approaches are successful, but they still show a gap between the CBM and ML parts. In such combinations, ML and MM parts communicate through a segmented framework, which necessarily brings limitations. First, such segmented framework brings less reusability: each framework is specific to the task it has been designed for, and can hardly be picked up for another purpose. Moreover, the mathematical formulation in such a combined approach is not well defined, relying mostly on ‘*ad hoc*’ communications between the MM and ML parts. This research gap is clearly shown in Sahu *et al.*’s figure 2, Procopio *et al.*’s figure 4, Rana *et al.*’s figure 2 and Khalegi *et al.*’s figure 5. The figure from Rana *et al.* is displayed as Figure 1.14.



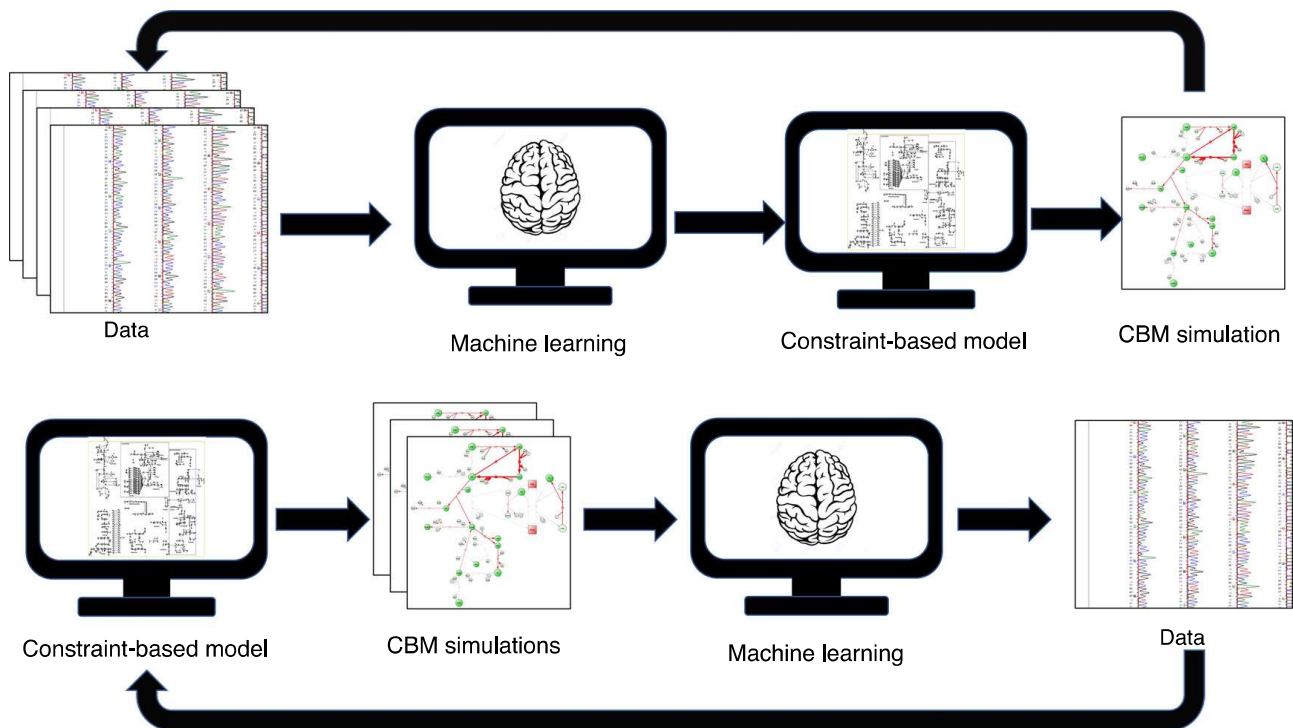


Figure 1.14: Integration example of experimental data with machine learning and constraint-based models (and simulation results) proposed by Rana *et al.* Top: Machine learning is applied to the input data to identify the important features for constructing reduced order constraint-based models; the CBM simulations can be iteratively matched with input data for convergence until the proper set of features are identified. Bottom: Machine learning is iteratively applied to CBM simulations to reconcile with experimental data. Interplay between the Top and Bottom parts can iteratively lead to convergence between CBM simulations, experimental data and machine learning based predictions. (Authors: Rana *et al.*[111]; license: CC BY 4.0)

An emerging field, namely hybrid modeling, aims to build models incorporating concepts and methods from both MMs and ML models. As I have reviewed so far, and as stated by Baker *et al.*[88], “the pros of one are the cons of the other, which suggests that research efforts should be directed towards enabling a symbiotic relationship between both”. In the next section 1.4.2.2, I will review how hybrid models attempt to achieve such complementarity, eventually leading to versatile and reusable models that incorporate ML and MM for metabolic modeling.

### 1.4.2.2 Hybrid models: towards a fusion of mechanistic and machine learning approaches

#### 1.4.2.2.1 Overview

Echoing the previously mentioned terminology of black- and white-box models, hybrid models are sometimes termed ‘gray-box’ models. They are based on the intuition that incorporating MM and ML together should bring more performance in terms of both predictive and explanatory power, respectively from ML statistical learning methods and MM mechanisms. As we have previously reviewed, pros and cons of each class of models seem complementary which justifies this approach. However, such marriage is not trivial and there are a number of ways to approach the hybrid modeling realm, which is a quite young and emerging field, especially for biology. Importantly, many combinations of CBM and ML including those mentioned above are termed ‘hybrid approach’ – this is not identical to hybrid *models*. Indeed, combining both MM and ML is a very widespread approach, but building

single model formulations incorporating MM and ML simultaneously, without previously described *ad hoc* combinations (section 1.4.2.1), is rather rare.

#### 1.4.2.2.2 Historical use of hybrid models

Historically, the first hybrid models to be widely used are found in the physics domain, notably to solve partial differential equations. The hybrid models, called Physics Informed Neural Networks (PINNs), are ANNs with specific loss functions that incorporate a term to fit experimental observations, a term for boundary constraints to be respected, and a term that computes the residual of the differential equations systems to be satisfied. In that sense, PINNs are designed to surrogate, approximately, the physical differential equations system through the learning of the neural networks parameters. After training, the PINNs are able to predict the differential equations system's output and parameters, for given inputs. Note that each term of the loss can have an associated weight which can be modulated to change how PINNs learn (e.g. with a low-confidence physical system, the loss weight applied on the term for residuals of the system might be reduced). Also, analyzing the resulting error from PINNs is of major importance, as it can indicate to what extent the PINNs predictions are approximate in regard to the physical system[118]. A major standardized computing environment is currently in active development, namely sciML[119], which implements ways to build and train PINNs and variations of such hybrid models, in the programming language Julia.

#### 1.4.2.2.3 Hybrid models for systems biology

In biology, the emergence of hybrid models is much more recent, and does not yet have standardized computing environments like SciML. However, the equivalent of PINNs for biology, Biologically-Informed Neural Networks (BINNs), were developed for reaction-diffusion models[120]. Moving away from the PINN inspiration, another way to build hybrid models is to assign to ML components some biological meaning. This was done with ANNs, assigning meaningful biological information to the neural architecture: Knowledge Primed Neural Networks (KPNNs) translate the known structure of protein signaling and gene regulation interaction into an ANN architecture, therefore increasing the interpretability of an ML model and leading to a new regulatory MM formulation from single-cell data[121]. A similar approach was performed with RNNs for signaling networks (with a specific activation function inspired from Michaelis-Menten), enabling the training of an hybrid model of signaling with single-cell data[122]. Finally, a third approach translated a graph of metabolic module interactions in a Graph Neural Network (GNN), to estimate single-cell metabolic fluxes[60]. Importantly, the four approaches cited are different from the CBM-ML synergies previously mentioned, in the sense they formulate single-block models incorporating MM and ML together, without using an *ad hoc* framework making the two parts communicate.

These approaches are all based on the use of ANNs as the ML model basis of the hybrid model. This choice can be understood because they have proved to be highly performant in a variety of tasks, show compelling achievements such as AlphaFold, and most importantly, they have a very high flexibility of architectures and loss formulations. Therefore, they seem like the best choice to easily couple MMs with ML.

#### 1.4.2.2.4 Hybrid model of GEMs: a research gap

The motivation for the original research work that will follow in chapters 2 and 3 is to fill the following research gap: GEMs lack hybrid model formulations, whereas a lot of approaches combine ML and GEMs in various ways. To my knowledge, there is only one hybrid model formulation that is filling the

same gap as this Ph.D. dissertation's work, described by Freischem *et al.* and Hasibi *et al.* [123, 124]. They built a mass flow graph from a GEM, then used it as a scaffold to learn with. However, such formulation is only used for gene essentiality prediction for now.

Importantly, our formulation has different purposes. In particular, some specific issues and limitations of GEMs used with FBA alone are tackled by the hybrid modeling of GEMs we propose. GEMs propose very large solution spaces: the number of possible metabolic phenotypes (flux distributions) is tremendous. In other words GEMs without additional constraints are under-determined. Therefore, measuring fluxes, such as the very critical uptake fluxes, is necessary for making meaningful predictions with GEMs. This is a limitation for experimentalists, for example when working in metabolic engineering projects, since flux measures are both time and resource expensive. More generally, this issue of GEMs imposes strong limitations to understand the cell's behavior. As stated in section 1.3.3.3, GEMs can provide insights on the metabolic activity of an organism when uptake fluxes are measured; but they cannot be reliably used to predict the organism's behavior to a new environment if uptake fluxes are not measured. Overall, we assessed that GEMs contain the reliable topology of metabolic networks, but the usual FBA methodology to exploit them is rather simplistic and not able to generalize over sets of conditions, which motivates our hybrid approach. In particular, the approach aims to predict metabolic phenotypes proposed by a GEM from different conditions, by ML methods. In that sense, regulations are modeled through ML, and metabolism is modeled through the GEM. Obviously, this is a strong oversimplification of the 'real-world' processes, where regulation and metabolism are not separated in two layers.

In more technical terms, we developed surrogates of FBA that can be integrated in learning procedures, with ANNs. Several surrogate methods were developed, with inspiration from PINNs (AMN-QP), the hybrid RNN from Nilsson *et al.* (AMN-Wt) and Hopfield networks [104] (AMN-LP). They all behave like 'interior-point' (see section 1.3.3.3.2) LP solvers using "neural" computations (*i.e.*, iterative vector and matrix multiplications). Moreover, we also use ML to surrogate MMs as described in recent reviews [67, 125], when using the 'AMN-Reservoir' approach in Chapter 2 section 2.3.5. Therefore, the hybrid models of GEMs that we developed exploit ANNs as the ML basis, to guide CBM formulations exploiting GEMs. The details of such formulation are presented in the next chapter. Figure 1.15 displays a simple schematic of the hybrid modeling of GEMs approach that I present in this dissertation.

In the present section 1.4 of the introduction, we have reviewed ML and MM capabilities, with an emphasis on their complementarity and the need to combine the two in systems biology. Hybrid modeling is an emerging field that aims to integrate both ML and MM methodologies in single formulations, instead of combining them separately. Developing hybrid models requires many methodological choices to be made, putting more or less emphasis on the ML or MM part, and focusing on different MM modeling pipeline steps. However, achieving the development of such hybrid models seems very promising for the systems biology endeavor of understanding life through integrative models, in its ability to encompass many mechanisms through MM and integrating diverse datasets with good predictive power (notably from -omics technologies) through ML. Eventually, we may formulate a whole-cell performant hybrid model, a step towards the holy grail of systems biology. Of course, this Ph.D. work does not have this ambition, but it presents original research that may inspire others towards that goal. The following and final section 1.5 will quickly summarize the content of this first chapter, and present the plan and content of the dissertation remainder.

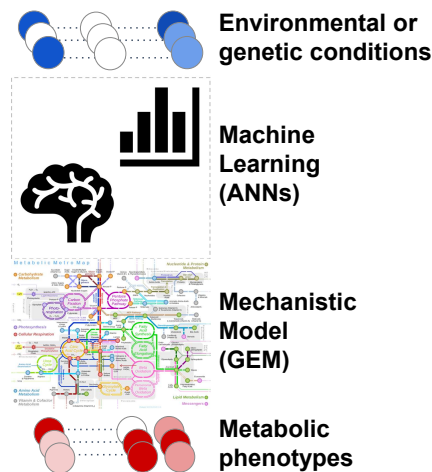


Figure 1.15: Simple schematic of the selected hybrid modeling approach for GEMs. From any kind of input data describing environmental or genetic conditions, a machine learning model (ANNs), predicts parameters of a mechanistic model (GEM) in order to predict corresponding metabolic phenotypes. Note that this model formulation is a single model, it does not separate the ML and MM parts and can therefore be considered a hybrid model. Details on the construction, training, and exploitation of the model are given in Chapters 2 and 3.

## 1.5 Research questions and content of the dissertation

In this introduction, I have reviewed the key knowledge and methodologies to understand and criticize the following chapters that will present original research works. In short, I have reviewed the discovery path from the first enzyme to genome-scale metabolic networks (in the appendix section A.1). Then, I described a variety of regulation mechanisms underlying the relationship between a cell's environment and its metabolic phenotype, and some approaches to integrate them in a holistic way. Moving on, I showed how modern research on metabolism aims to grasp complexity for controlling metabolic phenotypes and eventually computing with metabolism. Next, I focused on GEMs: their content, reconstruction process, various formulations, mathematical framework, capabilities and limits were described. After that, I compared MM and ML models that are complementary approaches, notably for metabolic modeling where numerous combinations of the two approaches have been performed. Finally, I introduced the field of hybrid modeling that aims to incorporate MM and ML methodologies in single model formulations, instead of combining the two separately.

In this Ph.D. dissertation, we will attempt to fill the previously mentioned research gap of hybrid models formulations for GEMs, even though a plethora of combined CBM-ML approaches exist. Therefore, the overall goal of this Ph.D. is to attempt such hybrid model development, and assess its capabilities and limits. In the remainder of the dissertation, the term 'Artificial Metabolic Networks' (AMNs) will be used to designate the hybrid models hereby developed. The following chapters will attempt to answer the following questions:

1. Can we develop AMNs that increase the GEMs predictive power while respecting their constraints?
2. What are the future improvements and most suited areas of application for AMNs?

In **Chapter 2**, I will start answering the first question by presenting a first-authored research article published in a peer-reviewed journal, entitled 'A neural-mechanistic hybrid approach improving the predictive power of genome-scale metabolic models'. This presents the basis of the modeling

concepts and mathematical frameworks that were developed for AMNs, and benchmarks their capabilities to surpass the predictive power of GEMs.

In **Chapter 3**, I will complete the answer to the first question by presenting further assessments of the innovative modeling concepts that were formulated in Chapter 2. This will show what challenges AMNs are facing and how to tackle them. Thereby, I will present a novel, more general formulation of an AMN core principle, as well as an enhanced way to fine-tune AMNs for better performance through hyperparameter optimization, and some important assessments on the capabilities of AMNs.

In **Chapter 4**, I will answer the second question with a general summary of previous chapters' results and a discussion of possible perspectives to enhance AMNs, as well as potential areas of applications. Here, I will also conclude my dissertation by an overall recap of the main novelties brought by this work and the potential it brings for future research on metabolic modeling.

Additionally, in the **appendices**, I include Supplementary Information sections for the current chapter as appendix A, for the published article presented in chapter 2 as appendix B, and for chapter 3 as appendix C. I also include 3 co-authored publications that I participated in: (i) a review published in *Current Opinion in Chemical Biology* entitled '*In silico*, *in vitro*, and *in vivo* machine learning in synthetic biology and metabolic engineering' in appendix D; (ii) a book chapter published in *Methods in Molecular Biology* entitled 'Cell-Free Biosensors and AI integration' in appendix E; and (iii) a research article published in *Nature Communications* entitled 'A versatile active learning workflow for optimization of genetic and metabolic networks' in appendix F.

## Chapter 2

# A hybrid neural-mechanistic approach improving the predictive power of genome-scale metabolic models

This Chapter has the same content as the article published in Nature Communications. The online published version can be found with the following [DOI \(clickable link\)](#).

**Authors:** Léon Faure<sup>1</sup>, Bastien Mollet<sup>2,3</sup>, Wolfram Liebermeister<sup>4</sup>, and Jean-Loup Faulon<sup>1,5,\*</sup>

**Affiliations:** <sup>1</sup>MICALIS Institute, INRAe, AgroParisTech, University of Paris-Saclay, Jouy-en-Josas, France, <sup>2</sup>Ecole Normale Supérieure, ENS-Lyon, Lyon, France, <sup>3</sup>UMR MIA, INRAe, AgroParisTech, Université Paris-Saclay, Palaiseau, France, <sup>4</sup>MaIAGE, INRAe, University of Paris-Saclay, Jouy-en-Josas, France, <sup>5</sup>Manchester Institute of Biotechnology, University of Manchester, Manchester, UK.

\*Corresponding author: [jean-loup.faulon@inrae.fr](mailto:jean-loup.faulon@inrae.fr), ORCID: [0000-0003-4274-2953](https://orcid.org/0000-0003-4274-2953)

**Keywords:** Artificial Neural Network, Metabolic Network, Mechanistic Modeling, Flux Balance Analysis, Scientific Machine Learning, Hybrid Modeling.

**Abbreviations:** AMN: Artificial Metabolic Network, ANN: Artificial Neural Network, CBM: Constraint-Based Modelling, FBA: Flux Balance Analysis, GEMs: GEnome-scale Metabolic models, KO: knock-out, LP(QP): Linear (Quadratic) Programming, ML: Machine Learning, MM: Mechanistic Modelling, MSE: Mean Squared Error, PINN: Physics Informed Neural Network, RNN: Recurrent Neural Network,  $R^2$ : Regression coefficient calculated on training set,  $Q^2$ : Regression coefficient calculated on cross-validation sets or independent test sets (not seen during training), ROC: Receiving Operating Characteristic, AUC: Area Under the (ROC) Curve.

### 2.1 Abstract

Constraint-based metabolic models have been used for decades to predict the phenotype of microorganisms in different environments. However, quantitative predictions are limited unless labor-intensive measurements of media uptake fluxes are performed. We show how hybrid neural-mechanistic models can serve as an architecture for machine learning providing a way to improve phenotype predictions. We illustrate our hybrid models with growth rate predictions of *Escherichia coli* and *Pseudomonas putida* grown in different media and with phenotype predictions of gene knocked-out *Escherichia coli* mutants. Our neural-mechanistic models systematically outperform constraint-based models and require training set sizes orders of magnitude smaller than classical machine learning

methods. Our hybrid approach opens a doorway to enhancing constraint-based modeling: instead of constraining mechanistic models with additional experimental measurements, our hybrid models grasp the power of machine learning while fulfilling mechanistic constraints, thus saving time and resources in typical systems biology or biological engineering projects.

## 2.2 Introduction

In this study, we present an approach that combines Machine Learning (ML) and Mechanistic Modeling (MM) to improve the performance of constraint-based modeling (CBM) on GEnome-scale Metabolic models (GEMs). Our hybrid MM-ML models are applied to common tasks in systems biology and metabolic engineering, such as predicting qualitative and quantitative phenotypes of organisms grown in various media or subjected to gene knock-outs (KOs). Our approach leverages recent advances in ML, MM, and their integration, which we briefly review next.

The increasing amounts of data available for biological research bring the challenge of data integration with ML to accelerate the discovery process. The most compelling achievement within this grand challenge is protein folding, recently cracked by AlphaFold[96], which in the last CASP14 competition predicted structures with a precision similar to structures determined experimentally. Following this foot step, one may wonder if in the future we will be able to use ML to accurately model whole cell behaviors. The curse of dimensionality[99], *i.e.* the fact that fitting many parameters may require prohibitively large data sets, is perhaps the biggest hurdle that prevents using ML to build cell models. Obviously, cells are far more complex than single proteins and since the amount of data needed for ML training grows exponentially with the dimensionality[99], as of today, ML methods have not been used alone to model cellular dynamics at a genome scale.

For the past decades, MM methods have been developed to simulate whole-cell dynamics (*cf.* Thornburg *et al.*[126] for one of the latest models). These models encompass metabolism, signal transduction, as well as gene and RNA regulation and expression. Cellular dynamics being tremendously complex, MM methods are generally based on strong assumptions and oversimplifications. Ultimately, they suffer from a lack of capacities to make predictions beyond the assumptions and the data used to build them.

Flux Balance Analysis (FBA) is the main MM approach to study the relationship between nutrient uptake and the metabolic phenotype (*i.e.*, the metabolic fluxes distribution) of a given organism, *i.e.*, *E. coli*, with a model iteratively refined over the past 30 years or so[127]. FBA searches for a metabolic phenotype at steady state, *i.e.*, a phenotype that is constant in time and in which all compounds are mass-balanced. Usually, such a steady state is assumed to be reached in the mid-exponential growth phase. The search for a steady state happens in the space of possible solutions that satisfies the constraints of the metabolic model, *i.e.*, the mass-balance constraints according to the stoichiometric matrix as well as upper and lower bounds for each flux in the distribution. The steady state search is performed with an optimality principle, with one principal objective (usually the 'biomass' production flux) and possibly secondary objectives (*i.e.*, minimize the sum of fluxes in parsimonious FBA, or the flux of a metabolite of interest). As we shall see later and as discussed in O'Brien *et al.*[55], FBA suffers from making accurate quantitative phenotype predictions.

The MM and ML approaches are based on two seemingly opposed paradigms. While the former is aimed at understanding biological phenomena with physical and biochemical details, it has difficulties handling complex systems; the latter can accurately predict the outcomes of complex biological processes even without an understanding of the underlying mechanisms, but require large training



sets. The pros of one are the cons of the other, suggesting the approaches should be coupled. In particular, MMs may be used to tackle the dimensionality curse of ML methods. For instance, one can use MMs to extend experimental datasets with *in silico* data, increasing the training set sizes for ML. However, with that strategy, if the model is inaccurate, ML will be trained on erroneous data. One can also embed MMs within the ML process, in this strategy, named hybrid-modelling, ML and MM are trained together and the model parameters can be estimated through training, increasing the model predictive capacities. To improve FBA phenotype predictions, ML approaches have been used to couple experimental data with FBA. Among published approaches, one can cite Plaimas *et al.*[128] where ML is used after FBA as a post-process to classify enzyme essentiality. Similarly, Schinn *et al.*[129] used ML as a post-process to predict amino-acid concentrations. Freischem *et al.*[124] computed a mass flow graph running FBA on the *E. coli* model iML1515[44] and used it with a training set of measured growth rates on *E. coli* gene KO mutants. Several ML methods were then utilized in a post-process to classify genes as essential vs. non-essential. As reviewed by Sahu *et al.*[52], ML has also been used to preprocess data and extract features prior to running FBA. For instance, data obtained from several omics methods can be fed to FBA, after processing multi-omics data via ML[107, 130, 131].

In all these previous studies, and as discussed in Sahu *et al.*[52], the interplay between FBA and ML still shows a gap: some approaches use ML results as input for FBA, others use FBA results as input for ML, but none of them embed FBA into ML, as we do in this study with the Artificial Metabolic Network (AMN) hybrid models.

The main issue with hybrid modeling is the difficulty of making MM amenable to training. Overcoming this difficulty, solutions have recently been proposed under different names in biology for signaling pathways and gene-regulatory networks (Knowledge Primed Neural Network[121], Biologically-Informed Neural Networks[120]) with recent solutions based on recurrent neural networks (RNNs)[122]. Hybrid models have also been developed in physics to solve partial differential equations, such as Physics Informed Neural Network[132] (PINN), available in open-source repositories like SciML.ai[119]. The goal of these emerging hybrid modeling solutions is to generate models that comply well with observations or experimental results via ML, but that also use mechanistic insights from MM. The advantages of hybrid models are two-fold: they can be used to parametrize MM methods through direct training and therefore increasing MM predictability, and they enable ML methods to overcome the dimensionality curse by being trained on smaller datasets because of the constraints brought by MM.

In the current paper we propose a MM-ML hybrid approach in which FBA is embedded within Artificial Neural Networks (ANNs). Our approach bridges the gap between ML and FBA by computing steady-state metabolic phenotypes with different methods that can be embedded with ML. All these methods rely on custom loss functions surrogating the FBA constraints. By doing so, our AMNs are mechanistic models, determined by the stoichiometry and other FBA constraints, and also ML models, as they are used as a learning architecture.

We showcase our AMNs with a critical limitation of classical FBA that impede quantitative phenotype predictions, the conversion of medium composition to medium uptake fluxes[55]. Indeed, realistic and condition-dependent bounds on medium uptake fluxes are critical for growth rate and other fluxes computations, but there is no simple conversion from extracellular concentrations, *i.e.*, the controlled experimental setting, to such bounds on uptake fluxes. With AMNs, a neural pre-processing layer aims to capture, effectively, all effects of transporter kinetics and resource allocation in a particular experimental setting, predicting the adequate input for a metabolic model to give the most accurate steady-state phenotype prediction possible. Consequently, AMNs provide a new paradigm



for phenotype prediction: instead of relying on a constrained optimization principle performed for each condition (as in classical FBA), we use a learning procedure on a set of example flux distributions that attempts to generalize the best model for accurately predicting the metabolic phenotype of an organism in different conditions. As shown in the results section, the AMN pre-processing layer can also capture metabolic enzyme regulation and in particular predict the effect of gene KOs on phenotype.

## 2.3 Results

### 2.3.1 Overview of AMN hybrid models

When making predictions using FBA, one typically sets bounds for medium uptake fluxes,  $\mathbf{V}_{in}$ , to simulate environmental conditions for the GEM of an organism (Figure 2.1, panel a). Each condition is then solved independently from each other by a Linear Program (LP), usually making use of a Simplex solver. In most cases, one sets the LP's objective to maximize the biomass production rate (*i.e.*, the growth rate), under the metabolic model constraints (*i.e.*, flux boundary and stoichiometric constraints). FBA computes the resulting steady-state fluxes,  $\mathbf{V}_{out}$ , for all the reactions of the metabolic network, which we use later in our reference 'FBA-simulated data', for the benchmarking of the hybrid models developed in this study. While FBA is computationally efficient and easy to use through libraries like Cobrapy[68], FBA cannot directly be embedded within ML methods, like neural networks, because gradients cannot be backpropagated through the Simplex solver.

To enable the development of hybrid models and gradient backpropagation, we developed three alternative MM methods (Wt-solver, LP-solver and QP-solver) that replace the Simplex solver while producing the same results (Figure 2.1, panel b). The three solvers, further described in the next subsection, take as input any initial flux vector that respect flux boundary constraints.

We next used the MM models as a component of AMN hybrid models that can directly learn from sets of flux distributions (Figure 2.1, panel c). These flux distributions used as learning references (*i.e.*, training sets) are either produced through FBA simulations or acquired experimentally. The AMN model comprises a trainable neural layer followed by a mechanistic layer (composed of Wt-solver, LP-solver or QP-solver). The purpose of the neural layer is to compute an initial value,  $\mathbf{V}^0$ , for the flux distribution to limit the number of iterations of the mechanistic layer. The initial flux distribution is computed from medium uptake flux bounds,  $\mathbf{V}_{in}$ , when the training set has been generated through FBA simulations, or medium compositions,  $\mathbf{C}_{med}$ , for experimental training sets. For all AMNs, the training of the neural layer is based on the error computation between the predicted fluxes,  $\mathbf{V}_{out}$ , and the reference fluxes, as well as on the respect of mechanistic constraints. It is important to point out that AMNs attempt to learn a relationship between  $\mathbf{V}_{in}$  (or  $\mathbf{C}_{med}$ ) and the steady-state metabolic phenotype, generalizing this relationship for a set of conditions and not just only one as in FBA. In the upcoming subsections, Figure 2.2 presents results for FBA-simulated training sets and Figures 2.3 and 2.4 results for experimental training sets.

Finally, we developed a non-trainable AMN-Reservoir to showcase how the predictive power of classical FBA can be improved (Figure 2.1, panel d). This architecture is based on a 2-step learning process with the specific goal of finding the best bounds on uptake fluxes for FBA, by extracting  $\mathbf{V}_{in}$  after training. Indeed, once the AMN has been trained on adequate FBA-simulated data, we can fix its parameters, resulting in a gradient backpropagation compatible reservoir that mimics FBA. The AMN reservoir can then be used to tackle the above-mentioned issue of unknown uptake fluxes: adding

a pre-processing neural layer and training this layer with an experimental dataset, one can predict uptake fluxes from the media composition. Results of the pre-processing neural layer can directly be plugged into a classical FBA solver and the neural layer can be reused by any FBA user to improve the predictive power of metabolic models with an adequate experimental set-up. We showcase AMN-Reservoir results in Figure 2.5 using experimental measurements acquired on *E. coli* and *P. putida*.

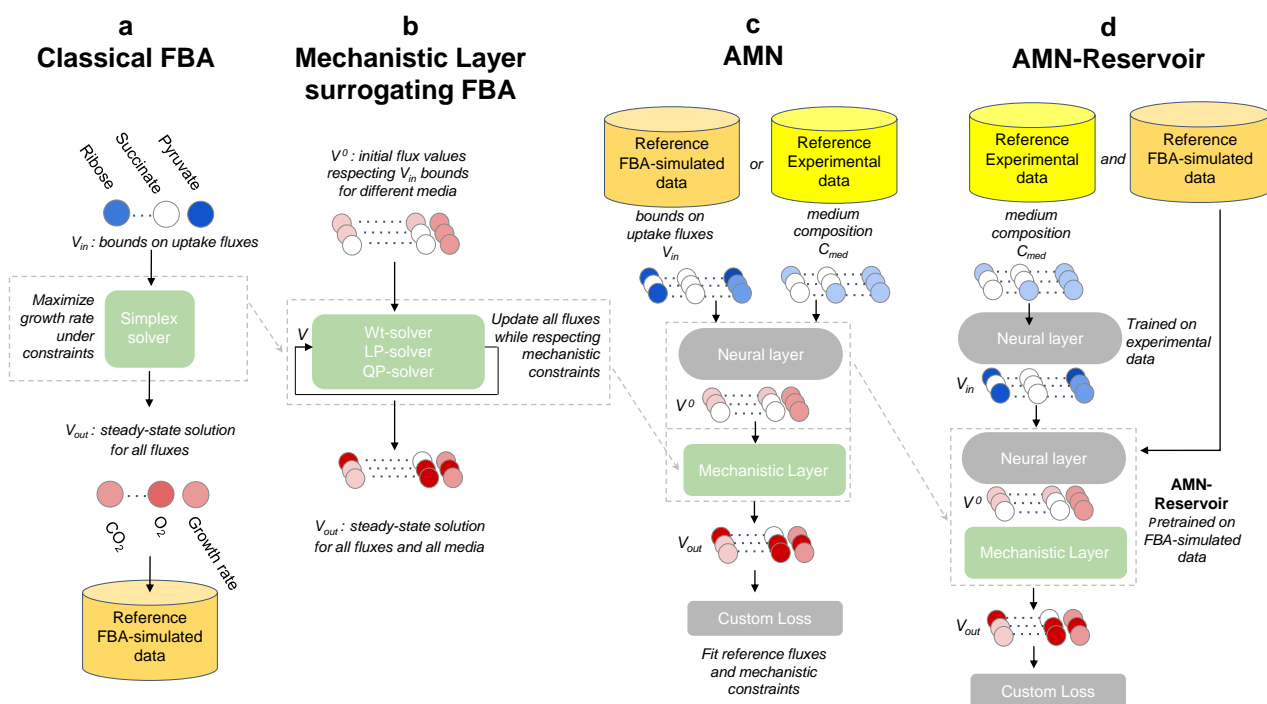


Figure 2.1: Computing and learning frameworks for FBA, alternative Mechanistic Models, AMN, and AMN-Reservoir. **a.** Computing framework for classical FBA. The process is repeated for each medium, computing the corresponding steady state fluxes. Blue circles represent different bounds on metabolites uptake fluxes and each red circle represents a flux value at steady-state. **b.** Computing framework for MM methods surrogating FBA. The methods can handle multiple growth media at once. Disregarding the solver (Wt, LP and QP), the MM layer takes as input an arbitrary initial flux vector,  $\mathbf{V}^0$ , respecting uptake flux bounds for different media, and computes all steady-state fluxes values ( $\mathbf{V}_{out}$ ) through an iterative process. **c.** Learning framework for AMN hybrid models. The input (for multiple growth media) can be either a set of bounds on uptake fluxes ( $\mathbf{V}_{in}$ ), when using simulation data (generated as in panel a), or a set of media compositions,  $\mathbf{C}_{med}$ , when using experimental data. The input is then passed to a trainable neural layer, predicting an initial vector,  $\mathbf{V}^0$ , for the mechanistic layer (a MM method of panel b). In turn, the mechanistic layer computes the final output of the model,  $\mathbf{V}_{out}$ . The training is based on a custom loss function (cf. Methods, 2.5) ensuring the reference fluxes are fitted (i.e.,  $\mathbf{V}_{out}$  matches simulated or measured fluxes) and that the mechanistic constraints (on flux bounds and stoichiometry) are respected. **d.** Learning framework for an ‘AMN-Reservoir’. The first step is to train an AMN on FBA-simulated data (as in panel c), after which parameters of this AMN are frozen. This AMN model, which purpose is to surrogate FBA, is named non-trainable ‘AMN-Reservoir’. In the second step, a neural layer is added prior to  $\mathbf{V}_{in}$  taking as input media compositions,  $\mathbf{C}_{med}$ , and learning the relationship between the compositions and bounds on uptake fluxes.

### 2.3.2 Alternative mechanistic models to surrogate FBA

Let us first recall that the methods described in this subsection are mechanistic models (MMs) that *replace* the Simplex-solver used in FBA and allow for gradient backpropagation, but without any learning procedure performed. As far as medium uptake fluxes are concerned, we consider in the following two cases: (i) when exact bound values (EB) for medium uptake fluxes are provided, and (ii) when only

upper bound values (UB) for medium uptake fluxes are given.

Our first method (Wt-solver), inspired by previous work on signaling networks[122], recursively computes  $\mathbf{M}$ , the vector of metabolite production fluxes, and  $\mathbf{V}$ , the vector of all reaction fluxes (cf. ‘Wt-solver’ in Methods 2.5 and ‘Wt-solver equations’ in appendix B, for further details). The vectors  $\mathbf{M}$  and  $\mathbf{V}$  are iteratively updated using matrices derived from the metabolic network stoichiometric matrix  $\mathbf{S}$  and from a weight matrix,  $\mathbf{W}_r$ , representing consensual flux branching ratios found in example flux distributions (i.e., reference FBA-simulated data or experimental measurements). Since the mass conservation law is the central rule when satisfying metabolic networks constraints, these ratios play a key role in the determination of the metabolic phenotype, i.e. the paths taken by metabolites in the organism. In this approach, we assume that the flux branching ratios remain similar between flux distributions with different bounds on different uptake fluxes. A simple toy model network is shown to demonstrate the functioning of the Wt-solver in Supplementary Figure S1 (appendix B).

While the Wt-solver is simple to implement it suffers from a drawback. As discussed in the Supplementary Information ‘AMN-Wt architecture’ (appendix B), a consensus set of weights leads to a solution when upper bounds (UB) for uptake fluxes are provided, but not when exact bounds (EB) for uptake fluxes are given (cf. Supplementary Figure S2; appendix B). Consequently, we cannot assume that the Wt-solver can handle all possible flux distributions in the EB case. To overcome this shortcoming, we next present two alternative methods that are much closer to the optimizations behind FBA and that can accommodate both EB and UB cases for uptake fluxes. The two methods address two distinct tasks in flux modeling: optimizing a flux distribution for maximal growth rate (LP-solver), as in classical FBA, and fitting a stationary flux distribution to partial flux data (QP-solver).

The second method (LP-solver), derived from a method proposed by Yang *et al.*[133], handles linear problems using exact constraint bounds (EBs) or upper bounds (UBs) for uptake fluxes ( $\mathbf{V}_{in}$ ). That method makes use of Hopfield-like networks, which is a long-standing field of research[103] inspired by the pioneering work of Hopfield and Tank[104]. As with the Wt-solver, the LP-solver iteratively computes fluxes to come closer to the steady-state solution ( $\mathbf{V}_{out}$ ). However, calculations are more sophisticated, and the method integrates the same objective function (i.e., maximize growth rate) as the classical FBA Simplex solver. The solver iteratively updates the flux vector,  $\mathbf{V}$ , and the vector,  $\mathbf{U}$ , representing the dual problem variables also named metabolites shadow prices[134] (cf. ‘LP-solver’ in Methods 2.5 and in appendix B for further details).

The third approach (QP-solver), is loosely inspired by the work on Physics-Informed Neural Networks (PINNs), which has been developed to solve partial differential equations matching a small set of observations[118]. With PINNs, solutions are first approximated with a neural network and then refined to fulfill the constraints imposed by the differential equations and the boundary conditions. Refining the solutions necessitates the computation of three loss functions. The first is related to the observed data, the second to the boundary conditions and the third to the differential equations. As detailed in Methods 2.5, we similarly compute losses for simulated or measured reference fluxes,  $\mathbf{V}_{ref}$ , the flux boundary constraints, and the metabolic network stoichiometry. As in PINN we next compute the gradient on these losses to refine the solution vector  $\mathbf{V}$ . Unlike with the LP-solver, we do not provide an objective to maximize in the present case, but instead reference fluxes, consequently the method is named QP because it is equivalent to solving an FBA problem with a quadratic program.

To assess the validity of the LP and QP solver, we used the *E. coli* core model[135] taken from the BiGG database[136]. To generate with Cobrapy package[68] a training set of 100 growth rates varying 20 uptake fluxes, following the procedure given in Methods 2.5. Results can be found in Supplementary Figure S6 (appendix B), showing excellent performances after 10,000 iteration steps.

### 2.3.3 AMNs: metabolic & neural hybrid models for predictive power with mechanistic insights

While the above solvers perform well, their main weakness is the number of iterations needed to reach satisfactory performances. Since our goal is to integrate such methods in a learning architecture, this drawback has to be tackled. As illustrated in Figure 2.1 panel c, our solution is to improve our initial guesses for fluxes, by training a prior neural layer (a classical dense ANN) to compute initial values for all fluxes ( $\mathbf{V}^0$ ) from bounds on uptake fluxes ( $\mathbf{V}_{in}$ ) or media compositions ( $\mathbf{C}_{med}$ ). This solution enables the training of all AMNs with few iterations in the mechanistic layer. In the remainder of the paper, we name AMN-Wt, AMN-LP and AMN-QP, the hybrid model shown in Figure 2.1 panel c, composed of a neural layer followed by a mechanistic layer, *i.e.*, a Wt, LP or QP solver.

The performances of all AMN architectures (Wt, LP, QP) and a classical ANN architecture (*cf.* Methods 2.5 ‘ANN architecture’, for further details) are given in Figure 2.2, using FBA-simulated data on two different *E. coli* metabolic models, *E. coli* core[135] and iML1515[44]. These models are composed respectively of 154 reactions and 72 metabolites, and 3682 reactions and 1877 metabolites (after duplicating bidirectional reactions). In all cases, the training sets were generated by running the default Simplex-based solver (GLPK) of Cobrapy[68] to optimize 1000 growth rates for as many different media. Each medium was composed of metabolites found in minimal media (M9) and different sets of additional metabolites (sugars, acids) taken up by the cell (more details in Methods 2.5 ‘Generation of training sets with FBA’). These training sets have as variables a vector of bounds on uptake fluxes (20 for *E. coli* core, 38 for iML1515) along with the Cobrapy[68] computed growth rate. For the ANN training set, to enable loss computation on constraints, we replaced the growth rate by the whole flux distribution computed by Cobrapy[68] (*cf.* Methods ‘ANN architecture’).

Figure 2.2 shows the loss values on mechanistic constraints and the regression coefficient ( $Q^2$ ) for the growth rates of the aforementioned models. All results shown are computed on 5-fold cross-validation sets. Additional information on hyperparameters and results on independent test sets are found in Supplementary Information (appendix B). In particular, Supplementary Figure S7 shows performances obtained with AMN-QP and the *E. coli* core model with different neural layer architectures and hyperparameters, justifying our choices for the neural layers of AMNs (one hidden layer of dimension 500 and a training rate of  $1e-3$ ). Similar results were found for AMN-LP and AMN-Wt. Additionally, in Supplementary Table S1 (appendix B), more extensive benchmarking is provided comparing MMs, ANNs and AMNs. This table shows performances for training, validation, and independent test sets of more diverse datasets, along with all training sets parameters and the models’ hyperparameters.

All AMN architectures exhibit excellent regression coefficients and losses after a few learning epochs, and this for both models *E. coli* core[135] and iML1515[44]. It is interesting to observe the good performances of AMN-Wt when UB training sets are provided. Indeed, while counterexamples can be found for which AMN-Wt will not work with EB training sets (*cf.* Supplementary Figure S2, appendix B), we argue in Supplementary Information ‘AMN-Wt architecture’ (appendix B) that AMN-Wt is able to handle UB training sets because the initial inputs (UB values for uptake fluxes) are transformed into suitable exact bound values during training (via the neural layer).

We recall that in Figure 2.2, ANNs were trained on all fluxes to enable loss computation (154 fluxes for *E. coli* core and 550 fluxes for iML1515), thus the number of training data points is substantially larger than for AMNs (154 000 or 550 000 for ANNs, instead of 1000 for AMNs). Despite requiring larger training sets, ANNs also need more learning epochs than AMNs to reach satisfying constraint losses and growth rate predictions for *E. coli* core (Figure 2.2, panels a and b) and do not handle well the

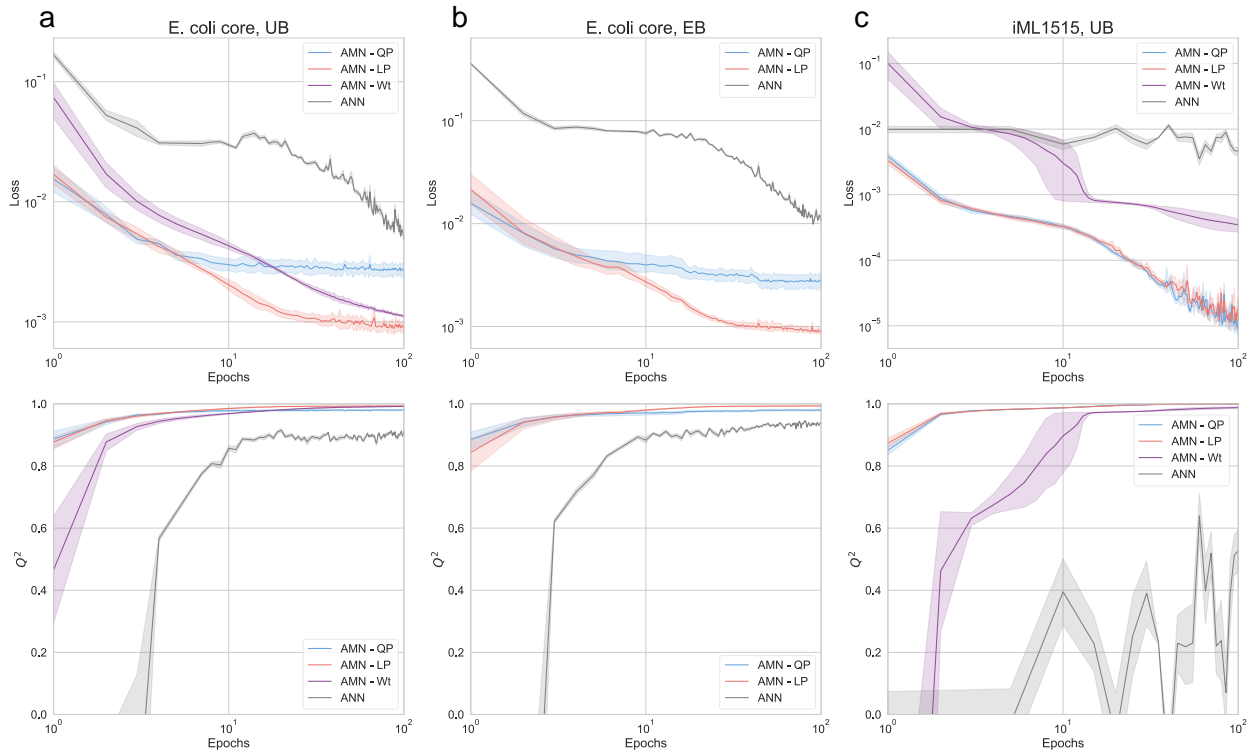


Figure 2.2: Benchmarking AMNs with different training sets and mechanistic layers. All results were computed on 5-fold cross-validation sets. Plotted is the mean and standard error (95% confidence interval) over the 5 validation sets of the cross-validation. Top panels show the custom mechanistic loss values, and bottom panels plot the  $Q^2$  values for the growth rate, over learning epochs ( $Q^2$  is the regression coefficient on cross-validation datapoints not seen during training). All AMNs have the architecture given in Figure 2.1 panel c, with  $\mathbf{V}_{in}$  as input, and a neural layer composed of one hidden layer of size 500. For all models, dropout = 0.25, batch size = 5, the optimizer is ‘Adam’, the learning rate is  $10^{-3}$ . The architecture for ANN (a classical dense network) is given in the Methods 2.5, it takes as input the uptake fluxes bounds  $\mathbf{V}_{in}$  and produce a vector  $\mathbf{V}_{out}$  composed of all fluxes with which the loss is computed. The panels a, b and c show results for different training sets: **a** and **b** for 1000 simulations training sets generated with the *E. coli* core model, respectively with UB and EB as inputs, whereas **c** is for a 1000 simulations training set generated with the iML1515 model, with UB as input (for more details on the training set generations, refer to Methods). As mentioned in subsection ‘Alternative mechanistic models to surrogate FBA’, AMN-Wt cannot be used to make predictions when exact bounds (EB) are used and is therefore not plotted in panel b of Figure 2.2. Source data are provided as a Source Data file (cf. Data availability, 2.7).

large iML1515 GEM (*i.e.*, the growth rate cannot be accurately predicted and the oscillatory behavior in Figure 2.2 panel c demonstrates 100 epochs are not enough to reach convergence).

### 2.3.4 AMNs can be trained on experimental datasets with good predictive power

To train AMNs on an experimental dataset, we grew *E. coli* DH5- $\alpha$  in 110 different media compositions, with M9 supplemented with 4 amino acids as a basis and 10 different carbon sources as possibly added nutrients. From 1 up to 4 carbon sources were simultaneously added to the medium at a concentration of  $0.4 \text{ g.L}^{-1}$  (more details in Methods 2.5 ‘Culture conditions’). We determined which compositions to test by choosing all the 1-carbon source media compositions and randomly picking one hundred of the 2-, 3- and 4-carbon sources media compositions (more details in Methods 2.5 ‘Generation of an experimental training set’). The growth of *E. coli* was monitored in 96-well plates,



by measuring the optical density at 600nm ( $OD_{600}$ ) over 24 hours. The raw  $OD_{600}$  was then passed to a maximal growth rate determination method based on a linear regression performed on  $\log(OD_{600})$  data (more details in Methods ‘Growth rate determination’).

The resulting experimental dataset of media compositions,  $\mathbf{C}_{med}$ , and growth rates,  $\mathbf{V}_{ref}$ , was used to train all AMN architectures (LP, QP, Wt). These architectures are those shown in Figure 2.1 panel c with  $\mathbf{C}_{med}$  as input. In all cases the mechanistic layer was derived from the stoichiometric matrix of the iML1515<sup>20</sup> *E. coli* reduced model (cf. Methods 2.5 ‘Making metabolic networks suitable for neural computations’). Following Figure 2.1 panel c,  $\mathbf{C}_{med}$  was entered as a binary vector (presence/absence of specific metabolites in the medium), the vector was then transformed through the neural layer into an initial vector,  $\mathbf{V}^0$ , for all reaction fluxes (therefore including the medium uptake fluxes) prior to be used in the mechanistic layer and the loss computations. Prediction performances are provided in Figure 2.3, alongside schematics for each of the architectures.

For displaying meaningful results and to avoid any overfitting bias, we show in Figure 2.3 predictions for points unseen during training. More precisely, we computed the mean and standard deviation of predictions over 3 repeats of stratified 10-fold cross-validations, each repeat having all points predicted, by aggregating validation sets predictions of each fold. Overall, results presented in Figure 2.3 have been compiled over  $3 \times 10 = 30$  different AMN models, each having different random seeds for the neural layer initialization and the train/validation splits.

As a matter of comparison, a decision tree algorithm predicting only the growth rate from  $\mathbf{C}_{med}$  (the ‘RandomForestRegressor’ function from the sci-kit learn package[137] having 1000 estimators and other parameters left with default values) reach a regression performance of  $0.71 \pm 0.01$  with the same dataset and cross-validation scheme, indicating AMNs can outperform regular machine learning algorithms.

As one can observe in Figure 2.3, the experimental variability on the measured growth rates is relatively high, and the  $Q^2$  values could be interpreted differently if taking this variability into account. To study this further, we estimated the best possible  $Q^2$  that can be reached at a given experimental variability. Precisely, for each experimental data point, we randomly drew a new point from a normal distribution with a mean and variance equal to what was experimentally determined for the original point. This point can be considered as an experimental ‘randomized’ point. After doing this for all points and computing the  $Q^2$ , repeating this process 1000 times, we obtain a mean  $Q^2 = 0.91$  with a standard deviation of 0.02. Consequently, the best possible  $Q^2$  accounting for experimental variability is 0.91, and the performance of  $Q^2 = 0.77$  (or 0.78) must be interpreted considering that value. Furthermore, substituting each point by a box defined by standard deviations of both measurement and prediction, we find that 79% for AMN-QP (76% for AMN-LP and 74% for AMN-Wt) of the boxes intersect the identity line indicating that these points are correctly predicted within the variances.

Our results show that AMNs can learn on FBA-simulated training sets and make accurate predictions while respecting the mechanistic loss, as shown in Figure 2.2. AMNs can also perform well on a small experimental growth rates dataset as shown in Figure 2.3. To demonstrate capabilities of AMNs beyond these tasks, we extracted from the ASAP database[138] a dataset of 17,400 growth rates for 145 *E. coli* mutants. Each mutant had a KO of a single metabolic gene and was grown in 120 media with a different set of substrates. Our AMNs training set, were therefore composed of medium composition and reaction KOs, both encoded as binary vectors, alongside the measured growth rates. More details can be found in Methods 2.5 ‘External training sets acquisition’. Results are presented in Figure 2.4 and compared with classical FBA results, which were obtained running Cobrapy using the same dataset and setting scaled upper bounds (cf. Methods 2.5 ‘Searching uptake fluxes upper bounds

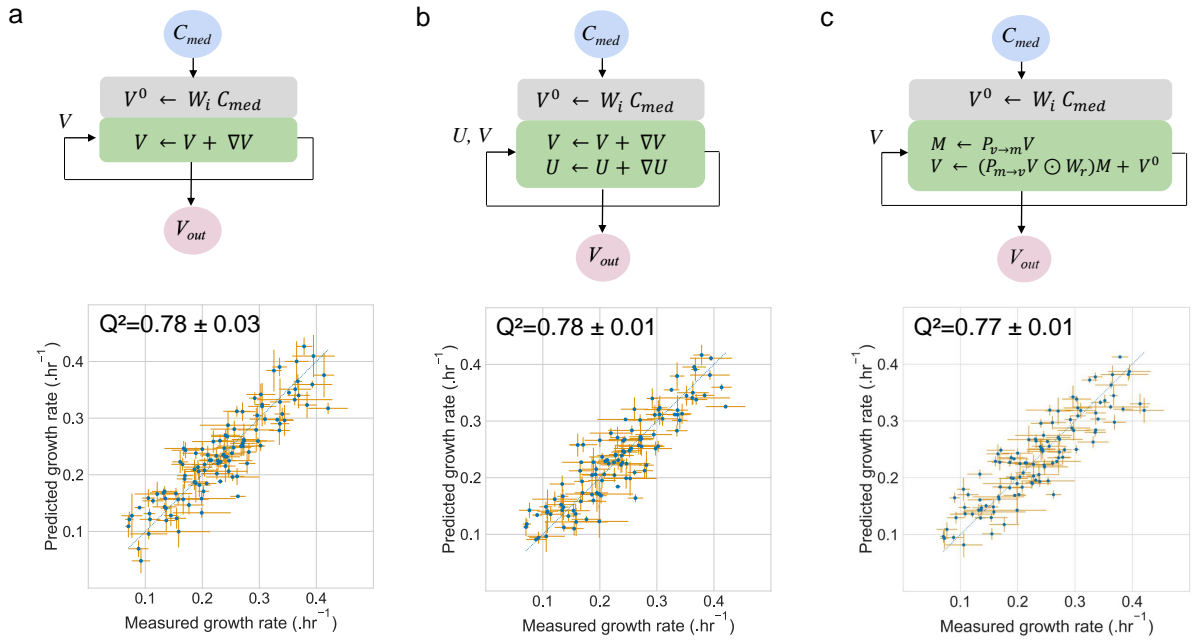


Figure 2.3: Benchmarking growth rate predictions by AMNs with experimental measurements. In all panels, the experimental measurements were carried out on *E. coli* grown in Mg with different combinations of carbon sources (strain DH5- $\alpha$ , model iML1515). Training and 10-fold stratified cross-validation were performed 3 times with different initial random seeds. All points plotted were compiled from predicted values obtained for each cross-validation set. In all cases, means are plotted for both axes (measured and predicted), and error bars are standard deviations. For the measured data, means and standard deviations were computed based on 3 replicates, whereas for predictions, means and standard deviations were computed based on the 3 repeats of the 10-fold cross-validation. **a.** Architecture and performance of AMN-QP. The neural layer (grey box) is composed of an input layer of size 38 ( $\mathbf{C}_{med}$ ), a hidden layer of size 500, and an output layer of size 550 corresponding to all fluxes ( $\mathbf{V}$ ) of the iML1515 reduced model. The mechanistic layer (green box) follows the neural layer and minimizes the loss between measured and predicted growth rate, as well as the losses of the metabolic network constraints. The model was trained for 1000 epochs with dropout = 0.25, batch size = 5, and the ‘Adam’ optimizer with a  $10^{-3}$  learning rate. **b.** Architecture and performance of AMN-LP. This model hyperparameters are identical to those of panel a. The neural layer computes the initial values for the 550 reaction fluxes (vector  $\mathbf{V}$ ), the initial values for the 1083 metabolite shadow prices (vector  $\mathbf{U}$ ) are set to zero. **c.** Architecture and performance of the AMN-Wt architecture. The model hyperparameters are those of the previous panels and the size of the  $\mathbf{W}_r$  matrix is 550x1083 (sizes of  $\mathbf{V}$  and  $\mathbf{U}$  vectors). Source data are provided as a Source Data file (cf. Data availability, 2.7).

in FBA) corresponding to medium uptake fluxes and constraining KO reactions to zero fluxes in the metabolic model. The AMN architecture (Figure 2.4 panel a) used with this dataset is similar to the architecture shown in Figure 2.1 panel c, with an added input for reaction KOs ( $\mathbf{R}_{KO}$ ). Importantly, we also added a term to the custom loss in order to respect the reaction KOs (cf. Methods 2.5 ‘Derivation of loss functions’ for more details).

The AMN regression performance in Figure 2.4 (aggregated predictions from a 10-fold cross-validation) reaches  $Q^2=0.81$  (Figure 2.4 panel b). For comparison, a decision tree algorithm predicting only the growth rates from  $\mathbf{C}_{med}$  and  $\mathbf{R}_{KO}$  (the ‘XGBRegressor’ function from the XGBoost package[139] with all parameters set to default values) yields a regression performance of 0.75, with the same cross-validation scheme and dataset.

The performance of classical FBA is poor, as no correlation can be found between measured and calculated growth rate (Figure 2.4 panel c). Such performance is expected as classical FBA relies on

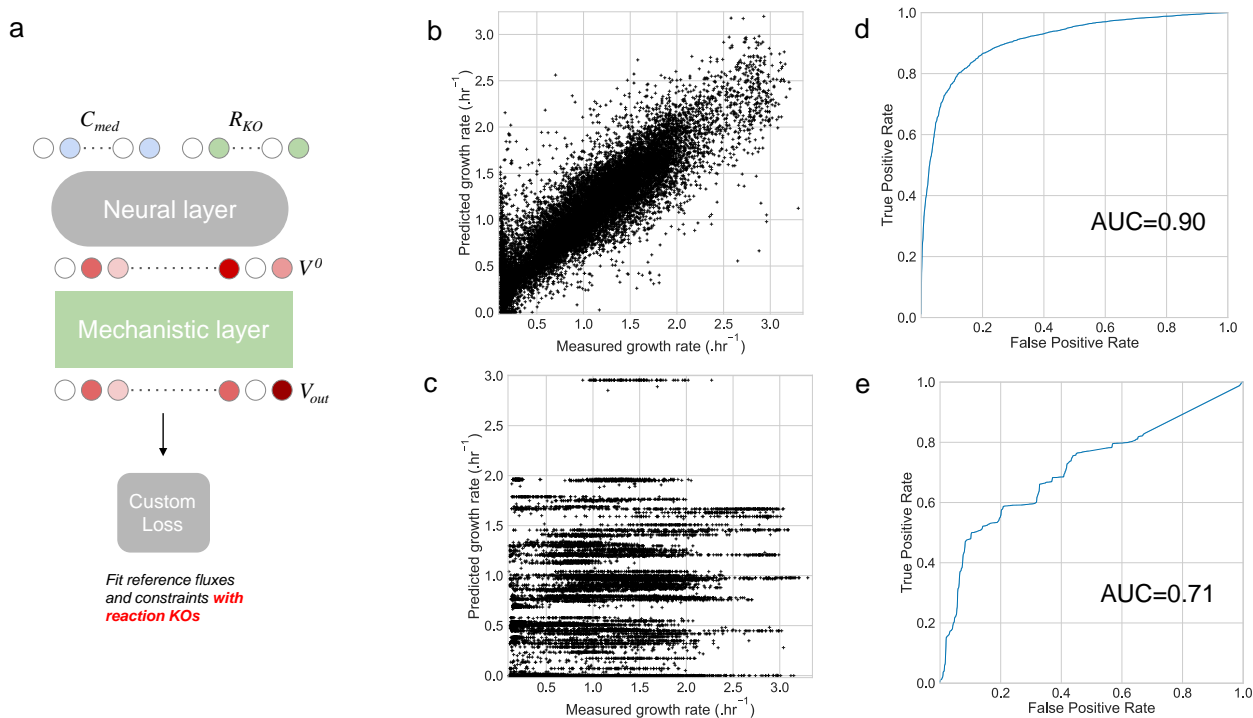


Figure 2.4: AMNs growth rate predictions for *E. coli* gene KO mutants. An AMN model was trained on a set of 17,400 measured growth rates of *E. coli* grown in 120 unique media compositions and 145 different single metabolic gene KOs. **a.** AMN architecture integrating metabolic gene KOs. This architecture is similar to Figure 2.1 panel c, except for a secondary input ( $R_{KO}$ ) for the neural layer, alongside the medium composition  $C_{med}$ . The  $R_{KO}$  input is a binary vector describing which reactions are KO. The custom loss function ensures that reference fluxes (*i.e.*, the *E. coli* mutants measured growth rates) and mechanistic constraints are respected and that reactions experimentally KO have in  $V_{out}$  a null flux value. The neural layer comprised one hidden layer of size 500 and the model was trained for 200 epochs with dropout = 0.25, batch size = 5, and the 'Adam' optimizer with a  $10^{-3}$  learning rate. **b.** AMN regression performance on aggregated growth rate predictions from a 10-fold cross-validation. The mechanistic layer used for this architecture was the QP solver. **c.** Regression performance of classical FBA with scaled upper bounds for compounds present in the medium and setting the upper bound and lower bound to zero for reactions that are KO (having a value of 0 in  $R_{KO}$ ). **d.** ROC curve of AMN results. We thresholded the measured growth rates (continuous values) in order to transform them into binary growth vs. no growth measures. **e.** ROC curve of classical FBA results. The same thresholding as for panel d was applied. Source data are provided as a Source Data file (*cf.* Data availability, 2.7).

fixed uptake fluxes. In contrast, FBA should perform better to predict growth vs. no growth (a classification task), this is due to the fact that the network structure of GEMs already provides a lot of information on reaction essentiality, growth yields on different substrates, and other qualitative insights about metabolism. Indeed, in the most recent GEM of *E. coli*, iML1515[44], an accuracy >90% was found for a dataset of growth assays based on media compositions (classification task predicting growth vs. no growth). Consequently, we also show in Figure 2.4 the performances of AMN and FBA for classification. Precisely, we treat the growth rate value predicted by either the AMN or FBA as a classification score for growth vs. no growth, to that end and following Orth *et al.*[140], we threshold the growth rate measures (continuous values) to binary values (1 when the growth rate is above 5% of the maximum growth rate of the dataset, 0 otherwise). With classifications, one can compute ROC curves, and these are shown in Figure 2.4, as panel d for AMN and panel e for classical FBA.

Overall, the results presented in Figure 2.4 show that for both regression and classification tasks, AMNs, which integrates learning procedures, outperforms classical FBA, which is based on maximizing



a biological objective only. Indeed, as mentioned in the introduction, one main issue of classical FBA is the unknown uptake fluxes, which have a large impact on the predicted growth rate value, while AMNs can handle this problem because of their learning abilities. To further showcase AMN capabilities, in particular when multiple fluxes are measured, we provide in Supplementary Figure S8 (appendix B) the performance of an AMN on a dataset from Rijsewijk *et al.*[24]. With this dataset, composed of 31 fluxes measured for 64 single regulator gene KO mutants of *E. coli* grown in 2 media compositions, our AMN reaches a variance averaged  $Q^2$  value of 0.91 in 10-fold cross-validation.

### 2.3.5 AMNs can be used in a reservoir computing framework to enhance the predictive power of traditional FBA solvers

As already mentioned in the introduction section, the uptake fluxes of *E. coli* nutrients, as well as their relation to external nutrient concentrations, remain largely unknown: the uptake flux for each compound may vary between growth media. In classical FBA calculations, this is usually ignored and the same upper bound (or zero, if a compound is absent) is used in all cases. Our results for KO mutants suggest that this strongly reduced regression performance of classical FBA, while in classification the effect is less severe. Nonetheless, for regression or classification the problem remains: how can realistic uptake fluxes be found?

In the following, we show a way to find these uptake flux values and improve the performances of classical FBA solvers (for both regression and classification). Once an AMN has been trained on a large dataset of FBA-simulated data, we can fix its parameters and exploit it in subsequent further learning in order to find uptake flux values that can be used in a classical FBA framework. Loosely inspired by reservoir computing[105], we call this architecture ‘AMN-Reservoir’ (Figures 2.1 panel d and 2.5 panel a). Let us note that we are not using usual reservoirs[105] with random weights and a post-processing trainable layer. As a matter of fact, we do not reach satisfactory performances when we substitute the AMN-Reservoir weights (learned during training) by random weights.

We benchmarked our AMN-Reservoir approach with two datasets. The first one is the one used in Figure 2.3, composed of 110 *E. coli* growth rates, and the second is a growth assay of *P. putida* grown in 296 different conditions[141] (more details in Methods 2.5 ‘External training sets acquisition’). The procedure used for the two datasets is the same. First, the AMN-Reservoir is trained on FBA simulations. For *E. coli* we used as an AMN-Reservoir the AMN-QP of Supplementary Table S1 (appendix B) trained on an iML1515 UB dataset, for *P. putida* we used the AMN-QP of Supplementary Table S1 (appendix B) trained on an iJN1463[141] UB dataset. Second, as shown in Figure 2.5 panel a, the whole experimental dataset is used to train the neural layer, setting up either a regression task for *E. coli* growth rates and a classification task for *P. putida* growth assays (growth vs. no-growth). After training on media compositions and measured growth rates (for both *E. coli* and *P. putida*), we extract the corresponding uptake fluxes ( $\mathbf{V}_{in}$ ). These uptake fluxes are then taken as input for a classical FBA solver for growth rate calculation, as shown in Figure 2.5 panel b.

The output of FBA was used to produce the results shown in 2.5 panels c and e. As a matter of comparison, we show the performance of FBA for the *E. coli* dataset (Figure 2.5 panel d) with scaled uptake fluxes bounds (*cf.* Methods 2.5 ‘Searching uptake fluxes upper bounds in FBA’), and for *P. putida* (Figure 2.5 panel f) where we used the same flux bounds as given in the reference study[141].

Overall, results shown in Figure 2.5 indicate that the usage of AMN-Reservoirs substantially increases the predictive capabilities of FBA without additional experimental work. Indeed, after applying the AMN-Reservoir procedure to find the best uptake fluxes, we raised the  $R^2$  on *E. coli* growth

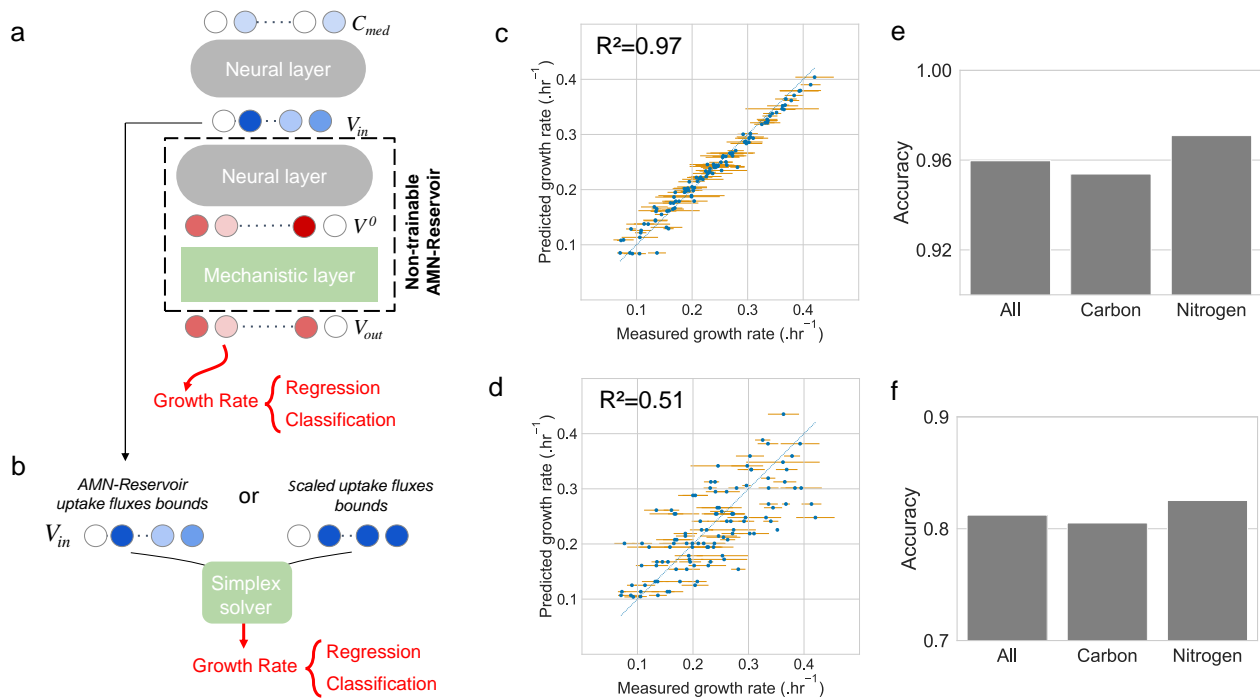


Figure 2.5: Reservoir computing for improving the predictive power of FBA modeling (strain *E. coli* DH5- $\alpha$ , model iML1515 and strain *P. putida* KT2440, model iJN1463). For panels c and d, plotted are the measured growth rates means and standard deviations, computed from replicates (*cf.* Methods 2.5). **a.** Learning architecture. The 2-step learning is similar to what is shown in Figure 2.1 panel d. Here an AMN with QP-solver is trained either on iML1515 (panels c and d) or iJN1463 (panels e and f) with FBA simulations. The AMN (with frozen parameters) is then connected to a prior trainable network that computes medium uptake fluxes ( $V_{in}$ ) from the medium composition ( $C_{med}$ ). From  $V_{in}$  the non-trainable reservoir returns all fluxes ( $V_{out}$ ) including the growth rate. Next, a regression or classification is carried out on the growth rate. For results presented in panels c and e, the neural layer comprised one hidden layer of size 500, the model was trained for 1000 epochs with dropout = 0.25, batch size = 5, and the Adam optimizer with a  $10^{-4}$  learning rate. **b.** Scheme showing the two possible inputs for Cobrapy (running a simplex solver), either using  $V_{in}$  extracted from panel a, or using scaled upper bounds on uptake fluxes. **c.** Regression performance of Cobrapy for the *E. coli* dataset when using  $V_{in}$ . **d.** Regression performance of Cobrapy when using scaled upper bounds on corresponding uptake fluxes. **e.** Accuracy performance of Cobrapy for the *P. putida* dataset when using  $V_{in}$ . **f.** Accuracy performance of Cobrapy when using original values of the study from J. Nogales *et al.* For the results in panels e and f, accuracies are given for the whole dataset ('All') composed of carbon source assays ('Carbon') and nitrogen source assays ('Nitrogen'). Source data are provided as a Source Data file (*cf.* Data availability, 2.7).

rates from 0.51 (panel d) to 0.97 (panel c) and that we raised the accuracy on *P. putida* growth assays from 0.81 (panel f) to 0.96 (panel e). We note that these uptake fluxes were found for the training set of the AMN-Reservoir, but we also show the performance of FBA with uptake fluxes found for cross-validation sets (Supplementary Figure S9, appendix B). As expected, it displays the same level of performance as the AMNs directly trained on experimental data (Figure 2.3).

## 2.4 Discussion

In this study we showed how a neural network approach, with metabolic networks embedded in the learning architecture, can be used to address metabolic modeling problems. Previous work on RNNs and PINNs for solving constrained optimization problems was re-used and adapted to develop three models (AMN-Wt, -LP and -QP) enabling gradient backpropagation within metabolic networks. The

models exhibited excellent performance on FBA generated training sets (Figure 2.2 and Supplementary Table S1, in appendix B). We also demonstrated that the models can directly be trained on an experimental *E. coli* growth rate dataset with good predictive abilities (Figure 2.3).

In classical FBA, all biological regulation mechanisms behind a flux distribution are ignored and flux computation relies entirely on bounds set on uptake or internal fluxes. Therefore, when performing classical FBA, one needs to set uptake bounds individually for each condition to reliably predict metabolic phenotypes. AMNs attempt to capture the overall effects of regulation via the neural layer while keeping the mechanistic layer for the metabolic phenotype. Indeed, as shown in Figure 2.4 and Supplementary Figure S8 (appendix B), gene KO's of metabolic enzymes or regulators can be taken into account via the neural layer. Such AMNs can potentially be trained on a variety of experimental inputs (wider than the carbon source composition shown in our studies) to grasp the effects of complex regulation processes in the cell and to better explain the end-point metabolic steady-state phenotype of an organism.

For improved adaptability, we also trained AMN-Reservoirs on large FBA-simulated training sets and used these to improve FBA computations on two experimental datasets (*E. coli* growth rates and *P. putida* growth assays). Figure 2.5 shows that our hybrid models substantially enhance classical FBA predictions both quantitatively and qualitatively, and this without any additional flux measurements.

One issue that impairs phenotype predictions with FBA is the lack of knowledge on media uptake fluxes and determining bounds on these fluxes is a core experimental work required for making classical FBA computations realistic. These bounds depend on cell transporters abundances, which may vary between conditions and depend on the cell's metabolic strategy. satFBA[76] is a variant of FBA that assumes fixed transporter levels and converts medium concentrations to possible uptake fluxes by kinetic rate laws, relying on a Michaelis-Menten value for each uptake reaction. In more sophisticated CBM approaches, such as molecular crowding FBA[142] or Resource Balance Analysis[73], constraints on the resource availability and allocation are added to obtain more biologically plausible metabolic phenotypes, but parameterizing such models requires additional data. To provide the necessary data to the aforementioned CBM methods and to validate results, fluxomics[143], metabolomics[59], or transcriptomics[60] have been used in the past. Because additional experimental work is needed with sophisticated CBM approaches, many users rely on classical FBA, which as we have seen, has limitations as far as quantitative predictions are concerned.

AMNs are used in this study for tackling the same issue as satFBA: predicting metabolites uptake fluxes from medium metabolite composition. To do so, where satFBA uses transporter kinetics with parameters that need to be acquired through additional measurements, AMNs use a pre-processing neural layer that is accessible for learning. Our AMN hybrid models get rid of additional experimental data for reaching plausible fluxes distributions. We do so by backpropagating the error on the growth rate or any other measured flux, to find complex relationships between the medium compositions and the medium uptake fluxes. To this end, we demonstrated the high predictive power of AMNs, and their re-usability in classical FBA approaches. Indeed, FBA developers and users may now make use of our AMN-Reservoir method for relating medium uptake fluxes to growth medium compositions. In this regard, a Source Data file (*cf.* Data availability, 2.7) gives uptake fluxes for the metabolites used in our benchmarking work with *E. coli* and *P. putida* (Figure 2.5), and these upper bounds for uptake fluxes can directly be used by Cobrapy to reproduce Figure 2.5 panels c and e.

Making FBA suitable for machine learning as we have done in this study opens the door to improve GEMs. For instance, in addition to estimating uptake fluxes, AMNs could be used to estimate the coefficients of the biomass reaction based on measurements. So far, these coefficients are derived based

on literature, but also using experimental data: growth rate, flux, and macromolecular fractions measures can help finding optimal coefficients[44]. However, these experiments are limited in number, and biomass coefficients are usually determined only once, for a single experimental setup, and are hardly extrapolated to all possible conditions. Some studies already underline this issue and attempt to efficiently integrate experimental data in the biomass reaction parametrization[57]. With AMNs, a trainable layer containing the biomass coefficients could be added, adapting the biomass reaction to any experimental setup. Another possible application of AMN is to enhance GEMs reconstruction based on quantitative prediction performance. Indeed, the method we developed for KOs could be adapted to screen putative reactions in a metabolic model so that its predictions match experimental data. This task should be performed after a manual curation, of course, to rely on existing literature knowledge and databases.

Returning to the curse of dimensionality issue mentioned in the introduction, we systematically studied at which training set sizes ‘black-box’ ML methods would yield performances similar to our AMN hybrid models. To that end, we trained a simple dense ANN model on training sets of increasing sizes. Results obtained with *E. coli* core[135] show that at least 500,000 labeled data (reference fluxes) are needed in the training sets to reach losses below 0.01 (*cf.* in Supplementary Figure S10, appendix B), which according to Figure 2.2 and Supplementary Table S1 (appendix B) are still one order of magnitude higher than all AMNs losses trained on only 1000 labeled data. This clearly demonstrates the capacity of hybrid models to reduce training set sizes by constraining the search space through the mechanistic layer. Other black-box models can also be used, indeed the experimental measurements used in Figures 2.3 and 2.4 can be fitted with decision tree algorithms (Random Forests[137] and XG-Boost[139]) with performances slightly under those of AMN. However, with these algorithms, nothing is learned regarding the mechanistic constraints and results produced by these methods cannot be fed back to classical FBA, as we do in Figure 2.5 with the AMN-Reservoir.

Beyond improving constraint-based mechanistic models and black-box ML models, AMNs can also be exploited for industrial applications. Indeed, since arbitrary objective functions can be designed and AMNs can be directly trained on experimental measurements, AMNs can be used to optimize media for the bioproduction of compounds of interest or to find optimal gene deletion and insertion strategies in typical metabolic engineering projects. In this latter case, reactions would be turned off via a trainable layer, which would be added prior to the mechanistic layers of our AMNs. Another potential application is the engineering of microorganism-based decision-making devices for the multiplexed detection of metabolic biomarkers or environmental pollutants. Here, AMNs could be used to search for internal metabolite production fluxes enabling one to differentiate positive samples containing biomarkers or pollutants from negative ones. Such a device has already been engineered in cell-free systems[41], and AMNs could be used to build a similar device *in vivo* by adding a trainable layer after the mechanistic layer whose purpose would be to select metabolite production fluxes that best split positive from negative samples.

## 2.5 Methods

### 2.5.1 Making metabolic networks suitable for neural computations

The set-up of our AMNs requires all reactions to be unidirectional; that is, the solutions must show positive-only fluxes (which is not guaranteed by usual GEMs). To split reactions of a given metabolic network into separate forward and reverse reactions, we wrote a standardization script that loads

an SBML model into Cobrapy[68] and screens for all two-sided reactions, then duplicating them into two separate reactions; and writes a new version of the model with bi-directional reactions split into separate forward and backward reactions. To avoid confusion, we add a suffix to these reaction names, either “for” or “rev” respectively designating the original forward reaction and the reversed reaction. The uptake reactions were also duplicated, even if encoded as one-sided, and their suffix was set to “i” for inflow reactions (adding matter to the cell), and “o” for outflow reactions (removing matter from the system).

As detailed in the next subsection, our unidirectional models are used to build flux data training sets. The duplicated iML1515[44] model is large, comprising 3682 reactions and 1877 metabolites. A substantial number of reactions in this model have zero fluxes for many different media, and it is unnecessary to keep these reactions during the training process. Prior to training, we therefore generated a reduced model by removing reactions having zero flux values along with the metabolites no longer contributing to any reactions. Using that procedure, we were able to reduce iML1515[44] model to only 550 reactions and 1083 metabolites.

## 2.5.2 Generation of training sets with FBA

Our reference flux data were obtained from FBA simulations, using the GNU Linear Programming Kit (‘GLPK’, a simplex-based method) on Cobrapy[68], with different models of different sizes. Throughout this paper, when ‘reference FBA-simulated data’ is mentioned, it refers to data computed with this method.

Reference FBA-simulated data for metabolic flux distributions were generated using models downloaded from the BiGG database[136]. The models were used to generate data using Cobrapy[68] following a precise set of rules. First, we identified essential uptake reactions for the models we used (*E. coli* core[135] and iML1515[44]) which we defined in the following way: if one of these reactions has its flux upper bound set to 0 mmol.gDW<sup>-1</sup>.h<sup>-1</sup>, the ‘biomass’ reaction optimization is impossible, even if all other uptake fluxes bounds are set to a high value, *i.e.*, 1000 mmol.gDW<sup>-1</sup>.h<sup>-1</sup>. In other words, we identified the minimal uptake fluxes enabling growth according to the models. For *E. coli* core[135] we found 7 of such obligate reactions (for the uptake of CO<sub>2</sub>, H<sup>+</sup>, H<sub>2</sub>O, NH<sub>4</sub>, O<sub>2</sub>, Phosphate, and Glycerol as the carbon source). For iML1515<sup>20</sup> we had the same 7 obligate reactions and additional salt and ions uptake reactions (for the uptake of Fe<sup>2+</sup>, Fe<sup>3+</sup>, Mn<sup>2+</sup>, Zinc, Mg, Calcium, Ni<sup>2+</sup>, Cu<sup>2+</sup>, Selenate, Co<sup>2+</sup>, Molybdate, Sulfate, K<sup>+</sup>, Sodium, Chloride, Tungstate, Selenite). With iML1515[44], we also added as obligate reactions the uptake of four amino acids (Alanine, Proline, Threonine and Glycine) in order to be consistent with our experimental training set where the four amino acids were systematically added to M9 (*cf.* subsection ‘Generation of an experimental training set’). During reference FBA-simulated data generation, the upper bounds on these obligate reactions were set to 10 mmol.gDW<sup>-1</sup>.h<sup>-1</sup>.

To generate different media compositions, we added to the obligate reactions a set of variable uptake reactions. For the *E. coli* core model[135] we added 13 variable uptake reactions (for Acetate, Acetaldehyde, Oxoglutarate, Ethanol, Formate, Fructose, Fumarate, Glutamine, Glutamate, Lactate, Malate, Pyruvate, and Succinate). For each generated medium, a set of variable uptake reactions was selected, drawn with a binomial distribution  $B(n, p)$  with  $n=13$  and  $p = 0.5$ ,  $p$  being a tunable parameter related to the ratio of selected reaction. Consequently, the mean number of selected variable uptake reactions was  $n \times p = 6.5$ . Next for each selected reaction, the upper bound continuous value of the reaction flux was randomly drawn from a uniform distribution between 2 and 10 mmol.gDW<sup>-1</sup>.h<sup>-1</sup>. For the iML1515[44] model, to limit the combinatorial search space, the selected variable uptake reactions were those of the experimental training set and consequently between 1 and 4 variable uptake reac-



tion were added (*cf.* subsection ‘Generation of an experimental training set’). The upper bound values for each selected variable reaction were chosen randomly between 0 and 2.2 mmol.gDW<sup>-1</sup>.h<sup>-1</sup> (0 excluded). The 2.2 threshold was chosen to produce predicted growth rates that were in the range of those observed experimentally. For the *P. putida* iJN1463[141] model, we used the same approach with variable uptake reactions selected from the experimental training set, and consequently 1 variable uptake reaction was added to obligate reactions (described as the minimal medium in the reference study[141]) for each element of the training set. The upper bound values for each selected variable reaction were chosen randomly between 0 and 10 mmol.gDW<sup>-1</sup>.h<sup>-1</sup> (0 excluded).

After generating the set of growth media for *E. coli* core[135], iML1515[44] and iJN1463[141] we ran FBA in Cobrapy[68] for each medium and recorded all steady-state fluxes including the growth rate (flux of the ‘biomass’ reaction). These fluxes were used as a training set for all models presented in Figure 2.2 and in Supplementary Table S1 (appendix B). All AMN architectures were trained on the ‘biomass’ flux (*i.e.*, the growth rate), while ANN architectures were trained on all fluxes. For all UB training sets, the variable uptake flux values were those used by Cobrapy[68] to generate the training set. For EB training sets, the variable uptake flux values were those calculated by Cobrapy[68] at steady state.

### 2.5.3 Derivation of loss functions

Loss functions are necessary to assess the performances of all MM solvers and all AMN architectures (AMN-Wt, -QP, and -LP) and also to compute the gradients of the QP solvers. In the following and subsequent subsections, all vectors and matrices notations are defined when they are first used and can also be found in Supplementary Table S2 (appendix B).

To compute loss, we consider a metabolic model with  $n$  reactions and  $m$  metabolites. Let  $V = (v_1, \dots, v_n)^T$  be the reaction flux vector and  $S$  the  $m \times n$  stoichiometric matrix of the model. We assume some metabolites can be imported in the model through a corresponding uptake reaction. Let  $V_{in}$  be the vector of  $n_{in}$  upper bounds (or exact values) for these uptake reactions, and let  $P_{in}$  the  $n_{in} \times n$  projection matrix such that  $V_{in} = P_{in}V$ . We further assume that some reaction fluxes have been experimentally measured, let  $V_{ref}$  be the vector of reference flux data (FBA-simulated or measured). With  $P_{ref}$  the  $n_{ref} \times n$  projection matrix for measured fluxes.  $V$  is calculated by solving the following quadratic program (QP):

$$\begin{aligned} \min & \left( \|P_{ref} V - V_{ref}\|^2 \right) \\ \text{s.t.:} & \\ & SV = 0 \\ & P_{in}V \leq V_{in} \\ & V \leq 0 \end{aligned} \tag{2.1}$$

For each solution,  $V$ , of eq. 2.1, four loss terms are defined.  $L_1$  is the loss on the fit to the reference data.  $L_2$  ensures the respect of the network stoichiometric constraint ( $SV = 0$ ).  $L_3$  ensures the respect of the constraints on input fluxes that depend on medium composition ( $P_{in}V \leq V_{in}$ ). Finally,  $L_4$  ensures the respect of the flux positivity ( $V \leq 0$ ). The losses are normalized, respectively by  $n_{ref}$  for the fit to reference data,  $m$  for the stoichiometric constraint,  $n_{in}$  for the boundary constraints, and  $n$  for the flux positivity constraints.

Summing the four terms, the loss  $L$  is:

$$L = L_1 + L_2 + L_3 + L_4 \quad (2.2)$$

$$= \frac{1}{n_{ref}} \|P_{ref}V - V_{ref}\|^2 + \frac{1}{m} \|SV\|^2 + \frac{1}{n_{in}} \|ReLU(P_{in}V - V_{in})\|^2 + \frac{1}{n} \|ReLU(-V)\|^2$$

More details about each loss term can be found in the Supplementary Information ‘QP-solver equations’ (appendix B).

When reaction KOs are added to the input of AMNs (as in Figure 2.4), we add a term to the loss function,  $L_5$ , for ensuring a null value for fluxes that have their reaction KO:

$$L_5 = \frac{1}{n_{KO}} \|ReLU(P_{KO}V - R_{KO})\|^2 \quad (2.3)$$

where  $R_{KO}$  is a vector of length  $n_{KO}$  describing which reactions are KO, and  $P_{KO}$  the projection matrix mapping the whole flux vector  $V$  to KO fluxes.

## 2.5.4 Wt-solver

The Wt-solver describes a metabolic state by two vectors  $V$  and  $M$ , representing respectively the reaction fluxes and the metabolite production fluxes. The initial value ( $V^0$ ) for vector  $V$  can be arbitrary as long as the uptake medium bounds are respected. Vectors  $V$  and  $M$  are iteratively computed until convergence using the following equations:

$$M = P_{v \rightarrow m}V \quad (2.4)$$

$$V = (P_{m \rightarrow v}V \odot W_r)M + V^0$$

where  $W_r$  is a consensus weight matrix representing flux branching ratios,  $P_{v \rightarrow m} = ReLU(S)$ ,  $P_{m \rightarrow v} = ReLU(\left[-\frac{1}{z_i s_{j,i}}\right])$ ,  $S$  is the stoichiometric matrix,  $s_{j,i}$  the stoichiometric coefficient of row  $j$  and column  $i$ ,  $z_i$  the number of strictly negative elements in column  $i$  of  $S$ , and  $\odot$  the Hadamard product operation. Additional details on the procedure and the associated matrices are provided in Supplementary Information section ‘Wt-solver equations’ (appendix B).

## 2.5.5 LP-solver

The LP method aims at solving linear constrained problem similar to the ones solved by FBA. It relies on the results from Yang *et al.* [133] where the authors used gradient descent on both primal and dual variables of the problem.

When the uptake fluxes are known (EB method), the FBA problem can be written as:

$$\begin{aligned}
& \max: c_{FBA}^T V \\
& \text{s.t.:} \\
& S_{int} V = -b_{FBA} \\
& V \leq 0
\end{aligned} \tag{2.5}$$

where  $S_{int}$  is the stoichiometric matrix with uptake fluxes zeroed out (*i.e.*, fluxes that add matter in the system). In other words,  $S_{int}$  is the internal stoichiometric matrix. Let us consider  $b_{FBA}$ , a vector of dimension  $m$  with  $b_i$  corresponding to uptake fluxes of medium metabolite  $m_i$  (either as an exact value for EB or an upper bound for UB) and  $c_{FBA}$ , the objective vector of dimension  $n$  (in this work this vector has non-zero elements only for reference fluxes like the ‘biomass’ reaction flux (growth rate)).

This problem can be written in its dual form with  $U$  being the dual variable of  $V$ :

$$\begin{aligned}
& \min: -b_{FBA}^T U \\
& \text{s.t.:} \\
& S_{int}^T U \leq c_{FBA}
\end{aligned} \tag{2.6}$$

As mentioned before, the problem given by eq. 2.5 can be solved conjointly with problem given by eq. 2.6 by iteratively updating  $V$  and its dual  $U$  through gradient descent:

$$\begin{aligned}
V^{(t+1)} &= V^{(t)} - dt \nabla V \\
U^{(t+1)} &= U^{(t)} - dt \nabla U \\
V^{(0)} &= P_{in}^T V_{in} \text{ and } U^{(0)} = 0
\end{aligned} \tag{2.7}$$

where  $t$  is the iteration number and  $dt$  the learning rate.

Note that initialization of LP with uptake fluxes is not mandatory with the method from Yang *et al.*[133] as it has been proven to converge to global optimum independently from the initial values of  $V$  and  $U$ . Detailed expressions and derivations of gradients for  $U$  and  $V$  are provided in Supplementary Information ‘LP-solver equations’ along with Figures S4 and S5 (appendix B).

## 2.5.6 QP-solver

The QP solver solves the quadratic program given by eq. 2.1. While the QP system can be solved by a simplex algorithm, solutions can also be approximated by calculating the vector  $V$  that minimizes the loss ( $L$  in eq. 2.2). The gradient  $\nabla V$  for vector  $V$  can thus be found by solving  $\frac{\partial L}{\partial V} = 0$  and, as in eq. 2.7,  $V$  is computed iteratively with iteration number  $t$  and learning rate  $dt$ .

Detailed expressions and derivations for the gradient  $\nabla V$ , when exact bounds (EBs) or upper-bounds (UBs) are provided for uptake flux medium, can be found in the Supplementary Information ‘QP-solver equations’ (appendix B).



### 2.5.7 ANN architecture

The ANN architecture is a ‘black box’ dense neural network. As with the other architectures the input layer corresponds to the medium uptake fluxes,  $\mathbf{V}_{in}$ , and the output layer corresponds to the set of all fluxes  $\mathbf{V}_{out}$ . In order to assess losses with the ANN architecture, which does not have any mechanistic layer, each entry of the training set contained all flux values (in other words,  $\mathbf{V}_{ref}$  contains all fluxes). Consequently, the training process with ANN consists in fitting all predicted fluxes to reference flux data (computing the MSE on all the fluxes). To compare results with the other architectures,  $R^2$  and  $Q^2$  are computed for the growth rate, and constraint losses are computed using predictions for all fluxes, using the formulation given in the subsection Methods 2.5 ‘Derivation of loss functions’.

### 2.5.8 AMN architectures

As shown in Figures 2.1 and 2.3, we propose three AMN architectures: AMN-Wt, AMN-LP and AMN-QP. The AMNs are run with training sets using exact values (EB) or only upper bound values (UB) for medium uptake fluxes. All AMNs take as their input a vector of bounds of size  $n_{in}$  for medium uptake fluxes ( $\mathbf{V}_{in}$ ) and then transform it via a dense neural network the input vector into an initial vector of size  $n$  for all fluxes ( $\mathbf{V}^0$ ), which is refined through an iterative procedure computing  $V^{(t+1)}$  from  $V^{(t)}$ . With all AMNs a  $n_{in} \times n$  weight matrix transforming  $\mathbf{V}_{in}$  to  $\mathbf{V}^0$  is learned during training, and we name this transforming layer the neural layer. With AMN-LP/QP,  $V^{(t)}$  is iteratively updated in a mechanistic layer by the gradient ( $\nabla V$ ) of LP/QP solvers (cf. previous subsections in Method). With AMN-Wt, the mechanistic layer computes  $V^{(t+1)}$  from  $V^{(t)}$  using the transformations shown in Figure S1 (appendix B), which include a  $n \times m$  weight matrix ( $W_r$ ). That weight matrix can be directly computed from training data when all fluxes are provided or can be learned during training, when only a fraction of fluxes are provided (like the growth rate with experimental datasets). In our implementation (cf. Code Availability, 2.8) AMN-Wt is embedded in a RNN Keras cell<sup>[144]</sup> and both matrices  $W_i$  and  $W_r$  are learned during training. ANN-Wt is further detailed in Supplementary Information ‘AMN-Wt architecture’ (appendix B).

With all AMN architectures, the values of  $\mathbf{V}$  corresponding to  $\mathbf{V}_{in}$  are not updated in the neural nor mechanistic layers when training with exact values for medium uptake (EB training sets).

### 2.5.9 ANN and AMN training parameters

For ANN and AMN architectures, we use the Mean Squared Error ( $L_1$  in eq. 2.2) for measured fluxes as the objective function to minimize during training. In all AMN architectures we add to the  $L_1$  loss function the terms corresponding to the 3 losses derived from the constraints of the metabolic model ( $L_2$ ,  $L_3$  and  $L_4$  in eq. 2.2).

The parameters used when training ANNs and AMNs, there are two types:

1. Reference data parameters. Reference data can either be FBA-simulated or experimental. For FBA-simulated data, we can tune the size of the training set to be generated. We can also modify the mean number of selected variable intake medium fluxes, and the number of levels (*i.e.* the resolution) of the fluxes. We can also modify the variable uptake reactions list, but this modifies the architecture of the model (initial layer size), so we kept the same list for each model in the present work. The lists can be found in the subsection ‘Generation of Training Sets with FBA’.

2. Model hyperparameters. During learning on FBA-simulated or experimental data, ANN and AMN have a small set of parameters to tune: the number and size of hidden layers, the number of epochs, the batch size, the dropout ratio, the optimizer and learning rate, and the number of folds in cross-validation. These numbers are provided in Supplementary Table S1 (appendix B) for models trained on FBA-simulated data and in the captions of Figures 2.3 to 2.5 for models trained on experimental data.

### 2.5.10 Searching uptake fluxes upper bounds in FBA

The goal of this optimization was to find the best scaler for fluxes to best match experimentally determined growth rates, by using 'out-of-the-box' FBA, simply informing the presence or absence of the flux according to the experimental medium composition. The optimal scaler used in Figures 2.4 and 2.5 was found using the Cobrapy software package[68] by simply searching for the maximum  $R^2$  between experimental and FBA-predicted growth rates for scalars ranging between 1 and 10.

### 2.5.11 Generation of an experimental training set

Ten carbon sources (Ribose, Maltose, Melibiose, Trehalose, Fructose, Galactose, Acetate, Lactate, Succinate, Pyruvate) were picked for being the variables of our training sets. These could ensure observable growth as a sole carbon source with a concentration of  $0.4 \text{ g.L}^{-1}$  in our M9 preparations. The selected carbon sources enter different parts of the metabolic network: 6 sugars enter the upper glycolysis pathway, and 4 acids enter the lower glycolysis pathway or the TCA cycle. With a binary (*i.e.*, presence or absence of each carbon source) approach when generating the combinations to test for making the experimental training set, we generated all possible combinations of 1, 2, 3 or 4 carbon sources simultaneously present in the medium. Naturally, we picked all 1-carbon source media combinations for experimental determination (only 10 points). Then, we randomly selected 100 more combinations to experimentally determine, by randomly picking 20 points from the 2-, 40 points from the 3- and 40 points from the 4-carbon source combinations sets. The python scripts to generate these combinations and pick the ones for making our experimental training set are available on our Github package (*cf.* Codes availability section). After picking the combinations to test, we experimentally determined the maximum specific growth rate of *E. coli* for each combination of carbon sources in M9 (*cf.* next two subsections). The mean over replicates for each media composition was computed as the corresponding growth rate value to make the final experimental training set (*cf.* Methods 2.5 'Growth rate determination').

### 2.5.12 Culture conditions

The base medium for culturing *E. coli* DH5- $\alpha$  (DH5a) was a M9 medium prepared with those final concentrations:  $100 \mu\text{M}$   $\text{CaCl}_2$ ,  $2 \text{ mM}$   $\text{MgSO}_4$ ,  $1 \times$  M9 salts ( $3 \text{ g.L}^{-1}$   $\text{KH}_2\text{PO}_4$ ,  $8.5 \text{ g.L}^{-1}$   $\text{Na}_2\text{HPO}_4 \cdot 2\text{H}_2\text{O}$ ,  $0.5 \text{ g.L}^{-1}$   $\text{NaCl}$ ,  $1 \text{ g.L}^{-1}$   $\text{NH}_4\text{Cl}$ ),  $1 \times$  trace elements ( $15 \text{ mg.L}^{-1}$   $\text{Na}_2\text{EDTA} \cdot 2\text{H}_2\text{O}$ ,  $4.5 \text{ mg.L}^{-1}$   $\text{ZnSO}_4 \cdot 7\text{H}_2\text{O}$ ,  $0.3 \text{ mg.L}^{-1}$   $\text{CoCl}_2 \cdot 6\text{H}_2\text{O}$ ,  $1 \text{ mg.L}^{-1}$   $\text{MnCl}_2 \cdot 4\text{H}_2\text{O}$ ,  $1 \text{ mg.L}^{-1}$   $\text{H}_3\text{BO}_3$ ,  $0.4 \text{ mg.L}^{-1}$   $\text{Na}_2\text{MoO}_4 \cdot 2\text{H}_2\text{O}$ ,  $3 \text{ mg.L}^{-1}$   $\text{FeSO}_4 \cdot 7\text{H}_2\text{O}$ ,  $0.3 \text{ mg.L}^{-1}$   $\text{CuSO}_4 \cdot 5\text{H}_2\text{O}$ ; solution adjusted at  $\text{pH}=4$  and stored at  $4^\circ\text{C}$ ),  $1 \text{ mg.L}^{-1}$  Thiamine-HCl and  $0.04 \text{ g.L}^{-1}$  amino acid mix so that each amino acid (L-Alanine, L-Proline, L-Threonine, Glycine) was at a final concentration of  $5 \text{ mg.L}^{-1}$  in the medium. The additional carbon sources that could be added were individually set to a final concentration of  $0.4 \text{ g.L}^{-1}$ . The pH was adjusted at 7.4 prior to a  $0.22 \mu\text{m}$  filter

sterilization of the medium. Pre-cultures were recovered from glycerol -80°C stocks, grew in Luria-Bertani (LB) broth overday for 7 hours, then used as 5µL inoculate in 200µL M9 (supplemented with variable compounds) in 96 U-bottom wells plates overnight for 14 hours. Then 5µL of each well was passed to a replicate of the plate on the next day for growth monitoring. The temperature was set to 37°C in a plate reader (BioTek HTX Synergy), with continuous orbital shaking at maximum speed, allowing aerobic growth for 24 hours. A monitoring every 10 minutes of the optical density at 600 nm ( $OD_{600}$ ) was performed. A figure for summarizing the experimental workflow is available in Figure S11 (appendix B).

### 2.5.13 Growth rates determination

The maximal growth rate was determined by sliding a window of 1 hour-size, performing a linear regression on the  $\log(OD_{600})$  data in each window. We then retrieve the maximum specific growth rate as the maximum regression coefficient over all windows. If several growth phases are visible, one can omit a part of the growth curve for the maximal growth rate determination (for this study we always retrieved the maximal growth rate on the first growth phase, so as we are certain that the media contains all added carbon sources). 8 replicates for each medium composition were performed (on a single column of a 96-well plate). Outliers were manually removed after visual inspection of the growth curves or clear statistical deviation of the computed growth rate from the remaining replicates. The numbers of replicates kept range from 2 to 8, with an average of  $4.6 (\pm 1.6)$  replicates per medium composition. Means and standard deviations over replicates were computed to be used for training AMNs and making figures. All raw data and the code to process it are available in the Github repository (cf. Code availability, 2.8).

### 2.5.14 External training sets acquisition

**Growth rates of *E. coli* metabolic gene KO mutants.** The dataset was downloaded from the ASAP database (Mutant Biolog Data I for K-12 mutant strains)[138]. That dataset was pre-processed by applying several filtering steps: removing substrates that do not appear in iML1515 as possible substrates for uptake fluxes, removing genes not found in iML1515, and removing all data duplicates to obtain a balanced and coherent dataset. The filtered dataset contains 17,400 growth rates: 145 *E. coli* mutants (each having a KO of a single metabolic gene) grown in 120 conditions (each with a different set of substrates) from Biolog phenotype microarrays[85]. The final training set can be found in the source data, provided as a Source Data file (cf. Data availability, 2.7).

For practical reasons, we converted the information about metabolic gene KOs information into binary vectors describing which reactions are directly affected by a gene KO, called  $\mathbf{R}_{KO}$  in Figure 2.4. This mapping was automated with iML1515's ability to link genes and reactions. For reactions performed by enzymes encoded by more than one gene, we make the assumption that when any of these genes is knocked-out, the reaction is also knocked-out.

For the FBA computation (Figure 2.4 panels c and e), we set an arbitrary upper bound on uptake fluxes (11 mmol.gDW<sup>-1</sup>.h<sup>-1</sup> was found to be the best value in terms of regression performance) for each substrate in the dataset when it is present (otherwise 0). To simulate a KO, we set the lower and upper bound of a reaction to zero.

To transform the measured growth rates from continuous values into binary values (for the ROC curves in Figure 2.4 panels d and e), following Orth *et al.*[140], we applied a threshold of 0.165 .h<sup>-1</sup>, which is equal to 5% of the maximum growth rate (3.3 .h<sup>-1</sup>) found in the dataset. Therefore, the classification

task can be seen as the ability for the model to classify growth rates below and above the threshold value.

***P. putida* growth assays.** The dataset used to generate Figure 2.5 (panels e and f) was taken from the study of J. Nogales *et al.*[141] presenting iJN1462 (an updated version called iJN1463 is available on BiGG[136]) for *P. putida*'s GEM. This state-of-the-art GEM of *P. putida* KT2440 contains a few hundred more genes and reactions from the previous models, allowing better coverage. The dataset corresponds to growth assays with 188 carbon and 108 nitrogen sources. For each condition, we verified that an uptake reaction flux was present in the iJN1463[141] model. 55 conditions contained a nutrient source without a corresponding uptake reaction in the model. For all those conditions, the AMN input would be the minimal medium. In order to avoid biasing the training set with 55 identical conditions, we kept one condition describing the minimal medium for carbon sources and one condition describing the minimal medium for nitrogen sources. The 55 conditions were added back to compute the final score. The training set can be found in the source data, provided as a Source Data file (*cf.* Data availability, 2.7).

The minimal medium assumed in our simulations was taken from J. Nogales *et al.*[141], reporting a set of uptake fluxes upper bounds. When testing a carbon (nitrogen) source, glucose ( $\text{NH}_4$ ) was removed from the minimal medium, and the respective nutrient source metabolite was added. Using these simulated growth media, accuracies on growth predictions using Cobrapy (Figure 2.5 panel f) were calculated considering as positive all non-zero growth predictions. Results presented in Figure 2.5 panel e were obtained by training a reservoir on simulations as explained in Methods 2.5 'AMN and ANN training parameters'. Thus, this reservoir was used to fit experimental data, and  $\mathbf{V}_{\text{in}}$  was directly used as an input for Cobra.

## 2.6 Statistics & reproducibility

As stated in the previous section Generation of training sets with FBA, the exchange reactions upper bounds were randomized to produce FBA-simulated training sets. No statistical method was used to predetermine the sample size, which was chosen based on time and resources. Blinding was not relevant to generate these training sets. When performing cross-validation data were excluded from these training sets depending on the fold size as indicated in Figure 2.2 and in Supplementary Table S1 (appendix B). Additionally, in Supplementary Table S1 (appendix B), 10% of the training set was removed and placed in a test set prior performing training and cross-validation.

As stated in the previous section Generation of an experimental training set, the media of the experimental training set were randomized by randomly drawings carbon sources combinations. The growth rate measures were computed as means over 2 to 8 technical replicates. In all figures displaying this dataset, we show the standard deviation over replicates as the error bars. No statistical method was used to predetermine the sample size of 110. This size was chosen based on time and resources. Blinding was not relevant to generate this training set.

As stated in the previous section External training sets acquisition, we found two additional publicly available datasets, for which the authors did not specify any statistical method to pre-determine the sample size. To our knowledge, there was no replication scheme for experiments, for both external training sets. Incompatible data were excluded in the pre-processing steps of the training set stemming from the ASAP database (*cf.* previous section). For more details on the statistics and reproducibility of these external training sets, please refer to the original studies.

## 2.7 Data availability

Metabolic models used in this study can be found with the following accessions on the BiGG database[136]: [e\\_coli\\_core](#), [iML1515](#), [iJN1463](#). Unidirectional versions of these models can be found on our repository, at [this link](#). The original dataset from the ASAP database[138] can be found under the accession [Mutant Biolog Data I](#). The original dataset from Nogales *et al.*[141] can be found in Supporting Information’s Table S2 (appendix B) of the study.

The source data underlying all figures presented in the main manuscript and Supplementary Information in appendix B (including training sets used in Figures 2.3 to 2.5), are provided with this paper as a downloadable archive, on this [clickable link](#). Additional datasets and raw data are available on our Github repository (*cf.* Code availability, 2.8), or from the corresponding author upon request.

## 2.8 Code availability

All scripts and data for generating results presented in this paper are available within a documented repository. For a citable and stable version of the repository supporting this article, refer to [Zenodo](#) ([clickable link](#)). Alternatively, to access future releases and interact with the repository authors, refer to [Github](#) ([clickable link](#)). The repository includes tutorials in Google Colab notebooks. The released codes make use of Cobrapy[68], numpy[145], scipy[146], pandas[147], tensorflow[148], scikit-learn[137] and keras[144] libraries. Figures were generated using the matplotlib[149] and seaborn[150] libraries.

## 2.9 Acknowledgements

JLF would like to acknowledge funding provided by the ANR funding agency grant numbers ANR-18-CE44-0015 (SynBioDiag project) and ANR-21-CE45-0021-01 (AMN project) and the UE HORIZON program (grant number 101070281). LF is supported by INRAE’s MICA department and by INRAE’s metaprogram DIGIT-BIO. BM is supported by an Ecole Normale Supérieure (ENS) Scholarship. We thank Aymeric Gaudin (CentraleSupélec Engineering School) for early development in reservoir computing with AMN, Ivan Radkevich (University of Paris Saclay) for his work on custom RNN cells and Tom Lorthios and Hadi Jbara (AgroParisTech and University of Paris Saclay) for their help on collecting data for experimental training sets. We thank Anne Giralt and Laetitia Laversa (INRAE) for reading and improving our manuscript.

## 2.10 Author contributions

LF and JLF wrote the core of the text of the manuscript. JLF designed the study and wrote the Wt and QP-solver and all the AMN and AMN-Reservoir codes used to produce results presented in Figures 2.1 to 2.5. BM wrote the LP-solver and the corresponding part in the Methods section and Supplementary Information. LF benchmarked all codes, wrote the codes transforming SBML models into unidirectional networks and processing experimental data, and handled the Colab and Git implementations. LF also performed all experimental work reported in Figure 2.3, acquired the data and run the AMN and AMN-Reservoir codes to produce Figures 2.4 and 2.5 and wrote the corresponding Methods section. WL contributed to designing the project and was involved in the discussions and writing the manuscript. All authors read, edited, and approved the manuscript.

## **2.11 Competing interests**

The authors declare no competing interests.



## Chapter 3

# Further assessment and improvements of hybrid models to exploit GEMs

In the previous chapter, I demonstrated the basic abilities of AMNs, with three key aspects: (i) we can surrogate FBA with different kinds of ‘neural’ solvers which can be included in ANNs thanks to their gradient backpropagation compatibility, termed ‘mechanistic layers’; (ii) we can build predictive models, AMNs, that use an ANN with a custom ‘mechanistic loss’, in order to learn the best input for the mechanistic layer, by generalizing from a set of conditions; and (iii) we can use a two-step learning process, termed ‘reservoir computing’, that relies on first training an AMN on FBA simulations, then re-train it with an experimental dataset to find the best FBA inputs in order to match experimental data. We showcased the capabilities of AMNs with (i) *in silico* datasets generated through FBA subjected to random combinations of upper bounds applied on uptake fluxes, (ii) an experimental dataset of 110 *E. coli* growth rates grown in mixed carbon sources (designed and acquired by myself), (iii) three external phenotyping datasets: *E. coli* growth rates for 120 single metabolic gene KOs mutants grown in 145 substrate conditions; 296 *P. putida* growth assays; as well as an *E. coli* fluxomics dataset, obtained with 64 single regulator gene KOs mutants grown in two substrate conditions, with 31 measured fluxes.

We clearly showed that the predictive performance of GEMs was surpassed without requiring more experimental measures, with the AMNs formulation. However, there is still a lot to investigate on different parts of the methodology, to further assess its capabilities and limits, and eventually give leads on how to improve it. Importantly, this chapter assesses how much AMN predictions are reliable in terms of GEMs constraint respect, which was not done in the previous Chapter, but is critical in the scope of developing a reliable hybrid model. Indeed, one advantage of hybrid models like AMNs is to bring mechanistic insights, that can be only reached if predictions respect GEMs constraints.

Each of the investigations will be divided in a section of this chapter, with an introduction stating what is the part of the methodology that is investigated, and why; then the methods and finally the results and their discussion. Before presenting my investigations, I will describe in a preamble the different training sets and metrics used in this chapter. Then, investigations will first focus on the ‘neural’ mechanistic solvers, notably by reformulating the QP-solver method, which was found to be incomplete. Then I will investigate what is the behavior of such mechanistic solvers when included in AMNs, as ‘mechanistic layers’, which is not obvious as this approach is rather new and innovative. Next, I investigate how the AMNs architectures can be automatically searched by hyperparameter optimization, leading to performance improvements. After that, I assess the performances of alternative ML models to surrogate FBA, underlining the importance of both training sets contents and the type of models for overall performance. Finally, the last section will assess AMN performance with *in vivo*



datasets that are splitted in specific ways. To conclude, I add some remarks on the present chapter's results and open the discussion for Chapter 4, which will give take-home messages from the whole dissertation, and some leads for the method's improvements.

## 3.1 Preamble: training sets and metrics presentation

### 3.1.1 Training sets

Throughout this chapter, the same training sets can be used for different tasks. A summary table of all those training sets is provided here, with corresponding appendix figure numbers, to access visual depictions. Note that in this chapter I will only consider 'UB' training sets for sake of simplicity, since the performance difference was not significant compared to 'EB' training sets, and these were not compatible with the AMN-Wt method (see section 2.5.2 and appendix B Table S2).

Context	Inputs	Outputs	GEM	Name	Comments	Appendix Figure number
<i>in silico</i> (FBA simulations)	Upper bounds on uptake fluxes ( $V_{in}$ )  OR  Upper and lower bounds on all fluxes (UB, LB)	Growth Rate  OR  All simulated fluxes (including Growth Rate)	<i>E. coli</i> core	<b>core-glucose</b>	Only the glucose uptake flux is varied	C.1
				<b>core-random</b>	13 upper bounds on uptake fluxes are drawn with binomial law (n=13, p=0.5)	
				<b>core-extended</b>	Extends the number and possible values of upper bounds on uptake fluxes compared to 'core-random'	
			iML1515	<b>iML-glucose</b>	Only the glucose uptake flux is varied	n/a
				<b>iML-random</b>	10 upper bounds on uptake fluxes are drawn with binomial law (n=10, p=0.5)	
				<b>iML-extended</b>	Extends the number and possible values of upper bounds on uptake fluxes compared to 'iML-random'	
				<b>iML-singles</b>	One out of 10 uptake fluxes upper bounds is drawn for each training set entry	C.1
				<b>iML-expsim</b>	For each '110GR' medium composition, 10 FBA simulations are performed with different drawings	
<i>in vivo</i> (experimental data)	Medium composition ( $C_{med}$ )	Growth Rate	<b>110GR</b>	Mixed carbon sources (1 to 4) media compositions, with corresponding growth rates	C.2	
	Medium composition ( $C_{med}$ ) and Reaction KO ( $R_{KO}$ )		<b>Cov-BiologKO</b>	17,400 growth rates of 120 metabolic gene KO mutants grown in 145 conditions		
	Medium composition ( $C_{med}$ ) and Regulator gene KO ( $G_{KO}$ )	31 fluxes (including Growth Rate)	<b>Rijs-RegKO</b>	128 sets of 31 fluxes of 64 regulator gene KO mutants grown in 2 conditions		

Table 3.1: Summary table of the training sets used in this chapter. For some training sets, a figure is shown in the appendix C.1 that displays a heatmap of the inputs and outputs, and other metrics (Figure C.1 and C.2). In Table C.1, the location of files containing rules to draw upper bounds on uptake fluxes is available.

### 3.1.2 $R^2$ and $Q^2$

Throughout this chapter, recurrent metrics will be used to evaluate and assess the different performance of AMNs and other models. The most frequently used metric is the coefficient of determination,  $R^2$ . Importantly, in the context of trainable models, the term  $R^2$  is used when computing the coefficient of determination on trained data points *computations*, in contrast to the term  $Q^2$  that is used when the coefficient of determination is computed on untrained data points *predictions*. Note that these 'unseen data' predictions can be aggregated from cross-validation sets, taken from independent test sets, or averaged over several cross-validations. Unless specified otherwise in a figure legend or method section, I always display results from aggregated predictions from validation sets

of a cross-validation. The  $R^2$  or  $Q^2$  metric measures the quality of a linear regression, in our case between the true and prediction data (without a y-intercept). The formula for  $R^2$  (and  $Q^2$ ) is the following equation 3.1.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{i=1}^n (y_i - f_i)^2}{\sum_{i=1}^n (y_i - m)^2} \quad (3.1)$$

With  $SS_{res}$  the residual sum of squares,  $SS_{tot}$  the total sum of squares,  $n$  the number of measures,  $f$  the 'true' data,  $y$  the predicted data, and  $m$  the mean of  $f$ .

### 3.1.3 SV norm, V norm, and Euclidean distance

I will frequently use metrics specifically designed for this chapter, that are meant to quantitatively measure how metabolic fluxes distributions (also termed 'flux vector', denoted  $V$ ) respect GEMs constraints. As stated in Chapter 1 section 1.3.3.1, the SV constraint is the most critical metabolic network constraint. In Chapter 2, we designed a loss term in QP-solver and in the custom loss formulations, in order for AMNs to predict metabolic flux distributions that fulfill this constraint, which is defined as  $L_2$  in Chapter 2's Methods (2.5), section 'Derivation of loss function' (equation 2.2). A slightly tweaked version of this loss term is used in this chapter as a metric to quantify the respect of the SV constraint in metabolic fluxes distributions that are computed by the different models. I term SV norm this metric that can be computed as follows in the equation 3.2, for a flux vector  $V$  and a stoichiometric matrix  $S$  of  $m$  columns (number of metabolites) and  $n$  rows (number of reactions).

$$\frac{1}{m} \|S.V\| \quad (3.2)$$

With  $S.V$  the matrix product of  $S$  with  $V$  and  $\|S.V\|$  the Euclidean norm of this product (square root of the sum of squares). Note that the only difference with Chapter's 2  $L_2$  is that I removed the square. Also, note that the 'Loss constraint' metric used in Chapter 2 and appendix B is not equivalent to SV norm. Indeed, 'Loss constraint' is the sum of  $L_2$ ,  $L_3$  and  $L_4$ , divided by 3, so the stoichiometry is only one of the constraints ensured by 'Loss constraint'.

Another important metric to analyze metabolic flux distributions predicted by AMNs is the V norm, which is simply defined as the Euclidean norm of the flux vector  $V$  normalized by its length, *i.e.* the square root of the sum of squares (of each flux in the flux vector), divided by the length of  $V$  (the number of fluxes in the distribution,  $n$ ). This value quantifies the norm (*i.e.* the 'size') of the flux vector, *i.e.* a rough estimate of how many fluxes are non-zero and how large are the flux values.

Finally, in order to compare two flux vectors, I use in this chapter the Euclidean distance, which is computed as the square root of the sum of squares of the differences between each flux value of the two vectors. This is a very basic and widespread way of quantitatively comparing two vectors. Note that most flux vectors compared with this distance metric are sparse, and the Euclidean distance might not be the best suited distance metric for such vectors. For example, the Wasserstein or cosine distance could be more suited[151].

### 3.1.4 Metrics interpretability

The metrics SV norm and V norm are designed to evaluate the SV constraint respect and the norm of V. The numerical values of such metrics can be hard to interpret. Therefore, the next Figure 3.1 gives examples of how increasing a single flux value of a steady-state flux distribution can change these metrics values, for both *E. coli* core[135] and iML1515[44] GEMs.

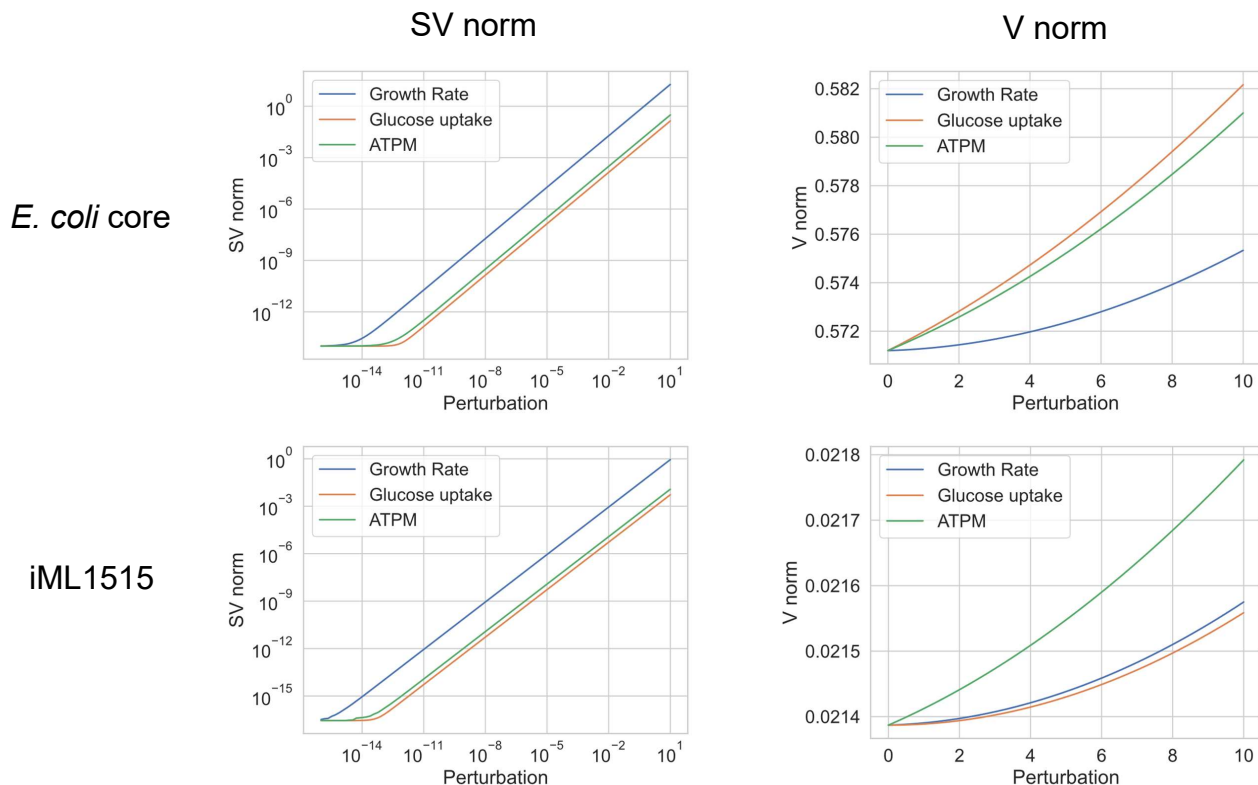


Figure 3.1: SV norm (left panels) or V norm (right panels) metrics response to different flux increases applied on a steady-state flux distribution of either ‘core-random’ (top panels) or ‘iML-random’ (bottom panels). Either the Growth Rate, Glucose uptake or ATPM flux were perturbed with 100 different perturbations (X-axis) and the SV norm or V norm metric (Y-axis) was measured afterwards. The 100 increases applied on a steady-state flux distribution range from  $10^{-16}$  to 10. For SV norm the perturbations were evenly sampled on a geometric space, whereas for V norm the perturbations were evenly sampled on a linear space. SV norm is very sensitive to small perturbations, whereas V norm is relatively robust to large perturbations.

First of all, it is important to notice the magnitude order difference between the metrics computed with *E. coli* core[135] and iML1515[44] models. The selected flux distribution from *E. coli* core[135] has a SV norm around  $10^{-14}$  and a V norm around 0.57, whereas the one from iML1515[44] has a SV norm around  $10^{-17}$  and a V norm around 0.02. These values differences are explained by intrinsic differences between the two metabolic models, probably by the number of fluxes (154 for *E. coli* core[135], 3682 for iML1515[44]). For complete distributions of such metrics, computed for many steady-state flux distributions (here, only one distribution for each model is used), please refer to the training set figures shown in appendix C.1.

Also, Figure 3.1 shows that the SV norm metric is very sensitive to small perturbations: adding  $10^{-4}$  to the glucose uptake of a stationary flux distribution of the *E. coli* core[135] model shifts its SV norm from less than  $10^{-12}$  to more than  $10^{-6}$ . Figure 3.1 also shows that the V norm is relatively robust to large perturbations: adding 10 to the glucose uptake of a stationary flux distribution of the *E. coli* core[135] model shifts its V norm from 0.571 to 0.582. This difference in sensitivity between the two

metrics should be kept in mind to interpret results of this chapter.

## 3.2 An improved QP-solver formulation to better respect GEMs' constraints

### 3.2.1 Introduction

This first section aims to describe technical improvements on the QP-solver method, and investigations of the behavior of such solvers. As a reminder, the reader should keep in mind that these solvers are non-trainable FBA surrogates, they are iterative procedures that should lead to steady-state solutions without any generalizing (*i.e.* learning) abilities. These procedures start with a given initial flux distribution, then iteratively modified to reach a steady-state flux distribution satisfying a GEMs' constraints. In particular, the QP-solver (and its reformulation, QP-bnds-solver) rely on minimizing a loss function by gradient descent. Importantly, this gradient descent is not related to any learning procedure, it is solely modifying the flux distribution to minimize the loss function (there are no parameters, the gradient is derived directly from the flux distribution).

Developing surrogates of FBA (*cf.* Figure 2.1 panel b) was a core research work of the AMNs development. Indeed, we designed AMNs based on the idea to include FBA surrogates as a process inside a ML architecture. However, these solvers were designed specifically for the study, and their behavior is not extensively studied in the previous chapter. Therefore, I will focus in this section on such mechanistic solvers, especially on the QP-solver which was found to be incomplete.

The QP-solver method (*cf.* appendix B, section 'QP-solver equations') is based on several terms, each designed with a particular purpose: a fitting term with  $L_1$ , a term to encourage the respect of  $SV=0$  with  $L_2$ , a term to encourage the respect of uptake fluxes upper bounds with  $L_3$ , and a term to encourage flux positivity with  $L_4$ . The QP-solver then operates by gradient descent of the sum of  $L_1$ ,  $L_2$ ,  $L_3$  and  $L_4$  terms. The issue with such formulation is two-fold: (i) it does not respect non-zero lower bounds, such as the ATPM reaction, and (ii) it does not respect upper bounds on other fluxes than the uptake fluxes. For example, a gene KO (encoded by an upper bound of zero on a flux) cannot be directly simulated with such formulation. These are clear issues that should be tackled. For that, I reformulated the QP-solver method, termed QP-bnds-solver for emphasizing the fact that it better respects the boundary constraints.

Moreover, the QP-solver relies on a tradeoff (*i.e.* a 'concurrency') of the different terms that are summed in the final loss to be minimized. This tradeoff is yet unexplored: in this section I will attempt to optimize it by tuning weights applied on each term of the loss to compute the final loss value, and observe the relationships between weights and their effect on solvers' behaviors. This is a basic approach to tune a multi-objective optimization problem, which is an active research field with more cutting-edge methods[152, 153].

### 3.2.2 Methods

#### 3.2.2.1 QP-solver reformulation

The original QP-solver formulation is defined in appendix B., section 'QP-solver equations'. The terms found to be problematic in this formulation are terms  $L_3$  that account only for uptake fluxes (and not

all fluxes) upper bounds to be respected, and  $L_4$  that account only for flux positivity (and not for non-zero lower bounds) respect. Therefore, I introduce two new loss terms, namely  $L_6$  (replacing  $L_3$ ) and  $L_7$  (replacing  $L_4$ ) that are respectively accounting for all upper bounds respect, and all lower bounds respect.

$$L_6 = \frac{1}{n} \|ReLU(V - UB)\|^2 \quad (3.3)$$

$$L_7 = \frac{1}{n} \|ReLU(LB - V)\|^2 \quad (3.4)$$

With  $UB$  and  $LB$  two vectors of the same size as  $V$ , respectively containing all upper and all lower bounds (see equation 1.2). From these two new terms, I formulate a new minimization performed by the QP-bnds-solver, in equation 3.5:

$$\min (L_1 + L_2 + L_6 + L_7) \quad (3.5)$$

To formulate QP-bnds-solver, we follow a similar procedure for this minimization in equation 3.5, as in the original QP-solver formulation for minimizing equation (S15) in appendix B. This yields the following gradient terms for  $L_6$  and  $L_7$ , to be used in QP-bnds-solver:

$$\nabla V_6 = -\frac{2}{n} D_{UB} ReLU(V - UB) \quad (3.6)$$

$$\nabla V_7 = -\frac{2}{n} D_{LB} ReLU(LB - V) \quad (3.7)$$

$$D_{UB} = \frac{ReLU(V - UB)}{ReLU(V - UB)}; D_{LB} = \frac{ReLU(LB - V)}{ReLU(LB - V)} \quad (3.8)$$

With  $D_{UB}$  and  $D_{LB}$  using the Hadamard division, as in appendix B equation (S22).

### 3.2.2.2 Optimization of solvers' terms weighting

To optimize the solvers (both QP-solver and QP-bnds-solver), I attempt to apply different weights on their different loss terms. These weights are multiplying the values of each term prior to the gradient computation, *i.e.*, on equation (S15) of appendix B for the original QP-solver, and on equation 3.5 for the QP-bnds-solver. For more clarity, different names are given to the weights applied on losses, according to the table 3.2 below, with the ranges of values they were allowed to be sampled in.

Loss term	Purpose	Weight name	Weight range
$L_1$ ('Loss fit')	Fit reference fluxes	'Loss weight (fit)'	0.01 to 0.9
$L_2$	Satisfy $SV=0$	'Loss weight (SV)'	0.01 to 0.9
$L_3$	Ensures uptake fluxes upper bounds	'Loss weight (Vin)'	0.01 to 100.0
$L_4$	Ensures flux positivity	'Loss weight (Vpos)'	0.01 to 100.0
$L_6$	Ensures all upper bounds	'Loss weight (UB)'	0.01 to 100.0
$L_7$	Ensures all lower bounds	'Loss weight (LB)'	0.01 to 100.0

Table 3.2: QP-solver and QP-bnds-solver loss terms, associated purpose, name of weights applied on different terms, and corresponding range searched during optimization of the solvers.

When testing the solvers, I observed that loss terms accounting for upper and lower bounds ( $L_3$ ;

$L_4$ ;  $L_6$ ;  $L_7$ ) were showing small values and tweaking their weights was not significantly changing the solvers results. In contrast, I observed that loss terms accounting for fitting reference fluxes ( $L_1$ ) and stoichiometric constraints ( $L_2$ ) were showing much higher values and tweaking their weights had a clear impact on the solver's results. The values of loss terms directly act on each iteration of the solver, since the loss is minimized based on such values, making terms with higher values more important for the direction that a solver takes. Therefore, I divided the weights search into two independent searches, based on the assumption that solvers' loss terms are subjected to two distinct tradeoffs: the fitting term  $L_1$  and the SV term  $L_2$ , which are the two most important terms of the solvers; and the upper bounds term  $L_3$  or  $L_6$  and the lower bounds term  $L_4$  or  $L_7$ , which are less important terms of the solvers.

The framework to optimize the weights applied on loss terms of the solvers is based on the optuna package[154]. It was set to search the best set of weights to apply on loss terms, with a two-objectives optimization: (i) the minimization of the Mean Squared Error (MSE) between reference and computed fluxes (here only the growth rate), *i.e.* the  $L_1$  value, called 'Loss fit'; and (ii) the minimization of the SV norm metric. The two-objectives optimization was performed with default optuna's method, *i.e.* the Tree-structured Parzen Estimator (TPE)[155]. It relies on Gaussian Mixture Models, fitted with the weights applied on loss terms as independent variables, and both objective values 'Loss fit' and SV norm as dependent variables, in order to efficiently sample the space of weights. 10 conditions from the 'core-glucose' training set were randomly picked. The solvers were then runned for 10,000 iterations on those 10 conditions. This was repeated 500 times to search for the best set of weights.

The first search explores the weight values for 'Loss weight (fit)' and 'Loss weight (SV)'. Results for that approach are displayed in Figure 3.2. The second search explores the weight values for 'Loss weight (Vin)' or 'Loss weight (UB)' and 'Loss weight (Vpos)' or 'Loss weight (LB)'. Results for that approach are displayed in the appendix C.3, Figure C.3.

### 3.2.2.3 Comparison of QP-solver and QP-bnds-solver for respecting GEMs' constraints

In order to compare how QP-solver and QP-bnds-solver respect GEMs' constraints, I used FBA as reference data, with 10 randomly sampled entries of the 'core-glucose' training set. For each training set entry, the upper and lower bounds of the FBA problem were used to derive loss terms for both QP-solver and QP-bnds-solver. I remind that the 'core-glucose' training set has only its glucose uptake flux varied between different entries. All other bounds are set to the default value, except for a series of minimal uptake fluxes ensuring positive growth computation by FBA (see Chapter 2 Methods 2.5, section 'Generation of training sets with FBA' 2.5.2).

## 3.2.3 Results and discussion

### 3.2.3.1 Tradeoff between the fitting and SV loss terms

First of all, we observe on Figure 3.2 similar performances for both formulations, in terms of fit on fluxes and respect of stoichiometric constraints. Strikingly, we observe a clear tradeoff operating between the  $L_1$  and  $L_2$  terms, that is visible as a Pareto-front-like frontier. This frontier seems mostly influenced by weight ratios. A ratio of 1 between the 'Loss weight (fit)' and 'Loss weight (SV)' seems optimal, as it shows an equilibrium between the two terms (purple points, with a 'Loss fit' around 0.4 and a SV norm around 0.6). The best value for those two weights was found to be 0.8.

A similar figure was generated for the other loss terms: 'Loss Vin' and 'Loss Vpos' (in the original

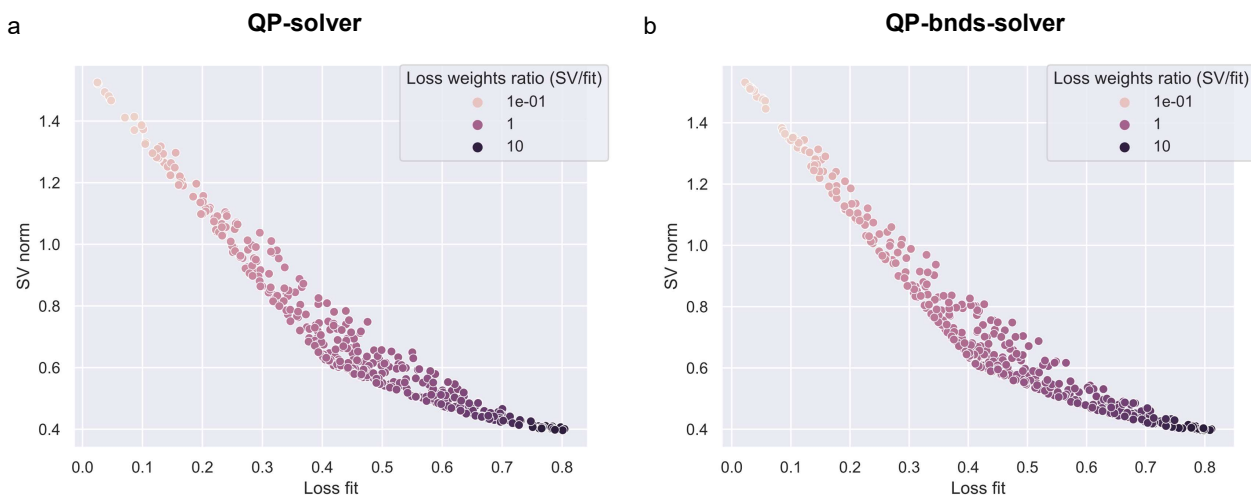


Figure 3.2: Pareto plots of QP-solver (a) and QP-bnds-solver (b) show the solvers' tradeoff between satisfying stoichiometric constraints ( $SV=0$ ) and fitting the reference fluxes. For both plots, each of the 500 points displays mean metrics for 10 conditions of the 'core-glucose' training set, computed after a mechanistic method run of 10,000 iterations. Plots show unweighted 'Loss fit' values on the X-axis, and unweighted SV norm values on the Y-axis. Each point's color corresponds to the ratio between the 'Loss weight (SV)' and the 'Loss weight (fit)'. Both solvers behave similarly, showing a clear pareto-like frontier between the 'Loss fit' and the SV norm when varying the weights applied on loss terms accounting for fitting reference fluxes and satisfying stoichiometric constraints.

QP-solver); or 'Loss LB' and 'Loss UB' (in QP-bnds-solver). Briefly, results show that weights ratios of 1 are yielding satisfactory performances, but not very clear trade offs were found like in Figure 3.2. The results can be found in appendix C.3.

### 3.2.3.2 No significant improvement of the MM solver by loss terms weights optimization

Importantly, I splitted in two the searches for optimal weights applied on the loss terms; based on an assumption, and also for the sake of clarity. But in practice, all of the terms compete with each other. Moreover, changing solvers' parameters (e.g. iterations number, gradient descent rate, selected conditions to solve) further changes the behavior of the solvers. That makes QP-solver and QP-bnds-solver hardly fine-tuned for all conditions and setups. In other words the optimization approach presented here is quite limited to the particular scope in which it is performed (*i.e.* what are the conditions to solve, with how many iterations at which gradient descent rate). If that scope is changed, the optimization approach should be performed again.

Overall, I could not make any significant improvement on the overall performance of the solvers, between previously used weights on loss (all at 1) and the weights found with the optimization approach. However, we have shown here that QP solvers rely on a subtle tradeoff between loss terms and that should be kept in mind for further research on such solvers and for the remainder of the dissertation. This is especially important when considering that the QP-solver formulation inspired the custom mechanistic loss of AMNs.

### 3.2.3.3 The QP-bnds-solver enhances the respect of GEMs' constraints

The new formulation QP-bnds-solver does not display better performance in terms of fitting fluxes and respecting the SV constraint. However, the following results will demonstrate how the QP-bnds-

solver outperforms the original QP-solver: it is making more reliable computations in terms of GEMs' constraints respect. This is shown in the next Figure 3.3, and in the appendix C.3, with Figure C.4 that shows data heatmaps to visualize extended results (for other fluxes than those shown in Figure 3.3).

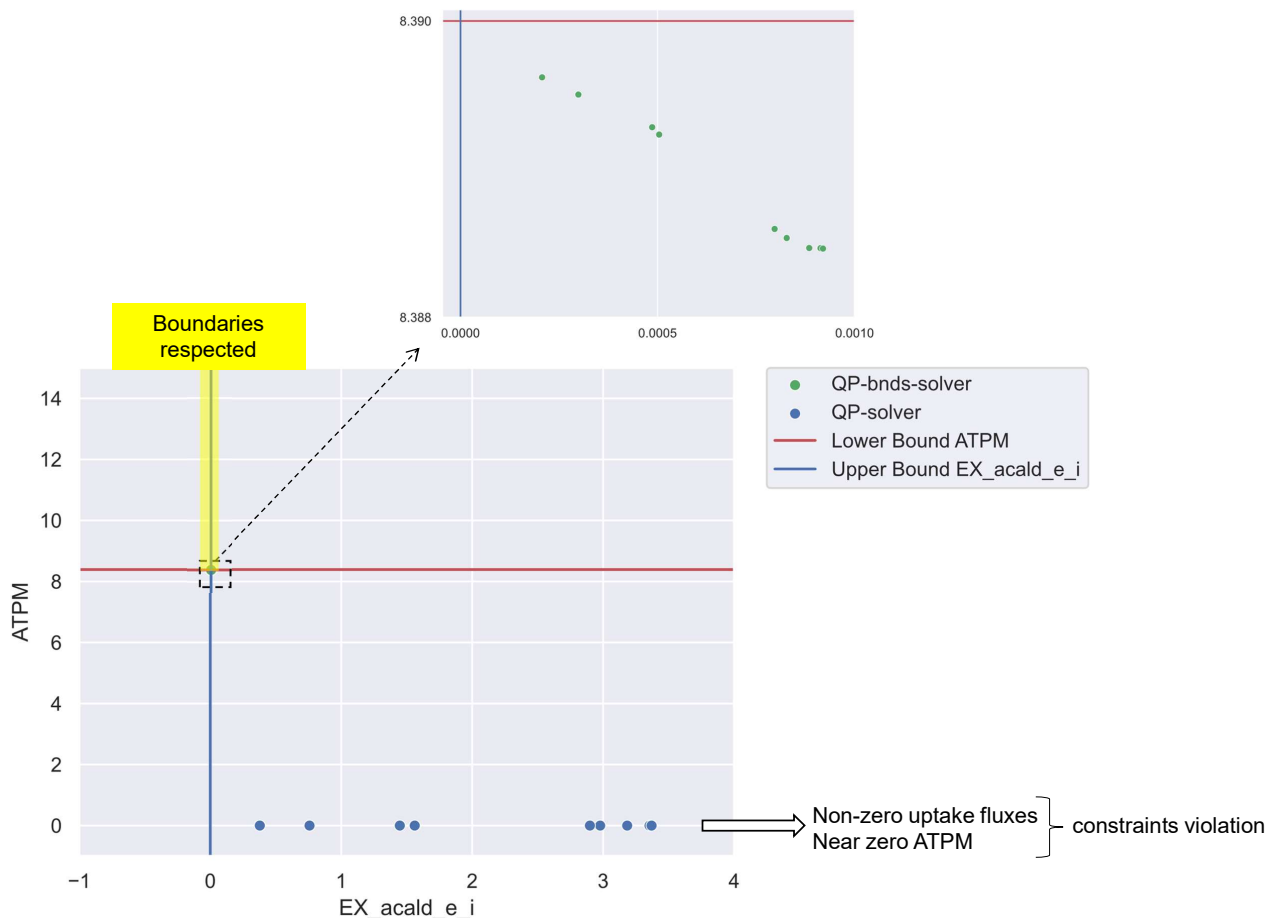


Figure 3.3: QP-bnds-solver better respects GEMs constraints. The red line represents the lower bound of ATPM and the blue line represents the upper bound of the acetaldehyde uptake flux (EX\_acald\_e\_i). The allowed region from these two bounds constraints (whether it is for FBA, QP-solver and QP-bnds-solver solutions) is highlighted in yellow. Blue points corresponding to QP-solver solutions show near zero ATPM fluxes and non-zero acetaldehyde uptake fluxes, which violates constraints. By zooming in the dotted black box, one can better observe the dispersion of green points, corresponding to QP-bnds-solver solutions, which are not in the allowed region but extremely close to it.

Figure 3.3 displays the ability of QP-bnds-solver to take into account both the non-zero lower-bounds (ATPM here) and respect all upper bounds, where the original QP-solver fails. QP-solver shows acetaldehyde uptake flux (EX\_acald\_e\_i) values that should be at zero having values between 0.3 and 3.4, and ATPM values near zero. In contrast, QP-bnds-solver shows acetaldehyde uptake flux near zero, and ATPM values close to its lower bound constraint of 8.39. Note that the upper bounds respect difference between the two solvers is only shown for the EX\_acald\_e\_i flux here, which was picked as an uptake flux that QP-solver does not respect. Extended results with all uptake fluxes are available in Figure C.4.

Importantly, one should note that in chapter 2, we always used QP-solver in a context where it did not respect ATPM lower bounds. However, it did respect the upper bounds of uptake fluxes, since we always considered all uptake fluxes in  $V_{in}$ . Here, by using the 'core-glucose' training set, I consider FBA data with only glucose and essential uptake fluxes in  $V_{in}$ . This makes the QP-solver formulation fall into its flaw, to show that QP-bnds-solver is more versatile and general.



We have assessed that an alternative formulation of the QP-solver can be formulated, with improved respect of constraints. However, the method seem to rely on subtle tradeoffs between the different loss terms, which seem hardly fine-tuned, and they still require very large ( $>1e5$ ) iterations number to perform well (*cf.* appendix B, figure S6e).

### 3.2.3.4 AMN-QP-bnds formulation better respects GEMs constraints and is more versatile

In light of previous results, using the QP-bnds-solver appears preferable than the original QP-solver. In Chapter 2, we use the QP-solver as a mechanistic layer of AMN-QP; but also as a foundation for the custom mechanistic loss common to all AMNs formulations, given in equation 2.2. The same issues of QP-solver mentioned above are expected in AMNs, which should be tackled.

Therefore, I also formulate AMN-QP-bnds, which differs from AMN-QP in two ways: (i) the mechanistic layer is QP-bnds-solver instead of QP-solver; (ii) the custom loss is derived from QP-bnds-solver instead of QP-solver. This derivation is identical to what was done with the original custom loss, one can refer to Chapter 2 Methods 2.5, section 'Loss functions derivation' for more details. In short, the original mechanistic loss  $L$ , given in equation 2.2 is reformulated in the following equation of the new mechanistic custom loss  $L_{bnds}$ , used in AMN-QP-bnds:

$$L_{bnds} = L_1 + L_2 + L_6 + L_7 \quad (3.9)$$

These changes also bring important improvements in the workflow of AMN-QP-bnds, which is displayed in the following figure 3.4.

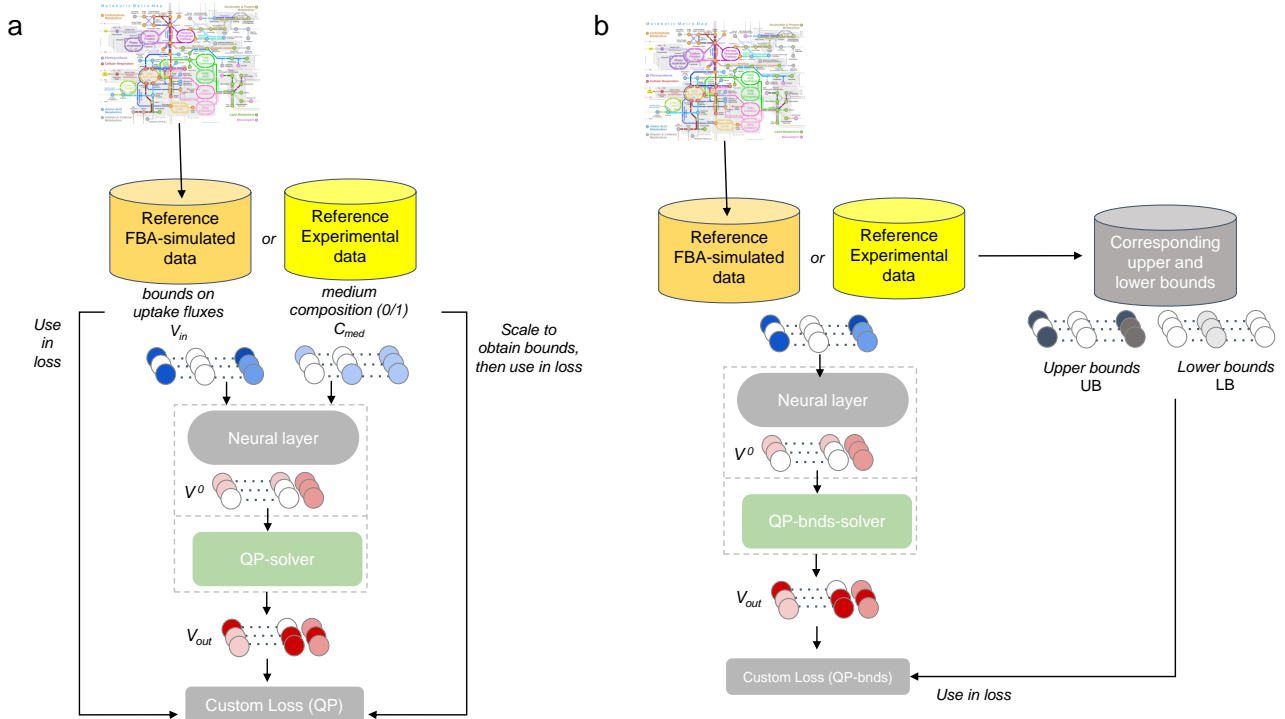


Figure 3.4: Improved workflow of AMN-QP-bnds compared to AMN-QP. The metabolic network schematic indicates the use of a GEM at a given step of the workflow. (a) Workflow of the AMN-QP formulation presented in Chapter 2, showing with the black arrows on the sides how to obtain the upper bounds on uptake fluxes from the inputs, to be used in the custom loss term  $L_3$  (based on the QP-solver). (b) Workflow of the AMN-QP-bnds formulation, showing how the upper and lower bounds for all fluxes can be derived from any kind of data, in order to be used in the custom loss terms  $L_6$  and  $L_7$  (based on the QP-bnds-solver).

Figure 3.4 shows how the workflow of the AMN-QP-bnds (panel b) improves the original one (panel a): the input is not anymore restricted to be either  $V_{in}$  or  $C_{med}$ . Therefore, AMN-QP-bnds can handle 'Cov-BiologKO' and 'Rijs-RegKO' datasets more easily, without changes to the custom loss (like what was done for AMN-QP to integrate reaction KOs, cf. Chapter 2 Methods 2.5, section 'Loss functions derivation', term  $L_5$ ), and without changes to the overall data flow of the model. However, even though any kind of data can be used as input, an extra pre-processing step has to be performed with AMN-QP-bnds in order to derive the corresponding upper and lower bounds for each entry of the dataset. Note that these can be the default lower and upper bounds of the GEM, or automatically generated bounds with scripts available in the github repository (cf. appendix C.2). Note that the AMN-QP-bnds workflow improvement could be made with AMN-LP and AMN-Wt by changing the custom loss formulation in these models. But it will not be done in the context of this dissertation, where I will keep the original formulations, for more consistency with the previous chapter's results.

When including QP-solver and other mechanistic solvers inside AMNs, we had to keep the number of iterations relatively low, enabled by adding a dense neural layer prior to the solver (which is then termed mechanistic layer). This approach raises important questions on how the mechanistic layer behaves when included in a learning model, which is a rather innovative approach. The next section will attempt to investigate such behavior.

## 3.3 How much do mechanistic layers improve AMN predictions?

### 3.3.1 Introduction

This second section aims to investigate the effect of increasing iterations in mechanistic layers on AMNs prediction performance. As a reminder, AMNs are learning models that have a neural layer predicting the best input for a mechanistic layer, based on a custom mechanistic loss. In this section, AMNs output full flux distributions, with upper bounds on uptake fluxes as input. Growth rates found in FBA simulations were used as reference. In other words, from a biological standpoint, AMNs generalize from a set of upper bounds on uptake fluxes (independent variables) and corresponding growth rates (dependent variable), to predict full metabolic flux distributions of a GEM.

In order to build AMNs, mechanistic layers were used together with ANNs. The reason underlying this methodological choice was three-fold: (i) reduce the number of iterations needed for mechanistic layers to perform well, by predicting an input for mechanistic layer with ANNs, (ii) enable AMNs to generalize over a set of conditions, with learning procedures, in contrast to FBA and other mechanistic solvers that are performed independently with each condition, and (iii) ANNs can pass a gradient, so they can be directly connected to mechanistic layers. Such integration raises important questions, because the behavior of mechanistic layers changes when included in learning architectures (mechanistic layers are limited in iterations, and their initial solution is predicted by ANNs).

Moreover, when the QP-solver (or QP-bnds-solver) are integrated in AMN-QP (or AMN-QP-bnds) as a mechanistic layer, they lose the term for fitting reference fluxes, and are only based on remaining loss terms, which further changes the behavior of the solver. Since the solvers performances have been assessed with the fitting term, their behavior is rather unknown without such fitting term. This removal of the fitting term had to be made, otherwise the predictions would need the reference data

to be performed, which prevents the model from being predictive. Importantly, note that AMN-LP and AMN-Wt do not require such a tweak to the mechanistic layer.

In the previous chapter, we did not assess the dependence of AMNs performance on their mechanistic layers. Indeed, important questions may arise from using such an approach: (i) how much the mechanistic layers improve the prediction performance in terms of fitting reference fluxes and satisfying the GEM constraints?, (ii) do mechanistic layers make the AMNs predictions closer to what FBA could predict, for what computational cost?, and (iii) how much does a mechanistic layer transform its input, in other words how close the ‘first guess’  $V^0$  is to the final prediction  $V_f$ ?

In this section I will attempt to answer these questions, by measuring AMNs performances with increasing iteration numbers of the different mechanistic layers.

### 3.3.2 Methods

#### 3.3.2.1 Training set and models architectures

The training set used in this section is ‘core-random’ in all cases. Please refer to the Preamble (section 3.1) of this chapter for more information.  $V_{in}$  (uptake fluxes upper bounds) was always used as input, the growth rate was always used as output (*i.e.* it was the only reference flux to fit).

Three AMN formulations were tested, AMN-QP-bnds, AMN-LP, and AMN-Wt. These differ in their mechanistic layers, and AMN-QP-bnds makes use of a different custom loss as presented in Figure 3.4. All architectures had the following neural layer parameters 1 hidden layer of size 100 with ReLU activation function and 0.2 of dropout rate. In all cases, a 10-fold cross-validation with 5 epochs was performed, and aggregated predictions from the validation sets were used to display performance results. The optimizer was always ‘adam’, with a batch size of 5.

#### 3.3.2.2 Varying mechanistic layers iterations

For AMN-QP-bnds and AMN-LP, the number of iterations in the mechanistic layers tested were 0, 4, 32 and 128. For AMN-Wt, the number of iterations tested were 1, 4, 32 and 64. For technical reasons, the AMN-Wt architecture could not be run with 0, nor with 128 iterations (the model training made python crash).

Most metrics displayed in this section are described in the Preamble (3.1):  $Q^2$ , SV norm, V norm, and the Euclidean distance between  $V_0$  and  $V_f$ . In this section I also describe the effect of increasing iterations on computation time, by simply indicating the computational time, in seconds, required for the whole 10-fold cross-validation to be performed. This computational time was obtained on an Intel Core i7-7500U CPU, paced at 2.7 GHz.

### 3.3.3 Results and discussion

#### 3.3.3.1 The mechanistic layer improves predictions of AMNs, with a large advantage for AMN-LP

A striking effect of mechanistic layers on AMNs predictions was found to be the SV norm reduction, alongside an increase in computational time, as shown in Figure 3.5. All AMNs consistently reduce the SV norm found in predictions by increasing the number of iterations in the mechanistic layer. However, AMN-LP has a clear advantage over others: the magnitude order of SV norm is around  $10^{-5}$  with 4 iterations of the mechanistic layer, whereas AMN-QP-bnds and AMN-Wt respectively reach

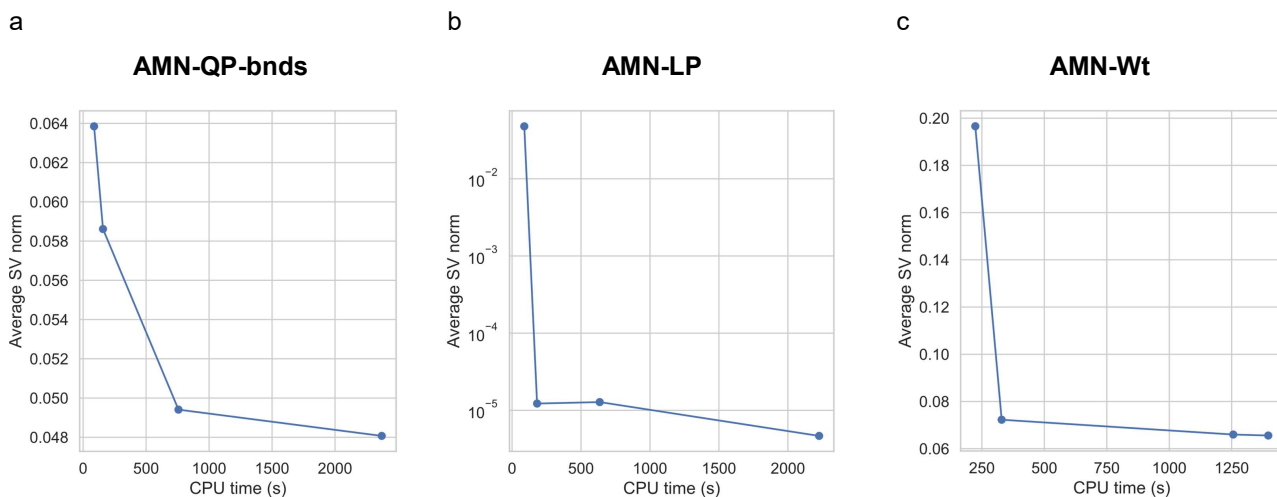


Figure 3.5: Increasing number of iterations in the mechanistic layers of AMNs reduces the SV norm but increases computational time. The SV norm was averaged over all predictions of an AMN, and the CPU time is expressed in elapsed seconds for the 10-fold cross-validation to be performed (see Methods). The three panels display results for AMN-QP-bnds for 0, 4, 32 and 128 mechanistic layer iterations (a), AMN-LP for 0, 4, 32 and 128 iterations (b) and AMN-Wt for 1, 4, 32 and 64 iterations (c). The AMNs differ by their mechanistic layer, based on different solvers. Looking at the SV norm values obtained for the different AMNs, a large advantage goes to AMN-LP, reaching SV norms around  $10^{-5}$  with roughly 150 seconds to perform the 10-fold cross-validation.

minimal SV norms of 0.048 (128 iterations) and 0.066 (64 iterations). This advantage of AMN-LP over other AMNs is more striking on the appendix figure C.6, which displays the same results as Figure 3.5 on a single plot.

Other measures were performed with increasing mechanistic layer iterations, notably observing a consistent increase of the V norm, and no significant effect on the growth rate  $Q^2$ . These results are shown and discussed in the appendix C.5.

Overall, we can conclude that the different mechanistic layers inside AMNs yield different behaviors, for AMN-LP, -QP-bnds and -Wt. In AMN-LP, increasing the mechanistic layer iterations has a strong effect on SV norms, and this effect is drastically smaller with AMN-Wt and AMN-QP-bnds. Therefore, the increase in computational time brought by the mechanistic layer seems to bring a significant advantage to AMN-LP only. Moreover, current implementations have limits which will be discussed next.

### 3.3.3.2 Current implementations have strong limits

The first limitation of mechanistic layers is the significant increase in computational time with increasing iterations, which should be kept in mind when considering the possible performance improvement brought by high iterations numbers. In the appendix C.4, I show that AMN-Wt's computation time logarithmically increases with mechanistic layer iterations, whereas AMN-LP and AMN-QP-bnds' computation time linearly increases with iterations. Note that this effect is amplified when using larger metabolic models, such as iML1515[44]. Importantly, using any mechanistic layer with iML1515[44] was found to be computationally too expensive for all tested AMNs (making python crash for unresolved technical issues), which is a very strong limit of the current mechanistic layers implementations. In Chapter 2 (Figures 2.3 and 2.4), we display results with AMNs that have a tweaked mechanistic layer in order to function with iML1515[44], by lowering the solvers' rates and using a maximum of 4 iterations. This makes mechanistic layers very slightly modifying  $V^0$  to produce  $V_f$ .

AMNs can be tuned in many ways, not only with increasing mechanistic layer iterations as it was done here. For example, the custom mechanistic loss and the neural layer can be tuned to obtain different model performances, without modifying the mechanistic layer. In the next section, I will present hyperparameter optimization results, by fine-tuning the neural layer and the custom mechanistic loss to enhance AMNs performances.

## 3.4 Fine-tuning AMNs by hyperparameter optimization

### 3.4.1 Introduction

This third section aims to investigate the effect of neural layer hyperparameters and custom loss terms weighting on AMNs prediction performance. The investigation is made through hyperparameter optimization, which also brings technical improvements. In this section, AMNs do not have a mechanistic layer so the neural layer directly predicts the flux distribution. This was decided in light of the previous section, to reduce computational time and avoid incompatibilities with large GEMs. Also in the present section, AMNs output full flux distributions, with upper bounds on uptake fluxes or medium compositions as inputs. Growth rates found in FBA simulations or experimental measurements were used as reference. In other words, from a biological standpoint, AMNs generalize from a set of *in silico* or *in vivo* conditions (independent variables) and corresponding growth rates (dependent variable), to predict full metabolic flux distributions of a GEM.

As stated in Chapter 1 (section 1.4.1.3), ANNs provide a large flexibility in terms of architecture and learning parameters. Indeed, one can select different layer types, of different sizes, with different activation functions. Moreover, the model can be trained with different learning parameters (solver, batch size, epochs) and loss formulations, for example applying a different set of weights on the loss terms. With AMNs, we also have this freedom of tweaking the ANN part and the loss of the model, which will be explored in this section.

In the present section, I vary the architecture ANNs to use as neural layers of AMNs, as well as weights applied on the custom loss terms, using a two-objective hyperparameter optimization framework built with optuna[154]. This framework is searching either ANN architectures or weights applied on the custom loss terms, in order to optimize both the respect of stoichiometric constraints and the fitting performances of AMNs.

### 3.4.2 Methods

#### 3.4.2.1 Training sets and models architectures

The training sets used in this section are 'core-random', 'iML-singles', and '110GR'. Please refer to the Preamble section (3.1) of this chapter for more information.  $V_{in}$  (upper bounds on uptake fluxes) or  $C_{med}$  (media compositions) was used as input, the Growth Rate (measured after FBA or experimentally) was always used as output (*i.e.* it was the only reference flux to fit).

AMN-QP-bnds was always used, without a mechanistic layer. All architectures had varying neural layer hyperparameters and weights applied on the custom loss (the space of possibilities will be described in the next subsection). The optimizer was always 'adam', with a batch size of 10.

### 3.4.2.2 Hyperparameters space and optimization framework

The default hyperparameters of neural layers (before optimization) were the following: 1 hidden layer of size 100 with ReLU activation function and 0.2 of dropout rate. The space of possible architectures to search for during hyperparameter optimization is defined in Table 3.3.

The default weights applied on the custom loss terms (before optimization) were all 1, applied on  $L_1$ ,  $L_2$ ,  $L_6$  and  $L_7$ . Ranges searched during hyperparameter optimization are given in Table 3.3. Importantly, note that the weights are applied on the custom mechanistic loss terms used to train neural layers, and they are not applied on a mechanistic layer loss terms (no mechanistic layer is used in the section).

The following table 3.3 gives an overview of the possible values for the different hyperparameters searched during the optimization. All these different ranges come from manual testings ensuring they do not produce aberrant results.

Type	Name	Range
Neural layer	Hidden layer number	0 to 5
	Hidden layers sizes	10 to 150
	Activation functions	ReLU or tanh
	Dropout rates	0 to 0.5
Mechanistic loss terms weights	Weight applied on $L_1$	1 to 1
	Weight applied on $L_2$	0.1 to 10
	Weight applied on $L_6$	0.01 to 10
	Weight applied on $L_7$	0.01 to 10

Table 3.3: Hyperparameters for optimization of either the neural layer or the mechanistic loss terms weighting, with each hyperparameter name and corresponding range searched during the optimization process.

Optuna's hyperparameter optimization was first targeted at searching the best neural layers architectures with default weights applied on the custom loss terms. Then, using the best architecture found, it was runned again to search the best weights applied on the custom loss terms. Both of the hyperparameters optimization were screening 100 combinations, each running a 10-fold cross-validation with 50 epochs, retrieving aggregated predictions from the validation sets to compute the metrics. For both searches, a two-objective optimization was performed, to (i) maximize the  $Q^2$  and (ii) minimize the custom loss. I always used optuna's default Tree-structured Parzen Estimator (TPE)<sup>[155]</sup> as the hyperparameter space sampler (described in section 3.2.2.2).

## 3.4.3 Results and discussion

### 3.4.3.1 Optimization of neural layer hyperparameters and weights for custom loss terms can lead to better AMN performance

On Figure 3.6, one can observe consistent improvements of AMN-QP-bnds' performances by hyperparameter optimization. For 2 out of the 3 tested training sets ('core-random' and 'iML-singles'), panel a displays increased performance between the green point (original neural layer) and the red point (optimized neural layer), with higher  $Q^2$  and lower custom loss. Also, searching for the best set of weights to apply on the custom loss terms further increased performance, displayed by yellow points of panel b.

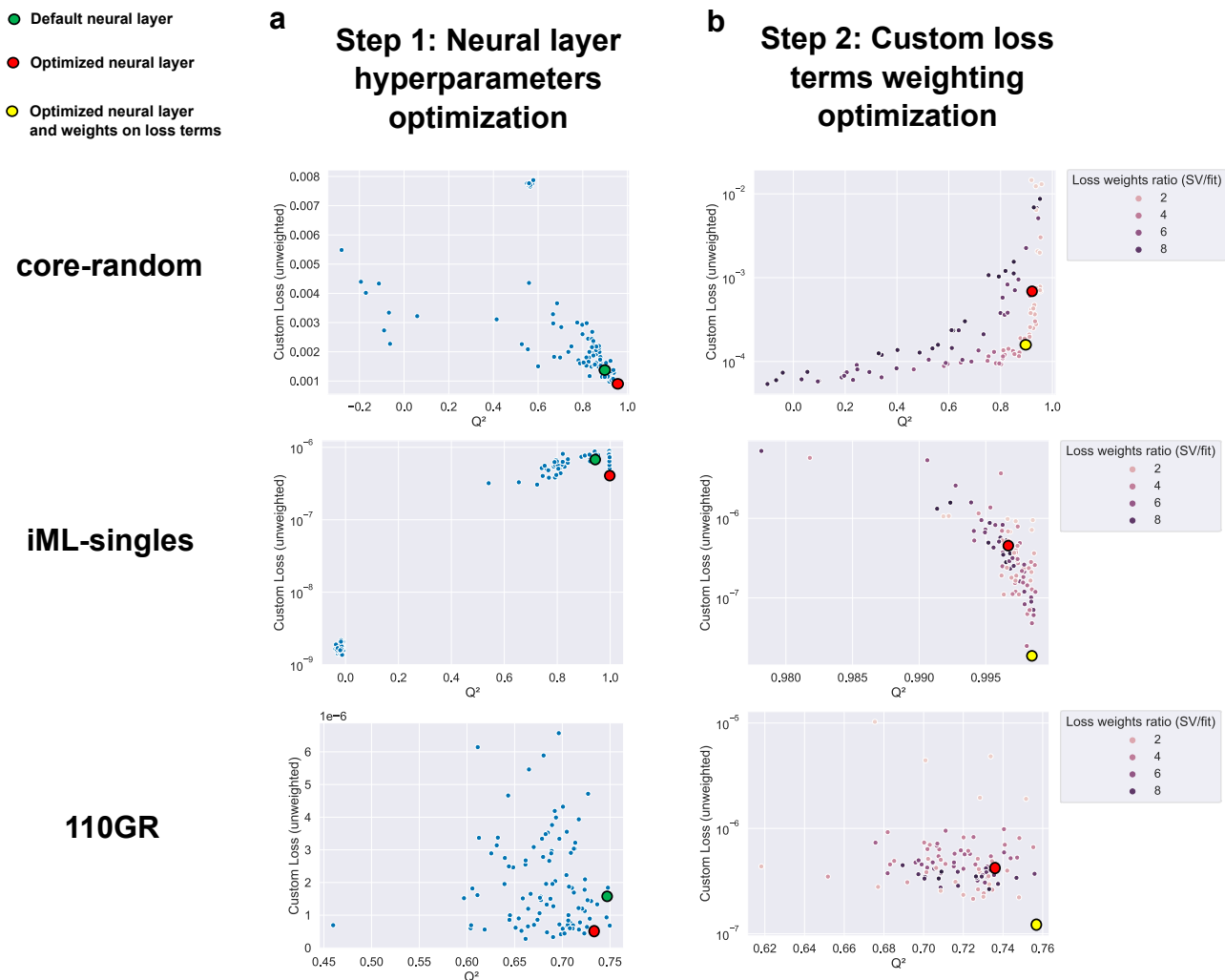


Figure 3.6: Hyperparameter optimization of the neural layer and custom loss terms weighting of AMNs. Each plot displays all trials of a hyperparameter optimization run, with the  $Q^2$  on the growth rate (X-axis) and the unweighted custom loss value (Y-axis). Each panel (a or b, the columns) displays the hyperparameter optimization results for 3 different training sets, from top to bottom: 'core-random', 'iML-singles', and '110GR'. Each point represents a 'trial' of the hyperparameter optimization, sampling a neural layer architecture in step 1 (a) or a set of weights for the custom loss terms (b). Large green points display the AMN performance with the original 'default' neural layer, and large red points indicate the performance with the best neural layer found (with high  $Q^2$  and low custom loss), with all weights on the custom loss terms at 1. Large yellow points display the AMN performance with the best neural layer found *and* the best set of weights applied on custom loss terms. This point is considered best, which yields the 'Optimized' hyperparameters for AMN-QP-bnds in the next Figure 3.7. (a) First step of the hyperparameter optimization, searching for the best neural architecture of AMN-QP-bnds (without mechanistic layer). (b) Second step of the hyperparameter optimization, searching for the best set of weights to apply on the custom loss terms of AMN-QP-bnds. The points are colored according to the ratio between the weight applied on the SV term ( $L_2$ ) and the fitting term ( $L_1$ ) of the custom loss. A general observation with this figure is the consistent performance increase brought by hyperparameter optimization with AMNs.

For easier comparison of AMN-QP-bnds' performance before and after hyperparameter optimization, the next figure 3.7 shows a simple bar chart of the  $Q^2$ , SV norm, and V norm.

To access details of which neural architectures and set of weights were found for each training set, and extended results comparing the default and optimized performance of AMN-QP-bnds, please refer to the appendix C.5.



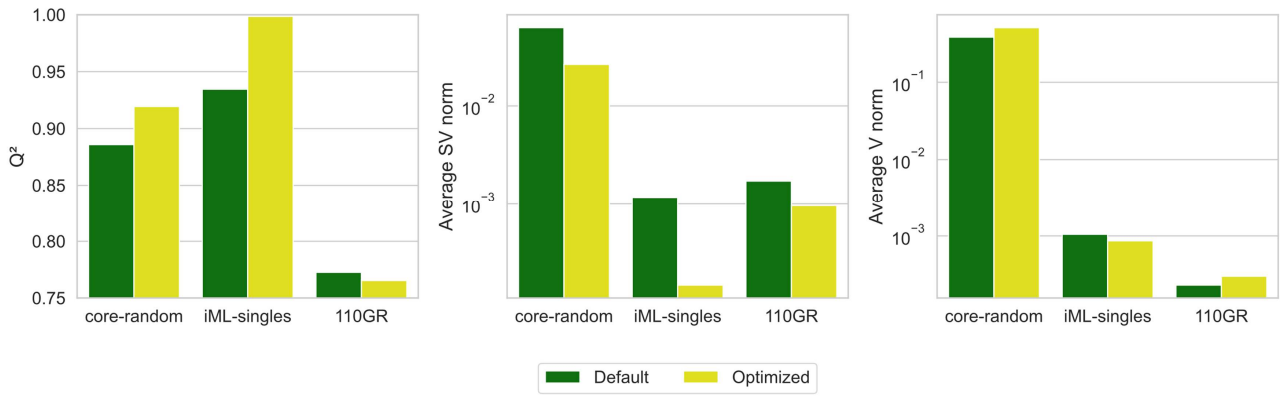


Figure 3.7: Comparison of AMN-QP-bnds' performance before ('Default', in green) and after ('Optimized', in yellow) hyperparameter optimization is performed. Each bar plot shows results for the 3 training sets tested ('core-random', 'iML-singles', '110GR'). From left to right, the metrics used for the different bar plots are the  $Q^2$  computed on the growth rate, the average SV norm, and the average V norm over all predictions. The  $Q^2$  was improved in two out of three cases (increasing values); the SV norm was always improved (decreasing values); no clear effect on V norm was observed.

### 3.4.3.2 Hyperparameter optimization depends on the training set

Added to the increased performance of AMNs after hyperparameter optimization, which is a technical improvement, the presented results can bring interesting observations. Different training sets not only yield different AMN performances, but they will also be associated with different hyperparameters optimal values. For example, we observe different effects of varying weights on the custom loss terms (Figure 3.6 panel b): with the 'core-random' training set, one can observe some tradeoff between the custom loss and  $Q^2$  values; whereas for the 'iML-singles' and '110GR' training set, one can observe a clear and single optimum found with a single set of weights. This underlines that different training sets will yield different optimal neural layer architectures and weighting schemes applied on loss terms. This can be considered both as an advantage that enables one to fine-tune an AMN for a particular purpose, but also a drawback as it requires one to search the hyperparameter space for each training set. These points will be further discussed in the next Chapter.

### 3.4.3.3 AMNs predictions are not respecting constraints like FBA computations

Importantly, the present results also show the significant gap between the SV norm and V norm found in FBA computations and these metrics found in AMN predictions. As shown in the following table 3.4, the difference of SV norm is several orders of magnitudes, questioning the realistic nature of AMN flux predictions in terms of stoichiometric constraints. The results presented in table 3.4 are computed from AMN-QP-bnds predictions, after hyperparameter optimization (same results as for yellow bars on previous Figure 3.7), or from FBA computations acquired for the training set generation. For extended results of metrics computed with FBA results, refer to the appendix Figure C.1, where one can observe the entire distributions of SV norms and V norms for both training sets.

The fact that AMNs reach significantly higher SV norms than what FBA computes is problematic: that means AMNs predictions do not respect GEMs constraints as well as FBA computations. The fact that AMNs reach slightly lower V norms is not problematic in itself, but indicates a probable reason explaining the SV norm issue, which will be further explained. Even though this issue was not fixed in the context of this dissertation, I will now give some hypotheses explaining this unwanted behavior. In the next Chapter 4, I will give leads on how the issue could be solved.

Looking into individual AMN predictions showing high SV norm and low V norm, I observed more



Training set	FBA or AMN	Average SV norm	Average V norm
core-random	FBA	8.36e-16	0.74
	AMN	2.63e-2	0.51
iML-singles	FBA	1.09e-10	1.0e-2
	AMN	1.49e-4	8.6e-4

Table 3.4: Comparison of SV norm and V norm metrics found in FBA computations and AMN predictions.

fluxes close to zero compared to fluxes computed with FBA. This observation was even more striking with increasing values for the weight applied on  $L_2$  (the term encouraging the respect of  $SV = 0$ ). Therefore, this issue is hypothesized to come from a ‘bias’ of the AMN when it attempts to respect  $SV = 0$ . This would be explained by the fact that a valid solution of  $SV = 0$  is  $V = 0$ . Therefore, the AMN would be ‘cheating’ to ensure  $SV = 0$ , which drives the model to predict many fluxes at zero. This induces lower V norm metrics than what can be found in FBA computations.

Importantly, such unwanted behavior was observed in many cases, but not all. For example, AMN-LP with 128 iterations trained on ‘core-random’ (Figure 3.5) yields satisfactory V norm metrics and lower SV norm than what was obtained with AMN-QP-bnds (even after hyperparameter optimization). Also, using a Reservoir Computing (RC) approach tackles this issue, as it relies on performing FBA to obtain final flux distributions, as it is done in Figure 2.5, appendix B Figure S9, and Figure 3.13. These points will be further discussed in Chapter 4.

In the next section, I will investigate alternative ML models capabilities in the task of surrogating FBA. This was motivated with the hypothesis that ANNs might not be the best architecture for that task. Indeed, the issue discussed above might also be explained from intrinsic limitations of ANNs in the FBA surrogation task.

## 3.5 Exploration of alternative ML methods to better surrogate FBA

### 3.5.1 Introduction

This fourth section aims to investigate the performance of alternative Machine Learning (ML) methods in the task of surrogating FBA. I will show purely exploratory results, not focusing on any technical improvements of the method. The previous section 3.4.3.3 shed light on a concerning issue found in AMNs predictions: the SV norm is orders of magnitude above what is found with FBA (and the V norm is lower). One solution to tackle this issue might be to use other ML methods than ANNs. Indeed, we solely used ANNs so far, and they might have intrinsic shortcomings to surrogate FBA. Consequently, this section investigates their ability to surrogate FBA through performance comparisons. It will focus on three ML methods: ANNs, Multi-task Elastic Net, and XGBoost.

Importantly, the task for which ML methods are tested is different from the usual task of AMNs. In most cases, AMNs have partial flux data to learn on (most often, the growth rate is the only flux to be fitted), and a custom mechanistic loss pushes AMNs predictions to fulfill constraints and produce realistic flux distributions. This is the case for Figures 2.2 to 2.5 in the previous chapter, as well as Figures 3.5 to 3.7 in the present chapter. In contrast, the ML methods will be tested here with *all* fluxes

given as reference data to fit, and these methods will not have a custom mechanistic loss like AMNs. They will simply minimize the mean squared error between all ‘true’ fluxes and all predicted fluxes. In short, ML methods tested here aim to predict full flux distributions from upper bounds on uptake fluxes, like AMNs; but they differ in the data used as reference and the loss function. In other words, from a biological standpoint, the ML methods tested here generalize from a set of upper bounds on uptake fluxes (independent variables) and all fluxes obtained by FBA (dependent variables), to predict full metabolic flux distributions of a GEM. In that sense I attempt in this section to build purely ML surrogates of FBA that are not ‘hybrid’, but should reproduce the behavior of FBA faithfully.

## 3.5.2 Methods

### 3.5.2.1 Training sets and models architectures

The training sets used in this section are ‘core-glucose’, ‘core-random’, ‘core-extended’, ‘iML-glucose’, ‘iML-random’, ‘iML-extended’ and ‘iML-singles’. Please refer to the Preamble section (3.1) of this chapter for more information.  $V_{in}$  was always used as input, all fluxes were always used as output (*i.e.* the whole flux distribution computed by FBA is the label).

The 3 tested ML models were picked for interesting comparisons: (i) ANNs with the same architectures as the AMNs’ optimal ones found after hyperparameter optimization of AMN-QP-bnds in the previous section; but this time fitted on all fluxes without constraints terms, instead of the growth rate fit and constraints terms; (ii) Multitask Elastic Net regression (from sci-kit learn), a method found to be performant in the tested task with a relatively simple formulation; (iii) XGBoost, a gradient-boosted decision tree algorithm that is performant in many tasks and scalable:

- ANNs, with the ‘ANN-core’ and ‘ANN-iML’ architectures. These architectures were taken from the AMN-QP-bnds’s neural layers architectures found to be optimal in the previous section’s hyperparameter optimization. These optimal architectures were found with the objective of maximizing the  $Q^2$  obtained on the growth rate and minimizing the custom mechanistic loss values. Therefore, it is Important to note that such architectures are not optimal for fitting all fluxes as it is done here. ‘ANN-core’ has 3 hidden layers of size 102, 97 and 16, with hyperbolic tangent activation function for the two first hidden layers and ReLU for the last one. The optimal dropout rates are set to 0.015, 0.022 and 0.017. ‘ANN-iML’ has no hidden layer, so  $V_{in}$  is directly connected to all fluxes. The optimizer was always ‘adam’, with a batch size of 10 and 50 epochs. Note that in the following results, ‘ANN’ will be used to designate both ‘ANN-core’ (for results obtained with ‘core-random’) and ‘ANN-iML’ (for results obtained with ‘iML-singles’).
- Multitask Elastic Net regression, called ‘MTEN’ in short. This model is detailed in its [documentation \(clickable link\)](#). It basically relies on an ensemble of elastic nets, one for each output (each flux). Each of them is a linear regression with L1 and L2 regularization terms. I used an existing implementation from scikit-learn[137].
- XGBoost[139], called ‘XGB’ in short. This model is detailed in its [documentation \(clickable link\)](#). XGBoost stands for extreme gradient boosted trees, it relies on boosting procedures (splitting the dataset to learn submodels) and tree-based machine learning (like random forests).

All models were benchmarked by 10-fold cross-validation, aggregated predictions from the validation sets were used to display performance results.

### 3.5.3 Results and discussion

#### 3.5.3.1 MTEN and XGB are promising ML models to surrogate FBA

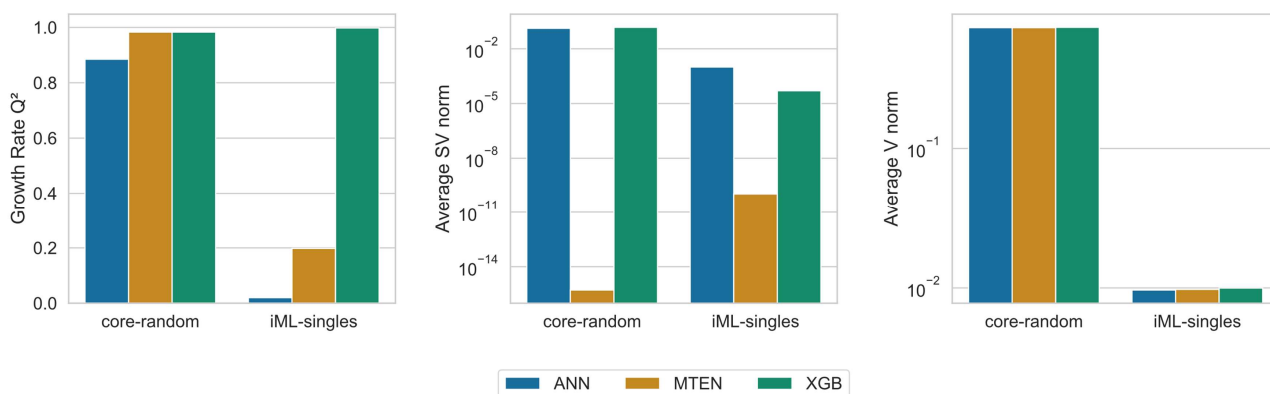


Figure 3.8: Alternative ML models to surrogate FBA. ANN (blue), MTEN (orange) and XGB (green) were used with two training sets, 'core-random' and 'iML-singles'. From left to right, bar plots display the ML models performance in terms of (i)  $Q^2$  computed between the FBA growth rate solution and the ML model prediction; (ii) the average SV norm found in the ML model predictions; and (iii) the average V norm found in the ML model predictions.

Three striking observations can be made from Figure 3.8. First, learning on all fluxes with ML methods instead of solely on the growth rate in AMNs, always yields more realistic V norm values (closer to what FBA has computed), for all ML methods tested here: for 'core-random', average V norms were always close to 0.7 and for 'iML-singles', they were always close to  $10^{-2}$ . For both models, these values fall in the range of what FBA predicts: the average V norm was found at 0.74 for 'core-random' and  $10^{-2}$  for 'iML-singles' (values from table 3.4, full distributions are available in appendix Figure C.1).

Second, both ANNs ('ANN-core' and 'ANN-iML') show limited fitting performance with systematically lower growth rate  $Q^2$  than other methods. Importantly, note that this could be eventually remedied with hyperparameter optimization, since we used an architecture optimized for a different task in the present case. In light of previous results, hyperparameter optimization could also slightly reduce the SV norm metric obtained with ANNs. This poor performance of ANNs, compared to the performance obtained for the same architecture in the previous section 3.4 for a different task, underlines the fact that architectures found with hyperparameter optimization cannot be directly transferred to another modeling task.

Third, the ANN and XGB models reach higher SV norm values than MTEN, which reaches the lowest SV norm values, by far. Most importantly, the SV norms obtained with MTEN are close to what was found in FBA computations (see Table 3.4 for SV norms computed on FBA results for 'core-random' and 'iML-singles'). Therefore, I conclude that MTEN has more potential to be used as a surrogate, since it seems able to satisfy SV constraints very well. Consequently, I extended the performance assessment of MTEN with more training sets, to observe its capabilities in more diverse contexts.

#### 3.5.3.2 MTEN surrogates FBA with different performances depending on the training set

From Figure 3.9, we see that the training set content plays a crucial role in MTEN performance. Strikingly, the SV norm is lower, so the stoichiometric constraints are better respected, with 'core-random', 'core-extended', 'iML-random' and 'iML-extended'; compared to 'core-glucose' and 'iML-glucose'. These

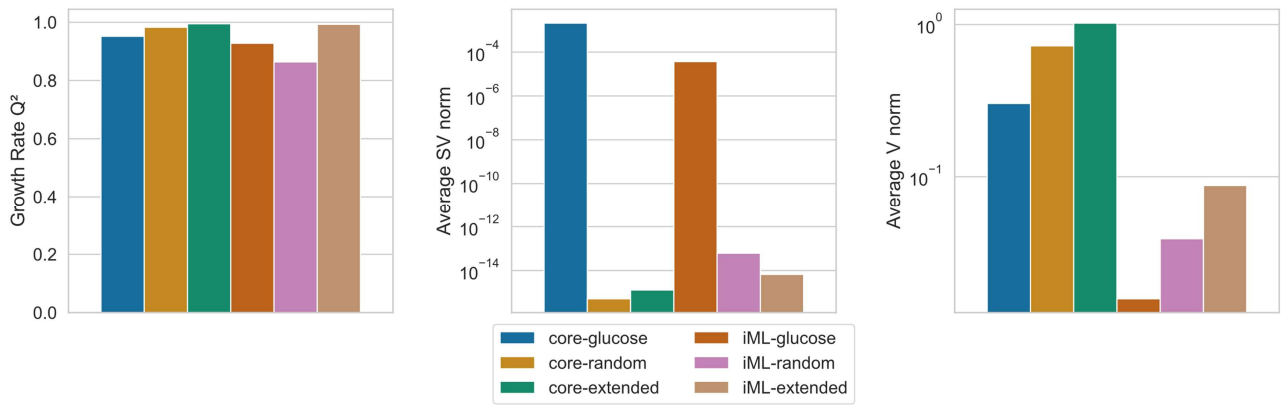


Figure 3.9: MTEN performance to surrogate FBA with six different training sets. MTEN was used with six training sets: 'core-glucose' (blue), 'core-random' (orange), 'core-extended' (green), 'iML-glucose' (red), 'iML-random' (pink) and 'iML-extended' (light brown). From left to right, bar plots display the ML models performance in terms of (i)  $Q^2$  computed between the FBA growth rate solution and the ML model prediction; (ii) the average SV norm found in the ML model predictions. Strikingly, different training sets yield different performances, especially with the SV norm that is drastically lower with more complex training sets (those where upper bounds are drawn for more diverse uptake fluxes than glucose).

results indicate that varying the glucose uptake flux only to generate a training set does not provide enough statistical patterns for MTEN to perform well. This suggests that training sets acquired through wider samplings of uptake fluxes upper bounds yield enough information in inputs to predict outputs, which is not the case for training sets acquired through narrow samplings such as 'core-glucose' or 'iML-glucose'.

Added to the SV norm differences, different training sets also yield different fitting performance. Even if all growth rate  $Q^2$  obtained are above 0.8 which is satisfactory, there are clear differences between training sets. In the appendix Figure C.11, one can observe the complete scatter plot of growth rate predictions by MTEN against reference growth rates obtained by FBA; for the three iML1515[44] training sets used.

From this section's results, we can conclude that MTEN has great potential to surrogate FBA, but the training set content is very crucial for its performance. Such importance of the training set content has been briefly assessed for AMNs, in section 3.4.

This influence of the training set content will be further investigated in the next section, focusing on Chapter 2's results shown in Figures 2.4 and appendix B section 'AMNs benchmarking with gene knockouts and multiple measured fluxes'. Indeed, we have shown there the good performance of AMNs when integrating metabolic or regulator gene KOs, with the growth rate or multiple fluxes measured. Now, we shall explore how such performance depends on training sets' contents.

## 3.6 AMNs' ability to predict the effect of media and genetic conditions on metabolic fluxes depends on training set contents

### 3.6.1 Introduction

This fifth section aims to investigate the effect of varying experimental training sets content on AMNs prediction performance. I will show purely exploratory results, not focusing on any technical improvements of the method. The investigation is made through modifications brought to the training set itself or how it is used in cross-validation. In this section, AMNs do not have a mechanistic layer so the neural layer directly predicts the flux distribution, as in section 3.4. Also in the present section, AMNs output full flux distributions, with medium compositions and genetic perturbations as inputs. Experimentally determined growth rates or a set of 31 measured fluxes (including the growth rate) were used as references. In other words, from a biological standpoint, AMNs generalize from a set of media compositions and genetic perturbations (independent variables); and corresponding reference fluxes (dependent variables), to predict full metabolic flux distributions of a GEM.

In section 3.4 of this chapter, we have seen that different training sets yield different performances with AMNs, and hyperparameter optimization can lead to better architectures, each suited for a specific training set. In section 3.5, when exploring the performance of MTEN, we have also assessed the importance of training data on the prediction performance of ML models. Therefore, it is important to assess how changing the structure of training sets changes the AMN performance. In particular, this section aims to tweak the training sets used to generate results obtained in Figures 2.4 and appendix B section 'AMNs benchmarking with gene knockouts and multiple measured fluxes'. We will use the 'Cov-BiologKO' (used for Figure 2.4) and 'Rijs-RegKO' (used for appendix B section 'AMNs benchmarking with gene knockouts and multiple measured fluxes') training sets, modifying their structure in meaningful ways. In particular, 'Cov-BiologKO' was re-organized with a specific cross-validation scheme, in order to obtain validation splits with unseen conditions (media compositions or reaction KOs); and 'Rijs-RegKO' was splitted to obtain subsets with only glucose or only galactose as carbon source; and it was used with only the growth rate as the reference flux to fit instead of the 31 fluxes of the dataset.

### 3.6.2 Methods

#### 3.6.2.1 Training sets and models architectures

The training sets used in this section are 'Cov-BiologKO' and 'Rijs-RegKO'. Please refer to the Preamble section (3.1) of this chapter for more information.

'Cov-BiologKO' will be used in 2 alternative ways: (i) organized in such a way that validation sets of a 145-fold cross-validation each contain an unseen combination of reaction KOs; (ii) organized in such a way that validation sets of a 120-fold cross-validation each contain an unseen combination of substrates (*i.e.* media composition). In all cases,  $C_{med}$  and  $R_{KO}$  were used as inputs, and the measured growth rate was used as output (*i.e.* it was the only reference flux to fit). The alternative ways (i) and (ii) will be respectively called 'Cov-BiologKO-unseen-KO' and 'Cov-BiologKO-unseen-media'. Note that these alternative ways do not change the content of the training set but solely their organization and cross-validation approach. In short, the training set was re-organized by grouping the same  $R_{KO}$  (*i.e.* the same metabolic gene KO) together sequentially, for alternative way (i); and the training set was re-

organized by grouping the same  $C_{med}$  (*i.e.* the same Biolog growth condition) together sequentially, for alternative way (ii). Importantly, for the two alternative ways, the training set was not shuffled during the cross-validation process, unlike all other cross-validation results shown in this dissertation.

‘Rijs-RegKO’ will be used in 3 alternative ways: (i) a subset with only the data of glucose conditions, (ii) a subset with only the data of galactose conditions, (iii) using only the growth rate as output of the training set, instead of the 31 fluxes. Results for alternative ways (ii) and (iii) can be found in the appendix C.7. For ‘Rijs-RegKO’,  $C_{med}$  and  $G_{KO}$  were always used as inputs. The alternative ways (i), (ii) and (iii) will be respectively called ‘Rijs-RegKO-galactose’, ‘Rijs-RegKO-glucose’ and ‘Rijs-RegKO-GR-only’. For the alternative ways (i) and (ii), 31 measured fluxes were used as outputs; for the alternative way (iii) only the measured Growth Rate was used as output (*i.e.* it was the only reference flux to fit). In short, the training set was splitted between galactose and glucose conditions, respectively for alternative ways (i) and (ii). For the alternative way (iii), the training set was kept as a whole but only the measured growth rate was retained, and the remaining 30 measured fluxes were discarded from the training set.

All alternative ways of using ‘Cov-BiologKO’ and ‘Rijs-RegKO’ are summarized in the following table 3.5.

Original training set	Alternative way number	Alternative way name	Comment
Cov-BiologKO	(i)	Cov-BiologKO-unseen-KO	Validation sets contain unseen reaction KOs
	(ii)	Cov-BiologKO-unseen-media	Validation sets contain unseen media compositions
Rijs-RegKO	(i)	Rijs-RegKO-galactose	The training set contains only galactose conditions
	(ii)	Rijs-RegKO-glucose	The training set contains only glucose conditions
	(iii)	Rijs-RegKO-GR-only	Only the growth rate is used as the reference flux to fit

Table 3.5: Summary of the alternative ways to use ‘Cov-BiologKO’ and ‘Rijs-RegKO’ training sets.

In all cases, AMN-QP-bnds was used, without a mechanistic layer. The ‘adam’ optimizer was always used. For ‘Cov-BiologKO’, aggregated predictions from the validation sets are used as final prediction values. For ‘Rijs-RegKO’, the mean aggregated predictions from the validation sets over 10 repeats of a 10-fold cross-validation were compiled as final prediction values.

For models trained with ‘Cov-BiologKO’ training set, in regular and alternative ways, neural layers had the following parameters: 1 hidden layer of size 500 with ReLU activation function and 0.2 of

dropout rate. The batch size was set to 100 and epochs to 10.

For models trained with 'Rijs-RegKO' training set, in regular and alternative ways, neural layers had the following parameters: 2 hidden layers, each of size 400 with ReLU activation function and 0.2 of dropout rate. The batch size was set to 10 and epochs to 200.

FBA alone was used with 'Cov-BiologKO', by simulating KOs found in  $R_{KO}$  and using arbitrary upper bounds on uptake fluxes for substrates found in  $C_{med}$ , for each training set entry. The biomass reaction of iML1515[44] was optimized and the value directly used as the prediction. This is the same process as in Chapter 2 (for Figure 2.4 panel c).

### 3.6.3 Results and discussion

#### 3.6.3.1 AMNs better predict the effect of reaction KOs and media conditions that appeared in the training data

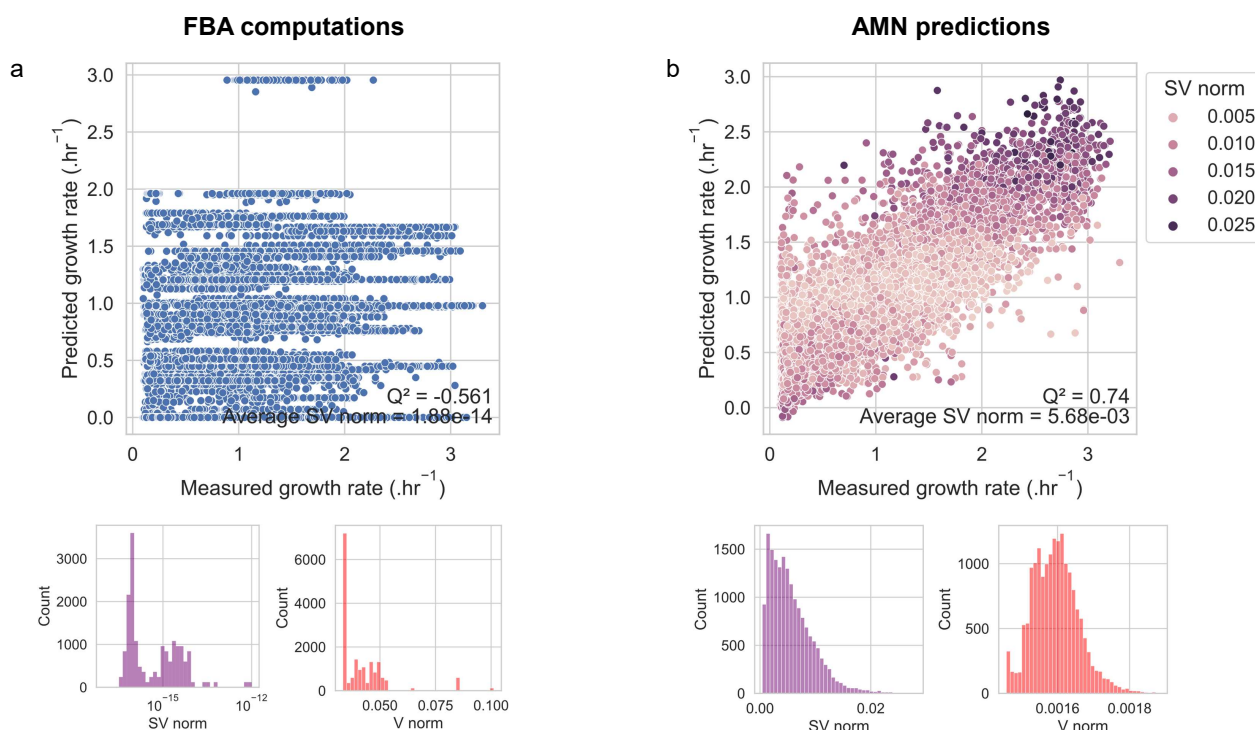


Figure 3.10: AMN-QP-bnds outperforms FBA with the 'Cov-BiologKO' training set but do not respect stoichiometry constraints. (a) FBA alone with arbitrary bounds on uptake fluxes (as in Chapter 2's Figure 2.4 panel c) (b) AMN-QP-bnds (regular approach, as in Chapter 2's Figure 2.4 panel b). Both panels display, from top to bottom and left to right: A scatter plot of the growth rate predictions (Y-axis) against the experimentally measured growth rates (X-axis), with points' colors corresponding to the SV norm value computed on the prediction; an histogram of the SV norm metric for all predictions and an histogram of the V norm metric for all predictions. For FBA computations, the average V norm is 0.042 and for AMN predictions the average V norm is 0.0016. Strikingly, AMN-QP-bnds outperforms FBA in terms of  $Q^2$  but stoichiometric constraints are not well respected, with an average SV norm of 5.68e-03, against an average SV norm of 1.88e-14 for FBA.

On Figure 3.10 we can observe that FBA alone shows bad fitting performance but extremely low SV norm values. In contrast, AMN-QP-bnds shows good fitting performance but relatively high SV norm values, and much smaller V norm compared to FBA alone. Looking at the detail of the prediction vector, I observed that many fluxes in the AMN-QP-bnds predictions were close to zero, explaining the small V norm observed, as discussed in section 3.4.3.3.



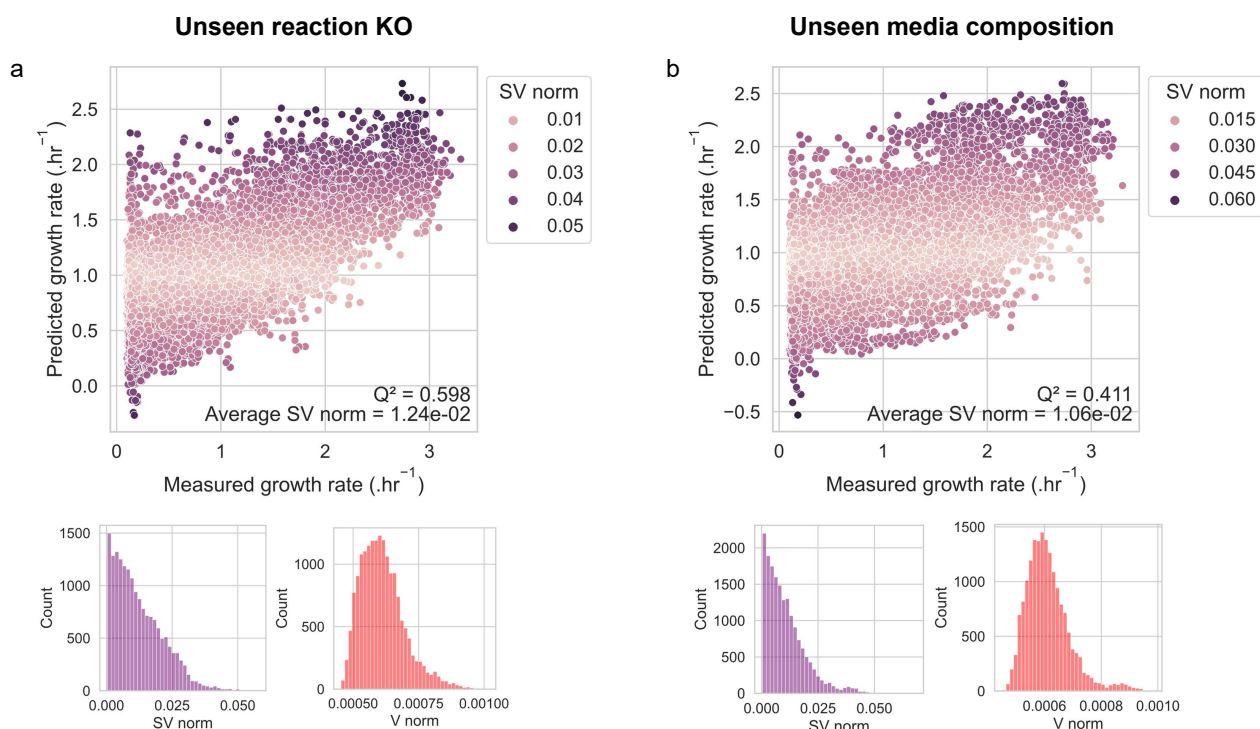


Figure 3.11: AMN-QP-bnds performance with unseen combinations of reaction KOs and unseen media compositions. AMN-QP-bnds performance with (a) ‘Cov-BiologKO-unseen-KO’, displaying aggregated predictions from a 145-fold cross-validation, with each validation set containing an unseen combination of reaction KOs and (b) ‘Cov-BiologKO-unseen-media’, displaying aggregated predictions from a 120-fold cross-validation, with each validation set containing an unseen media composition. Same plot organization as previous figure’s. The  $Q^2$  is reduced with those alternative ways, compared to the regular approach on panel b of Figure 3.10. This indicates that AMNs better predict the effect of gene KO and media compositions when they are found in the training data.

On panels a and b of Figure 3.11, we can observe the SV norm and V norm metrics having similar order of magnitudes compared to the regular way, displayed on panel b of Figure 3.10. More strikingly, alternative ways of using ‘Cov-BiologKO’ display lower  $Q^2$ . Results of Figure 3.11 can be interpreted as the AMN-QP-bnds predictive power that is only based on the media composition (panel a) or the reaction KOs (panel b). Therefore, we can conclude that most of the predictive power displayed on panel b of Figure 3.10 comes from the media composition, and a smaller part also originates from the reaction KOs. However the difference in performance here should be kept in perspective of the very different cross-validation scheme (120-fold and 145-fold for alternative ways, 10-fold for the regular way).

### 3.6.3.2 AMNs can predict the effect of a regulator gene KO only if statistical patterns are found in the reference fluxes

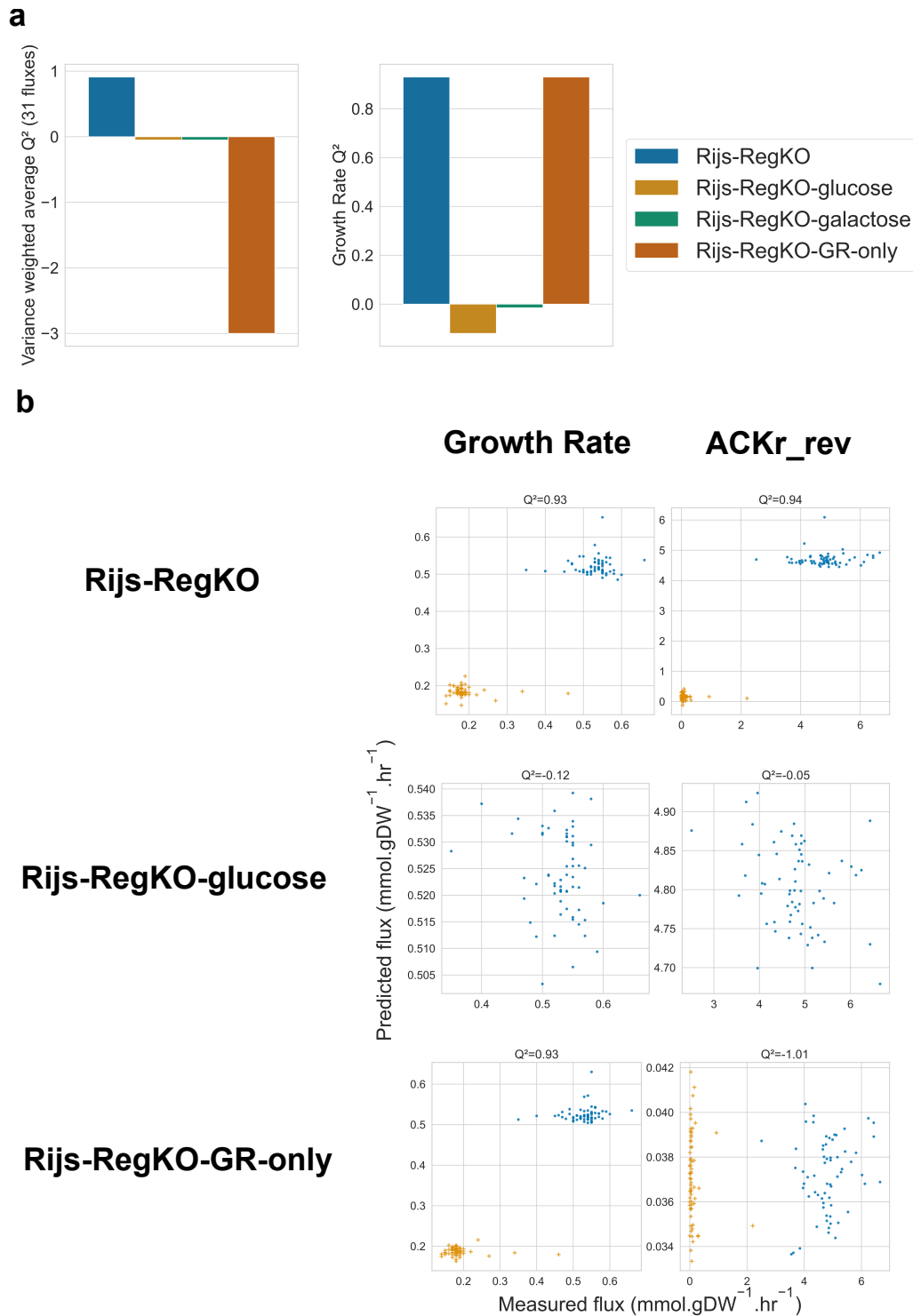


Figure 3.12: AMNs can predict the effect of a regulator gene KO only when statistical patterns are found in the training data. (a) Comparative bar chart of AMN-QP-bnds performance with different ways of using the training set 'Rijs-RegKO': regular (blue, 'Rijs-RegKO'), glucose only conditions (light orange, 'Rijs-RegKO-glucose'), galactose only conditions (light orange, 'Rijs-RegKO-galactose'), and growth rate reference flux only (dark orange, 'Rijs-RegKO-GR-only'). Refer to Table 3.5 for details on those training sets. Bar plots show the variance weighted  $Q^2$  computed for all 31 reference fluxes (left) and the  $Q^2$  computed only with the growth rate (right). (b) Scatter plots of the Growth Rate and ACKr\_rev flux predictions (Y-axis) against the experimentally measured fluxes (X-axis), with points' style corresponding to the carbon source (yellow crosses: glucose, blue dots: galactose). Scatter plots are displayed for the regular and alternative ways (ii) and (iii). Such alternative ways do not yield satisfactory performance, indicating that using only one carbon source condition or only the growth rate as a reference flux do not provide enough statistical patterns for the AMN to perform well.

Strikingly, as observed on the left bar plot of panel a Figure 3.12, the variance weighted aver-

age  $Q^2$  computed on the 31 reference fluxes is close to 1 only when using the full training set in the regular way, 'Rijs-RegKO'. When considering only glucose or galactose, with 'Rijs-RegKO-glucose' and 'Rijs-RegKO-galactose', the variance weighted average  $Q^2$  falls close to zero, and when considering only the growth rate as a reference flux to fit, with 'Rijs-RegKO-GR-only' the variance weighted average  $Q^2$  falls to -3. When looking at the  $Q^2$  computed solely on growth rates, on the right bar plot of panel a Figure 3.12, the  $Q^2$  only falls around zero when considering glucose or galactose conditions only. When looking at the detailed prediction results on panel b Figure 3.12, it is clear that the growth rate and ACKr\_rev reference fluxes are clustered according to the carbon source. As a result, most of the statistical patterns in the training set rely on such clusters. Indeed, we clearly see that the AMN is unable to accurately predict the fluxes when the training set contains only glucose or only galactose conditions. This is not surprising: with such subsets of the training set, each data point has a different regulator gene KO, which prevents the AMN from learning any statistical relationships. Looking into the detailed flux predictions of the growth rate and ACKr\_rev (panel b, 'Rijs-RegKO-glucose', and appendix Figure C.13), one can clearly observe that AMN predictions fall into the right value range compared to measured fluxes, but the precise value prediction is impossible for the AMN.

The 'Rijs-Reg-KO-GR-only' results of Figure 3.12 indicate that the AMN learning only with the growth rate as a reference flux to fit is unable to have predictive power on other fluxes. This might be related to the issue mentioned in section 3.4.3.3, since most flux predictions are close to zero for other fluxes than the growth rate (see appendix Figure C.15).

Figures with extended results for all alternative ways are available in the appendix C.7. Figure C.12 to C.15 display the V norm and SV norm distributions; as well as scatter plots for all 31 fluxes (not just the growth rate and ACKr\_rev).

## 3.7 Closing remarks

### 3.7.1 Chapter summary

In this chapter, I have formulated an improved version of the QP-solver, namely QP-bnds-solver, that inspired an improved AMN architecture, namely AMN-QP-bnds. It uses a custom loss that is more general and versatile, shown to better respect GEMs' constraints. After that, I investigated the behavior of MM solvers inside AMNs (termed mechanistic layers) showing their interesting capabilities for improving AMNs predictions, but also their limits: further investigations and improvements are required to reliably use them with all GEMs. Moving on, I showed that neural layers of AMNs can be fine-tuned through hyperparameter optimization, a process that heavily depends on the training set. In the same section, I also stressed an issue observed with AMN flux predictions showing high SV norms, probably due to many flux values close to zero, an issue that seems to originate from  $L_2$  (the loss term that ensures stoichiometric constraints) pushing predicted flux values towards zero instead of more realistic values. Next, I showed how different ML models can surrogate FBA with different performances, using different training sets. Finally, I showed the dependence of AMNs' performances on their training sets, with alternative ways of using two experimental datasets, drastically changing the obtained performance, concluding that AMNs can predict the effect of media compositions and genetic perturbation on metabolic fluxes if and only if those fluxes show enough statistical patterns to learn with.

In short, the important take-aways of the present chapter are the following: (i) alternative MM solvers can be formulated, as well as alternative custom mechanistic loss for AMNs, which change

the general behavior of the model, (ii) mechanistic layers have high potential to enhance AMNs predictions but still require improvements, (iii) hyperparameter optimization can enhance an AMN performance, but AMNs still predict flux distributions far from respecting GEM's constraints like FBA, (iv) MTEN performs very well for surrogating FBA, especially in terms of stoichiometric constraints, and (v) an AMN can predict the effect of media compositions and genetic perturbations depending on statistical patterns found within the training data.

### 3.7.2 Reservoir computing, a promising approach to tackle AMN issues?

From all the observations made in this Chapter, we can make important assessments on AMNs capabilities and caveats. First of all, including the  $L_2$  term in the custom loss, to ensure  $SV = 0$ , does not suffice for making realistic flux predictions in terms of stoichiometric constraints. Mechanistic layers seem like a very promising way to tackle the issue, but the current implementations still suffer drawbacks. The MTEN model also showed promising results in its capabilities of better respecting GEMs constraints. However, it was only assessed with all fluxes used as references to fit. Therefore, its potential as a reservoir to surrogate FBA is clear but its potential as a basis for a hybrid model similar to AMN is still uncertain.

All these assessments should drive us towards the use of the 'AMN-Reservoir' approach (Figure 2.2), that seems the most reliable approach. Indeed, if we redraw the Figure S9 of appendix B (section 'AMN-Reservoir prediction performance') with similar plots as those shown in this chapter, we obtain the following Figure 3.13.

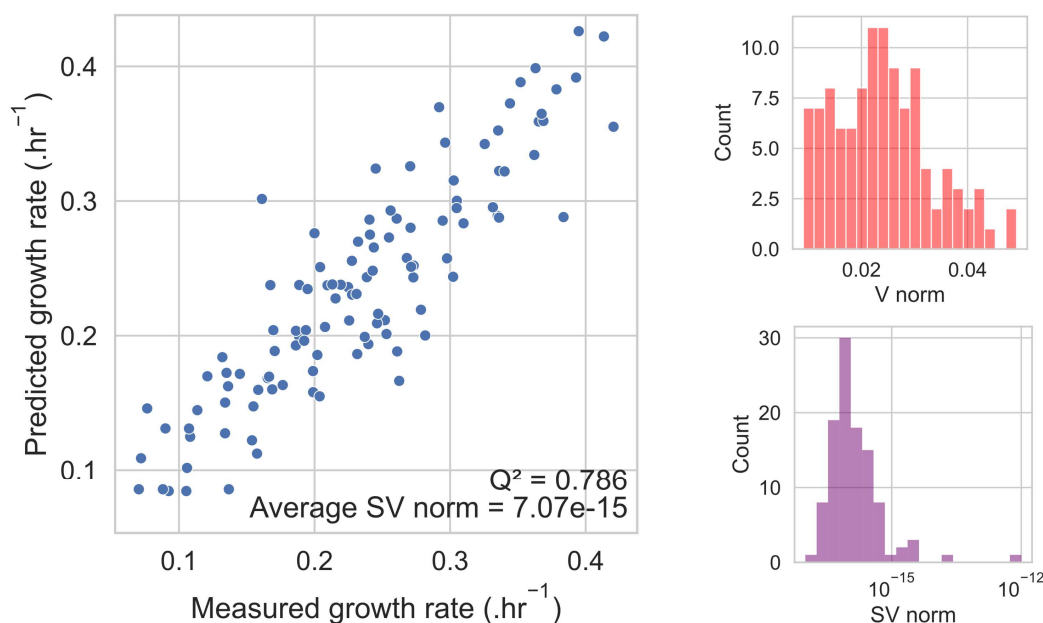


Figure 3.13: Performances of AMN-Reservoir using predicted  $V_{in}$  as input to FBA. The dataset used to train the AMN-Reservoir is '110GR' (also used for Figure 2.3 and Figure 2.5 panel c). The measured growth rates are plotted as the mean over technical replicates (*cf.* Chapter 2 Methods, 2.5) and FBA results are shown as the predicted growth rate. The hyperparameters and the pre-trained AMN-Reservoir were the same as for Figure 2.5 panel c. A 10-fold cross-validation was performed (instead of a training and prediction on the whole dataset as in Figure 2.5 panel c), and validation sets predictions were used to extract  $V_{in}$ , then used as input for FBA. On the right are shown histograms of the V norm and SV norm metrics computed for FBA results. As a reminder, FBA with arbitrarily chosen bounds yields a  $Q^2$  of 0.51 as shown in Figure 2.5 panel d (with similar values for SV and V norms).

This figure shows ideal results, as it not only yields good  $Q^2$  performance, but also perfectly respects GEM constraints and produces realistic norms of flux predictions, which is expected since we use FBA to compute these flux distributions. However, one can argue that this approach is not a hybrid model as defined in Chapter 1, as it relies on 3 different steps for making such predictions, as shown in the following Chapter 4, Figure 4.1.

We shall discuss these points further in the next Chapter, where I will summarize all previous chapters, and further discuss the present concluding remarks. I will also provide usage perspectives and improvement directions of hybrid modeling for GEMs, and conclude the dissertation by presenting where the metabolic modeling community should focus in the future, in my opinion.



# Chapter 4

## General discussion and perspectives

I will start this chapter with an overview of both the strengths and limitations of AMNs. First, I will summarize how AMNs improve GEMs' predictive power. Then, I will review how such improvement is based on an innovative approach of learning with constraints. Doing so, I will underline how this challenging approach requires further investigations, reminding some issues uncovered in the dissertation.

Next, I will discuss the potential ways to tackle such limitations, and the best potential areas of application for AMNs. Finally, concluding remarks close the dissertation with a global summary, answering the scientific questions set for the dissertation in chapter 1, section 1.5.

### 4.1 A novel and challenging approach to exploit GEMs

#### 4.1.1 Improving GEMs predictive power with AMNs

##### 4.1.1.1 From an optimality to a learning principle

In many metabolic modeling approaches, GEMs are involved and exploited through FBA, or closely related methods (examples are given in Chapter 1 section 1.3.3.3.3). From a given condition (*i.e.*, LP constraints to simulate media compositions, or genetic perturbations), such methods can find a single metabolic phenotype (*i.e.*, a flux distribution solution). Importantly, they solve each condition independently of each other. Whether the methods rely on a linear or quadratic program, they do not attempt to generalize from a set of instances, but rather consider each condition to solve as a different problem. In most approaches combining ML and GEMs (examples are given in Chapter 1 section 1.4.2), the ML part enables generalization from a set of conditions, but the GEMs are used without such generalizing abilities.

By formulating AMNs, we radically change this paradigm: relying on learning procedures, the model attempts to learn and generalize from statistical patterns found between input conditions and output reference fluxes, as well as custom loss terms accounting for GEMs constraints. This is the main novelty brought by the original research work shown in this Ph.D. dissertation. As we have seen in Chapter 2, this novelty drastically improves the predictive power of GEMs, when comparing it to FBA. In particular, when uptake fluxes are unknown but media compositions are known, AMNs can grasp the power of ML to predict the uptake fluxes in yet unseen conditions. We also showed that integration of metabolic gene KOs and regulator gene KOs with AMNs was possible and showed good performance. Finally, we showed that an AMN could be pre-trained on FBA simulations to faithfully



reproduce FBA, to then be included with frozen parameters in a larger learning model; that is trained on experimental datasets in order to find best FBA inputs for the experimental data to match FBA computations (*i.e.*, the ‘AMN-Reservoir’ or ‘Reservoir Computing’ approach).

Interestingly, comparing MFA (described in Chapter 1 section 1.3.3.2) and AMNs, one can assess that AMNs propose another level of generalization. Indeed, MFA approaches attempt to minimize an error between flux measures and GEMs predictions, but they do so for each condition independently. In contrast, AMNs are able to minimize a similar error for a set of conditions, thus building a predictive model.

#### **4.1.1.2 AMNs can predict full metabolic phenotypes with partial flux data**

AMNs have an advantage over regular ML models: they predict more than the data given as reference. In particular, they can predict full metabolic fluxes distributions with partial flux data measures, whereas regular ML models would only predict the fluxes that are measured. Let me consider a case where one wants to build a predictive model for 5 metabolic fluxes from media compositions. One could use regular ML models, using supervised learning to model the relationship between the media compositions and the 5 metabolic fluxes, probably reaching satisfactory predictive power. But one could also use hybrid models like AMNs, enabling the prediction of the full metabolic phenotype instead of the 5 measured fluxes. This would extend the size of predictions, which could be useful for later statistical analyses of metabolic fluxes, eventually uncovering new insights. Therefore, AMNs can bring more insights than regular ML models, with the same measuring efforts.

#### **4.1.1.3 Mechanistic layers can improve predictions quality**

Mechanistic layers are the second main novelty introduced in this dissertation. Indeed, the inclusion of “neural” solvers as a part of learning architectures is an innovative piece of work. To our knowledge, no such approach was yet performed in literature. The use of mechanistic layers, as seen in Chapter 3 section 3.3 for AMN-LP (Figure 3.5), can drastically improve the prediction quality of AMNs, especially in terms of GEMs constraints respect.

#### **4.1.1.4 A wider range of input or output data type**

The formulation of AMNs (especially with AMN-QP-bnds, Figure 3.4), enables the use of a GEM with any kind of dataset as input. A strong limitation of GEMs was to only accept flux measures as possible data to integrate: the AMN formulation tackles this issue very clearly. One can think of using as input variables some physical or chemical parameters other than metabolic fluxes, such as temperature, shaking speeds, vessel types, pH, salinity (which cannot be used with GEMs alone, at least with FBA). These environmental parameters strongly influence metabolic phenotypes, as shown in Figure A.5. Thus, AMNs enable one to grasp the power of ML and GEMs for predicting metabolic phenotypes depending on physico-chemical conditions that would not be usable in a regular FBA approach.

#### **4.1.1.5 A wide range of architectures**

The AMNs formulation also provides high flexibility over the possible model architectures to design. As already shown in Chapter 3 section 3.4, hyperparameters of the ML part of AMNs can be fine-tuned to reach better performance. In that sense, AMNs architectures can be designed for any dataset and

modeling task, which is an advantage over FBA and related approaches to exploit GEMs, that are not as tunable.

I just listed the main advantages of AMNs over usual approaches to exploit GEMs, and over usual ML approaches. However, we have seen in Chapter 3 that AMNs still show some limits. These will be summarized in the next section.

## **4.1.2 Learning with constraints: a challenging approach**

### **4.1.2.1 The custom mechanistic loss relies on tradeoffs between the terms**

In Chapter 3 section 3.4, I assessed that the weighting scheme on custom mechanistic loss terms of AMNs influences the final prediction performance. Therefore, these tradeoffs must be investigated when developing AMNs. Indeed, the proposed custom mechanistic loss has several terms competing, making the AMNs training a multi-objective optimization process. This will be further discussed in section 4.2.1.2 of this chapter.

### **4.1.2.2 The SV loss term is not sufficient to satisfy GEMs constraints in AMNs predictions**

Each of the AMNs loss terms has a purpose, and it was found that the SV loss term ( $L_2$ ) imposes clear limitations to the AMNs predictions quality. In Chapter 3 section 3.4.3.3, I assessed that the loss term encouraging  $SV = 0$  ( $L_2$ ) was not sufficient to make meaningful flux predictions (*i.e.*, flux distributions that respect GEMs constraints). Indeed, AMNs were found to predict flux distributions with significantly higher SV norms than what is found in FBA computations. I hypothesized this issue to come from flux predictions showing many values close to zero. This would be due to  $V = 0$  being a solution to  $SV = 0$ . The SV loss term seems to get stuck in that local minima, and does not help reaching realistic flux predictions. Instead, it ‘cheats’ to minimize SV and predicts most fluxes at zero. This issue is related to an important feature of AMNs: they predict flux distributions under ‘loose’ constraints. In contrast to FBA, in which the LP is solved under strict constraints, AMNs attempt to satisfy constraints in predictions in the best way possible, but not under strict constraints.

### **4.1.2.3 Mechanistic layers have limitations**

In Chapter 3 section 3.3, I showed that the mechanistic layer of AMN-QP-bnds and AMN-Wt had both a low effect on prediction performance. Moreover, I mentioned that AMN-LP, even if showing very promising performance with a high number of iterations (128) with *E. coli* core[135], could not be performed with larger models such as iML1515[44] due to technical issues. Finally, using mechanistic layers in AMNs drastically increases computational time, which represents a significant drawback when dealing with large datasets.

### **4.1.2.4 The AMN performance depends heavily on the training set**

In Chapter 3 section 3.4, with Figure 3.6, I assessed that different *in silico* training sets, generated with different samplings of uptake fluxes upper bounds to perform FBA with, were yielding different AMN performances. In this section I also assessed that different training sets were yielding different optimal architectures through hyperparameter optimization.

Moreover, in Chapter 3 section 3.6, with Figures 3.10, 3.11 and 3.12, I showed that the AMNs performance strongly depends on the content (and train/validation split, if cross-validation is performed) of experimental training sets. In particular, it was shown that the AMNs prediction performance was strongly dependent on statistical patterns found in the training sets. This should be taken into account for further development of AMNs, especially when designing and acquiring training sets.

More generally, the trainable nature of AMNs make them very performant in a specific space of experimental conditions. But after being trained, their reusability seem limited to that space. This issue is common to all ML models, as discussed in Chapter 1, section 1.4.1.2.4.

#### 4.1.2.5 The AMN-Reservoir approach is reliable but suffers from limitations

As stated in the closing remarks of Chapter 3 (section 3.7), the AMN-Reservoir approach seems to be the most reliable method. However, it requires to run simulations to train the reservoir with, then train the AMN-Reservoir on experimental data, and finally predict the metabolic phenotypes using FBA with AMN-Reservoir predicted upper bounds as inputs. These 3 separate steps make the workflow fragmented. In Chapter 2, this workflow was described in Figure 2.1 panel d, here I show a more detailed version of this workflow in Figure 4.1, to underline the fragmented nature of the Reservoir Computing approach.

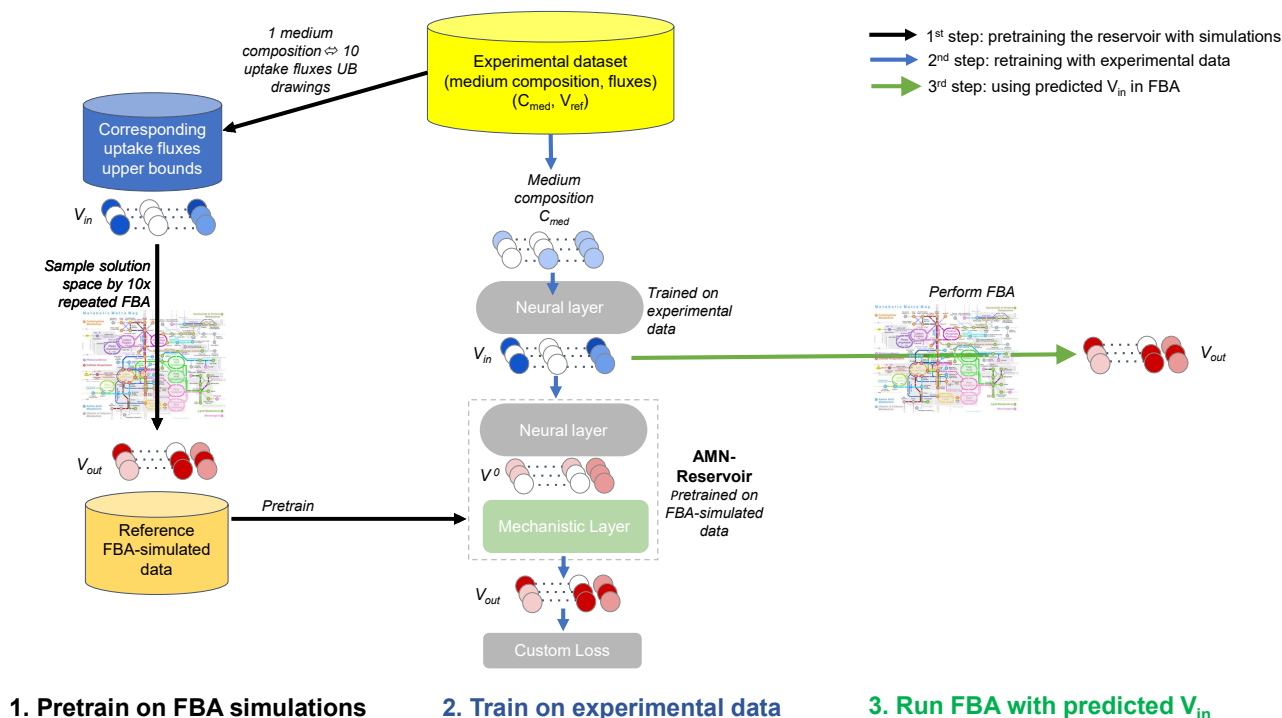


Figure 4.1: Reservoir Computing workflow to generate Figure 2.5. This Figure is an extended version of Figure 2.1 panel d, to show in more detail the different steps of the workflow for AMN-Reservoir computations to be made.

Even though the AMN-Reservoir approach makes more reliable predictions, it is not fitting the hybrid model definition given in Chapter 1. Indeed, it is rather a combined approach of ML with GEMs, as many other existing approaches (cited in Chapter 1 section 1.4.2), which reduces novelty compared to the regular AMN approach (Figure 2.1 panel c). Another possible limitation of the AMN-Reservoir approach is its ability to make predictions overcoming the GEMs limitations. Indeed, the reservoir being trained on FBA simulations, it may show the exact same shortcomings. For example, if we consider a condition in which the GEM predicts no growth of an organism, whereas growth is observed

experimentally, the reservoir trained with this erroneous data will fail to predict growth in similar conditions.

I have summarized here the main limitations of AMNs that were assessed in Chapter 3. These limitations may be tackled in various ways, which will be discussed hereafter. I will also discuss what are the best prospective areas of applications for AMNs.

## 4.2 Perspectives on hybrid models for GEMs: improvements and applications

### 4.2.1 Improved approaches and formulations

#### 4.2.1.1 Standardized training set acquisition

As stated in the previous section, both *in silico* and *in vivo* training sets contents were assessed to strongly influence the overall AMN performance. Therefore, there is a clear need to standardize the design, acquisition, and preprocessing of such training sets, for further developments of AMNs.

##### 4.2.1.1.1 *in silico* training sets

For *in silico* datasets exploited in this Ph.D. dissertation, we always used a random (binomial law, see section 2.5.2) drawing of upper bounds on uptake fluxes, which is an effective means of sampling the space of possible metabolic phenotypes by FBA. However, it is not much standardized, as any drawing scheme can be designed, with different probability laws applied on different uptake fluxes, each with different values range. Also, FBA optimizing the growth rate (biomass reaction) was always performed, which can be limiting the diversity of metabolic phenotypes found in the training set.

For more standardized approaches, one could take inspiration from past works where a sampling of metabolic phenotype space was performed by flux sampling methods[116], to evenly sample a constrained metabolic phenotype space. Another possible inspiration can be taken from the Design of Experiments (DoE) methods, to draw upper bounds in more standardized ways. Moreover, we could think of drawing upper and lower bounds for each reaction instead of solely uptake fluxes, further extending the metabolic phenotype space found in the training set. Finally, one could think of other optimization schemes to generate *in silico* training sets. For example, one could perform FBA with a diverse set of objectives instead of only the growth rate.

##### 4.2.1.1.2 *in vivo* training sets

For *in vivo* datasets exploited in this Ph.D. dissertation, different approaches have been used to build the combinatorial space to perform experiments. For '110GR', a DoE method was used, in particular a random sampling of a full-factorial design. The full-factorial design included all 1 to 4 carbon sources combinations. This approach yields a balanced dataset, in terms of input variables found in the training set. For 'Cov-BiologKO' and 'Rijs-RegKO' (the two external phenotyping datasets), all combinations of the experimental variables were performed. In particular, in 'Cov-BiologKO', all 120 metabolic gene KO mutants have been grown in the 145 media conditions. In 'Rijs-RegKO', all 64 regulator gene KO mutants have been grown in the 2 media conditions. These two approaches yield balanced and 'complete' datasets, which is an appealing feature of a training set. However, they lack a large combinatorial space to further explore. This could be limiting for AMNs in some specific contexts, such as

media optimization for bioproduction, where the training set should not cover the whole combinatorial space. In general, I conclude that the design of experimental training sets should be led with DoE methods, with experimental variables shaping the combinatorial space that are chosen according to our knowledge, as well as the project's goals and resources.

#### **4.2.1.1.3 Automated experimentation for large training sets acquisition**

One limit of AMNs is their requirement for large training sets, compared to regular FBA approaches. A good way to tackle this issue would be to grasp the power of automated experiments[156]. Indeed, this could be an effective means, possibly combined with DoE methods, to acquire large phenotyping datasets in standardized ways.

#### **4.2.1.1.4 Data preprocessing**

An important preprocessing step for ML systems, that has not been extensively explored in this Ph.D. dissertation, are normalization and other data standardization methods. These are known to significantly change the performance of an ML system in some cases (e.g., in a classification task[157]). For all results shown in this dissertation, the only data normalization that was performed is a feature scaling, dividing all values with the global maximum input value (to obtain feature values between 0 and 1). Therefore, more suited input and output data normalization methods could lead to better performance of AMNs.

#### **4.2.1.2 Improved custom mechanistic loss formulations**

The custom loss of AMNs suffers from two main issues: (i) a tradeoff between its terms, and (ii) a term that is not performing as expected, namely  $L_2$ .

The different terms tradeoffs can be partially tackled through hyperparameter optimization. Indeed, changing the weighting of loss terms has been shown to increase AMNs performance, as shown in Chapter 3 section 3.4. In such hyperparameter optimization, the weighting scheme was relatively basic. More sophisticated solutions exist to enhance tradeoffs within models that have such a loss with multiple terms. In literature, this issue (termed 'multi-objective optimization' or 'multi-task learning'), is known and methods exist to tackle this issue[152, 153], such as the multiple gradient descent algorithm.

The  $L_2$  term that should be accounting for GEMs stoichiometric constraints, has to be further investigated in order to enhance AMNs performance. Additional terms, or modification of existing terms could enhance the GEMs stoichiometric constraints respect. Or, the  $L_2$  term behavior might be enhanced by searching for better loss normalizations, in order to give more importance to the term without pushing flux predictions towards flux values close to zero.

Another way to improve the custom loss formulation is to inspire from the LP-solver formulation. One may be able to derive gradients from that method, and inspire from those in the custom loss instead of inspiring from the gradients of the QP-solver.

Finally, in literature, some purely ML processes specifically designed to surrogate constrained optimization problems could inspire a novel custom loss formulation[67]. With such a formulation, mechanistic layers would not be used. For example, one can use a "decomposition scheme alternating master steps (in charge of enforcing the constraints) and learner steps (where any supervised ML model and training algorithm can be employed)" as done by Detassis et al[158], separating the fitting and constraints enforcing processes. Such purely ML approach to surrogate a constrained op-

timization problem has very recently been applied for metabolic modeling, where authors surrogate a dynamic FBA (dFBA) problem with Convolutional Neural Networks (CNNs)[159].

#### 4.2.1.3 Improved mechanistic layers formulations

As shown in Chapter 3 section 3.3, mechanistic layers can improve AMNs predictions, notably with better respect of GEMs stoichiometric constraints. But mechanistic layers also suffer from strong limitations, at least in the current implementations: (i) only AMN-LP's mechanistic layer significantly improves predictions, (ii) their use largely increases computational time of AMNs, and (iii) they could not be used to enhance AMNs predictions with large GEMs like iML1515[44].

The possible above-mentioned improvements for the custom loss formulations could be directly applied to the QP-bnds-solver of AMN-QP-bnds, possibly improving the mechanistic layer capabilities. For AMN-Wt, further investigations are needed to improve the behavior of the mechanistic layer.

A possible way to reduce the computational time might be to enhance the current backend implementation of AMN-LP and AMN-QP-bnds mechanistic layers, as it is done in AMN-Wt. The latter relies on a RNNCell custom object of Keras[144], that is much more efficient than the QP-bnds-solver and LP-solver current implementations as mechanistic layers, relying on stacking solver's operations in a Keras model.

By reducing the computational time of mechanistic layers in AMNs, and increasing their computational efficiency, their use with larger models might be enabled. Otherwise, further improvements would have to be performed; or the use of reduced models would be required to exploit mechanistic layers.

#### 4.2.1.4 Replacing ANNs by other ML methods

Finally, one way to improve the custom loss behavior could be to set us away from ANNs as the ML part of AMNs. This would be a drastic change as it would make the use of mechanistic layers impossible, at least in their current formulations. But given the good performance of the Multi-task Elastic Net (MTEN) observed in Chapter 3 section 3.5, it might be worth trying to use a similar custom mechanistic loss as the one used for AMNs, with MTEN or other performant ML models. As a reminder, this good performance of MTEN was assessed with a fit on all fluxes, with a usual mean squared error loss. Thus, one perspective work can be to attempt a MTEN formulation that integrates a custom loss like AMNs, in order to use partial flux data to fit, and predict the remaining flux vector based on GEMs constraint.

#### 4.2.1.5 Improved AMN-Reservoir workflow and alternatives

As stated in the above section 4.1.2.5, the AMN-Reservoir approach is making reliable predictions but has two major drawbacks: (i) it relies on three separated learning steps, which makes it more of a combined MM and ML approach rather than a hybrid model, and (ii) it can suffer from the same limitations as the GEMs themselves. These issues are intrinsic to the approach, and will not be tackled. However, the AMN-Reservoir approach previously described in Figure 4.1 lacks generality, and a better workflow can be proposed.

I propose the following workflow for the AMN-Reservoir approach, shown in Figure 4.2. This new workflow brings several advantages: (i) the ML reservoir is not strictly limited to be an AMN-Reservoir, it just has to allow gradient backpropagation, (ii) the ML reservoir has lower and upper bounds for all fluxes as inputs and some simulated fluxes (not the whole distribution) as outputs, these fluxes being

measured in the experimental dataset to process afterwards; (iii) the sampling of the solution space (left part of Figure 4.2, black arrows) can rely on other processes than repeated FBA, *e.g.*, using flux sampling approaches (which could be more performant, as stated in section 4.2.1.1.1).

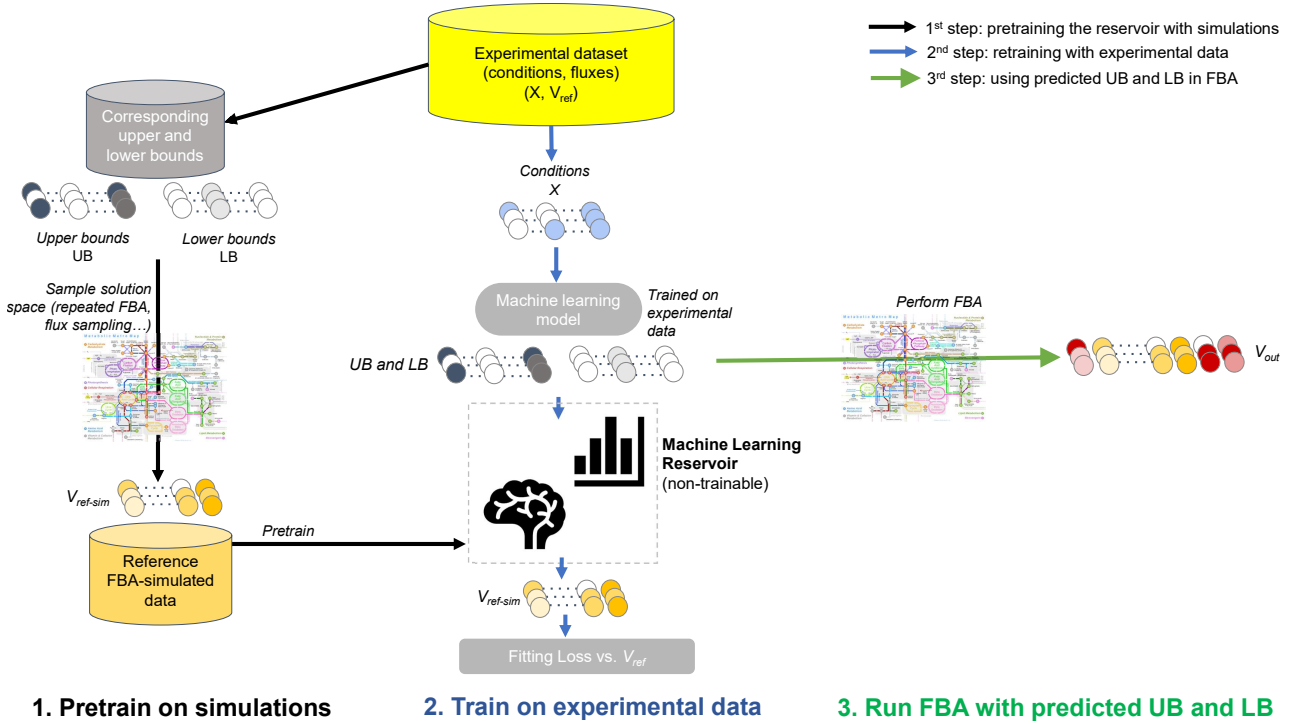


Figure 4.2: Proposition of an improved Reservoir Computing workflow. Compared to the original workflow of Figure 4.1, the workflow shows four main differences: (i) the input of the experimental dataset is not restricted to media compositions, it should simply describe any experimental conditions ( $X$ ); (ii) we derive upper and lower bounds for all fluxes from the experimental dataset instead of upper bounds for uptake fluxes ( $UB$  and  $LB$  instead of  $V_{in}$ ); (iii) the AMN-Reservoir is replaced by a more generic Machine Learning Reservoir that can be based on any method; (iv) this ML reservoir takes as inputs lower and upper bounds for all fluxes ( $UB$  and  $LB$ ) instead of upper bounds for uptake fluxes ( $V_{in}$ ), and outputs only measured fluxes ( $V_{ref-sim}$ ) instead of entire flux distributions ( $V_{out}$ ) like the AMN-Reservoir in Figure 4.1. Moreover, the output  $V_{ref-sim}$  is used in a regular fitting loss with the experimental reference data  $V_{ref}$ . Note that FBA outputs a full flux distribution (on the right,  $V_{out}$ , with yellow fluxes indicating the presence of  $V_{ref-sim}$ ).

In the case we wish to use a ML reservoir that does not allow gradient backpropagation, an alternative approach can be conducted. A learning loop (Figure 4.3), for example based on Active Learning (AL) or Genetic Algorithm (GA) procedures, can link a ML model that predicts upper and lower bounds from experimental conditions (step 1), to FBA performed on a GEM with those bounds (step 2), whose output is used for computing an error metric with experimental measurements (step 3), which in turn is used to update the machine learning model (step 4). This learning loop could provide very reliable predictions, and enable a high diversity of ML models to be tested. However, the mathematical details and examples of implementations will not be discussed in this dissertation.

The novel Reservoir Computing (Figure 4.2) and learning loop (Figure 4.3) workflows are two alternatives that should predict perfectly reliable metabolic phenotypes, in terms of GEMs constraints respect. However, they are not considered hybrid models like AMNs due to their fragmented nature, and 'ad hoc' implementation to make data communicate in the model. In literature, a lot of approaches use such combinations but very few attempt to build hybrid models like AMNs. The following section will describe the closest hybrid model of GEMs formulation found in literature and how we could inspire from it.



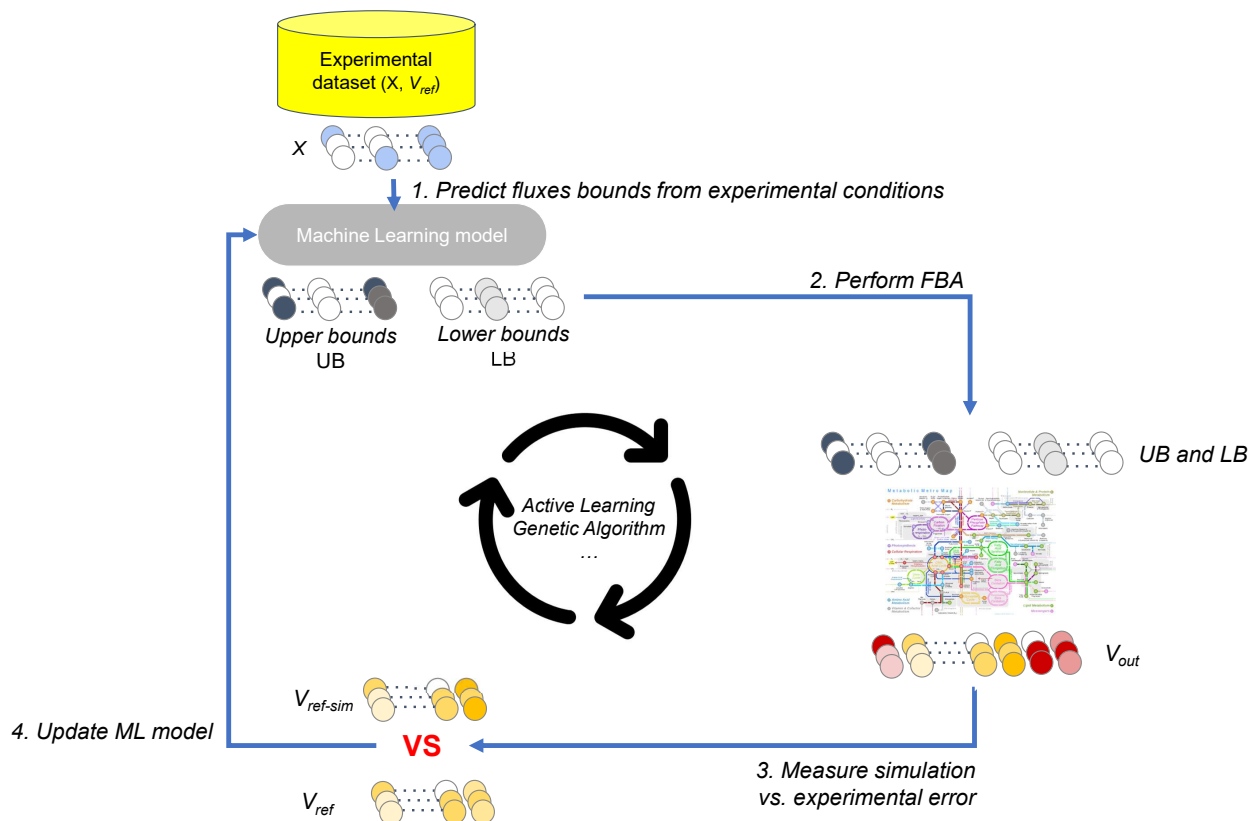


Figure 4.3: Proposition of a learning loop workflow, with FBA and ML separated. From experimental conditions ( $X$ ), a ML model predicts fluxes bounds (step 1),  $UB$  and  $LB$ ; which are used to perform FBA (step 2), producing  $V_{ref-sim}$  (yellow fluxes) as a part of  $V_{out}$ . These simulations from FBA are then compared to experimental measures (step 3), computing an error that is in turn used to update the ML model (step 4).

#### 4.2.1.6 Alternative hybrid modeling of GEMs: comparison of AMNs with existing approaches

The closest works that attempt a hybrid model formulation for GEMs have been directed by Pr. Diego Oyarzun[123, 124]. In these research articles, authors propose to translate a GEM into a mass flow graph (a directed graph with reactions as nodes and chemical mass flows as edges). Then, using wild-type flux data as training, they learn mass flows of the graph, to then predict gene essentiality with the same accuracy as the GEM alone. Although this formulation slightly overlaps with the present work, it differs in the data type that can be used as input, which is wider with AMNs. It also differ in the task that has been designed for the formulations to solve, with mostly uptake fluxes predictions with AMNs, instead of gene essentiality prediction (note that a slight change to the AMN formulation could enable such gene essentiality prediction). Finally, the mathematical approach to hybrid modeling is different, with the use of mechanistic layers and custom loss with AMNs, and no translation of the metabolic network into mass flow graphs. However, we could get inspiration from this formulation in order to improve AMNs: formulating the metabolic network as a mass flow graph drastically reduce the model's size, which can be very useful to ease the functioning of some AMNs formulations, such as AMN-LP with a high number of mechanistic layer iterations.

To my knowledge, no similar hybrid model formulation for GEMs has been formulated. Moving away from GEMs but worth citing is the work of Nilsson et al.[122] for gene regulatory networks. In this work, recurrent neural networks with the same architecture as the GRNs have been developed,



and were shown to be trainable with experimental data and yield satisfactory performances. This approach is similar to the AMN-Wt formulation, so we may find inspiration in this work to improve it.

The advantage of hybrid models over more fragmented and *ad hoc* combinations of ML and GEMs, lies in the single-block nature of their formulations. Indeed, hybrid models are single models, with a clearer mathematical definition, a more concise functioning, and they are more reusable. Therefore, hybrid models seem more promising and ambitious for the future than ML and GEMs combined approaches. However, they have common areas of application, as they are built with the same goal: using the predictive power of ML alongside the mechanistic insights of GEMs. These areas of application will be further described, with a focus on how AMNs can be used for each use case.

## 4.2.2 Areas of application

### 4.2.2.1 Omics data integration

A general, previously mentioned, advantage of AMNs, is their ability to integrate diverse input data types. This perk could be of great help for multi-omics data integration, that can be of many types, and whose integration can be very challenging and require machine learning methods, as underlined by Sahu et al.[52]. For example, with few fluxes measured and many transcriptomic data acquired in different conditions, one could use as inputs of AMN the transcriptomic data and the conditions in which they were acquired, to predict the full metabolic phenotype. Therefore, acquiring new transcriptomics data in new conditions would suffice to predict entire metabolic phenotypes, without requiring the measurement of fluxes. More specific areas of application will be detailed hereafter, that can possibly include -omics datasets integration.

### 4.2.2.2 Data-driven metabolic models curation

Many existing ML and GEMs combined approaches aim to develop new ways of curating metabolic models. For example, DNNGIOR aims to use deep learning for reconstructing metabolic networks of organisms that are poorly described in literature (notably from metagenomic samples)[160]; CHESHIRE aims to find missing reactions of GEMs by hypergraph learning[161]; and Szappanos et al. estimate genetic interaction networks from large-scale phenotyping data and GEMs[117].

One area of application for AMNs could be such data-driven metabolic model curation. By using large amounts of experimental data, and varying the GEM structure to derive different mechanistic layers and custom losses, one could automatically screen the best modifications to bring to the GEM, in order to best match the experimental data. For example, adding promiscuous reactions to the GEM could be automated with such an approach. Also, finding the best coefficients for the biomass reaction, and a more general lower bound for the ATPM reaction (and possibly other fluxes), could be performed. Note that these approaches could work with regular GEMs including metabolic processes only, but also more advanced ones taking into account transcription and translation for example.

For such data-driven metabolic models curation, flux predictions should perfectly respect the GEMs constraint. As seen previously, this is not the case in current implementations of AMNs for large models like iML1515[44]. Therefore, these formulations should be avoided and one should prefer the AMN-Reservoir (Figure 4.2) or learning loop (Figure 4.3) approaches previously described, until further improvements on AMNs are performed.

#### 4.2.2.3 Integration of time-series datasets

Time-series datasets are abundant for kinetic models of metabolism, describing the time dynamics of metabolites concentrations. A good example of a combined ML and GEM model for integrating time-series multi-omics data is the work of Costello and Martin[162]. Here, the ML is used to predict time-series metabolite concentrations, from a time-series proteomics dataset, outperforming a more traditional Michaelis-Menten kinetic model.

For AMNs, one way to integrate time-series seems more accessible, as it does not require kinetic model formulations. Dynamic FBA (dFBA) is a version of FBA where the time series are discretized, each timestep being considered a 'steady-state' solved by regular FBA. With dFBA, only the exchange reactions are controlled by a dynamical model simulating substrate availability dynamics. With a time-series dataset of phenotype measures (growth rate and/or other fluxes) acquired, one could design an AMN where the neural layer is time-series compatible, such as a LSTM layer. This LSTM layer would then predict, for each timestep, a flux vector for a mechanistic layer to produce a final metabolic phenotype. It is yet quite unclear how this would be implemented mathematically, making this approach completely prospective. But, given the high performance of ML models with time-series, it is worth citing this possible area of application, that has great potential.

#### 4.2.2.4 Guiding bioproduction efforts with hybrid models

Many examples of combined ML and GEMs approaches target the optimization of bioproduction as an end goal. For example, tryptophan metabolism was maximized by Zhang et al.[113] using both a GEM to identify genetic perturbation targets, and ML processes to predict the best genetic perturbation experiments to conduct. ML is a natural ally of such engineering projects: the combinatorial space of parameters to tune can be huge, if taking into account physical, chemical, growth media and genetic perturbation parameters. By active learning and other approaches, one can save many resources and find the best combinations to optimize a bioproduction[163]. However, these ML approaches lack a mechanistic insight: this is where AMNs could play a role. Indeed, AMNs could be easily integrated in active learning workflows, and the ability of AMNs to predict unmeasured fluxes could give new metabolic insights to ease and streamline some complex bioproduction optimization processes.

#### 4.2.2.5 Biocomputing with hybrid models

As stated in Chapter 1 section 1.2.2.2, metabolism can be considered to have signal processing abilities. Hybrid models may help to exploit such signal processing abilities for biocomputing. Coming back to Reservoir Computing, described in Chapter 1 section 1.4.1.3.4, the metabolic activity of an organism in response to changing environments could be used as a biological reservoir. This idea originates from a simple analogy between the bucket of water (physical reservoir) and the metabolic activity of *E. coli* (biological reservoir). Hybrid models like AMNs could help to define such biological reservoir abilities, and find the best conditions to expose an organism to, in order for the reservoir to grasp more computing power.

#### 4.2.2.6 Towards a whole-cell hybrid model?

As stated in Chapter 1, the most ambitious endeavor of systems biology is to build performant whole-cell models that would simulate living systems as whole, predicting many biological functions for a wide range of conditions. Efforts towards the development of such models are numerous, such as

the *E. coli* whole-cell modeling project[49], or the 3D minimal cell dynamical model from Thornburg et al.[126]. These ambitious projects integrate an array of processes, from metabolism, gene regulation and transport, to complexation, growth and division. Importantly, the more processes are included, the more they rely on large amounts of parameters that can be hardly determined for many conditions. Moreover, they often suffer from simplifications in order to be computationally tractable: for example, Thornburg et al.'s model only considers glucose as a possible *E. coli* sugar source, which is very far from true.

I personally believe that hybrid models could partially tackle the issues mentioned above. Machine learning methods could ease the integration of diverse omics datasets, in order to parametrize whole-cell models with wide arrays of conditions to generalize from. Pure ML models would not be appealing for building such whole-cell models, since one goal of such formulation is to easily obtain insights on the biological functions of an organism exposed to certain conditions. Therefore, the MM part of a whole-cell hybrid model would be very important: once parametrized through powerful ML approaches, they would be accurately simulating the biological processes included in the model. In a nutshell, my view of an ideal whole-cell hybrid model would be the following: from any environmental and genetic conditions (inputs), a ML model would predict the best set of parameters for many MMs describing different biological processes, that would be merged to produce the final model output as a global behavior of the cell, with macroscopic phenotypic observations (e.g., growth rate, cell average size, motility...) as well as smaller scales behavior (e.g., transport, metabolism, gene regulation...). It would grasp the power of ML to decipher the complexity of omics datasets; in order to efficiently guide mechanistic models, that would make meaningful predictions based on our current knowledge of several biological processes. Such a model would be useful for many engineering tasks, as well as a tool to decipher more mechanisms to include in mechanistic models.

Concretely, how this model would be constructed is still unclear and open for discussion. Still, here is my personal take on what systems biology should focus on in the following years: developing insightful and predictive hybrid whole-cell models, integrating diverse datasets through ML methods, that enables the accurate simulation of more and more detailed small-scale biological processes, as well as larger scale phenotypic traits. Importantly, I believe the evaluation of such whole-cell models should be standardized, in order for the community to better grasp the capabilities of each model. In the same idea, a similar competition as for protein folding (CASP[164]) could emulate the advances of whole-cell models, eventually leading to a breakthrough like AlphaFold[96].

## 4.3 Concluding remarks

### 4.3.1 Chapters summaries

In Chapter 1, I have described the fascinating complexity of biological phenomena that are modeled by Artificial Metabolic Networks (AMNs). Then I delved into genome-scale metabolic models and the essential tools to exploit them, with a focus on flux balance analysis limitations. Finally, I introduced two captivating modeling realms, mechanistic modeling and machine learning. After a general description of both approaches, I reviewed combinations of the two for state-of-the-art metabolic modeling. Lastly, I described the emerging hybrid modeling field that aims to push such combinations one step further. Overall, I present the research gap filled by AMNs, and the motivation to develop them. These hybrid models fill a crucial research gap, pushing the boundaries of what we can achieve with genome-scale metabolic models.

Chapter 2 takes us on a journey to discover the basic concepts and mathematical formulations of AMNs, as well as their capacity to outperform flux balance analysis. First, I have shown that innovative mechanistic methods can surrogate flux balance analysis and blend with artificial neural networks, forming the basis of the AMN model. With concrete examples of experimental datasets, I have witnessed AMNs surpass traditional flux balance analysis in predictive power. Also, an AMN-Reservoir approach was performed to help us fine-tune uptake fluxes to match experimental data more effectively. Overall, AMNs were found to significantly enhance GEMs' predictive capabilities.

Chapter 3 is where we put AMNs to the test. I have explored various limits, starting with the QP-solver missing key GEM constraints and depending on a subtle balance of its loss terms. Mechanistic layers, though promising, currently increase computation time without substantial benefits for large genome-scale metabolic models. Our custom mechanistic loss has shown room for improvement, especially concerning the  $L_2$  (for ensuring  $SV = 0$ ) term, probably pushing predictions towards zero flux values. We should also keep in mind that AMNs' performance is highly reliant on training set structure and content, be it real-world or simulated data. Yet, I have uncovered avenues for improvement, such as optimizing AMNs' hyperparameters and exploring alternative ML systems like the multi-task elastic net. Finally, I assessed that the AMN-Reservoir approach, while promising to tackle major issues of AMNs, does not fit the precise definition of 'hybrid model' as defined in Chapter 1.

The present Chapter 4 wraps it all up. I have summarized AMNs' capabilities and limitations. Importantly, we must acknowledge a significant challenge: the current AMNs implementations struggle to reliably predict fluxes that conform to GEMs' constraints. To overcome this shortcoming, I have outlined numerous paths for improvement, like standardized training set generation, enhanced loss and mechanistic layer formulations, and a refined Reservoir Computing approach. I have also identified exciting areas where AMNs could shine, from automated metabolic model curation to optimizing bioproduction, and eventually the realm of whole-cell hybrid models.

### 4.3.2 Answers to scientific questions of Chapter 1 section 1.5

Overall, we have embarked on a journey exploring an innovative way to harness the power of GEMs: a hybrid neural-mechanistic approach, with machine learning and flux balance analysis at its heart. I will now give short answers to the scientific questions I formulated in the last section of Chapter 1.

1. Can we develop AMNs that increase the GEMs predictive power while respecting their constraints?

Even though the predictive power of AMNs is impressive when compared to FBA alone, it still needs further investigations to more reliably respect large GEMs constraints, and eventually be used by the metabolic modeling community as a standard method.

2. What are the future improvements and most suited areas of application for AMNs?

Many improvements in that direction can be made, so I believe that hybrid models of GEMs have great potential in a vast array of projects, from metabolic model curation to bioproduction optimization, and even for the development of more performant whole-cell models, the holy grail of systems biology.

### 4.3.3 Final words

To end this dissertation, let us have a step back on the research work that was presented here, to conclude with a more biology-centered, non-technical point of view.

Billions of years of evolution have led living organisms to rely on processes of amazing complexity. A myriad of molecular mechanisms act in concert, so that living systems constantly adapt to changing environments, survive and reproduce. Naturally, the human quest to understand life led to ways to handle such complexity.

Two main modeling approaches are seemingly opposed, but share the purpose of deciphering complexity. One attempts to gather pieces of knowledge and build models of mechanisms; whereas one attempts to gather high numbers of observations and build statistical models. The first is overambitious given our current knowledge of biological processes, whereas the second may lack insights for understanding underlying mechanisms. Therefore, this original research work has shown an attempt towards a merging of both approaches.

This hybrid approach, as it was formulated, is appropriate for predicting metabolic phenotypes from environmental and genetic conditions. Indeed, the intricate and highly complex mechanisms between an organism environmental and genetic resources, and its metabolic activity, are not very well understood. Consequently, we chose to rely on statistical observations to model these high complexity phenomena.

However, the past 200 years of biochemistry research has accumulated extensive knowledge about the possible metabolic reactions happening in an organism: complete metabolic maps of model organisms like *E. coli* can nowadays be exploited. Therefore, we include such knowledge of metabolism in our hybrid approach, to re-use and exploit 200 years of research in the best possible way.

Even if we cannot hope to model some extremely complex biological processes like consciousness with modern machine learning techniques, they have shown amazing capabilities, like the accurate prediction of protein structures by AlphaFold[96]. This is inspiring to formulate whole-cell hybrid models, able to simulate diverse biological activities from a vast array of conditions; by grasping the power of modern machine learning and leveraging tremendous amounts of -omics datasets.

The future of biological research will be computer-guided, and the potency of Artificial Intelligence in more and more complex tasks appeals for pursuing AI integration within biological models. Here, I have presented a step towards that path, which will eventually lead to a better understanding and exploitation of organisms. Hopefully, this could accelerate biotechnological and medical breakthroughs, for a safer and more sustainable future.

# Appendix A

## Supplementary Information for Chapter 1

### A.1 Historical breakthroughs in metabolism research: from single enzymes to complete metabolic maps

This first subsection will recall the historical metabolism research breakthroughs, to underline the tortuous research path that brought humanity from unknowingly exploiting microorganisms' metabolism, to the modern days wide and deep understanding of metabolism. A chronological timeline schematic is displayed in Figure A.1, which the reader can refer to for a visual summary of this subsection.

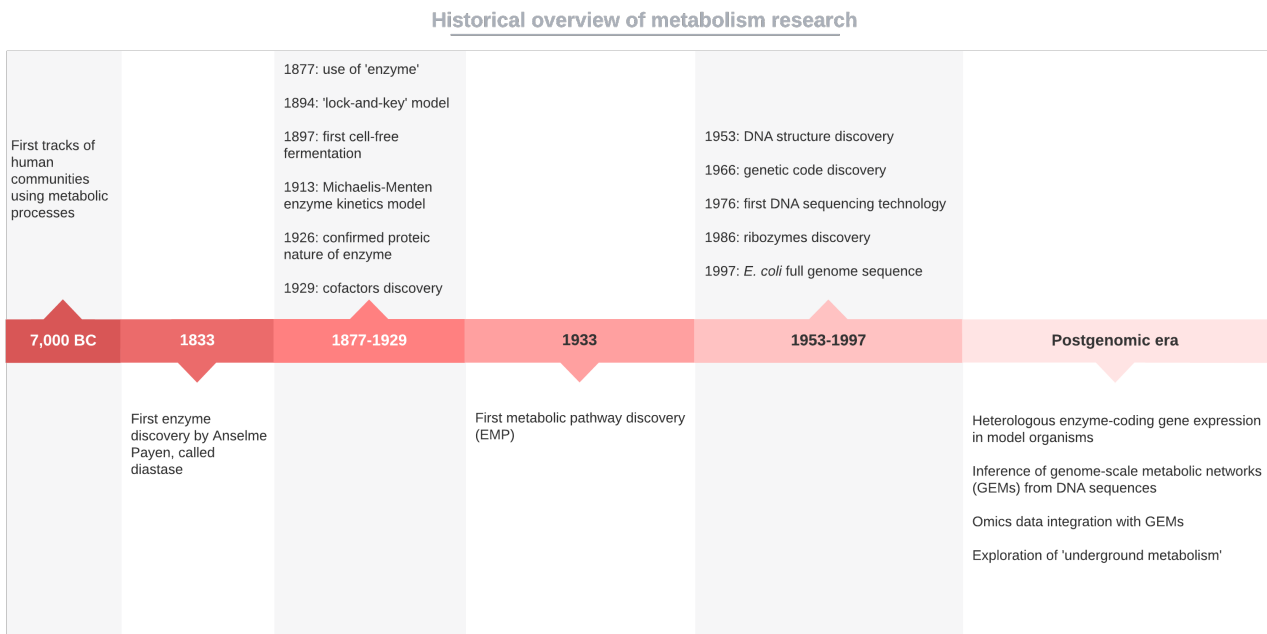


Figure A.1: Timeline of historical breakthrough in metabolism research, and modern research approaches in the postgenomic era.

#### A.1.1 Discovery of enzymes

The first tracks of human communities using metabolic processes dates back to 7,000 BC<sup>[165]</sup>. At the time, fermented beverages and foods were produced unknowingly of the metabolic activity of

microorganisms. Today, microorganisms are engineered for many industrial applications, in an increasingly standardized and rational way[33]. This first section aims to summarize how scientists built their knowledge of metabolism, and present the state-of-the-art research on metabolism.

A first important milestone is the discovery of the first enzyme - at the time termed 'ferment' - by the french chemist Anselme Payen, in 1833[166]. He called 'diastase' a mixture of germinating barley, after observing its catabolic properties: transforming starch into sugars. This was quickly followed by more enzyme discoveries, among which pepsin - breaking down proteins in peptides - and invertase - hydrolyzing sucrose into fructose and glucose[31]. In 1877, Wilhelm Kühne coined the term 'enzyme', originating from ancient Greek and meaning 'in yeast'. Shortly after, the first model of enzyme-substrate interaction was formulated by Emil Fischer in 1894: the widely known 'lock-and-key' conception explaining substrate specificity of enzymes. Surprisingly, at the time, most scientists believed that enzymes were performing their activity thanks to a 'vital force' inside microbes, notably from Louis Pasteur research and the 'vitalist' movement. That was until Eduard Büchner, in 1897, showed that a dead yeast extract could perform the same transformation as a live one, putting an end to the 'vitalist' view of enzyme activity. Note that Büchner's work is therefore the first cell-free fermentation in history, rewarded by a Nobel prize. In 1913, extending the lock-and-key model to a more quantitative enzyme activity model, a foundational work was proposed by Maud Menten and Leonor Michaelis: the Michaelis-Menten kinetic model. This model was slightly refined over the years, and is still used in many modern kinetic models of metabolism. This formulation introduced important thermodynamic constants for enzyme kinetics, that shall be quickly reminded here:  $K_m$ , the Michaelis constant of an enzyme, indicating the concentration of substrate for which the reaction speed is half of its maximum ( $\frac{V_{max}}{2}$ ); and  $k_{cat}$ , the turnover number or catalytic constant of an enzyme, indicating the maximal number of substrate catalyzed by one enzyme in one second. Around the same period, the first cofactor (called 'cozymase' at the time, later known as NAD) was discovered by A. Harden and H. von Euler-Chelpin, rewarded by a Nobel prize in 1929. This work shed light on the dependency of many enzymes on small molecules and ions. However, the nature of enzymes themselves was still unclear at that point, until James B. Sumner deciphered a protein structure from crystallizing urease in 1926, leading to a consensus on the proteic nature of enzymes. It took several decades before the community realized that some biocatalysts were protein-RNA complexes, or RNA alone in the case of ribozymes, discovered by S. Altman and T. Cech and rewarded by a Nobel prize for their work[167]. A typical enzyme 3D structure, Glucosidase, is represented in Figure A.2, with its catalytic activity, active site, substrate and products displayed.

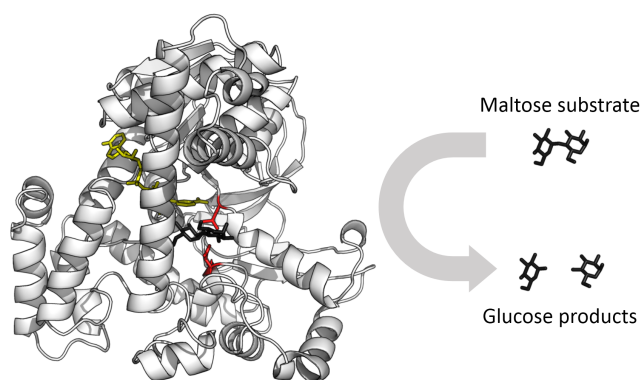


Figure A.2: Glucosidase 3-dimensional structure, catalyzing the hydrolysis of maltose into two glucose sugars. Active site residues are displayed in red, maltose and glucose in black, and the NAD cofactor in yellow. (Author: Thomas Shafee, License: CC BY 4.0)



### A.1.2 Discovery of metabolic pathways

Discovering enzymes naturally led to the discovery of metabolic pathways, the first one being glycolysis, i.e. a very common pathway breaking down 1 glucose into 2 pyruvate molecules. In the early 1900s, Louis Pasteur was the first to get hints on the glycolysis process, notably observing glucose consumption reduction when *Saccharomyces cerevisiae* (baker's yeast) was put under aerobic conditions. It was a joint effort of many scientists to decipher the glycolysis puzzle, each discovery unraveling a small part of the pathway. Among those scientists, the most important ones gave their name to the glycolysis: the Embden–Meyerhof–Parnas (EMP) pathway. Embden was the first to propose a full description of glycolysis, in 1933, based on all the previous work from the community[168]. Importantly, today's use of the term 'glycolysis' may extend to other pathways that break down glucose in pyruvate, such as the Entner-Doudoroff (ED) pathway. A description of the EMP glycolytic pathway is shown in Figure A.3.

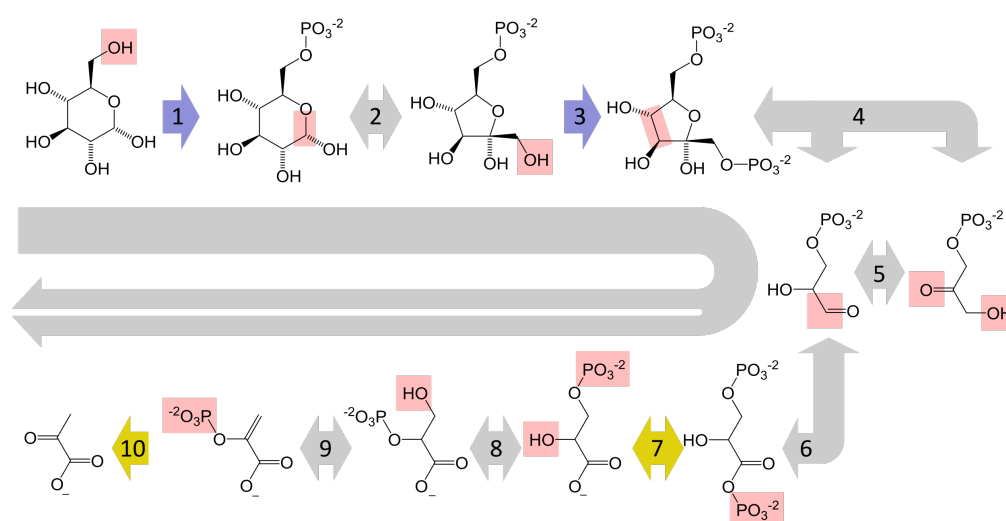


Figure A.3: Glycolysis: the first metabolic pathway discovered. Conversion of glucose to pyruvate is performed by enzymatic reactions (arrows) that consume (purple) or produce (yellow) ATP. Such enzymatic reactions induce chemical modifications (red boxes) on intermediate metabolites, from glucose (top-left) to pyruvate (bottom-left). Steps 6 to 10 are occurring twice per glucose molecule. (Author: Thomas Shafee, License: CC BY 4.0)

Over time, the metabolism research community efforts for (i) identifying enzyme structures and functions, and (ii) identifying metabolic pathways (i.e., chained series of enzymatic reactions found in organisms) gradually expanded our knowledge of metabolism. This accumulation of knowledge was accelerated by the 'DNA revolution', starting after Watson and Crick's discovery of the DNA structure in 1953[31]. After that, the genetic code was deciphered in 1966 and DNA sequencing technologies were developed, with F. Sanger who paved the way in 1977. The first complete genome sequence of *E. coli* was published in 1997[169], which follows genome sequences of a bacteriophage in 1976 and other bacteria since 1995. This period marks the beginning of the post-genomic era, which revolutionizes biology, including the research on metabolism. Importantly, the ability to isolate a DNA sequence coding for an enzyme, and clone it in model organisms such as *E. coli*, enabled scientists to further characterize individual enzymes in a more precise way. Later, sequencing tremendous amounts of DNA, and relating them to enzymes, enabled the development of bioinformatics tools that can infer metabolic networks from genome sequences[1]. Eventually, the accumulation of enzyme and pathway



data brought up challenges on how to organize and exploit this knowledge.

### A.1.3 Metabolism at the organism scale

As stated before, more and more investigations into diverse enzyme catalytic activities were performed. However, the community lacked a standardized way of classifying and naming enzymes, until 1961. At that time, the Enzyme Commission (EC) numbers were created in order to classify enzymes in a standardized way[170]. The classification was then iteratively refined, the last update in 2018 adding a 7th category of enzymes, translocases. An EC number is a 4-part hierarchical numerical ID, each number narrowing down the capabilities of the enzyme: the main class (oxidoreductases, transferases...), subclass, sub-subclass and serial number (a unique identifier for an enzyme). Furthermore, enzymes are now cataloged in information-rich databases such as the Braunschweig Enzyme Database (BRENDA)[171].

Many metabolic pathways characterizations followed the EMP pathway, and connections between pathways were uncovered. This eventually led to more and more complete metabolic maps of more and more organisms. Figure A.4 displays the major metabolic pathways and how they are connected, agnostically of the organism it can be found in, conceptualized as a metro map. This gives a general idea of the complexity, modularity, and communality of pathways across all living organisms. Nowadays, metabolic pathways are cataloged in databases such as BioCyc[172] or the Kyoto Encyclopedia of Genes and Genomes (KEGG)[173]. The wide and deep understanding of metabolism, and the extensive description of metabolic pathways, led to more and more complete metabolic models, sometimes described at the genome-scale and called genome-scale metabolic models (GEMs or GSMMs). These models are also stored in databases, such as BiGG[136].

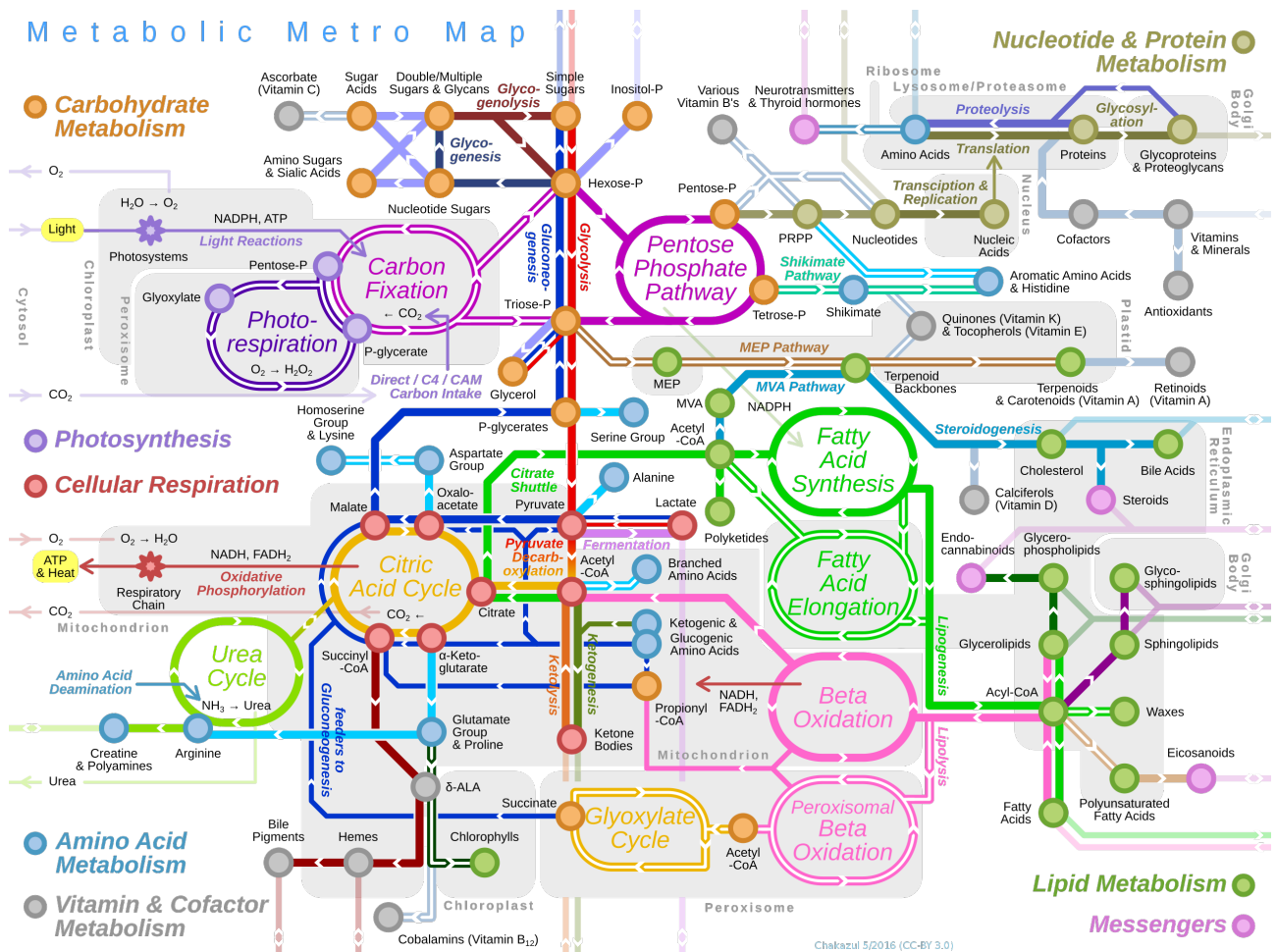


Figure A.4: 'Metabolic metro map' representation of the major metabolic pathways (Author: Chakazul, License: CC BY 4.0, 2016)

### A.1.4 Questioning our knowledge of metabolism

The 'written-in-stone' nature of metabolic networks, based on evolution-driven strong enzyme-substrate specificity, has been recently challenged. Researchers have shown that evolution has driven metabolic networks to more complexity and connectivity[39]. Such complexity is hypothesized to bring more robustness (to genetic variations) and adaptation (to changing environments or diverse ecological niches) abilities to the network. Interestingly, studies suspect that primitive enzymes were more promiscuous than modern 'evolved' enzymes[174, 175]. Recently, enzyme promiscuity was found to be a more widespread phenomenon than previously thought, questioning the lock-and-key model of enzyme-substrate specificity. From these results, some metabolic models attempt to include non-canonical enzymatic reactions that happen in organisms (extending the 'classical' metabolism with 'underground' metabolism)[176].

Since the acceptance of the central dogma of molecular biology, metabolism has most often been considered as an 'end-point' phenotype[177]. Indeed, with most enzymes being proteins and proteins being at the end of the information flow in this central dogma, metabolism would be mostly controlled by genetic regulation processes. However, after 50 years of research, the central dogma proposed by Crick and his peers is challenged. Proteins are now known to strongly interact with DNA replication, transcription of DNA in RNA, and translation of RNA in proteins, as shown in the 'new central dogma of molecular biology' proposed by Change Tan et al.[178]. A schematic representation of this new dogma is displayed in Figure A.5.

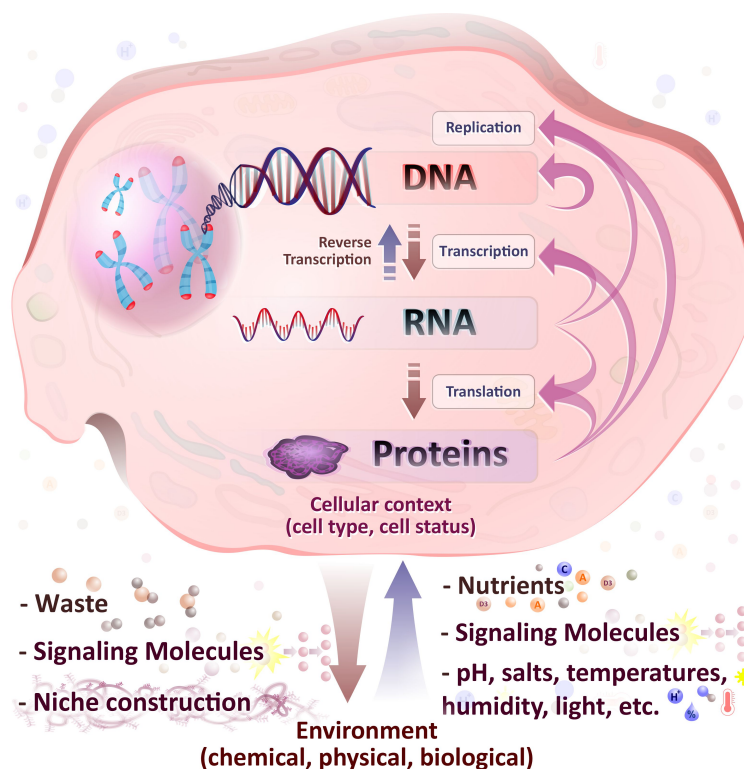


Figure A.5: Schematic view of the ‘New central dogma of molecular biology’ proposed by Tan and Anderson[178]. Dashed arrows represent sequence information transfer, curved purple arrows represent the kind of molecules needed for the corresponding information transfer, triangular bottom arrows represent the interactions between the cell and its environment. (Authors: Tan and Anderson, reused with the permission of the authors. Unpublished work - no editors concerned by copyrights).

## A.2 GEMs reconstruction process

### A.2.1 Automated or manual curations depend on the GEM’s scope

Genome annotations enabled the detection of metabolic reactions in an automated fashion, directly from DNA sequences, mostly through the gene-protein-reactions (GPR) rules functional annotations. In some cases when the organism is widely and precisely described in literature, manual curation can be used in order to increase the ‘certainty’ of the GEM. Briefly, manual curations consist in manually adding, removing, or changing parts of a GEM, based on expert knowledge. Unfortunately, except for a few microorganisms such as *E. coli* (having the most precisely reconstructed GEM as of today), one cannot rely on many manual curations. Therefore, most of the GEMs reconstructions rely on automated genome annotation, to find the corresponding metabolic reactions possibly happening in an organism. Recent studies also propose deep learning-powered GEM reconstruction methods from incomplete genomes (especially useful in metagenomic samples, for example)[160]. Importantly, some GEMs reconstruction rely on more diverse data than genome annotations, since they contain more information than just the stoichiometric matrix representing the metabolic network. In practice, what kind of data will be used depends on the designed GEM’s scope.

As stated in Chapter 1’s Preamble (section 3.1), the delimitation of what is considered metabolism necessarily relies, in the end, on a subjective choice. This has consequences on how we conceive metabolism, and when considering what to include in GEMs as well. In the most recent GEMs reconstruction, not only the small molecule transformations by enzymes have been integrated, but also transcription, tRNA charging, and translation reactions, thus enlarging the usual scope of tradi-

tional GEMs to macromolecular processes[54]. Moreover, promiscuity is more and more considered in the metabolism research community and this brings GEM formulations including the ‘underground metabolism’[175, 179]. As stated in the previous subsection, there is also a great diversity of GEMs computing frameworks available. Consequently, this opening of GEMs and their computing frameworks to a wider conception of metabolism requires the integration of more diverse datasets, such as phenotypic stress responses including protein folding sensitivity to temperature or metalloprotein oxidation and repair, just to name a few (the diversity of data to integrate is, effectively, quite endless). Increasing the ‘completeness’ of GEMs has been the main endeavor of the GEM reconstructing community. Indeed, the latest GEM of *E. coli* called iML1515[44], released in 2017, is very different from the 2003-released iJE660. The most striking difference is that it contains 1,515 open-reading frames (i.e. RNA-encoding genes), more than doubling the amount from the 2003 version. New genes notably include periplasm-related processes, reactive oxygen species processes, and expanded cofactor metabolism.

### **A.2.2 Refining the previous GEM: enhancing the stoichiometric matrix and metadata links**

To give a concrete and detailed example of the GEM reconstruction process and global content, I will now describe how iML1515[44] has been reconstructed, and what it contains precisely[44]. This model emerges from iterative refinement that started in 2003 with iJE660, each new version pushing the previous one to a new level of completeness. During the last refinement step, to build iML1515[44], the previous reconstruction (iJO1366) was analyzed and 54 of its reactions were identified and corrected, from enzymatic assays and other experimental data. Also, new gene functions were added by the so-called model-driven gap-filling approach, which is very important in GEM reconstruction and refinement. It basically consists in finding disagreements between model predictions and experimental observations, most often by automated algorithms interpreting growth assays[180]. With this approach and literature reviews, 184 genes and 196 reactions were added. Also, iML1515[44] displays new metadata links for each gene to the protein structure accession in the Protein Database (PDB), enabling a possibly deeper investigation of the gene-protein-reaction associations (GPR), up to the catalytic site domain level. Previous points are the main improvements proposed by the iML1515[44] reconstruction; other performance assessments can be found in the original study. For further descriptions of its general content, please refer to Chapter 1 section 1.3.2.

## **A.3 Pathway analysis: deriving topological insights from metabolic networks**

Analyzing the topology of pathways can lead to insightful conclusions and novel hypothesis formulations on the metabolic networks capabilities, and can provide powerful tools to compare metabolic networks together. Pathway analysis of metabolic networks rely on convex analysis, a mathematical domain studying linear inequalities systems, which is the case of metabolic networks. Indeed, with constraints of equations 1.1 and 1.2, one can apply convex analysis algorithms. Note that for such algorithms to function,  $LB_i$  must be all positive (i.e. the reactions must be unidirectional, which is done by duplicating bi-directional reactions). The main algorithms worth citing here aim to enumerate Elementary Flux Modes (EFMs) and Extreme Pathways (EPs). EFMs are the set of all possible paths in the

metabolic network, i.e. the set of valid steady-state solutions (solely based on network topology, not on actual flux values). Consequently, EFMs actually constitute the convex solution cone that satisfies equations 1.1 and 1.2. Note that EFMs can be enumerated with the constraint of including a specific reaction in the network. Visual representations of a convex solution cone for 3 fluxes is displayed on Figure 1.6. The EFMs enumeration algorithm is computationally intensive and can return a very large number of elements[1]. A less computationally intensive enumeration algorithm searches for EPs. EPs are defined as the unique and minimal subset of EFMs that are needed to compute any EFMs (with linear combinations of EPs). They are therefore the convex basis vectors, i.e. the limits of the convex solution cone, represented by arrows on figure 1.6, panel (c). Tackling the issue of the increasing size of GEMs, some more sophisticated approaches such as Singular Value Decomposition (SVD) were developed to handle large sets of EPs for pathway analysis[181]. EFMs and EPs have been extensively used to analyze topological properties of metabolic networks. The conclusions brought by these analyses range from minimal medium predictions, to evaluation of pathway redundancies (equivalent to network robustness), or the characterization of correlated reactions or metabolites. Note these analyses are universal and only depend on the metabolic network topology, unlike most of the approaches reviewed in Chapter 1 section 1.3.3 that involve experimental measurements.

## A.4 Open discussion on MM and ML capabilities

### A.4.1 Two seemingly opposed approaches

A terminology designating ML and MM models is particularly evocative of its core difference: MMs are often designated as white-box models, whereas ML models are often designated as black-box models. This terminology underlines the fact that, in MMs, we construct the model with *a priori* knowledge of the system, and each component is clearly pre-defined based on expert knowledge rather than instrumental observations. In contrast, ML are built without *a priori* knowledge and entirely rely on *a posteriori* instrumental observations to train the model. In other words, MMs are knowledge-driven whereas ML models are data-driven (they are sometimes referred to as ‘models of mechanisms’ and ‘models of data’). In this subsection I will review core differences between MM and ML that make them appear as seemingly opposed, and focus on these differences in the scope of metabolic modeling.

On one hand, MMs can be built only based on knowledge, e.g. through manual curations that emanate from literature reviews. Moreover, MMs can be used without datasets, and enable us to draw insightful conclusions or test hypotheses quantitatively, as described for metabolic networks in the previous section on pathway analysis, for example. On the other hand, ML models are usually built without using any form of knowledge. Consequently ML models always require training sets, i.e. observations of the phenomenon to model; and this data acquisition is critical regarding the ML model capabilities and scope. Indeed, the training set acquisition does require some insight (or at least some intuition) on the phenomenon that is modeled, given the full reliance of an ML model on its training set. If the wrong measures are made to define the training set, the ML model may have no predictive power, or predict the phenomenon from the wrong angle. A trivial example would be to attempt to build a ML model for weather forecasting, and train it with data of vehicle speeds; or build a ML model for lung cancer development prediction based on coffee drinking data. Even if some predictive power could be obtained, the relationships that could be learned do not imply a direct causality, but rather a confounding factor. This issue is very important: ML models being solely based on deciphering statistical patterns, it relies on the data acquisition process and input/output segmentations to have

a meaningful causality to be modeled. On the other hand, the interpretability of MMs is very good as they are based on predefined components that have a meaning and purpose[87]. But, as stated above, the predictive power of MMs is in general quite low compared to ML models. This is due to the *a priori* knowledge used to build MMs that is much constraining. Indeed, MMs are less flexible as the *a priori* knowledge can limit the types of inputs, the mathematical structure, and the types of outputs that the model can handle.

#### **A.4.2 MM and ML for systems biology**

The very goal of MMs and ML models is different: the former aim to understand underlying mechanisms of a system, while the latter aim to faithfully reproduce the behavior of a system without understanding it. Coming back to the endeavor of systems biology to understand life as a whole, including with GEMs, MMs and ML models offer two very different paradigms. We have seen that the traditional reductionist approach cannot grasp the complexity of biological systems when considered as whole, mainly because of unknown emerging properties. Consequently, MMs will by essence lack some components to have good predictive power. In contrast, ML models will have a better predictive power, but they will lack any component that helps the understanding of the system. A key difference between MM and ML is the paradigm of scientific method they are based on: while MMs focus on deductive, hypothesis-driven discoveries, ML focus on inductive, data-driven discoveries. However, efforts to make these two paradigms of reasoning conciliate were formulated, as it seems necessary in the postgenomic era[182]. Indeed, Kell and Oliver propose as early as 2004, a synergy between the inductive paradigm, that brings insights based on observations, and the deductive paradigm, that shall test such putative insights through hypothesis-driven discovery. Moreover, they suggest that similar synergy should be attempted between reductionism and holism, when modeling complex systems such as life at the organism scale.

In systems biology, it is an active debate whether the modeling community should direct their efforts towards MMs or ML. According to some modelers, MMs should be preferred as they are enabling the identification of causes and effects, and are more compatible with the classical scientific method (hypothesis-driven research). This is the case of R. D. Phair who writes that “modeling thrives on the unknown and does not require that we know all the parts”[87]. Others warn of the possibly overambitious endeavor to use MMs for systems as complex as life, since we do not have enough knowledge yet of the different components and their interactions. And some propose new paradigms merging MM and ML[88, 162, 183], which are further described in Chapter 1, section 1.4.2.



## **Appendix B**

### **Supplementary Information for Chapter 2**



# A neural-mechanistic hybrid approach improving the predictive power of genome- scale metabolic models

Léon Faure<sup>1</sup>, Bastien Mollet<sup>2,3</sup>, Wolfram Liebermeister<sup>4</sup>, and Jean-Loup Faulon<sup>1,5,\*</sup>

<sup>1</sup>MICALIS Institute, INRAE, AgroParisTech, University of Paris-Saclay, Jouy-en-Josas, France, <sup>2</sup>Ecole Normale Supérieure, ENS-Lyon, Lyon, France, <sup>3</sup>UMR MIA, INRAE, AgroParisTech, Université Paris-Saclay, Palaiseau, France, <sup>4</sup>MaIAGE, INRAE, University of Paris-Saclay, Jouy-en-Josas, France, <sup>5</sup>Manchester Institute of Biotechnology, University of Manchester, Manchester, UK.

\*Corresponding author: [Jean-loup.Faulon@inrae.fr](mailto:Jean-loup.Faulon@inrae.fr), ORCID [0000-0003-4274-2953](https://orcid.org/0000-0003-4274-2953)

## Supplementary Information

<b>Wt-solver equations</b>	<b>2</b>
Figure S1. Computing steady-state fluxes with the Wt-solver	4
<b>AMN-Wt architecture</b>	<b>5</b>
Figure S2. Different weights for different uptake fluxes provided by exact bounds (EBs)	5
Figure S3. Branch point metabolite flux ratio	6
<b>LP-solver equations</b>	<b>8</b>
Figure S4. Matrices used with the LP (EB) method	11
Figure S5. Matrices used with the LP (UB) method	12
<b>QP-solver equations</b>	<b>13</b>
<b>MM solvers benchmarking</b>	<b>16</b>
Figure S6. MM solvers architectures and performances	16
<b>AMNs benchmarking varying hyperparameters</b>	<b>17</b>
Figure S7. Hyperparameters for AMN's neural Layer	17
<b>AMNs benchmarking with independent test sets and additional metabolic models</b>	<b>18</b>
Table S1. Benchmarking MMs, ANNs and AMNs	18
<b>AMNs benchmarking with gene knockouts and multiple measured fluxes</b>	<b>21</b>
Figure S8. AMN performance on multiple fluxes dataset	22
<b>AMN-Reservoir prediction performance</b>	<b>23</b>
Figure S9. Performances of AMN-Reservoir using predicted $V_{in}$ as input to FBA	23
<b>ANN training set sizes</b>	<b>24</b>
Figure S10. Loss and regression coefficient for training sets of increasing sizes	24
<b>Experimental workflow</b>	<b>25</b>
Figure S11. Experimental workflow pipeline	25
<b>Terminology</b>	<b>26</b>
Table S2. Vectors and matrices notations used in figures and equations	26
<b>Supplementary references</b>	<b>28</b>

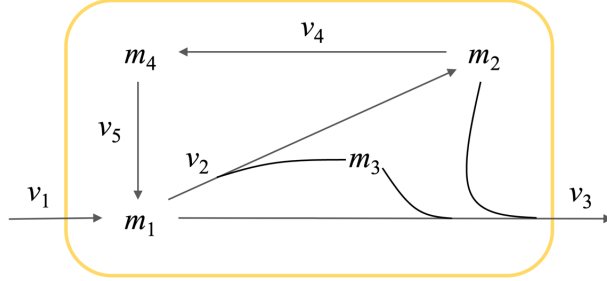
## Wt-solver equations

The Wt-solver equations, inspired by the work of Nilsson *et al.*<sup>1</sup> for signaling modeling, are detailed below for the specific example given in Figure S1.

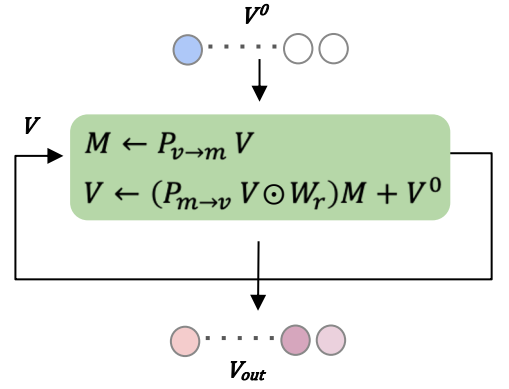
With the usual FBA method (Figure S1.a), we obtain the flux distribution maximizing the flux of the  $v_3$  reaction (representing a classical ‘biomass’ reaction) with an uptake reaction  $v_1$  by solving a linear program (Figure S1b).

In the Wt-solver method, we first translate the model into a neural-network-like architecture (Figure S1c). Precisely, in Figure S1c we start from an initial set of given fluxes ( $v_1 = 0.1$ ) and then propagate knowledge about the fluxes through the entire network, each layer corresponding to one step in a discrete flux propagation. Mathematically, each layer is composed of two simple operations that update the  $M$  and  $V$  vectors, respectively representing metabolites production rates and reaction fluxes. Those operations are repeated until convergence. The network shown in Figure S1c is the unrolled representation of an RNN-like network depicted in Figure S1d. The matrices used in the Wt-solver are given in Figure S1e. The weight matrix  $W_r$  can be computed from the steady state flux values if they are known, or learned through training from provided reference fluxes (*cf.* section ‘AMN-Wt architecture’). For instance, taking the example of Figure S1, the weight matrix  $W_r$  can be either computed from steady state fluxes, computed by FBA (*cf.* legend of Figure S1f), or learned after setting  $v_1 = 0.1$  and searching weights for which  $v_3 = 0.5$ . We note, the steady state fluxes after iterating 30 times (or more, here we stopped at 30 because the reference data values were reached) are equal to those obtained in the reference data (*cf.* Figure S1 panels b and f).

**a**



**d**



**b**

**Solve with Linear Programming:**

$$\text{Max}(c^T V)$$

Subject to:

$$SV = 0$$

$$0 \leq V \leq V_{in}$$

where:  $V_{in} = (0.1, -, -, -, -)^T$

$$V = (v_1, \dots, v_5)^T$$

$$c = (0, 0, 1, 0, 0)^T$$

$$\text{and } S = \begin{pmatrix} v_1 & v_2 & v_3 & v_4 & v_5 \\ 1 & -1 & -0.13 & 0 & 1 \\ 0 & 1 & -0.07 & -1 & 0 \\ 0 & 1 & -0.36 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{matrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{matrix}$$

**Steady state solution:**

$$V_{out} = (0.1, 0.18, 0.5, 0.15, 0.15)^T$$

**e**

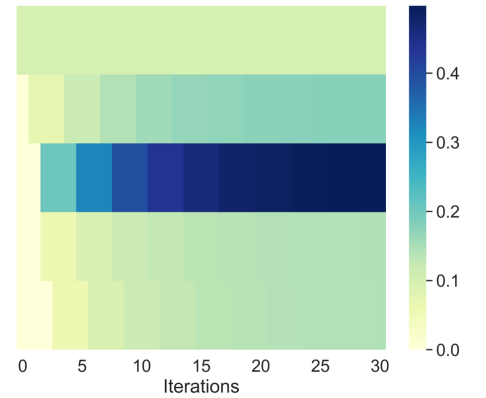
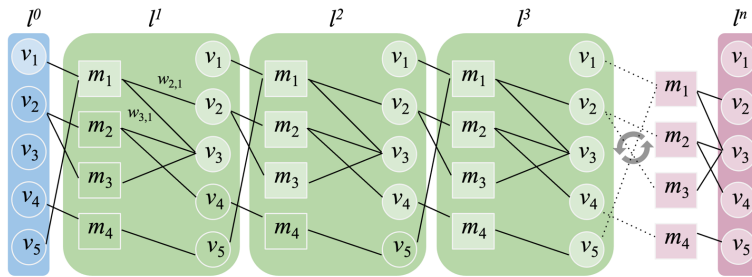
$$P_{v \rightarrow m} = \text{ReLU}(S) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$P_{m \rightarrow v} = \text{ReLU}\left(\left[\frac{-1}{z_i s_{j,i}}\right]\right) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0.39 & 0.21 & 1.08 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

**f**

$$W_r = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.74 & 0 & 0 & 0 \\ 0.26 & 0.20 & 1 & 0 \\ 0 & 0.80 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

**c**



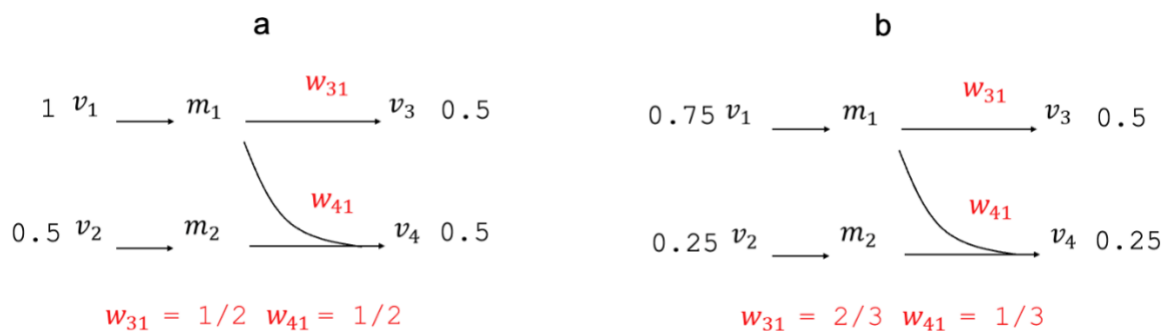
$$V^{30} = (0.1, 0.18, 0.5, 0.15, 0.15)^T$$

### Figure S1. Computing steady-state fluxes with the Wt-solver

**a.** Simple toy stoichiometric model. The model is unidirectional, all flux values are positive.  $v_1$  represents a nutrient uptake flux and  $v_3$  the objective (e.g. the biomass flux). This toy network was inspired from the upper glycolysis pathway found in iML1515<sup>2</sup>, downloaded from BiGG<sup>3</sup>. **b.** Steady-state solution fluxes maximizing  $v_3$ . At steady state, the reaction fluxes ( $v_i$ ) must satisfy stationarity conditions that guarantee mass balance of all metabolites, this is depicted by the equation  $SV = 0$ , where  $S$  is the stoichiometric matrix representing the connectivity of the model and  $V$  the vector of fluxes to be calculated.  $V_{in}$  is the medium represented by a vector of nutrient uptake fluxes (here  $v_1 = 0.1$ , symbol “—” indicates that no value is provided, in practice one uses an ‘infinity’ value to represent an unbounded flux). The steady-state solution  $V_{out}$  is calculated by solving a linear program maximizing the objective  $c^T V = v_3$ , here the Cobrapy package was used to compute  $V_{out}$ , by making use of a Simplex-solver algorithm. **c.** Unrolled neural network built from the stoichiometric model. In the initial layer ( $l^0$ ) only  $v_1$  has a value. In layer 1,  $v_1$  value is passed to  $m_1$ , the production flux for metabolite  $m_1$ . Subsequently a fraction ( $w_{21}$ ) of  $m_1$  goes to  $v_2$  and the other fraction ( $w_{31}$ ) to  $v_3$ . In layer 2,  $v_1$  continues to feed  $m_1$ ,  $v_2$  is passed on to  $m_2$  and  $m_3$ , and then goes to  $v_3$  and  $v_4$ . In layer 3,  $m_4$  receives input from  $v_4$  which in turn activates  $v_5$ . The unrolling is iterated until the values for the metabolites production rates and reaction fluxes converge. **d.** Recurrent neural network (RNN) representation. Vectors  $M$  and  $V$  respectively represent metabolites production rates and reaction fluxes. At each iteration step,  $M$  and  $V$  are computed using matrices  $P_{v \rightarrow m}$  and  $P_{m \rightarrow v}$  of panel e. When a metabolite is the substrate of several reactions (like  $m_1$  and  $m_2$ ), each reaction gets a fraction of the metabolite production flux, this is depicted in matrix  $W_r$  ( $r$  indicates this matrix is used in recurrence). The matrix  $W_r$  can be computed from the steady state fluxes of all reactions or learned through training. The operator  $\odot$  stands for element-wise matrix product (Hadamard product). **e.** Neural network matrices.  $P_{v \rightarrow m}$  is the matrix to compute metabolite production fluxes from reaction fluxes,  $P_{m \rightarrow v}$  is a matrix to compute reaction fluxes from the production rates of the reaction substrates.  $\text{ReLU}(x) = \max(0, x)$ ,  $s_{j,i}$  corresponds to the value of  $S$  at the  $j^{\text{th}}$  row (metabolite) and  $i^{\text{th}}$  column (flux) and  $z_i$  is the number of strictly negative elements in column  $i$  of  $S$ . **f.:** Providing the FBA steady state solution, the weight matrix is computed as follows:  $w_{21} = v_2/(v_1+v_5)$ ,  $w_{31} = v_3/(v_1+v_5)$ ,  $w_{23} = v_3/v_2$  and  $w_{24} = v_4/v_2$ . Heatmap obtained for  $n=30$  iterations of the RNN of panel d running with the toy model of panel a.

## AMN-Wt architecture

As shown in Figure 1 in the main paper, the AMN-Wt architecture (like AMN-LP or AMN-QP) takes as input a vector ( $V_{in}$ ) representing exact bounds (EBs) or upper bounds (UBs) for uptake fluxes. In both cases an initial vector  $V^0$  is computed via the weight matrix  $W_i$  ( $V^0 = W_i V_{in}$ ). We recall from the previous section that AMN-Wt also comprises a weight matrix  $W_r$  which can be learned through training. In the EB case,  $W_i$  is not trainable and is just a mapping of  $V_{in}$  into  $V$ . Consequently, in this case only the matrix  $W_r$  is learned during training. In the UB case, the weights in  $W_i$  are trainable and transform the upper bounds  $V_{in}$  into exact bounds in  $V^0$ . Consequently, with UB, both  $W_i$  and  $W_r$  are learned during training. It turns out with EB that a single set of weights (matrix  $W_r$ ) cannot handle all the elements of a training set when the network contains internal reaction fluxes depending on at least two metabolite uptake fluxes. Figure S2 below shows such an example. Consequently, AMN-Wt cannot be used to process EB training sets.



**Figure S2. Different weights for different uptake fluxes provided by exact bounds (EBs)**

In the two cases all flux values ( $v_i$ ) satisfy the steady state constraints ( $SV = 0$ , cf. Figure 1b). Following the equations provided in Figure 1d, the production rates for  $m_1$  and  $m_2$  are respectively 1 and 0.5 in (a) and 0.75 and 0.25 in (b). The reaction for flux  $v_4$  is taking two substrates  $m_1$  and  $m_2$  and the value for  $v_4$  is the minimum metabolite production rate (i.e., the rate limiting among  $m_1$  and  $m_2$ ). Consequently, the value for  $v_4$  is 0.5 (a) and 0.25 (b). Therefore, the fraction ( $w_{41}$ ) of  $m_1$  contributing to  $v_4$  is 1/2 (a) and 1/3 (b). The weights are different in panel a and b as they depend on the uptake flux values.

In the UB case, the uptake flux upper bounds are first transformed into exact bounds with the matrix  $W_i$  learned during training. In this instance, a vector  $V^0$  is calculated via  $W_i$  for each element of a training set, and as observed in Table S1, large training sets can be processed with a single set of weight  $W_r$ . Returning to the example of Figure S2, assuming we measure fluxes  $v_3$  and  $v_4$ , then for any non-null weights  $w_{31}$  and  $w_{41}$ , it is easy to find exact bounds for  $v_1$  and  $v_2$ :  $v_1 = v_3 / w_{31}$  and  $v_2 = 2v_4 - v_3 w_{41} / w_{31}$ . More generally, as shown in Table S1, when running AMN-Wt for many training sets with upper bounds for uptake fluxes for both *E. coli* core<sup>4</sup> and iML1515<sup>2</sup> models, we always find solutions with losses around 0.001 and for which the regression coefficients between AMN predicted growth rates and Simplex calculated growth rates are above 0.98.

Remains the question of whether or not a single set of weights is realistic from a metabolic kinetics point of view. Weights arise at branching points where metabolite fluxes can contribute to two or more reactions. For instance, taking the example of Figure S2, the metabolite production flux  $m_1$  is spliced into  $w_{21}m_1$  and  $w_{31}m_1$ . The question we therefore have to answer is: given different production rates for branch point metabolites, are the weights conserved?

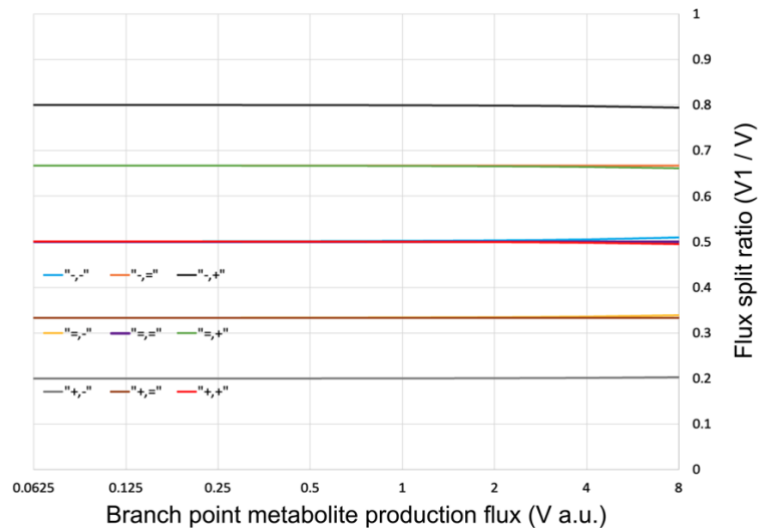
Without lack of generality and for simplicity, we consider below a branch point metabolite with different production fluxes contributing to two reactions. Let  $V$  be the production flux of that metabolite, and let  $V_1$  and  $V_2$  be the fluxes of the two reactions, we necessarily have  $V = V_1 + V_2$ . Now according to Michaelis-Menten equations:

$$V_1 = \frac{V_{max,1} S}{S + K_{m,1}}, V_2 = \frac{V_{max,2} S}{S + K_{m,2}} \quad (S1)$$

where  $V_{max,1} = E_1 k_{cat,1}$  ( $V_{max,2} = E_2 k_{cat,2}$ ),  $E_1$  ( $E_2$ ) being the concentration of the enzyme catalyzing the reaction,  $S$  the concentration of the branch-point metabolite, and  $k_{cat,1}$  ( $k_{cat,2}$ ) and  $K_{m,1}$  ( $K_{m,2}$ ) the turnover rate and the Michaelis constant of the reaction. Using these notations, we have:

$$V = \frac{V_{max,1} S}{S + K_{m,1}} + \frac{V_{max,2} S}{S + K_{m,2}} \quad (S2)$$

We note that  $S$  can be computed from  $V$  by solving the quadratic eq. S2. As shown in Figure S3 below, a numerical simulation for different values of the kinetics parameters shows that the ratio ( $V_1 / V$ ) remains constant (slope of Figure S3 equals 1) even when the production flux  $V$  is changed by several orders of magnitude. Consequently, a single set of weights can fit a training set as long as the kinetics parameters  $V_{max,1}$  ( $V_{max,2}$ ) and  $K_{m,1}$  ( $K_{m,2}$ ) do not change with the production flux  $V$ .



**Figure S3. Branch point metabolite flux ratio**

Here a branch point metabolite contributes to two reactions with fluxes  $V_1$  and  $V_2$ . The kinetics parameters for reaction  $V_1$  are arbitrarily set to  $V_{max,1} = 1000$  (a.u) and  $K_{m,1} = 100$  (a.u). The values for  $V_{max,2}$  and  $K_{m,2}$  are those of  $V_{max,1}$  and  $K_{m,1}$  halved, equal or doubled. Nine cases are considered from  $V_{max,2} = \frac{1}{2} V_{max,1}$ ,  $K_{m,2} = \frac{1}{2} K_{m,1}$  ("-, -") to  $V_{max,2} = 2 V_{max,1}$ ,  $K_{m,2} = 2 K_{m,1}$  ("+, +").

According to Figure S3, flux split ratios are conserved for nutrients leading to different metabolite production fluxes if the Michaelis-Menten kinetics parameters ( $V_{\max}$  and  $K_m$ ) of the enzymes catalyzing the reactions involved in the split remain constant. However, Chubukov *et al.*<sup>5</sup> have shown experimentally that it was not the case (for *B. subtilis*) and different nutrients do provide different ratios. This behavior is due to varying enzyme activities, which themselves depend on enzyme concentrations, post-translational modification, and gene regulations. Chubukov *et al.*<sup>5</sup> showed with experimental evidence that different nutrients give rise to different concentrations for many enzymes, implying that nutrients do have an effect on gene regulations. We note that even though weights in  $W_r$  do not have a physical meaning, AMN-Wt still exhibits excellent performances, showing that the consensual  $W_r$  matrix and the initial  $V^0$  vector (computed through a neural layer from the upper bound  $V_{in}$ ) are performant enough. We also note that the weight issue does not arise with AMN-LP and AMN-QP as these architectures do not rely on flux split ratios.

## LP-solver equations

We recall that the LP-solver makes use of Hopfield-like networks, which is a long-standing field of research<sup>6</sup> inspired by the pioneering work of Hopfield and Tank<sup>7</sup>. Later, these Hopfield-like networks were showcased to perform well for solving linear programs<sup>8</sup> and simpler and more efficient solutions were developed over the years<sup>9,10</sup>. It is important to point out at this stage that these Hopfield-like networks are non-trainable networks and differ from classical neural networks used in ML. The Hopfield-like networks are instead recurrent procedures iteratively updating the solution of linear programs.

The constrained linear optimization problems EB and UB are specific cases of the general problem described in Yang *et al.*<sup>9</sup> which can be written as:

$$\begin{aligned} \min: & c^T x \\ \text{s.t. } & Ax = b \\ & Bx \leq d \end{aligned} \tag{S3}$$

where  $x$  is the vector of unknown (size  $n$ ) to be calculated,  $c$  is the objective vector (also of size  $n$ ),  $A$  is a  $(m \times n)$  matrix of rank  $m$  (and therefore non-null) and  $B$  a  $m \times n$  matrix. Translated to FBA problems,  $x$  is the flux vector ( $V$ ),  $c$  the vector corresponding to the objective function,  $A$  is related to the stoichiometric matrix,  $B$  is a matrix that extracts exchange fluxes from the full flux vector and  $d$  is a vector of constraints on exchange fluxes.

Consequently, to solve a FBA problem  $A$ ,  $B$ ,  $c$ ,  $d$ ,  $b$  and  $x$  take the following values in the EB case:

$$A = S_{int}, B = -I_n, b = -b_{FBA}, d = 0, c = -c_{FBA}, x = V$$

According to Yang *et al.*<sup>3</sup>, gradients for  $V$  and its dual  $U$  in EB case can be written as:

$$\begin{aligned} \nabla V &= (I_n - P) [c_{FBA} - S_{int}^T R] + QV \\ \nabla U &= \frac{1}{2} (U - R) \end{aligned} \tag{S4}$$

where:

$$R = \text{ReLU}(U + S_{int}V + b_{FBA})$$

$$Q = S_{int}^T (S_{int} S_{int}^T)^{-1}$$

$$P = Q S_{int}$$

As  $S_{int} S_{int}^T$  has to be invertible, it is important that  $\text{rank}(S_{int}) = (S_{int})$ . To ensure this point,  $S_{int}$  was converted to its row echelon form and rows with null values were removed. Consequently, prior to any computation of the LP solver, one has to preprocess the matrices in row echelon form.



The linear problem in the case where uptake fluxes values are unknown (UB method) can be written as:

$$\begin{aligned} \max: & c_{FBA}^T V \\ \text{s.t.} & S_{int,In} V \geq -b_{FBA,0} \\ & SV = 0 \end{aligned} \quad (S5)$$

The matrix  $S_{int,In}$  is obtained by concatenation of  $S_{int}$  and  $-I_n$  (see Figure S5c). This matrix ensures the respect of two inequalities; entry fluxes are inferior to the upper bound, fluxes are positive. Consequently,  $b_{FBA,0}$ , is the concatenation of the upper bounds vector  $b_{FBA}$  and a zero vector of size  $n$ .

The dual form of eq. S5 being:

$$\begin{aligned} \min: & -b_{FBA,0}^T U \\ \text{st:} & S_{int,In}^T U \leq c_{FBA} \\ & U \leq 0 \end{aligned} \quad (S6)$$

Thus,  $A, B, c, d, b$  and  $x$  take the following values:  $A = S, B = -S_{int,In}, b = 0, d = b_{FBA,0}, c = -c_{FBA}, x = V$

We note  $c = -c_{FBA}$  since  $\min(-c_{FBA}V)$  is equivalent to  $\max(c_{FBA}V)$ . The matrices  $A, B$  and vectors  $b, d$  take different forms depending if we are in the EB or UB cases. More details can be found in Figure S4 and Figure S5.

Similarly, to the EB case, gradients for  $U$  and  $V$  are:

$$\begin{aligned} \nabla V &= (I_n - P)[c_{FBA} - S^T R] + Q(S_{int,In} V + b_{FBA,0}) \\ \nabla U &= \frac{1}{2}(U - R) \end{aligned} \quad (S7)$$

where:

$$\begin{aligned} R &= ReLU(U + SV) \\ Q &= S^T(SS^T)^{-1} \\ P &= QS \end{aligned}$$

Yang *et al.*<sup>9</sup> proved in their paper that  $x$ , the variable of the primal problem, and  $y$ , the variable of the dual problem, ( $V$  and  $U$  using FBA notations) can be calculated iteratively starting with arbitrary values for  $V^{(0)}$  and  $U^{(0)}$ :

$$\begin{aligned} V^{(t+1)} &= V^{(t)} - dt \nabla V \\ U^{(t+1)} &= U^{(t)} - dt \nabla U \end{aligned} \quad (S8)$$

where  $t$  is the iteration number,  $dt$  the learning rate, and the derivatives are equation (S4) for EB and (S7) for UB.

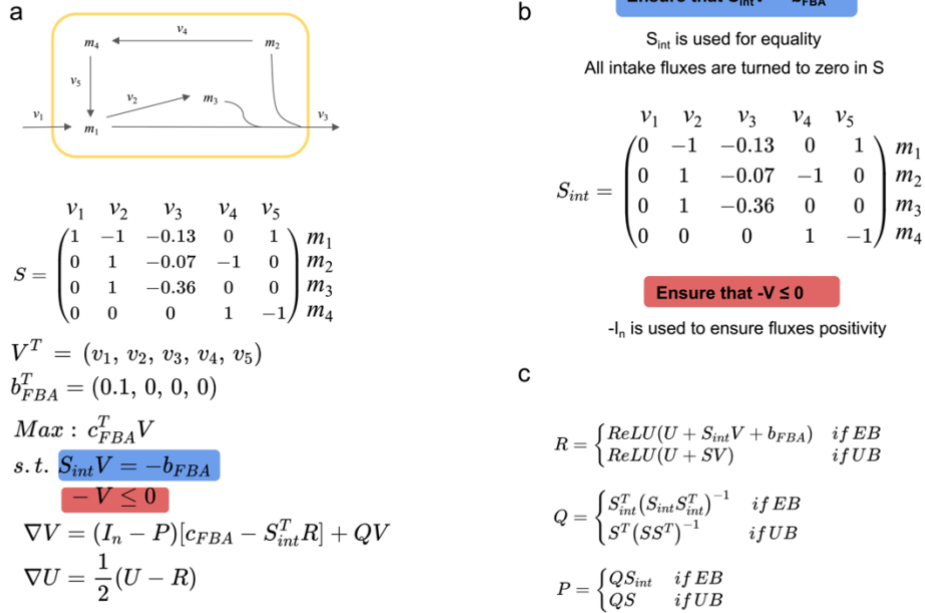
The equivalence between (S3) and (S8) is proved by the first Lemma of Yang *et al.*<sup>9</sup> which states that  $V^*$  is a solution of (S3) if and only if there exist  $U^*$  such that:

$$(I - P)(-c - B^T U^*) - Q(A V^* - b) = 0 \quad (S9)$$

$$ReLU(U^* + B V^* - d) - U^* = 0$$

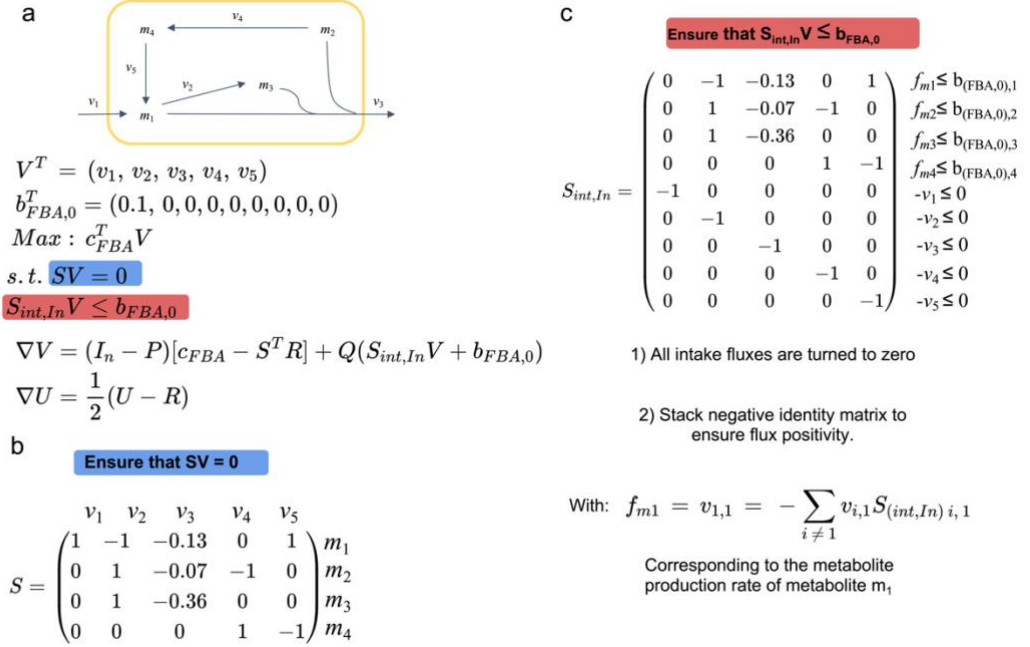
with  $I$  the identity matrix of adequate size for  $P$ . According to the 2<sup>nd</sup> Theorem in Yang *et al.*<sup>9</sup>, any initialization will converge to an equilibrium point. This Theorem also states that the convergence trajectories are asymptotically stable if there is a unique equilibrium point. In practice it is almost never the case with metabolic networks because the optimum is rarely unique.

Note that the work from Yang *et al.*<sup>9</sup> allows quadratic optimization, thus this method could be used with a fitting term similar to the one described in the QP method.



**Figure S4. Matrices used with the LP (EB) method**

**a.** We show here an example for the same model as in Figure S1 in the scenario of known uptake fluxes (EB). All the information about uptake fluxes is contained in  $b_{FBA}$ .  $P$ ,  $Q$  and  $R$  are defined in panel c (in this case using the formulations for exact bounds, EB).  $\nabla V$  and  $\nabla U$  are gradients respectively for the fluxes and metabolites shadow prices (as in the eq. S4) **b.** In the EB case, the only inequality constraint to verify is the positivity of fluxes (highlighted in red), so we use  $-I_n$ , the identity matrix of size  $n$  (number of fluxes), multiplied by  $-1$ , as the matrix  $B$  in Yang et al. formulation. To verify the equality constraints, coefficients of  $v_1$  are zeroed out in  $S_{int}$  because only the input flux  $v_1$  of the metabolite  $m_1$  is known, thus the matrix  $A$  in Yang et al. formulation is  $S_{int}$  in our formulation, and  $S_{int} V = -b_{FBA}$  ensures the respect of equality constraints. **c.** Reminder of  $P$ ,  $Q$  and  $R$  when using exact (EB) or upper bounds (UB).



**Figure S5. Matrices used with the LP (UB) method**

**a.** The example is the same as in Figure S1 and Figure S4. In this scenario, uptake fluxes are unknown (UB), consequently upper bounds are contained in  $b_{FBA,0}$  and their values are fixed arbitrarily. Note that, in  $b_{FBA,0}$ , an upper bound is different from zero if and only if the corresponding metabolite is in the medium (only  $m_1$  here).  $P$ ,  $Q$  and  $R$  are shown in Figure S4, panel c (in this case using the formulations for upper bounds, UB).  $\nabla V$  and  $\nabla U$  are gradients respectively for the fluxes and metabolites shadow prices (as in the eq. S7) **b.** The only equality constraint to verify in this problem is the physical law of mass conservation:  $SV=0$ . Therefore, the stoichiometric matrix  $S$  is used as the matrix  $B$  in Yang et al. formulation **c.**  $S_{int,In}$  was constructed to ensure two inequalities: first the uptake fluxes should be inferior to their upper bounds (UB), which is verified with the 4 first rows (corresponding to  $S_{int}$ , defined in Figure S4); and all fluxes should be positive, which is verified by  $-I_n$  (the same as in Figure S4), that is stacked to  $S_{int}$ . Consequently,  $S_{int,In}$  is used as the matrix  $A$  in Yang et al. formulation.

## QP-solver equations

We recall below the quadratic program (QP) exposed as eq. 1 in Methods ‘Derivation of loss functions:

$$\min( \|P_{ref} V - V_{ref}\|^2) \quad (S10)$$

$$\text{s. t.} \quad S V = 0$$

$$P_{in} V \leq V_{in}$$

$$V \geq 0$$

To solve this problem, a loss function with four terms was built. As mentioned in the Methods section ‘Loss functions derivation’, the first term is related to the fit to the reference targeted values and the three additional losses terms are related to the boundary, stoichiometric and flux positivity constraints of the metabolic network.

The first loss is simply the Mean Square Error (MSE) between predictions ( $V$ ) and FBA-simulated or measured reference data ( $V_{ref}$ ):

$$L_1 = \frac{1}{n_{ref}} \|P_{ref} V - V_{ref}\|^2 \quad (S11)$$

The second loss is linked to the network stoichiometric constraint ( $S V = 0$ ), which in its normalized form (loss per constraint) is:

$$L_2 = \frac{1}{m} \|SV\|^2 \quad (S12)$$

The third loss evaluates how well boundary constraints are respected ( $P_{in} V \leq V_{in}$ ):

$$L_3 = \frac{1}{n_{in}} \|ReLU(P_{in} V - V_{in})\|^2 \quad (S13)$$

The last loss enforces all fluxes to be positive:

$$L_4 = \frac{1}{n} \|ReLU(-V)\|^2 \quad (S14)$$

We note that when exact bounds are provided,  $P_{in} V = V_{in}$ , and  $L_3$  becomes obsolete as the values of  $V$  corresponding to  $V_{in}$  are not updated by the LP/QP solvers and AMN programs.

Thus, the sum of those four terms is the loss  $L$  given by eq. 2 in Methods ‘Loss functions derivation’.

Note that  $L$  can also be computed as the MSE between the vectors:

$$\left( P_{ref} V, \frac{\|SV\|}{\sqrt{m}}, \frac{\|ReLU(P_{in} V - V_{in})\|}{\sqrt{n_{in}}}, \frac{\|ReLU(-V)\|}{\sqrt{n}} \right) \text{ and } (V_{ref}, 0, 0, 0)$$

While the QP system can be solved by a simplex algorithm, solutions can also be approximated calculating  $V$  corresponding to:

$$\min \left( \frac{1}{n_{ref}} \|P_{ref} V - V_{ref}\|^2 + \frac{1}{m} \|SV\|^2 + \frac{1}{n_{in}} \|ReLU(P_{in} V - V_{in})\|^2 + \frac{1}{n} \|ReLU(-V)\|^2 \right) \quad (S15)$$

The vector  $V$  can thus be found solving:

$$\frac{\partial \left( \frac{1}{n_{ref}} \|P_{ref} V - V_{ref}\|^2 + \frac{1}{m} \|SV\|^2 + \frac{1}{n_{in}} \|ReLU(P_{in} V - V_{in})\|^2 + \frac{1}{n} \|ReLU(-V)\|^2 \right)}{\partial V} = 0 \quad (S16)$$

As mentioned in Methods 'QP solver',  $V$  satisfying eq. S15 can be found iteratively:

$$V^{(t+1)} = V^{(t)} - dt \nabla V \quad (S17)$$

$$V^{(0)} = P_{in}^T V_{in}$$

where  $t$  is the iteration number,  $dt$  the learning rate.

$\nabla V$  is computed as follow:

$$\nabla V = \frac{2}{n_{ref}} P_{ref}^T (P_{ref} V - V_{ref}) + \frac{2}{m} S^T SV + \frac{2}{n_{in}} P_{in}^T D_{in} ReLU(P_{in} V - V_{in}) - \frac{2}{n} D_V ReLU(-V) \quad (S18)$$

It is easy to verify that for the first term of  $\nabla V$  we have:

$$\nabla V_1 = \frac{\partial \left\| \frac{P_{ref} V - V_{ref}}{n_{ref}} \right\|^2}{\partial V} = \frac{2}{n_{ref}} P_{ref}^T (P_{ref} V - V_{ref}) \quad (S19)$$

for the second term:

$$\nabla V_2 = \frac{\partial \left\| \frac{SV}{m} \right\|^2}{\partial V} = \frac{2}{m} S^T SV \quad (S20)$$

for the third term:

$$\nabla V_3 = \frac{\partial \left\| \frac{ReLU(P_{in} V - V_{in})}{n_{in}} \right\|^2}{\partial V} = \frac{2}{n_{in}} P_{in}^T D_{in} ReLU(P_{in} V - V_{in}) \quad (S21)$$

where  $D_{in} = \frac{ReLU(P_{in} V - V_{in})}{ReLU(P_{in} V - V_{in})}$  using an *Hadamard* division:  $\frac{A}{A} = \left( \frac{a_{ij}}{a_{ij}} \right)$  ( $= 0$  when  $a_{ij} = 0$ )

and for the fourth term:

$$\nabla V_4 = \frac{\partial \left\| \frac{ReLU(-V)}{n} \right\|^2}{\partial V} = -\frac{2}{n} D_V ReLU(-V) \quad (S22)$$

where  $D_V = \frac{ReLU(-V)}{ReLU(-V)}$  using an *Hadamard* division.

Summing eqs. S19-S22 we find:

$$\nabla V = \frac{2}{n_{ref}} P_{ref}^T (P_{ref} V - V_{ref}) + \frac{2}{m} S^T S V + \frac{2}{n_{in}} P_{in}^T D_{in} ReLU(P_{in} V - V_{in}) - \frac{2}{n} D_V ReLU(-V) \quad (S23)$$

In the EB case where exact medium uptake fluxes are known, the QP system is:

$$\min(\|P_{ref} V - V_{ref}\|^2) \quad (S24)$$

$$\text{s.t.} \quad S V = 0$$

$$P_{in} V = V_{in}$$

$$V \geq 0$$

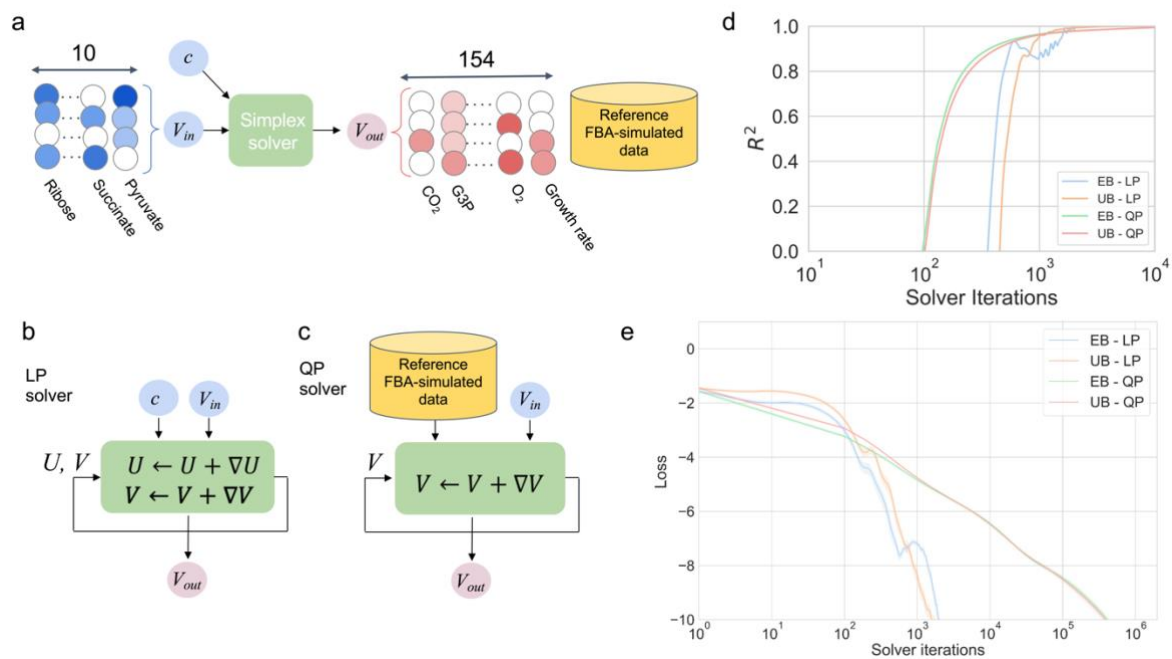
In such an instance, reaction fluxes having a mapping in  $V_{in}$  remain constant and are not updated, therefore:

$$\nabla V = \left( \frac{2}{n_{ref}} P_{ref}^T (P_{ref} V - V_{ref}) + \frac{2}{m} S^T S V - \frac{2}{n} D_V ReLU(-V) \right) \odot (1_n - P_{in}^T 1_{\{n_{in}\}}) \quad (S25)$$

where  $\odot$  stands for *Hadamard* product ( $A \odot B = a_{ij} b_{ij}$ ) and  $1_n$  (*respectively*  $1_{\{n_{in}\}}$ ) is a vector of dimension  $n$  (*respectively*  $n_{in}$ ) with constant coefficients equal to 1.

## MM solvers benchmarking

As it has been described along this paper, AMNs are composed of both a mechanistic layer and a neural layer. As illustrated in Figure S6d and S6e, mechanistic solvers require more than 10,000 iterations, which brings issues to the gradient backpropagation (vanishing or exploding gradients) as well as an increased training time. Further results about mechanistic layers alone are given in the Table S1, for the ‘MM’ model types.



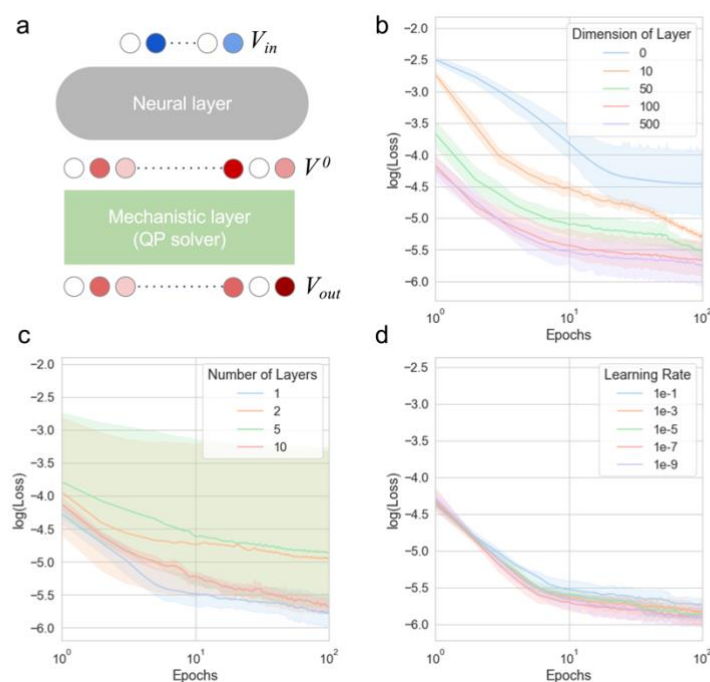
**Figure S6. MM solvers architectures and performances**

**a.** Schematic procedure for the Simplex solvers. From  $V_{in}$ , which is a vector describing the bounds of some uptake fluxes, the solvers reach a steady-state solution,  $V_{out}$ , optimizing the objective function  $c$ , satisfying the constraints and bounds of the network. Solutions obtained using the simplex-based method in Cobrapy<sup>11</sup>, are taken as reference data. **b.** Schematic for LP-solver architecture. This solver surrogates the simplex-based algorithm. Following Yang et al. and as further detailed in the Methods ‘AMN architectures’, the full flux distribution  $V$  is updated by  $\nabla V$  and the metabolites shadow prices  $U$  by  $\nabla U$  through products of matrices derived from the stoichiometry of the network. **c.** Schematic for QP-solver architecture. Here target reference fluxes ( $V_{ref}$ ) are given to the solver. The computed fluxes are fitted to the reference targets by means of a custom loss function integrating also the input constraints along with the stoichiometric constraint of the metabolic network. The flux vector  $V$  is updated by  $\nabla V$  which is the gradient minimizing the loss function (cf. Methods ‘AMN architectures’ for further details). **d.**  $R^2$  vs Solver iteration. **e.** Loss vs. Solver iterations. QP takes 1 million iterations to reach close to zero values, whereas LP takes 10,000 iterations. In d and e, plotted is the mean and standard error (95% confidence interval) across all elements of the set of 100 simulations.



## AMNs benchmarking varying hyperparameters

We benchmarked hyperparameters for the neural layer of the AMN-QP model shown in Figure 2b (trained on a 1,000 *E. coli* core simulations dataset). Results are presented in Figure S7 panels b-d. The main conclusion of this search is that increasing the number of layers was inducing overfitting (displaying worst performance on validation sets than with a single layer) and that a minimal hidden dimension was 100 for better performance.



**Figure S7. Hyperparameters for AMN's neural Layer**

**a.** AMN architecture. In panels b-d the AMNs with QP-solver were trained on a simulated training set of 1000 samples for 100 epochs using the Adam optimizer. The metabolic model was *E. coli* core. Three hyperparameters were tested: dimension of the hidden layer(s), number of hidden layers and learning rate. Plotted is the mean and standard error (95% confidence interval) of the loss on test set across 5-folds cross-validation. **b.**  $\log(\text{Loss})$  vs epoch. **c.**  $\log(\text{Loss})$  vs epoch. **d.**  $\log(\text{Loss})$  vs epoch.

From Figure S7 we note that: (i) increasing dimension of the hidden layer increases the decay of the loss function, (ii) the number of hidden layers exhibits huge variability across cross-validation folds suggesting overfitting, and (iii) no major influence was detected for the learning rate. The default architecture of the neural layer in all AMN was therefore set as one layer of dimension higher than 50 (we increased this hyperparameter with the model's size, see Table S1) and a training rate of 1e-3

## AMNs benchmarking with independent test sets and additional metabolic models

The performances of all AMN architectures (Wt, LP, QP) are given in Table S1 using FBA simulated data on two different *E. coli* metabolic models, *E. coli* core model<sup>4</sup> and iML1515<sup>2</sup> along with iJN1463 *P. putida* model. These *E. coli* models are composed respectively of 154 reactions and 72 metabolites, and 3682 reactions and 1877 metabolites (after duplicating bidirectional reactions). The *P. putida* model is composed of 2135 reactions and 1637 metabolites (after duplicating bidirectional reactions). In all cases the default Simplex-based solver (GLPK) of Cobrapy was run to optimize growth rates for different media. Each medium was composed of metabolites found in minimal media (M9) and additional metabolites (sugars, acids) crossing the cell membrane (more details in Methods ‘Generation of training sets with FBA’). For comparison purposes, Table S1 also provides results for MM architectures (no neural layer) and ANN architectures (no mechanistic layer).

**Table S1. Benchmarking MMs, ANNs and AMNs**

(1) All SBML models describing different *E. coli* strains were downloaded from the BiGG database, ‘Core’ stands for the *E. coli* core model, EB (UB) stands for exact bounds (upper bounds) for medium uptake fluxes, the iML1515 model was reduced following the procedure described in Methods ‘Making metabolic networks suitable for neural computations’. (2) Training set size (number of elements multiplied by number of labelled data per element) and range for the number of metabolites added to the minimal medium. (3) YES or NO if the model contains a neural layer or a mechanistic layer. (4) MM stands for Mechanistic Model, ANN stands for Artificial Neural Network (a dense neural architecture) and AMN for Artificial Metabolic Network. ANN and AMN architectures are described in Methods. Neural Layer Hyperparameters display the number of hidden layers, the size of the hidden layer, and the training rate. Mechanistic Layer Hyperparameters display the number of iterations performed by the solver. (5) Number of trainable parameters and epochs, in all cases dropout = 0.25, batch size = 5, the optimizer is Adam and the loss function is the mean squared error between predicted and reference fluxes to which for AMN loss constraints are added, see Methods ‘Loss functions derivation’ for additional details. (6) Regression coefficient and Loss values for training set ( $R^2$ ), and cross-validation sets ( $Q^2$ ) between reference growth rate and predicted growth rate. (7) Regression coefficient and Loss values for growth rates for independent test sets not seen during training. Test set sizes are 10% of training set sizes. For (6) and (7) the performance is displayed as the mean over 5 folds (or over a training set when no cross-validation scheme is performed, i.e., for the MM performances). n/a: not applicable or not computed.

SBML strain	Size	Neural layer	Architecture	Nbr param.	Training $R^2$	Validation set $Q^2$	Test set $Q^2$
Bound	Range	Mechanistic layer	Neural Layer Hyperparameters Mechanistic Layer Hyperparameters	Nbr epochs	Loss constraint	Loss constraint	Loss constraint
(1)	(2)	(3)	(4)	(5)	(6)	(6)	(7)
Core	100	NO	MM_LP	n/a	$1.00 \pm 0.000$	n/a	n/a
EB	1-6	YES	n/a 10 <sup>4</sup>	n/a	$3.2e-9 \pm 3.2e-8$	n/a	n/a
Core	100	NO	MM_LP	n/a	$1.00 \pm 0.000$	n/a	n/a
UB	1-6	YES	n/a 10 <sup>4</sup>	n/a	$5e-7 \pm 2.8e-6$	n/a	n/a
Core	100	NO	MM_QP	n/a	$1.00 \pm 0.000$	n/a	n/a
EB	1-6	YES	n/a	n/a	$7.8e-6 \pm 6.1e-6$	n/a	n/a

			10 <sup>6</sup>				
Core	100	NO	MM_QP	n/a	1.00 ± 0.000	n/a	n/a
UB	1-6	YES	n/a	n/a	7.1e-6 ± 5.7e-6	n/a	n/a
			10 <sup>6</sup>				
Core	1540	YES	ANN	8904	0.83 ± 4.8e-2	0.66 ± 1.4e-1	n/a
EB	1-6	NO	1, 50, 1e-3	500	1.8e-1 ± 2.2e-3	2.6e-1 ± 2.1e-2	n/a
			n/a				
Core	1540	YES	ANN	8904	0.82 ± 4.8e-2	0.39 ± 2.5e-1	n/a
UB	1-6	NO	1, 50, 1e-3	500	1.1e-1 ± 8.2e-3	2.1e-1 ± 7.4e-2	n/a
			n/a				
Core	154 000	YES	ANN	8904	0.91 ± 1.5e-2	0.98 ± 1.1e-2	n/a
EB	1-6	NO	1, 50, 1e-3	100	2.3e-1 ± 2.3e-2	1.0e-2 ± 1.1e-2	n/a
			n/a				
Core	154 000	YES	ANN	8904	0.86 ± 2.1e-2	0.94 ± 3.9e-2	n/a
UB	1-6	NO	1, 50, 1e-3	100	1.8 ± 2.0	4.4e-3 ± 1.0e-3	n/a
			n/a				
Core	1000	YES	AMN_LP	17 808	0.98 ± 7.9e-3	0.98 ± 7.4e-4	0.98
EB	1-6	YES	1, 50, 1e-3	500	2.8e-3 ± 0.6e-3	2.8e-3 ± 0.5e-3	3.0e-3
			4				
Core	1000	YES	AMN_LP	25 152	0.98 ± 9.7e-3	0.97 ± 1.0e-2	0.99
UB	1-6	YES	1, 50, 1e-3	500	2.5e-3 ± 0.4e-3	2.5e-3 ± 0.4e-3	3.1e-3
			4				
Core	1000	YES	AMN_QP	8904	0.99 ± 4.2e-3	0.99 ± 4.7e-3	0.98
EB	1-6	YES	1, 50, 1e-3	500	2.3e-3 ± 0.5e-3	2.3e-3 ± 0.5e-3	3.0e-3
			4				
Core	1000	YES	AMN_QP	8904	0.97 ± 9.9e-3	0.97 ± 1.3e-2	0.97
UB	1-6	YES	1, 50, 1e-3	500	2.5e-3 ± 0.6e-3	2.5e-3 ± 0.6e-3	2.0e-3
			4				
Core	1000	YES	AMN_Wt	13 622	0.99 ± 1.3e-3	0.99 ± 2.2e-3	1.0
UB	1-6	YES	1, 50, 1e-3	500	0.9e-3 ± 0.000	0.9e-3 ± 0.000	0.000
			4				
iML1515	11000	YES	ANN	295 050	0.88 ± 4.3e-2	0.76 ± 1.0e-1	n/a
UB	1-6	NO	1, 500, 1e-3	100	2.2 ± 0.8	4.7 ± 4.2	n/a
			n/a				
iML1515	550 000	YES	ANN	295 050	0.98 ± 3.3e-2	0.67 ± 3.5e-1	n/a
UB	1-6	NO	1, 500, 1e-3	100	4.0e-4 ± 3.0e-4	3.1e-3 ± 4.5e-3	n/a
			n/a				
iML1515	11000	YES	AMN_LP	839 266	1.0 ± 1.0e-3	1.0 ± 1.0e-3	1.0
UB	1-4	YES	1, 250, 1e-3	100	0.000 ± 0.000	0.000 ± 0.000	0.000
			4				

iML1515	11000	YES	AMN_QP	295 050	$1.0 \pm 1.4e-3$	$1.0 \pm 1.4e-3$	1.0
UB	1-4	YES	1, 500, $1e-3$ 4	100	$0.000 \pm 0.000$	$0.000 \pm 0.000$	0.000
iML1515	11000	YES	AMN_Wt	634 238	$1.0 \pm 0.1e-3$	$0.99 \pm 0.4e-3$	1.0
UB	1-4	YES	1, 500, $1e-3$ 4	100	$0.000 \pm 0.000$	$0.000 \pm 0.000$	0.000
iJN1463	4860	YES	AMN_QP	1 168 135	$0.99 \pm 2e-3$	$0.99 \pm 2e-3$	0.99
UB	1-1	YES	1, 500, $1e-3$ 4	500	$0.000 \pm 0.000$	$0.000 \pm 0.000$	0.000

The MM architectures show good performances both in terms of growth rate computation and loss on constraints. There is no learning process involved with MMs, therefore no reason to compute results for validation and test sets. The ANN architectures (cf. Methods ‘ANN architecture’ for further details) exhibit poor performances and have small predictive capacities (high Loss) for cross validation sets of sizes in the range of those of AMN (~1000 reference data in training), for that reason performances for test sets were not assessed. We also note that losses remain higher with ANNs even for training set sizes 1000 times larger than those used with AMNs. All AMN architectures exhibit excellent regression coefficients and losses for training sets, validation sets and test sets, and this for both models *E. coli* core<sup>4</sup>, iML1515<sup>2</sup> and iJN1463<sup>12</sup>.

## AMNs benchmarking with gene knockouts and multiple measured fluxes

To assess the performance of AMNs on datasets where more than one flux is measured, we extracted a dataset from Rijsewijk *et al.*<sup>13</sup> which consists of 128 experiments, each containing 31 measured fluxes.

The dataset was composed of 2 media compositions (glucose or galactose as carbon source), for 64 regulator gene KO mutants ( $G_{KO}$ ). These regulator genes were found on RegulonDB<sup>14</sup> and their corresponding regulated metabolic reactions encoded in iML1515 were compiled. Each regulator was found to have at least one regulated reaction in iML1515. The final training set to use with AMNs was composed of 2 inputs: 1 binary vector of size 2 for media compositions ( $C_{med}$ ) and 1 binary vector of size 64 for gene KOs ( $G_{KO}$ ). Unlike for the *E. coli* KO dataset used in Figure 4 in the main paper, we did not add a term to the custom loss since the effect of deleting the regulation of a reaction is *a priori* unknown (at least quantitatively, in terms of effect on the fluxes distribution).

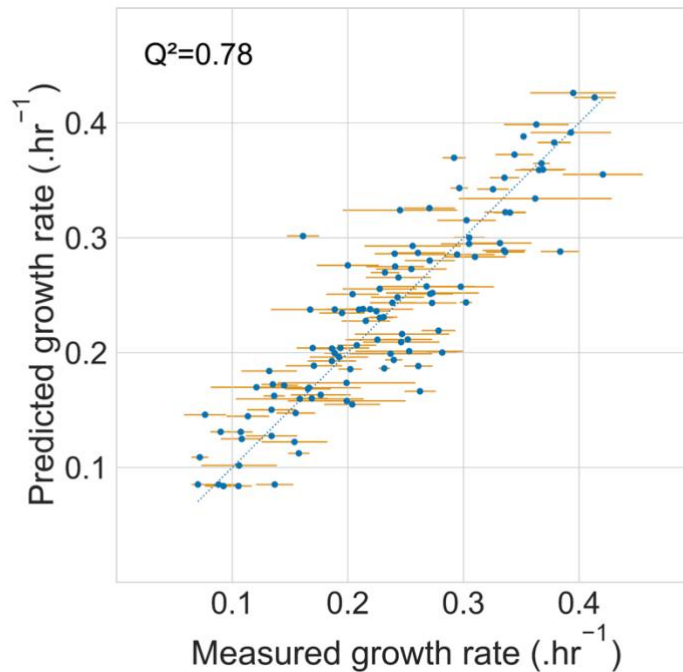
Overall, the performance is satisfactory for most fluxes, but 7 fluxes (empty slots in Figure S8b) have a  $Q^2$  close to zero or negative. However, these low-performance predictions do not impact the variance-weighted average  $Q^2$  (red line, 0.91) because the corresponding measured fluxes have low variance, thus, there are limited statistical patterns for the model to learn on these fluxes (we recall that the variance weighted-average consists in a weighted average of all 31 fluxes'  $Q^2$ s, with a weight applied on each flux's  $Q^2$  corresponding to the variance found in the flux's measure)s.

Learning on many fluxes is more challenging for the AMNs than on a single flux, but it still seems to make accurate predictions with this dataset for the majority of fluxes.



## AMN-Reservoir prediction performance

Figure 5 in the main paper displays the performance of classical FBA with  $V_{in}$  extracted from the AMN-Reservoir, after training on the whole dataset. Another possibility, is to use the AMN-Reservoir in a more predictive manner, obtaining  $V_{in}$  during predictions on the validation sets of a cross-validation. We show in Figure S9 below the performance of FBA when using such predicted  $V_{in}$  from unseen data, as the regression performance on the 110 *E. coli* growth rates dataset. We note the regression coefficient we obtained is similar to those obtained when training AMNs directly on experimental data (Figure 3, main paper).



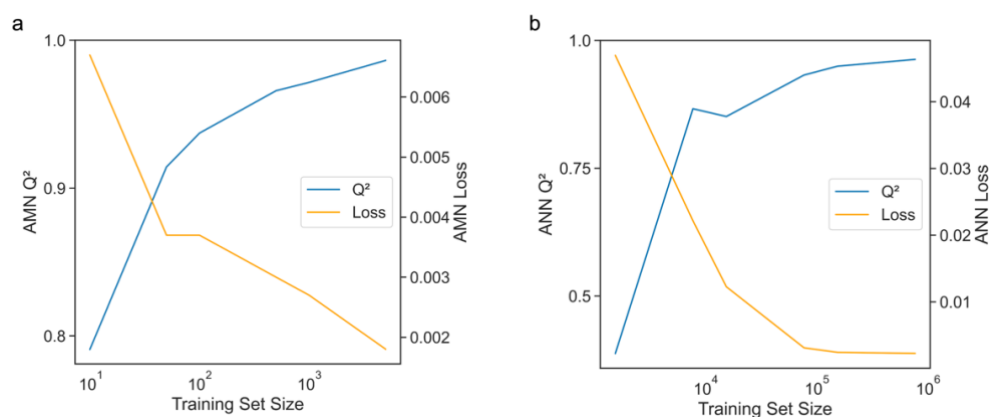
### Figure S9. Performances of AMN-Reservoir using predicted $V_{in}$ as input to FBA

The dataset used to train the AMN-Reservoir was the 110 *E. coli* growth rates used for Figure 3 and Figure 5 panels c and d. The measured growth rates are plotted as the mean and standard deviation over technical replicates (cf. Methods). The hyperparameters and the pre-trained AMN-Reservoir were the same as for Figure 5 panel c. A 10-fold cross-validation was performed (instead of a training and prediction on whole dataset as in Figure 5c), and validation sets predictions were used to extract  $V_{in}$  then use it as input for FBA. FBA results are shown here as the predicted growth rate.

## ANN training set sizes

To compare the performances of an ANN ‘black box’ model with AMNs, we trained a simple dense ANN model and an AMN-QP model on training sets of increasing sizes. The training sets were generated using *E. coli* core<sup>4</sup> as in Methods ‘Generation of training sets with FBA’. We recall (see Methods ‘ANN architecture’) that to assess losses for an ANN model (which does not have any mechanistic layer), each entry of the training sets contains all flux values; this enables one to compute the losses given in Methods ‘Loss function derivation’. Consequently, with ANN for each element in the training set we provided as labeled data all reaction fluxes (154 for *E. coli* core<sup>4</sup>), while with AMN-QP we provided as labeled data only the flux of the biomass reaction.

To enable comparison between AMN and ANN, in Figure S10 given below the training set size is the number of labeled data provided. We find that the results obtained for AMN-QP are consistent with those presented in Table S1: training set size of 1000 yields a  $Q^2$  above 0.95 while the loss remains below 0.003. We also observe that the ANN architecture requires training set sizes several orders of magnitude larger to reach losses that are still above those obtained with AMN-QP: training set sizes of more than 500,000 are needed to obtain a loss below 0.01, while  $Q^2$  above 0.9 are reached for training set sizes above 50,000. Finally, it is worth noticing that while ANN can be trained on simulated data as in Figure S10 they cannot directly be used with experimental data as it is practically not possible to measure all the reaction fluxes of a strain grown with different media compositions.



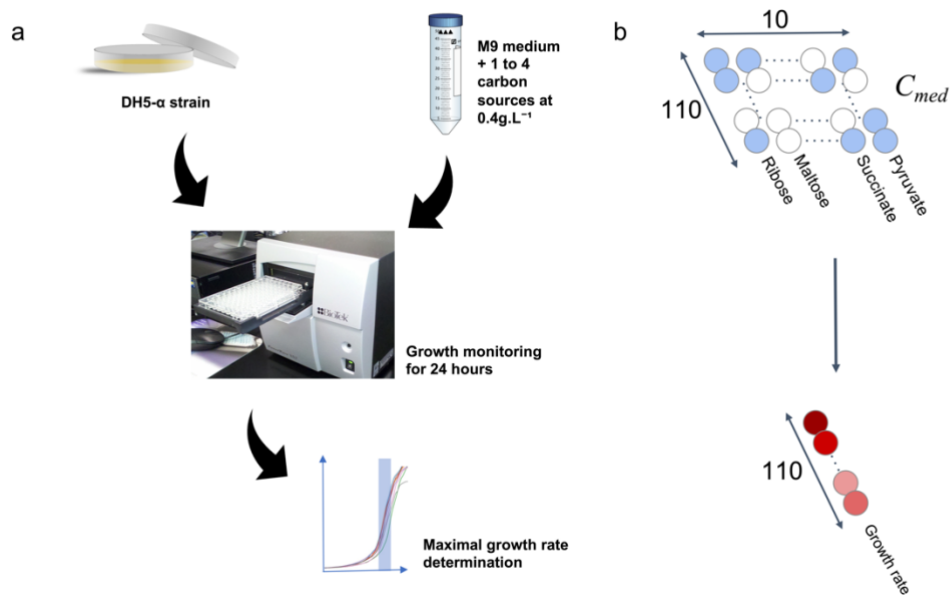
**Figure S10. Loss and regression coefficient for training sets of increasing sizes**

In both cases training sets were generated for the *E. coli* core model using the procedure described in Methods ‘Generation of training sets with FBA’. AMN and ANN were trained for different medium metabolites uptake rates as inputs and, as reference (labeled) data, the biomass reaction flux for AMN and all fluxes for ANN. In both cases  $Q^2$  is the regression coefficient between the reference and the predicted biomass reaction fluxes during 5-fold cross-validation. Loss on constraints were computed as described in Methods ‘Loss functions derivation’ on 5-fold cross-validation sets. **a.** AMN. The model architecture is the one shown in Figure 3 with a QP-solver for the mechanistic layer. The neural layer is composed of an input layer of size 20 (all uptake fluxes of *E. coli* core), a hidden layer of size 50 and an output layer of size 154 (all reactions of *E. coli* core), the learning rate was set to  $1.0e-3$  and Adam was used as the optimizer. **b.** ANN. The ANN model has the same architecture as the neural layer of the AMN and no mechanistic layer. Raw data for this figure can be found in the *amn\_release* GitHub repository (Result folder).



## Experimental workflow

Details on the experimental protocol can be found in the Methods ‘Culture conditions’ and ‘Growth rate determination’. Figure S11 gives a visual overview of the workflow to generate the dataset showcased in Figures 3 & 5 (main paper).



**Figure S11. Experimental workflow pipeline**

**a.** The DH5- $\alpha$  strain of *E. coli* was cultured in M9 medium with different carbon source combinations (1 to 4 carbon sources simultaneously added, all at 0.4g.L<sup>-1</sup>). The optical density at 600nm (OD600) was monitored for 24 hours in a plate reader; reading 96-well plates each containing 10 media compositions, each in 8 replicates (remaining space was used as blanks, for the edges of the plate that show high evaporation). After data acquisition, the maximal growth rate was computed (cf. Methods ‘Growth rate determination’). **b.** The experimental workflow enables the generation of 110 data points each composed of  $C_{med}$  as the independent variables and growth rates as dependent variables.  $C_{med}$  is describing each medium carbon source composition as zeros - when absent - and ones - when present – yielding in the end a binary vector of length 10.

## Terminology

We provide below the list of all notations used in all equations and figures of our main manuscript and Supplementary Information.

**Table S2. Vectors and matrices notations used in figures and equations**

Notation	Description (units)
$V_{in}$	<b>Vector</b> of exact or upper bounds for uptake fluxes. See Figure 1a,c,d (mmol.gDW <sup>-1</sup> .h <sup>-1</sup> )
$V_{out}$	<b>Vector</b> of steady-state fluxes values predicted by a model, either fully mechanistic or AMN. See Figure 1. (mmol.gDW <sup>-1</sup> .h <sup>-1</sup> and .h <sup>-1</sup> )
$V_{ref}$	<b>Vector</b> of reference fluxes, either FBA-simulated or measured. In all results except Figure S8 (31 fluxes) and for the ANNs (all fluxes), it contains only the growth rate. (mmol.gDW <sup>-1</sup> .h <sup>-1</sup> and .h <sup>-1</sup> )
$V^0$	<b>Vector</b> of fluxes values before passing through the mechanistic model. Referred to as initial guess for the flux distribution. See Figure 1c,d. (mmol.gDW <sup>-1</sup> .h <sup>-1</sup> and .h <sup>-1</sup> )
$C_{med}$	<b>Vector</b> describing medium composition. See Figure 1c,d. (no unit)
$V$	<b>Vector</b> of reaction fluxes. Generic name. (mmol.gDW <sup>-1</sup> .h <sup>-1</sup> and .h <sup>-1</sup> )
$M$	<b>Vector</b> of metabolites production rates, used for Wt. (mmol.gDW <sup>-1</sup> .h <sup>-1</sup> )
$U$	<b>Vector</b> . Dual variable of $V$ when considering FBA's constrained linear problem. Also called metabolites' shadow prices.
$R_{KO}$	<b>Vector</b> of reactions that are KO. See Figure 4a. 0 if reaction is inactivated by the gene KO, 1 otherwise. (no unit)
$C_{FBA}$	<b>Vector</b> of reactions that are hypothesized to be maximized by the cell. In this work, always set to the biomass production reaction ( <i>i.e.</i> , the growth rate). Used for LP method. (mmol.gDW <sup>-1</sup> .h <sup>-1</sup> )
$S$	<b>Matrix</b> of stoichiometric coefficients given by the GEM. Its dimension is $m$ (number of metabolites) $\times$ $n$ (number of reactions). (no unit)
$W_r$	<b>Matrix</b> of weight representing consensual flux branching ratios. (no unit)
$P_{in}$	<b>Matrix</b> of mapping $V$ into the fluxes of $V_{in}$ . (no unit)
$P_{out}$	<b>Matrix</b> of mapping $V$ into the fluxes of $V_{out}$ . (no unit)
$P_{ref}$	<b>Matrix</b> of mapping $V$ into the fluxes of $V_{ref}$ . (no unit)
$P_{ko}$	<b>Matrix</b> of mapping from gene KO to inactivated reactions. (no unit)
$P_{v \rightarrow m}$	<b>Matrix</b> of mapping from reactions to metabolites. (no unit)
$P_{m \rightarrow v}$	<b>Matrix</b> of mapping from metabolites to reactions. (no unit)
$S_{int}$	<b>Matrix</b> of stoichiometric coefficients where uptake reactions have been zeroed out. (no unit)

In our study  $V_{in}$  are “uptake fluxes” also named “uptake reactions” the fluxes and corresponding reactions that introduce matter into the model, *i.e.* reactions that have no reactants and their product is a metabolite in the ‘medium’ compartment of the metabolic model. These reactions are also called “exchange reactions” in many studies, and this subsection aims to clarify the use of “uptake flux” in this study. The term “uptake” was preferred to “exchange” for two main reasons: (i) for simplicity to readers that are not familiar with metabolic models and (ii) for the better biological sense of “uptake

reactions” (designating an organism uptaking nutrients from its environment), even if these reactions introducing matter into the model are fully virtual reactions without any biological or physical sense.

Importantly, ‘uptake reactions’ are not referring to the membrane-crossing reactions, which are always left with default bounds in this study. In practice, when one makes *ab initio* predictions with classical FBA, one sets a non-zero upper bound for a reaction introducing matter in the system, to simulate the presence of a given metabolite in the medium. But, in most cases, this reaction flux optimized value will be equal to the membrane-crossing flux value, since one metabolite, in most cases, can only go to this reaction once it has been introduced in the medium compartment of the model. The only exception is in some models where the metabolites can interact in the medium compartment, or when several transport reactions are available, which is rare and not the case in the scope of the models used in this study.

## Supplementary references

1. Nilsson, A., Peters, J. M., Meimetis, N., Bryson, B. & Lauffenburger, D. A. Artificial neural networks enable genome-scale simulations of intracellular signaling. *Nat. Commun.* **13**, 3069 (2022).
2. Monk, J. M. *et al.* iML1515, a knowledgebase that computes Escherichia coli traits. *Nat. Biotechnol.* **35**, 904–908 (2017).
3. Norsigian, C. J. *et al.* BiGG Models 2020: multi-strain genome-scale models and expansion across the phylogenetic tree. *Nucleic Acids Res.* **48**, D402–D406 (2020).
4. Orth, J. D., Fleming, R. M. T. & Palsson, B. Ø. Reconstruction and Use of Microbial Metabolic Networks: the Core Escherichia coli Metabolic Model as an Educational Guide. *EcoSal Plus* **4**, (2010).
5. Chubukov, V. *et al.* Transcriptional regulation is insufficient to explain substrate-induced flux changes in Bacillus subtilis. *Mol. Syst. Biol.* **9**, 709 (2013).
6. Jin, L., Li, S., Hu, B. & Liu, M. A survey on projection neural networks and their applications. *Appl. Soft Comput.* **76**, 533–544 (2019).
7. Hopfield, J. J. & Tank, D. W. “Neural” computation of decisions in optimization problems. *Biol. Cybern.* **52**, 141–152 (1985).
8. Wang, J. & Chankong, V. Recurrent neural networks for linear programming: Analysis and design principles. *Comput. Oper. Res.* **19**, 297–311 (1992).
9. Yang, Y., Cao, J., Xu, X., Hu, M. & Gao, Y. A new neural network for solving quadratic programming problems with equality and inequality constraints. *Math. Comput. Simul.* **101**, 103–112 (2014).
10. Ghasabi-Oskoei, H. & Mahdavi-Amiri, N. An efficient simplified neural network for solving linear and quadratic programming problems. *Appl. Math. Comput.* **175**, 452–464 (2006).
11. Ebrahim, A., Lerman, J. A., Palsson, B. O. & Hyduke, D. R. COBRApy: COntstraints-Based

- Reconstruction and Analysis for Python. *BMC Syst. Biol.* **7**, 74 (2013).
12. Nogales, J. *et al.* High-quality genome-scale metabolic modelling of *Pseudomonas putida* highlights its broad metabolic capabilities. *Environ. Microbiol.* **22**, 255–269 (2020).
  13. Haverkorn van Rijsewijk, B. R. B., Nanchen, A., Nallet, S., Kleijn, R. J. & Sauer, U. Large-scale <sup>13</sup>C-flux analysis reveals distinct transcriptional control of respiratory and fermentative metabolism in *Escherichia coli*. *Mol. Syst. Biol.* **7**, 477 (2011).
  14. Tierrafría, V. H. *et al.* RegulonDB 11.0: Comprehensive high-throughput datasets on transcriptional regulation in *Escherichia coli* K-12. *Microb Genom* **8**, (2022).

# **Appendix C**

## **Supplementary Information for Chapter 3**

### **C.1 Training sets presentation**

The following figures (C.1 and C.2) show visual depictions of different training sets inputs (also called features, independent variables, or  $X$ ) and outputs (also called labels, dependent variables, or  $Y$ ).

## Training set inputs

## Training set output and metrics

Columns as uptake fluxes upper bounds  
Rows as data points

Distribution of **Growth Rate**,  
**V norm** and **SV norm** found with FBA

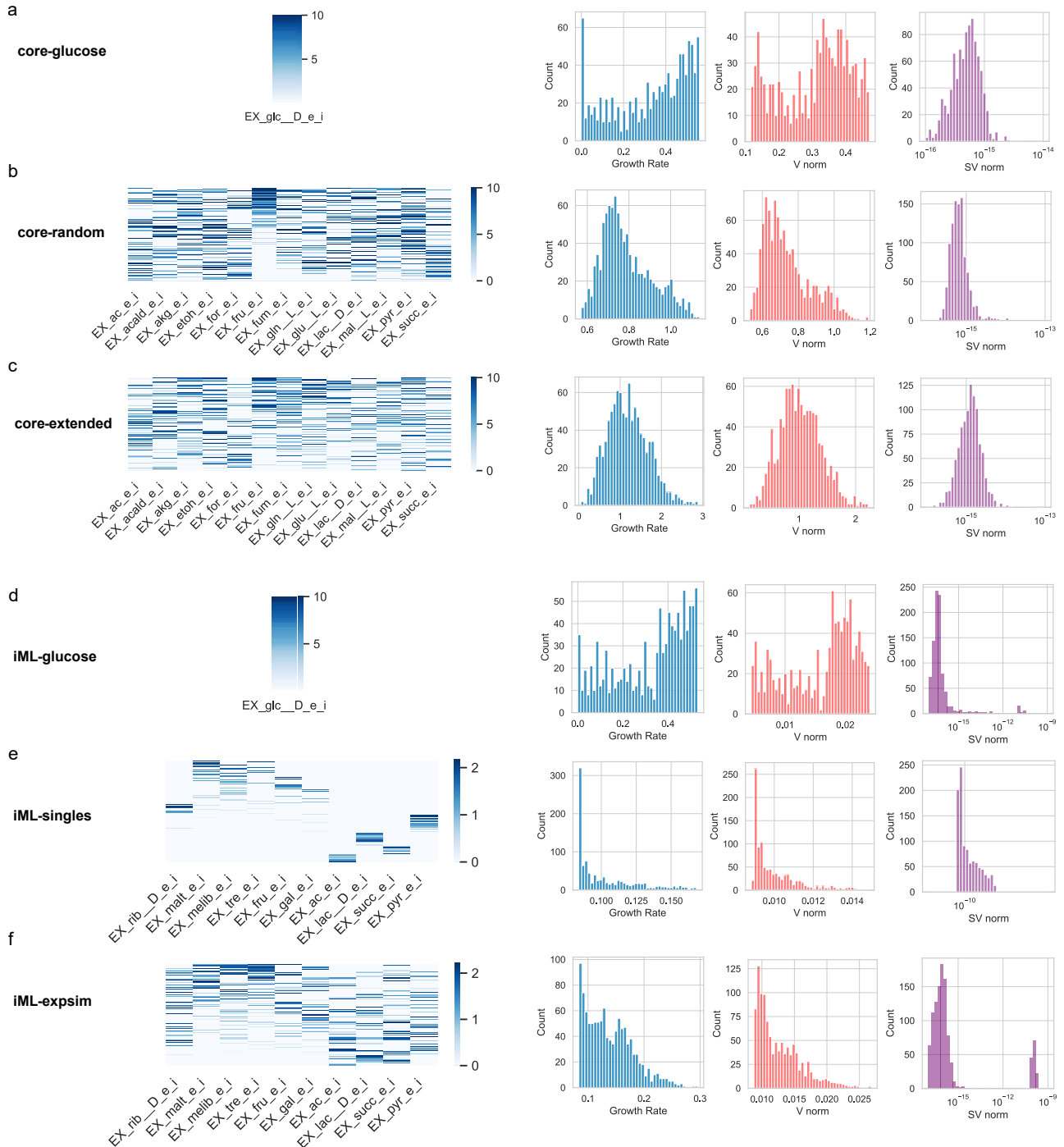


Figure C.1: *in silico* training sets generated by FBA on *E. coli* core or iML1515[44]. For visualization purposes, each heatmap's rows were ordered by decreasing optimized growth rate value obtained by FBA; and minimal medium (*i.e.*, essential uptake fluxes for the GEM to predict growth, see Chapter 2 Methods 2.5, 'Generation of training sets with FBA') was removed from the heatmaps.

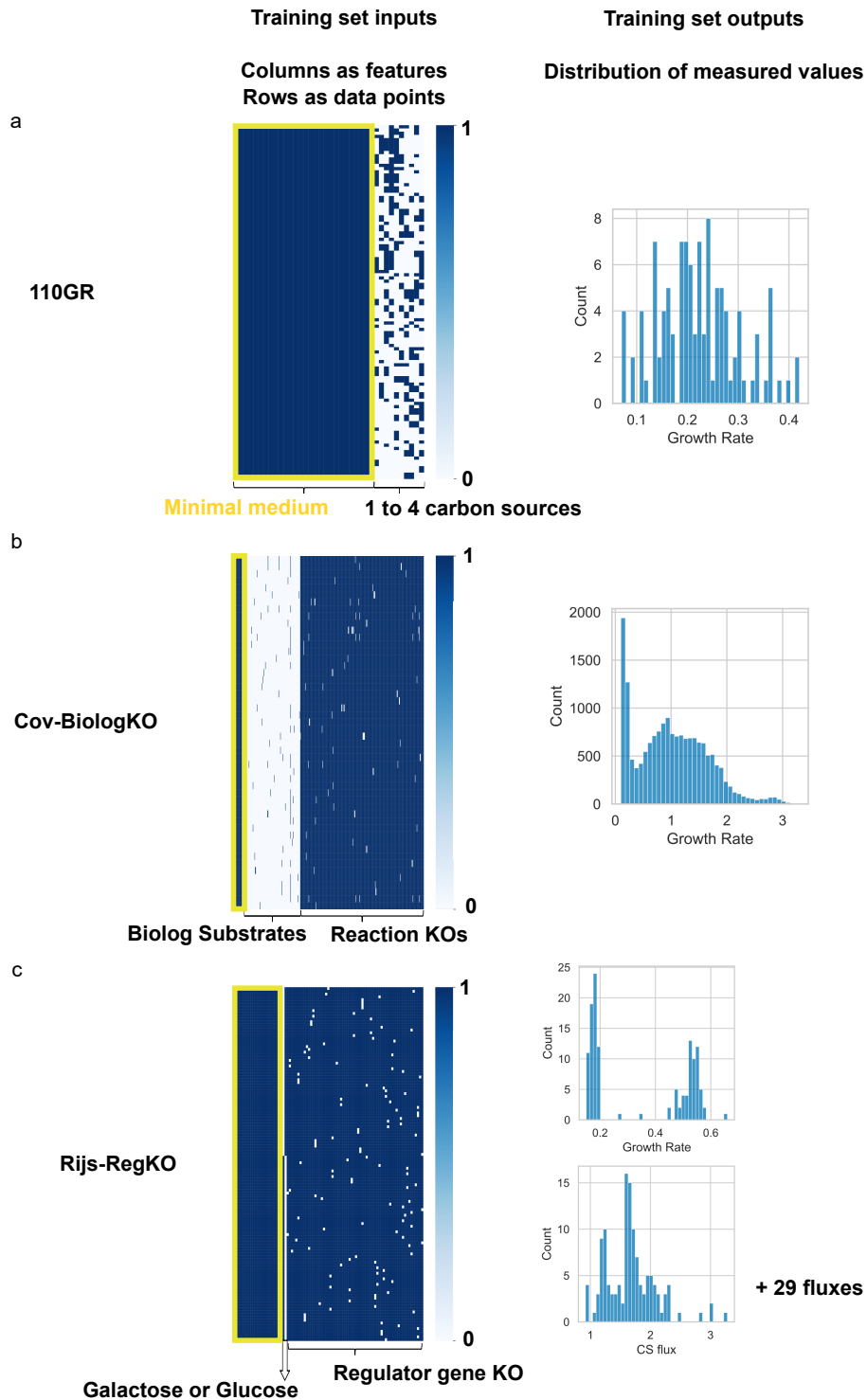


Figure C.2: *in vivo* training sets generated from experimental data. For visualization purposes, each heatmap's rows were ordered by decreasing growth rate value obtained experimentally. The minimal medium (i.e., essential uptake fluxes for the GEM to predict growth, see Chapter 2 Methods 2.5, 'Generation of training sets with FBA') is indicated by a yellow box for each experimental dataset.

In the following table C.1 is provided the path to the random drawing instruction files used to generate *in silico* training sets, or upper bounds used for *in vivo* training sets, in the github repository 'amn\_phd' (see following section 'Code and Data availability' C.2).



Context	Repository folder and file content	Name	File name
<i>in silico</i> (FBA simulations)	Data/Constraints/ Random drawing rules	core-glucose	e_coli_core_glucose_only.csv
		core-random	e_coli_core.csv
		core-extended	e_coli_core_extended.csv
		iML-glucose	iML1515_glucose_only.csv
		iML-random	iML1515_random.csv
		iML-extended	iML1515_extended.csv
		iML-singles	iML1515.csv (drawing_mode='random')
<i>in vivo</i> (experimental data)	Data/Experimental/ Default bounds for substrates found in media composition	iML-expsim	iML1515.csv (drawing_mode='expsim')
		110GR	iML1515_EXP_bounds.csv
		Cov-BiologKO	biolog_iML1515_EXP_bounds.csv
		Rijs-RegKO	rijs_iML1515_EXP2_bounds.csv

Table C.1: Drawing rules and default bounds file locations, used to generate *in silico* and *in vivo* training sets. The repository can be found at this [address \(clickable link\)](#). Note that 'iML-singles' and 'iML-expsim' use the same drawing rule file with a different 'drawing\_mode' argument used in the training set generation.

## C.2 Code and Data Availability

All codes and data to generate training sets, models and figures presented in Chapter 3 can be found in this [github repository \(clickable link\)](#).

Here is a non-exhaustive list of improvements made in this repository compared to the 'amn\_release' repository developed for Chapter 2's publication (see section 2.8):

- Saving AMN-Wt models is possible.
- UB and LB can be used as inputs of AMN-QP-bnds, enabling compatibility of 'Cov-BiologKO', 'Rijs-RegKO', and other datasets with a single model encoding.
- Dense layers are now tunable in size, activation function and dropout rate. When using several layers, these parameters are accessible for each layer independently. L1 and L2 regularization terms were also added as possible parameters.
- Weighing of each loss term is possible, both in mechanistic layers and in custom mechanistic loss.
- A hyperparameter optimization framework compatible with all AMNs and most parameters is now available.
- An easier saving and loading of training sets and models was performed, with easier tuning of attributes, thanks to the use of compressed pickles of the objects instead of parsed strings to retrieve an object's attributes.
- A more centralized catching of errors that come from incompatible parameters (it can be found in the `__init__` functions of TrainingSet, NeuralModel, and RC objects).
- More functions to analyze predictions and compare them to FBA training sets. All these are stored within the function file 'Library/analysis.py'.
- The scaler applied on X is now saved as an attribute of the NeuralModel object.

### C.3 QP-solver reformulation and optimization

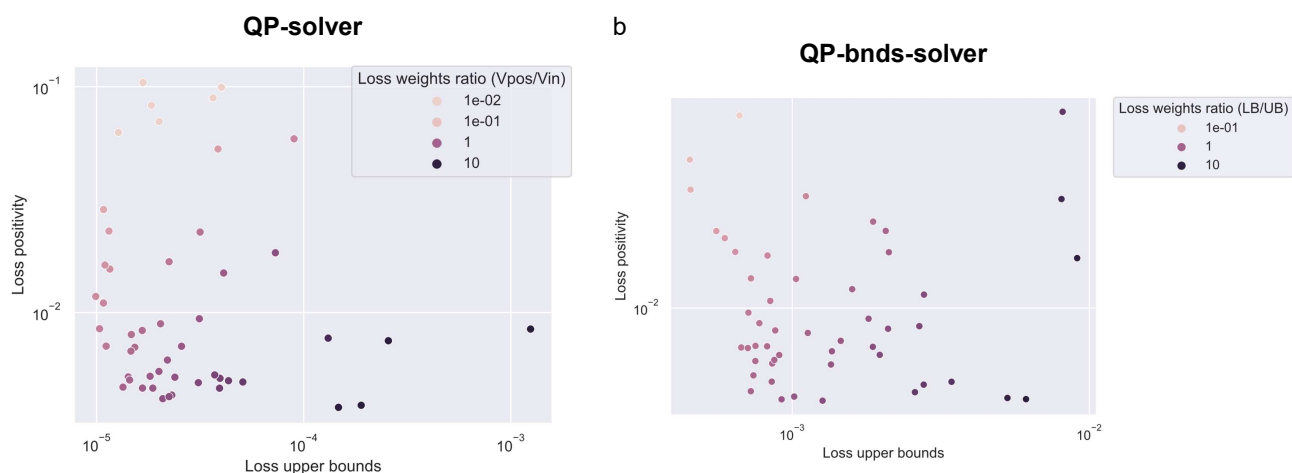


Figure C.3: Pareto plots of QP-solver (a) and QP-bnds-solver (b) do not show a clear solver's tradeoff between satisfying lower bounds and upper bounds constraints. For both plots, each of the 500 points displays mean metrics for 10 conditions of the 'core-glucose' training set, computed after a mechanistic method run of 10,000 iterations. Plots show unweighted 'Loss upper bounds' (either 'Loss  $V_{in}$ ' or 'Loss UB') on the X-axis, and unweighted 'Loss positivity' (either 'Loss  $V_{pos}$ ' or 'Loss LB') on the Y-axis. Each point's color corresponds to the ratio between the 'Loss weight ( $V_{in}$ )' or 'Loss weight (UB)' and the 'Loss weight ( $V_{pos}$ )' or 'Loss weight (LB)'. Both solvers behave similarly, showing no clear tradeoff and optimal weights ratio of 1.

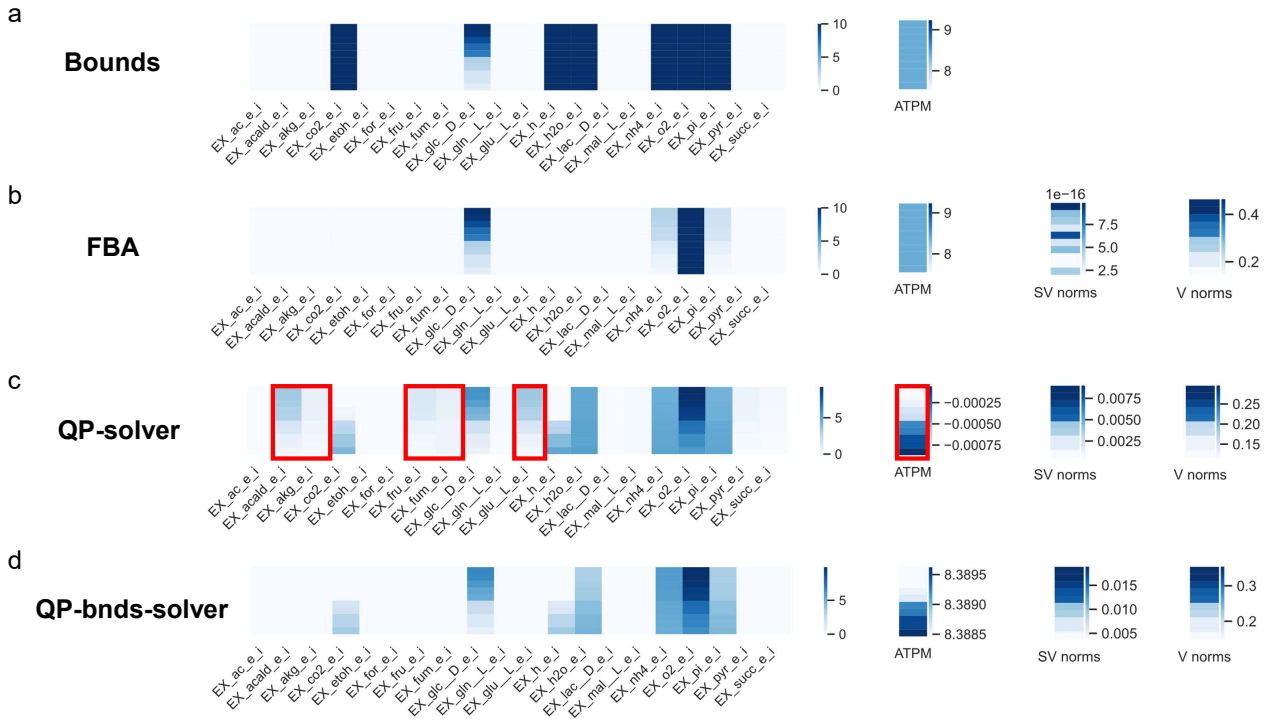


Figure C.4: QP-bnds-solver reformulation improves the respect of GEM constraints. For visualization purposes, each heatmap's rows were ordered by decreasing glucose uptake flux upper bound. Red boxes indicate a constraint violation by the QP-solver. (a) FBA bounds constraints to generate flux distributions with the *E. coli* core[135] model (10 samples from the 'core-glucose' training set). From left to right: 10 sets (rows) of 20 upper bounds (columns) on uptake fluxes; 10 values of lower bound on ATPM (the only non-zero lower bound). (b) FBA results. From left to right: uptake fluxes, ATPM flux values, SV norm metrics, V norm metrics. (c) Same as in the previous panel, here with QP-solver results (100,000 iterations). (d) Same as in the previous panel, here with QP-bnds-solver results (100,000 iterations). Note that for both QP-solver and QP-bnds-solver, the  $R^2$  on the growth rate was always very close to 1.0, and all loss terms except  $L_2$  were showing low values ( $L_1 < 1e-3$ ,  $L_3 = 0$ ,  $L_4 < 1e-5$ ,  $L_6 = 0$ ,  $L_7 < 1e-5$ ).

Figure C.4 clearly shows the constraints violation by QP-solver when used with conditions taken from the 'core-glucose' training set. Interestingly, the norms of V computed by QP-bnds-solver are closer to those of FBA compared to the original QP-solver. However, the SV norms are higher for QP-bnds-solver, which might be explained by the more stringent constraints that are 'harder' to respect for the method. Strikingly, the SV norms obtained through FBA are many orders of magnitude below the ones obtained with QP-solvers. Even if we push the method to 1,000,000 iterations, it will show SV values higher than  $1e-8$ . Note that with the LP-solver, we obtain SV norms closer to those obtained through FBA, with less iterations, as shown in appendix B section 'MM solvers benchmarking' Figure S6e.

## C.4 Mechanistic layers performance assessment

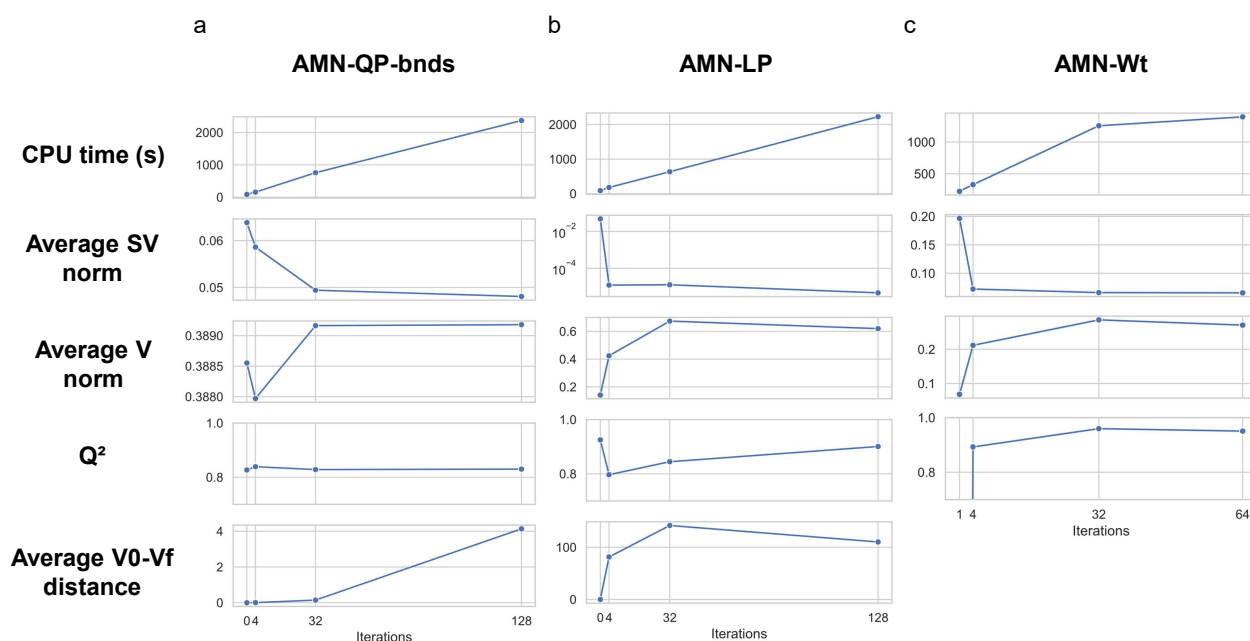


Figure C.5: Performance of AMN-QP-bnds (a), AMN-LP (b) and AMN-Wt (c) for increasing number of iterations of the mechanistic layer. The performance is evaluated with 5 metrics, from top to bottom: the computational time required to perform the 10-fold cross-validation (CPU time (s)), the Average SV norm found in predictions, the Average V norm found in predictions, the  $Q^2$  computed on the growth rate, and the average Euclidean distance between the predicted input of the mechanistic layer ( $V_0$ ) and the predicted output of the AMN ( $V_f$ ). The Euclidean distance could not be computed for AMN-Wt for technical reasons. The AMN-LP model has a clear advantage over other AMNs, since it lowers the SV norm below  $10^{-4}$  with only 4 iterations.

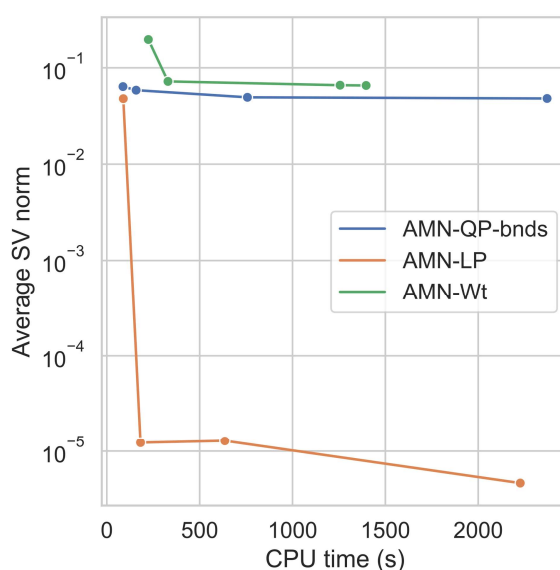


Figure C.6: Increasing number of iterations in the mechanistic layers of AMNs reduces the SV norm but increases computational time. Same data as for Figure 3.5, plotted on a single panel instead of 3 separate panels for each AMN. The advantage of AMN-LP over other AMNs, in its ability to reduce SV norm by its mechanistic layer, is very clear.

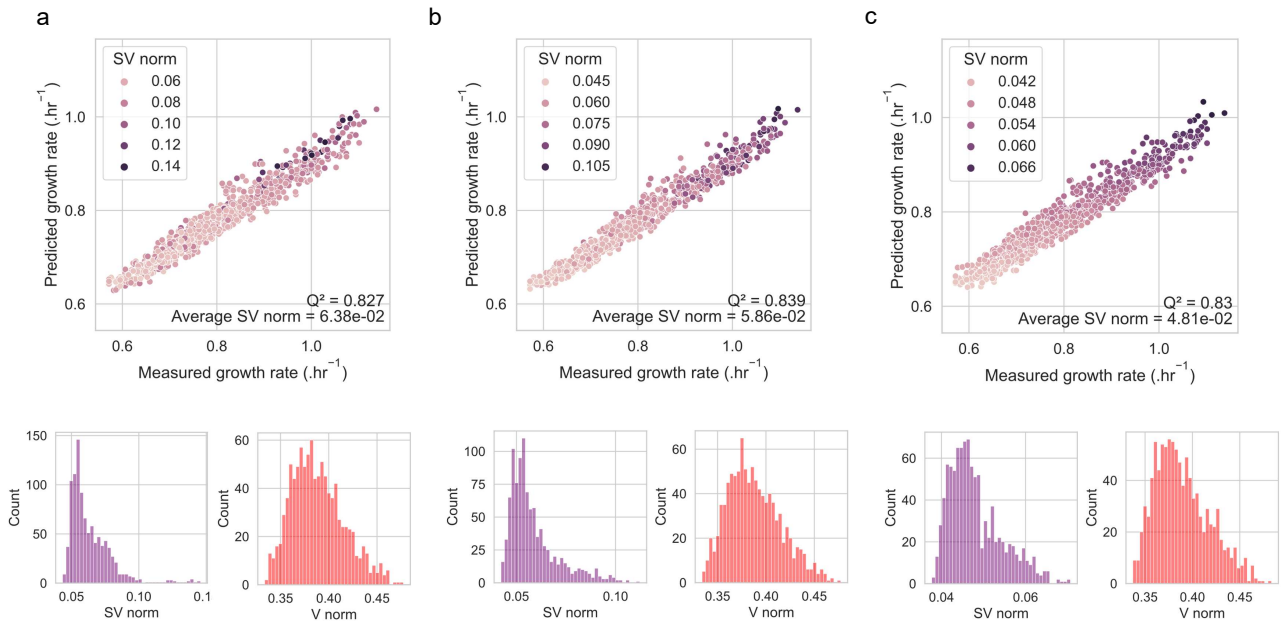


Figure C.7: Increasing the iteration number of the mechanistic layer has limited effect on AMN-QP-bnds prediction performance. All panels display, from top to bottom and left to right: A scatter plot of the growth rate predictions (Y-axis) against the measured growth rates after FBA, i.e. the reference growth rates (X-axis), with points' colors corresponding to the SV norm value computed on the prediction; an histogram of the SV norm metric for all predictions; an histogram of the Euclidean distances between predicted  $V^0$  and predicted  $V_f$ ; and an histogram of the V norm metric for all predictions. (a) 0 iterations, in this case  $V^0$  and  $V_f$  are the same flux vectors, and the mechanistic layer is deleted from the AMN. (b) 4 iterations, (c) 128 iterations.

On Figure C.7, one can observe the limited effect of increasing mechanistic layer iterations on the AMN-QP-bnds performances. Comparing the panel (a) and (c), one can observe a similar  $Q^2$  value and a slightly inferior average SV norm in panel (c). Moreover, the V norm does not seem affected by increasing iteration numbers. We can conclude that, in the tested range, increasing mechanistic layer iterations with AMN-QP-bnds does not improve the overall performance of the model, neither in terms of fitting performance, nor in terms of GEM's constraints respect.

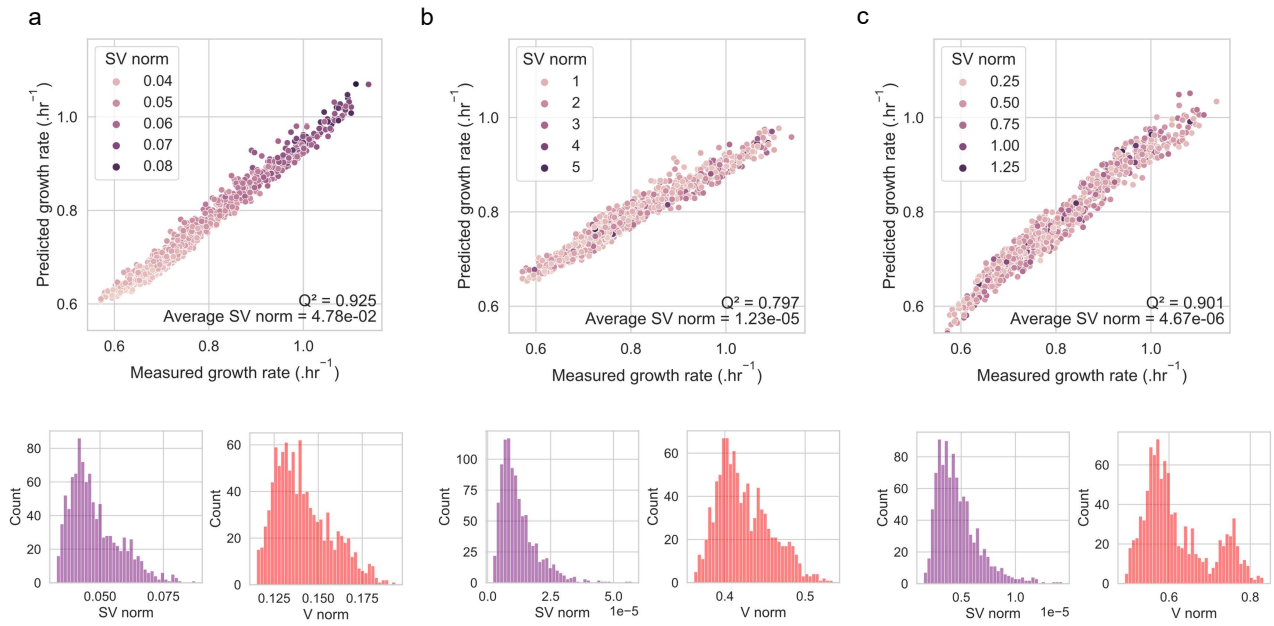


Figure C.8: Increasing the iteration number of the mechanistic layer has a strong effect on AMN-LP prediction performance. Same plot organization as previous figure's. (a) 0 iterations, in this case  $V^0$  and  $V_f$  are the same flux vectors, and the mechanistic layer is deleted from the AMN. (b) 4 iterations, (c) 128 iterations.

On Figure C.8, one can observe the strong effect of increasing mechanistic layer iterations on the AMN-LP performances. Comparing the panel (a) and (c), one can observe a similar  $Q^2$  value and a drastically inferior average SV norm in panel (c), by roughly  $1e4$  smaller. Moreover, the V norm is clearly affected by increasing iteration numbers, reaching more realistic values (Figure C.1 shows the distribution of V norms computed by FBA with the 'core-random' training set, that roughly ranges from 0.6 to 1.2). We can conclude that, in the tested range, increasing mechanistic layer iterations with AMN-LP drastically improves the overall performance of the model in terms of GEM's constraints respect. However, it does not improve the performance in terms of fitting reference fluxes.

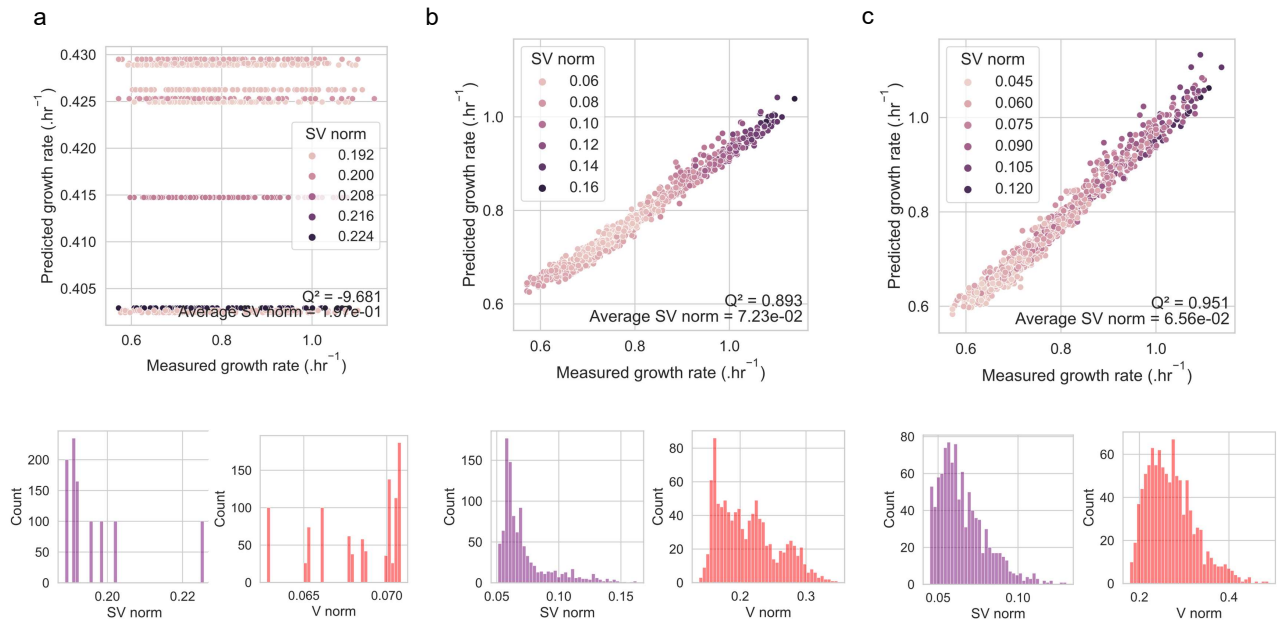


Figure C.9: Increasing the iteration number of the mechanistic layer has a mitigated effect on AMN-Wt prediction performance. Same plot organization as previous figure's. (a) 1 iteration (b) 4 iterations, (c) 64 iterations.

On Figure C.9, one can observe mitigated effects of increasing mechanistic layer iterations on the AMN-Wt performances. Panel (a) shows that 1 iteration is clearly not enough for the AMN to perform correctly, thus we will not consider this panel for comparison. Comparing the panel (b) and (c), one can observe similar  $Q^2$  and average SV norm values. Moreover, the V norm does not seem affected by increasing iteration numbers. We can conclude that, in the tested range, increasing mechanistic layer iterations with AMN-Wt does not significantly improve the overall performance of the model, neither in terms of fitting performance, nor in terms of GEM's constraints respect.

## C.5 AMNs hyperparameter optimization

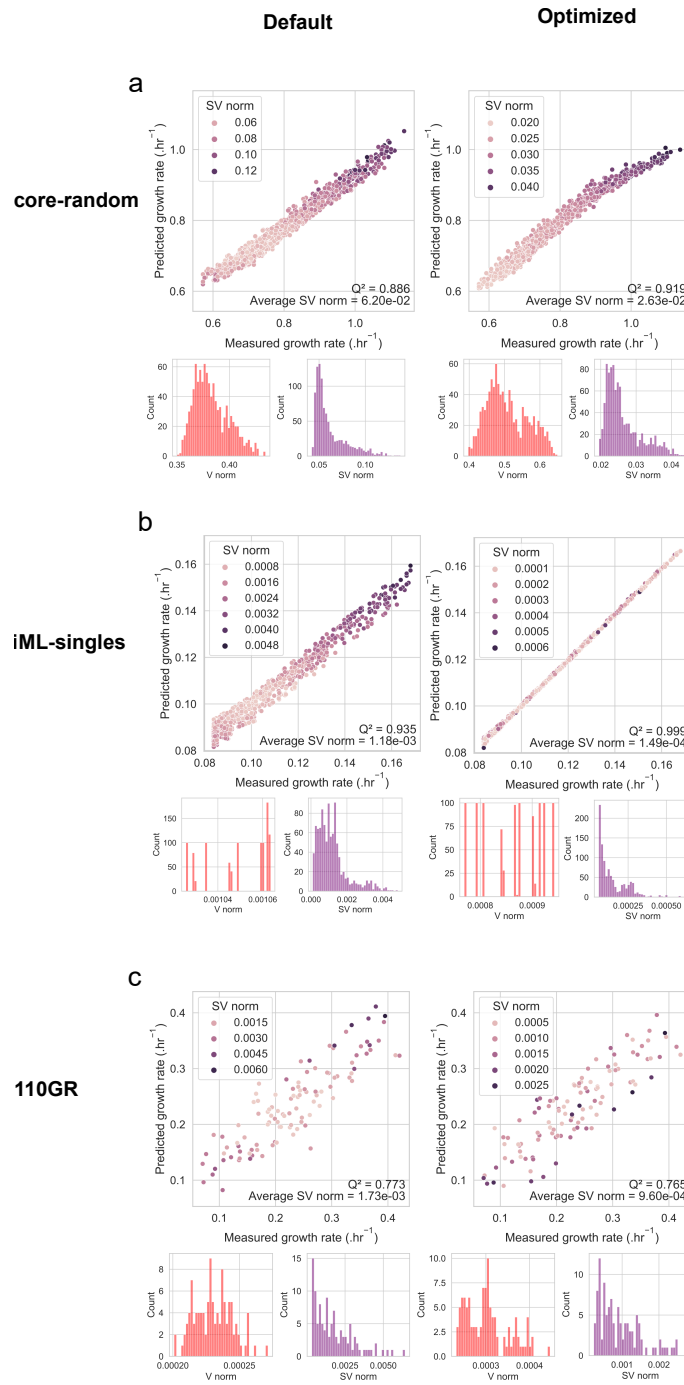


Figure C.10: Comparison of AMN-QP-bnds performance before ('Default', left plots), and after ('Optimized', right plots) hyperparameter optimization; for 3 training sets: (a) 'core-random', (b) 'iML-singles' and (c) '110GR'. In each case is displayed: a scatter plot of the growth rate predictions (Y-axis) against the measured growth rates after FBA, i.e. the reference growth rates (X-axis), with points' colors corresponding to the SV norm value computed on the prediction; an histogram of the SV norm metric for all predictions; and an histogram of the V norm metric for all predictions. In cases (a) and (b), the hyperparameter optimization increased the AMN performance by increasing the  $Q^2$  and decreasing the average SV norm; in case (c) it only improved the AMN performance by decreasing the average SV norm.

The best neural architecture found in the hyperparameter optimization of AMN-QP-bnds with 'core-random' (red points on Figure 3.6) was found to have 3 hidden layers of size 102, 97 and 16, with hyperbolic tangent activation function for the two first hidden layers and ReLU for the last one. The



optimal dropout rates were found to be 0.015, 0.022 and 0.017.

With that architecture, the best weights were found to be 1.6, 8.22 and 8.14, respectively applied on  $L_2$ ,  $L_6$  and  $L_7$ . This set of weights yields the yellow point on Figure 3.6 panel b, which shows a good tradeoff between  $Q^2$  and SV norm.

The best neural architecture found in the hyperparameter optimization of AMN-QP-bnds with ‘iML-singles’ (red points on Figure 3.6) was found to have no hidden layers. In that case  $V_{in}$  is directly connected to  $V_f$ .

With that architecture, the best weights were found to be 7.4, 2.7 and 9.3, respectively applied on  $L_2$ ,  $L_6$  and  $L_7$ . This set of weights yields the yellow point on Figure 3.6 panel b, which shows the best performance both in terms of  $Q^2$  and SV norm.

The best neural architecture found in the hyperparameter optimization of AMN-QP-bnds with ‘110GR’ (red points on Figure 3.6) was found to have 1 hidden layer of size 48 with hyperbolic tangent activation function. The optimal dropout rate was found to be 0.45. This architecture was found to be the best  $Q^2$  and ‘Loss SV’ tradeoff, as it yields a bottom-right point on panel a.

With that architecture, the best weights were found to be 2.7, 6.8 and 7.8, respectively applied on  $L_2$ ,  $L_6$  and  $L_7$ . This set of weights yields the yellow point on Figure 3.6 panel b, which shows a good tradeoff between  $Q^2$  and ‘Loss SV’.

## C.6 Alternative ML models to better surrogate FBA

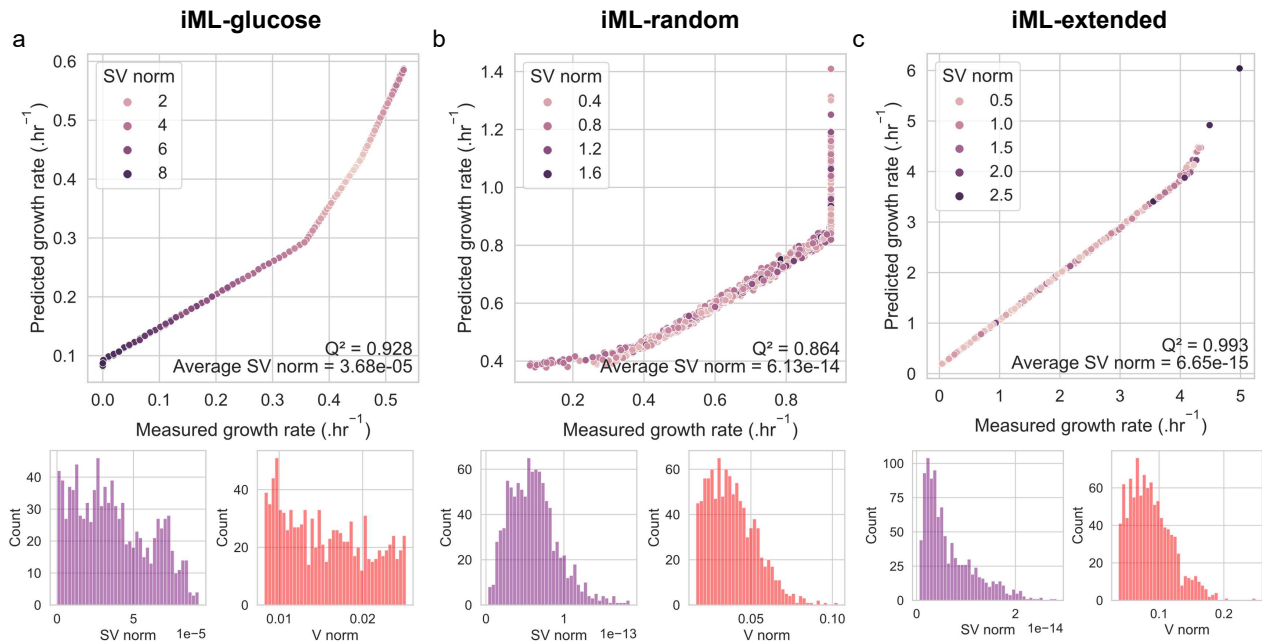


Figure C.11: MTEN performance with ‘iML-glucose’, ‘iML-random’, ‘iML-extended’ training sets. Each panel shows a scatter plot of the growth rate predictions (Y-axis) against the measured growth rates after FBA, i.e. the reference growth rates (X-axis), with points’ colors corresponding to the SV norm value computed on the prediction; an histogram of the SV norm metric for all predictions; and an histogram of the V norm metric for all predictions.

## C.7 Extended results for 'Rijs-RegKO'

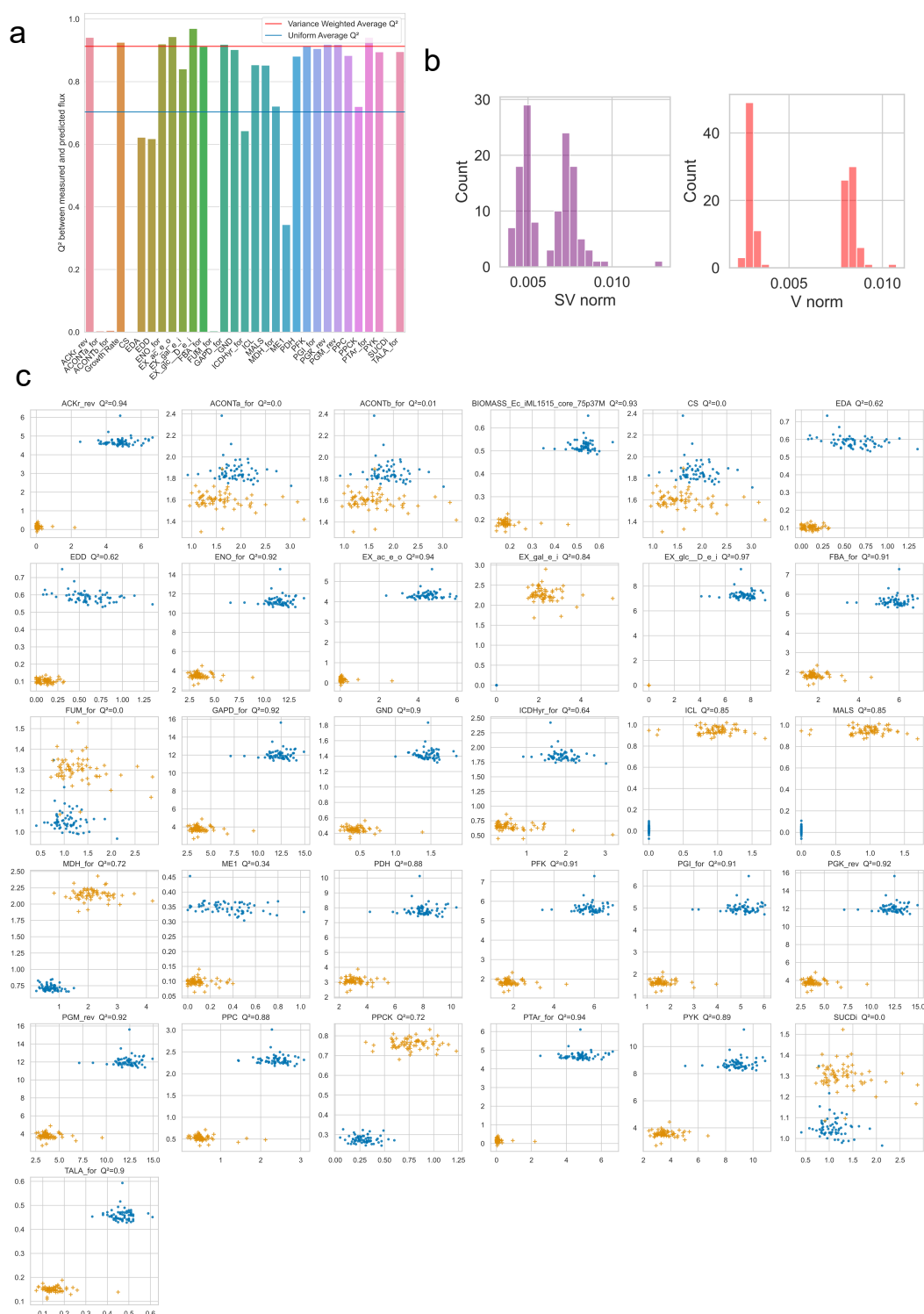


Figure C.12: AMN-QP-bnds performance with 'Rijs-RegKO' (regular approach, as in appendix B Figure S8). (a) Performance chart of AMN-QP-bnds.  $Q^2$  is displayed for each flux individually with the bars of the chart, and we also show the variance weighted (red line, 0.91) and uniform (blue line, 0.70) averages of  $Q^2$ s of all fluxes. (b) Histograms of SV norm and V norm metrics for all predictions. (c) Individual fluxes scatter plots of the predictions (Y-axis) against the experimentally measured flux (X-axis), with points' style corresponding to the carbon source (yellow crosses: glucose, blue dots: galactose).

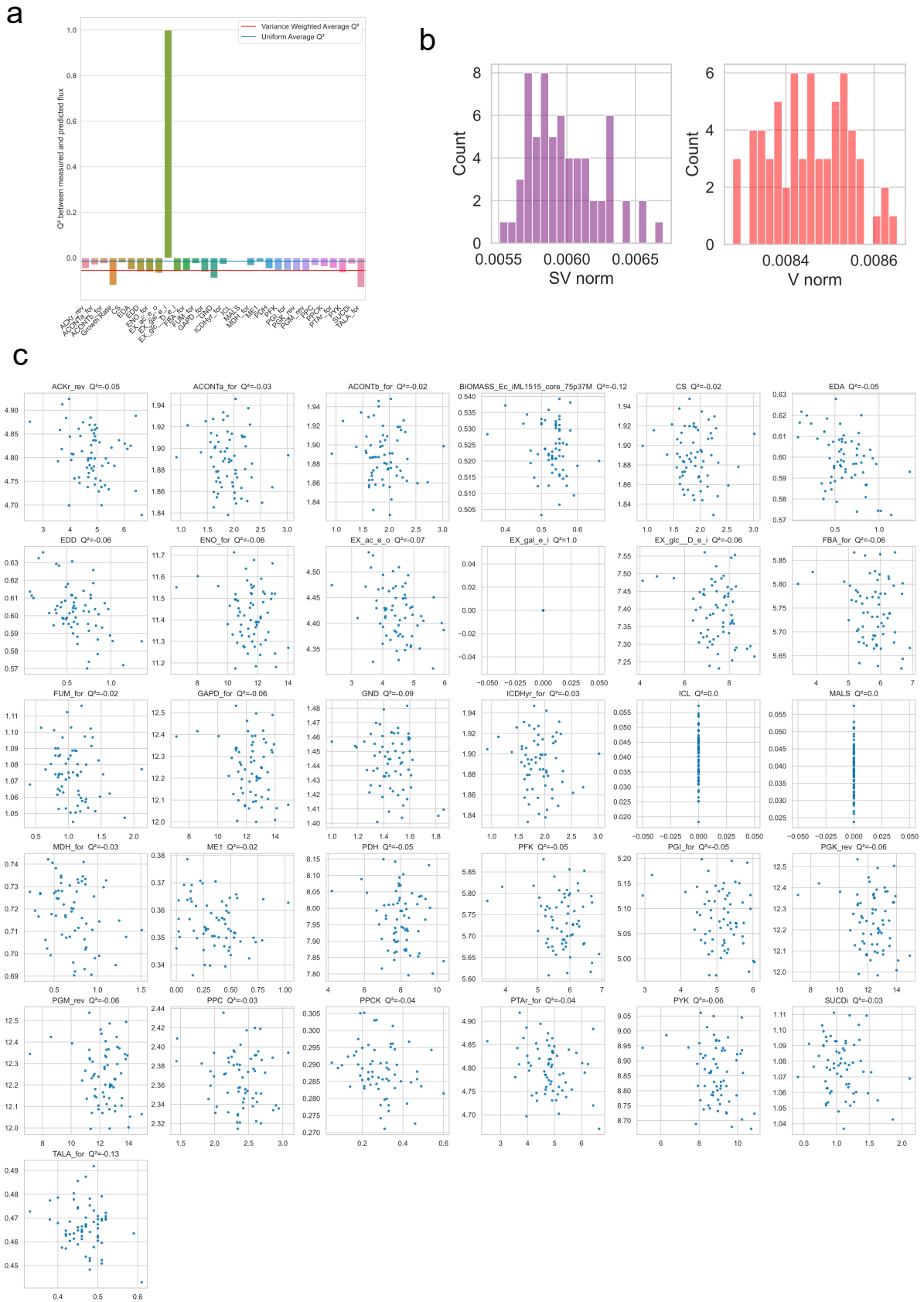


Figure C.13: AMN-QP-bnds performance with 'Rijs-RegKO-glucose'. Same plot organization as previous figure's. On panel (a), we show the variance weighted (red line, -0.05) and uniform (blue line, -0.01) averages of  $Q^2$ s of all fluxes.



Figure C.14: AMN-QP-bnds performance with 'Rijs-RegKO-galactose'. Same plot organization as previous figure's. On panel (a), we show the variance weighted (red line, -0.05) and uniform (blue line, 0.01) averages of  $Q^2$ s of all fluxes.

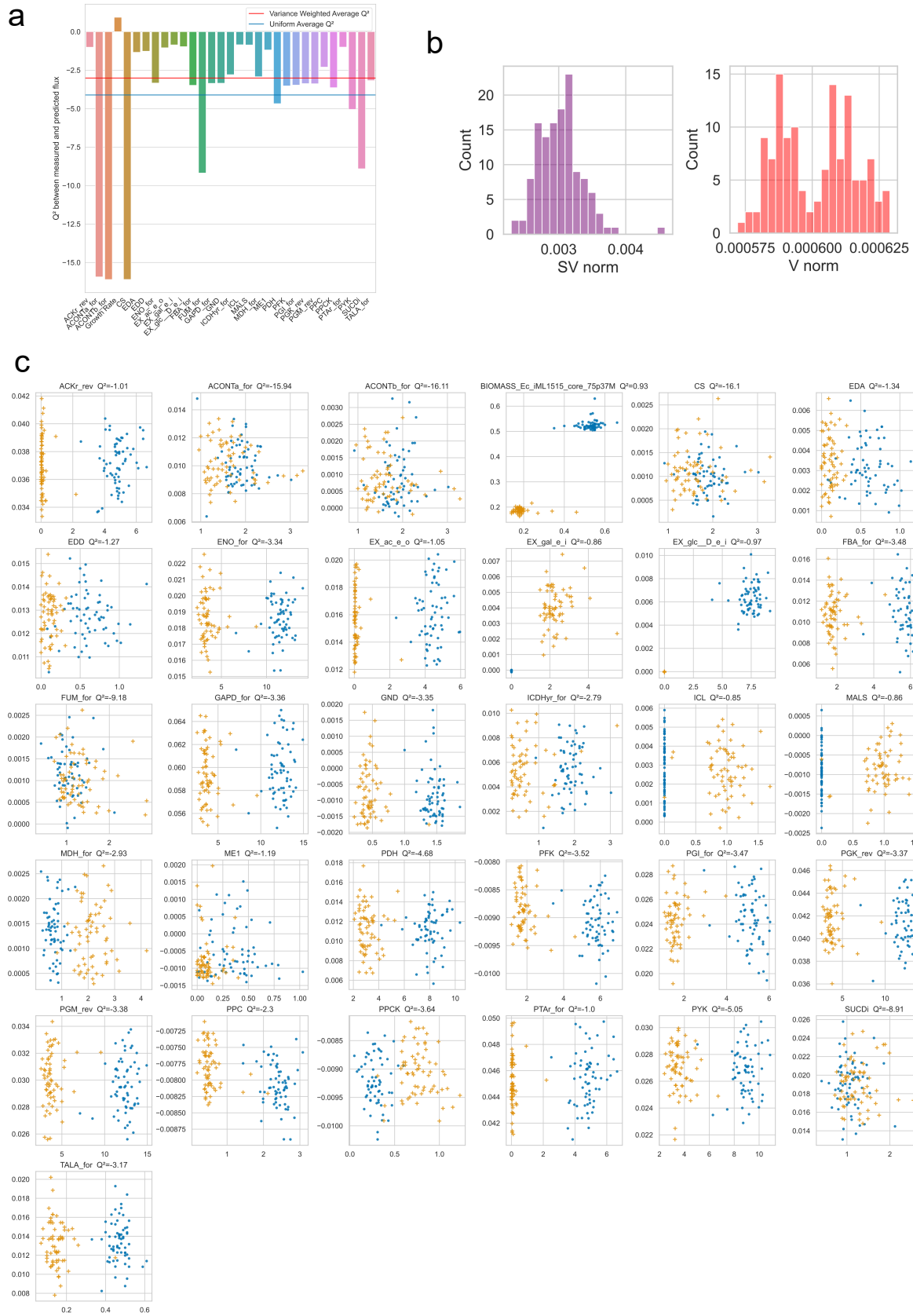


Figure C.15: AMN-QP-bnds performance with 'Rijs-RegKO-GR-only'. Same plot organization as previous figures. On panel (a), we show the variance weighted (red line, -3.0) and uniform (blue line, -4.0) averages of  $Q^2$ s of all fluxes.

## **Appendix D**

**Co-authored publication n°1: *In silico, in vitro, and in vivo* machine learning in synthetic biology and metabolic engineering**



# *In silico*, *in vitro*, and *in vivo* machine learning in synthetic biology and metabolic engineering

Jean-Loup Faulon<sup>a</sup> and Léon Faure

## Abstract

Among the main learning methods reviewed in this study and used in synthetic biology and metabolic engineering are supervised learning, reinforcement and active learning, and *in vitro* or *in vivo* learning.

In the context of biosynthesis, supervised machine learning is being exploited to predict biological sequence activities, predict structures and engineer sequences, and optimize culture conditions.

Active and reinforcement learning methods use training sets acquired through an iterative process generally involving experimental measurements. They are applied to design, engineer, and optimize metabolic pathways and bioprocesses. The nascent but promising developments with *in vitro* and *in vivo* learning comprise molecular circuits performing simple tasks such as pattern recognition and classification.

## Address

MICALIS Institute, INRAE, University of Paris-Saclay, Jouy-en-Josas, France

Corresponding author: Faulon, Jean-Loup ([Jean-Loup.Faulon@inrae.fr](mailto:Jean-Loup.Faulon@inrae.fr))

<sup>a</sup> [www.jfaulon.com](http://www.jfaulon.com).

Current Opinion in Chemical Biology 2021, 65:85–92

This review comes from a themed issue on **Machine Learning in Chemical Biology**

Edited by **Conor Coley** and **Xiao Wang**

For a complete overview see the [Issue](#) and the [Editorial](#)

<https://doi.org/10.1016/j.cbpa.2021.06.002>

1367-5931/© 2021 Elsevier Ltd. All rights reserved.

## Keywords

Synthetic biology, Metabolic engineering, Machine learning, Active learning, Reinforcement learning, Artificial neural networks, Perceptron.

## Introduction

We have seen in the past few years a growing interest in using machine learning for chemistry and biology, synthetic biology and metabolic engineering making no exception to this trend [1]. This study reviews three main techniques used when engineering biological systems. In section 2, we present an overview of supervised and semisupervised machine learning techniques,

providing examples on searching for promiscuous enzyme activities. In section 3, we discuss active learning (AL) and reinforcement learning (RL) methods, which are generally based on supervised learning, with training sets acquired on the fly in an iterative process. These methods are particularly amendable to the design-build-test-learn synthetic biology cycle. Examples are provided in the context of predicting enzymatic activities, optimizing metabolic pathways, and performing retro-biosynthesis. Engineering information processing devices in living systems is a long-standing venture of synthetic biology. Yet, the problem of engineering devices that perform basic operations found in machine learning remains largely unexplored. Section 4 presents attempts to construct *in vitro* and *in vivo* perceptrons which are the basic units of all artificial neural networks.

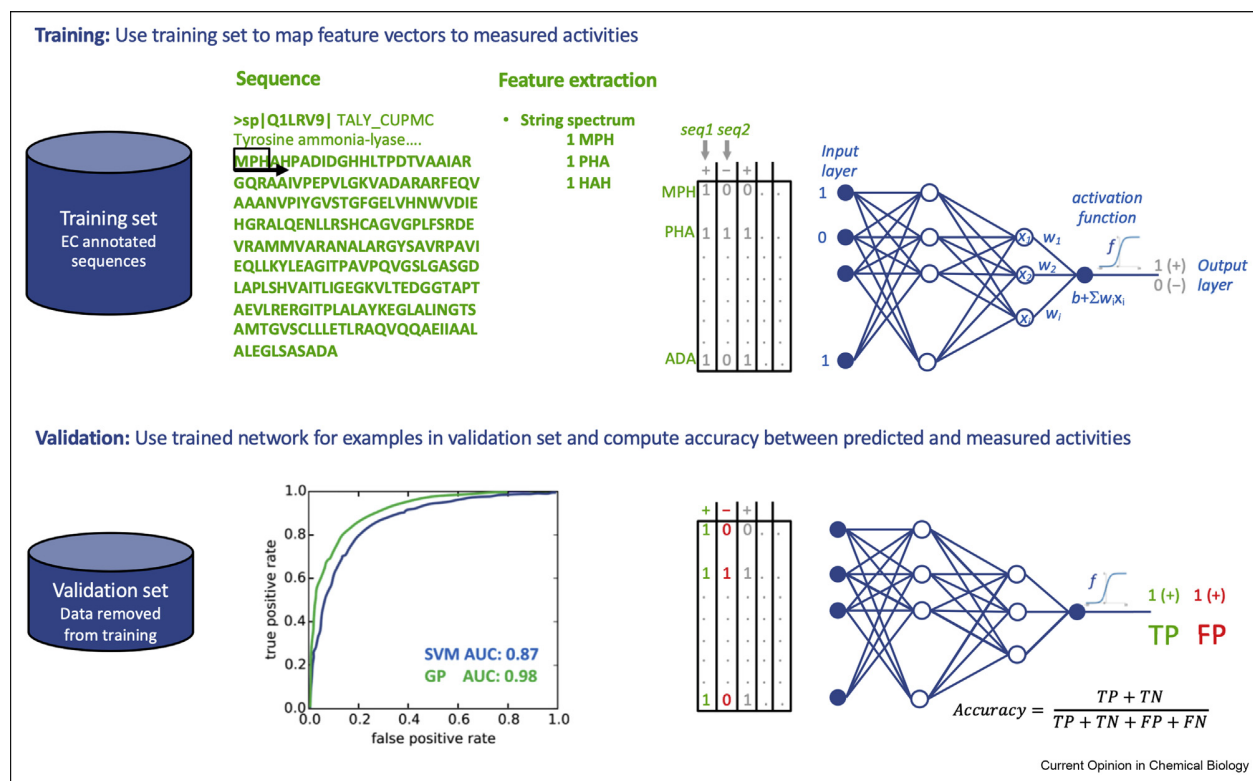
## Supervised and semisupervised learning

Supervised learning is one of the main machine learning methods that is being used in biology and in particular in bioinformatics where it has been extensively developed [2]. Focusing on biosynthesis, and to name a few, supervised learning enables one to predict enzyme activities [3–6], to propose protein structures [7], to engineer sequences (DNA, RNA, protein) [8–11], to complete metabolome [12], to optimize culture conditions [13], and to perform more unexpected tasks like predicting the lab-of-origin of engineered DNA [14]. The supervised learning workflow starts by compiling a training set where each object being studied (promoter sequence, ribosome-binding site sequence, protein sequence, pathways,...) has been labeled (with strength, activity, flux,...). In life sciences, labels generally correspond to experimental measurements. Disregarding the machine learning technique used, the workflow is composed of two main steps: (1) training and (2) validation (cf. Figure 1). Training is not performed on the entire data set but a fraction of it, the rest being set aside for validation.

The core of supervised learning is of course the learning step, where a mapping between the objects and the labels is established. Several mapping techniques have been used in synthetic biology and metabolic engineering including support vector machine (SVM) for classification or support vector regression (SVR) [3], random forests [15], Gaussian processes (GPs) [16], and



Figure 1



Typical machine learning process to predict and validate enzyme activity from sequence. We assume here we have collected a set of sequences having a given EC number (at any level of the EC nomenclature) along with a set of sequences not having that EC number. We wish to learn if any given sequence belongs to the chosen EC class or not. This is a classification problem. See main text for additional details.

artificial neural networks (ANNs) including deep learning networks [4–13]. While Figure 1 illustrates the process with ANNs, with all machine learning techniques one must first transform the objects into vectors of integers or reals. Feature extraction is generally the method preferred to compute these vectors. For biological sequences, string spectrum [17] (count of kmers on the top left side of Figure 1), motif counts [4] (like Pfam domains), and one-hot encoding and embeddings [5] are common features that are used. With feature vectors in hands, all machine learning techniques mentioned previously search for a linear or nonlinear function mapping features to labels. When using an ANN (right side of Figure 1), the function is a recursive weighted sum starting with the feature vector (input layer), propagating to hidden layers to reach an output layer composed of only one node. As we wish a 0 or 1 answer, the value of the last layer is generally calculated through a sigmoid function. Learning here consists in finding the weights ( $w_i$ ) for each weighted sum minimizing the difference between the values (0, 1) calculated at the last layer and the values in the training set. During validation, values are predicted using the trained

network, and true or false positives or negatives are recorded. Receiver operating characteristic (ROC) curves (bottom left side of Figure 1) can also be calculated on the validation set changing the threshold between positives and negatives. The numbers provided in Figure 1 are areas under the ROC curves that have been reported in the past when using SVMs [3] and GPs [16].

In some instances, it is necessary to merge different types of features together, for instance when building a classifier to determine if a given sequence will metabolize a given substrate. To merge biological and chemical information, one strategy is to compile feature vectors separately for each object and then merge these into a tensor product [3], the tensor representing the interactions between the objects (sequence-chemical). Such a strategy has shown to outperform other techniques for drug–target interactions [18] and enzyme–substrate interactions [3].

When dealing with classification problems, like for instance finding if a particular sequence is a promiscuous enzyme [19], we need positive and negative examples in



the training set. Yet, very often, we are faced with the issue that only positives can be found, as failures are hardly reported in the literature. Here comes therefore the problem of generating negative examples. In the past, this issue has been tackled using ad hoc methods to generate negatives arbitrarily, for instance, randomly drawing sequences that are not annotated as those in the training set or sequences and chemicals that are distant (similarity wise) to those in the training set [19]. A more thorough method is using semisupervised learning [20]. In semisupervised learning, the data set is composed of two classes, a class of labeled examples (either positive or negative when performing classification) for which measurement has been carried out and a class of unlabeled examples. The learning process consists in finding the best partition between positives and negatives by shuffling unlabeled data points either in the positive class or in a negative class.

Although numerous studies are making use of machine learning in the life sciences, in the context of biosynthesis and bioengineering, only a few studies have triggered experimental validation. One can cite a work making use of a semisupervised GP [16], which predicted three native *Escherichia coli* BL21 enzymes capable of synthesizing L-acetyl-leucine. These enzymes (ECBD0907, ECBD4067, and ECDB4269) are known to transform glutamate into acetyl-glutamate, ornithine into acetyl-ornithine, and glutamate and methyl-oxoalate into oxoglutarate and leucine. When overexpressed, two (ECBD4067 and ECDB4269) of the three enzymes increased the production of acetyl-leucine. Not only this study demonstrated that machine learning could be used to find promiscuous enzyme activities but also revealed that acetyl-leucine was produced in *E. coli*, which was not known before that study. In our second example, the DeepEC deep learning method [5] was used to find alternative EC numbers for YgbJ, an L-threonate dehydrogenase (annotated 1.1.1.411 in Uniprot). For YgbJ, DeepEC predicted an oxidoreductase activity on D-glycerate (1.1.1.60). A follow-up enzymatic assay revealed that YgbJ was indeed able to metabolize both D-glycerate and L-threonate.

### Active learning and reinforcement learning

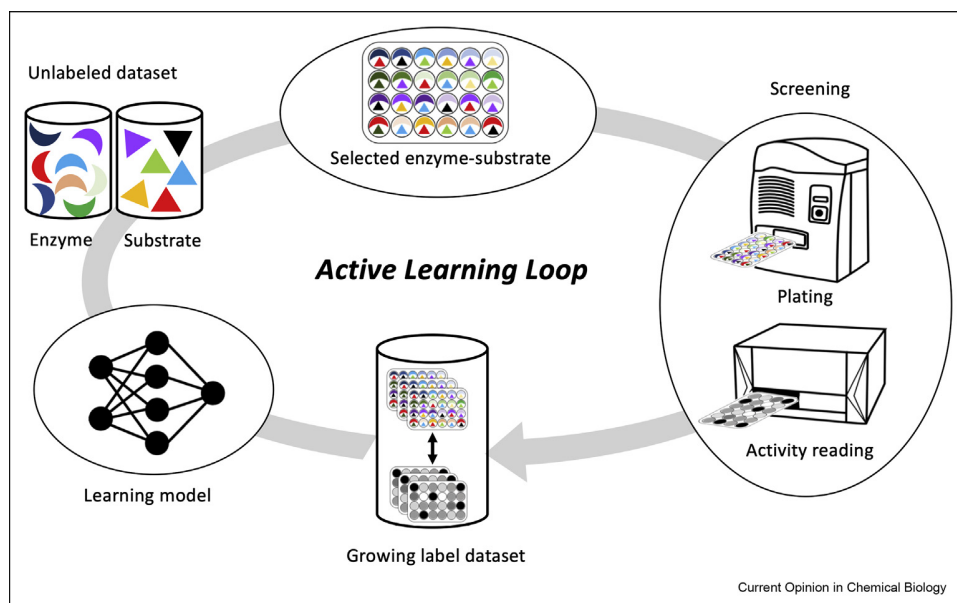
AL is a special case of supervised machine learning, where a learner (any learning algorithm mentioned in the previous section) can interactively query an oracle (a human, a robot, a computer simulation) to ask new data points to be labeled [21]. The process is iterative, and the training set is acquired and growing on the fly. Because the learner chooses the examples to be labeled, the number of examples can be made lower than the number required in normal supervised learning while maintaining performances. For instance, searching novel substrates for a small set of four promiscuous enzymes, it was shown that SVMs trained on

a set of substrates selected by AL performed with 80% accuracies using 33% fewer compounds than when trained on the whole set of substrates [22]. AL is particularly appealing in the context of bioengineering because it reduces the number of experiments to be performed. In addition, AL perfectly fits the design—build—test—learn cyclic process developed in synthetic biology as it proposes a solution to the learning step of the cycle [23].

AL is illustrated in Figure 2 to search alternative substrates metabolized by promiscuous enzymes. The process starts by asking to label (i.e. perform measurements) an initial set of data points (enzyme x substrate pairs). Each data point is described by features; one can use, for instance, chemical fingerprints for substrates and string spectrum or one-hot encoding for sequences. The initial set can be generated by choosing enzyme x substrate pairs at random or better using fractional factorial design [24] to evenly sample the space of possibilities. Measurements are then acquired, eventually using robotic screening equipment, and the pairs with activity measurements are added to the labeled data set. Next, a machine learning algorithm is trained on the labeled data set and used to predict labels from features for all the pairs in the unlabeled set (or a sample of pairs if the whole unlabeled set is too large). Methods mentioned in section 2 like the tensor product can be utilized to perform the predictions. Based on predictions carried out on the unlabeled data set, a new set of enzyme x substrate pairs is selected for the next round of measurement. The selected pairs are screened, the new measurements are added to the growing training set, and the trained model is retained. The process is iterated until the performances of the learner cannot be improved.

The AL process illustrated in Figure 2 is generic; it can be and has been applied to other biosynthesis and metabolic engineering—relevant problems like finding the expression level of enzymes in a pathway producing a target molecule [25–28] or finding buffer composition maximizing cell-free productivity [29]. As an example, coupling robotic equipment with AL, Hamedirad et al. [27] optimized the lycopene biosynthetic pathway evaluating less than 1% of possible variants while outperforming random screening by 77%. One critical step in AL is the selection of new data points to be labeled. AL makes use of two selection modes, exploitation and exploration. In exploitation mode and when maximizing an objective, AL is seeking predicted label values ( $\mu$ ) higher than those already in the training set, whereas in exploration mode, AL is searching for predictions having high variances ( $\sigma$ ), these corresponding to data points that are far away from those being in the training set. As shown in the study by Borkowski et al. [29], the combination of exploitation and exploration via an upper confidence bound formula like  $\mu + k \sigma$  (where

Figure 2



Active learning process applied to search for alternative substrates of promiscuous enzymes.

k is a constant) is efficient in large combinatorial spaces. Indeed, that study demonstrated that fewer than 10 AL iterations were sufficient for a 34-fold cell-free productivity increase, while optimizing buffer composition in a combinatorial space  $> 4 \cdot 10^6$ .

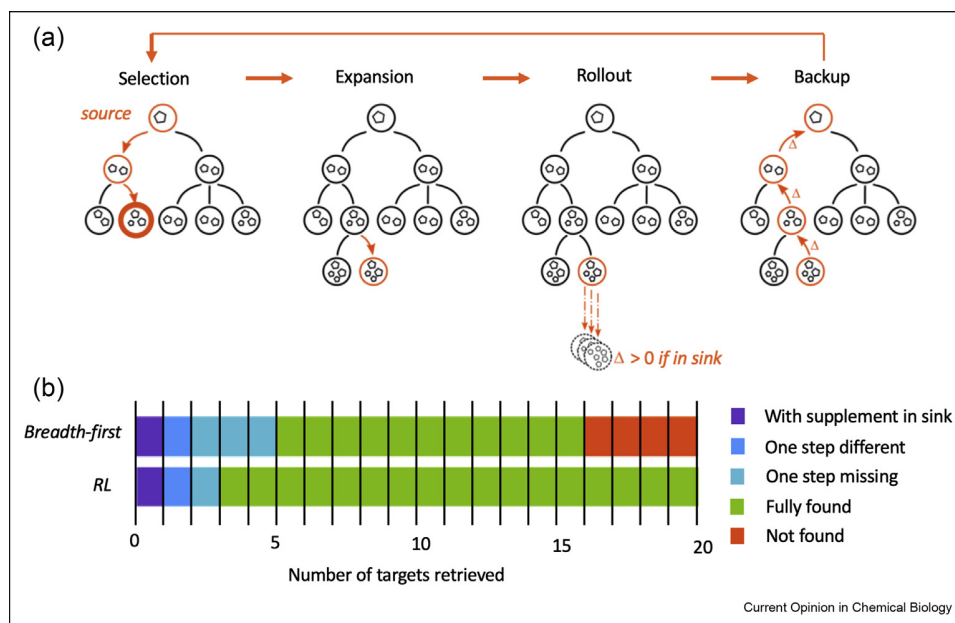
RL is another technique that can be coupled with simulations or experimental measurements. RL was popularized by the Google DeepMind AlphaGO program [30]. It has since been used for retro-biosynthesis [31], synthesis planning of synthetic pathways with green processes [32], and bioprocess optimization [33,34]. The Monte Carlo tree search RL method, developed for the AlphaGO program [30], is outlined in Figure 3a in the context of retro-biosynthesis. Retro-biosynthesis consists in finding heterologous enzymatic reactions transforming the native metabolites of a host strain into a target molecule. Traditional breadth-first search retrosynthesis algorithms [35] proceed from the target (source) to the strain (sink) applying retro reaction rules (rules for reactions that have been reversed). The process is iterated layer by layer until a pathway is found ending in the sink. Monte Carlo tree search does not proceed breadth-first but instead makes use of 4 phases (selection, expansion, rollout, and back-propagation), repeated until a number of iterations are reached. During expansion and rollout, policy networks can be used to return the most appropriate transformations for the set of molecules of the selected node. Supervised or semisupervised methods described in section 2 can be used to train these policy networks.

Figure 3b shows that number of targets successfully retrieved by RL (reported in Ref. [31]) is larger than the number obtained with a classical breadth-first algorithm [35].

### ***In vitro* and *in vivo* learning**

In all the applications we have seen so far, learning is performed *in silico*. In this section, we are interested in performing learning *in vitro* or *in vivo*; the main challenge is therefore to be able to construct molecular devices processing information the same way as the basic blocks of machine learning programs. Two main goals motivate this innovative learning approach. The first, rather theoretical, is to probe to which extent cellular networks can be engineered to learn. The second, more pragmatic, is to develop diagnostic tools for pollutants or diseases [45] making use of *in vitro* or *in vivo* molecular circuits performing learning tasks like classification. Constructing electronic-like devices *in vivo* has been a long-standing endeavor of synthetic biology, and many logic gates [36], switches [37], amplifiers [38], latches [39], and memory devices [40] can be found in the literature. Quite complex logic circuits can now be constructed [39], but building a circuit that would mimic the behavior of a machine learning code is still out of reach. Timing consideration is also a major issue as it takes generally half an hour (the time taken to transcribe and translate genes) to pass information from one circuit layer to another when implemented *in vivo*. Considering the time already taken to train an *in silico* machine learning model, trying to do this *in vivo* appears

Figure 3



Reinforcement learning (RL) with Monte Carlo tree search (MCTS) illustrated with retro-biosynthesis. **(a)** MCTS method. Circles represent nodes and pentagon molecules. Selection: Starting from the root node (here, a chemical state containing the target compound), the best child nodes are iteratively chosen until a leaf node is reached. Typical selection policies are based on exploitation and exploration. Exploitation is computed from a reward value ( $\Delta$ ) received in previous iterations of the algorithm (nodes with high values are favored), and exploration is based on the number of times a node has been visited (nodes with low number of visits are favored). Expansion: Possible transformations are applied on the selected node generating new children. Rollout: If the node is not terminal (the molecules are not in the sink or the maximum number of iterations is not reached), a transformation is sampled from available transformations and the process is repeated. If the node is terminal, a reward (if in sink) or penalty (if not in sink) is returned. Rollout is repeated until a maximum number of steps or the maximal depth of the tree is reached. Backpropagation or update: The reward obtained after exploring the expanded node is returned to its parents to update their values ( $\Delta$ ) and visit counts. **(b)** Performances for a golden set of 20 experimental pathways (cf. [31]). With supplementation (purple) means a supplement has to be provided in the media to identify the correct experimental pathway. One step different (dark blue) means only one step differs from the described pathway, for example, by using a different co-substrate. One step missing (light blue) means the search algorithm found a pathway identical to the experimental one, except one step which was shortcut. Fully found (green) means the experimental pathway was found without restriction. Not found (orange) means the experimental pathway was not found.

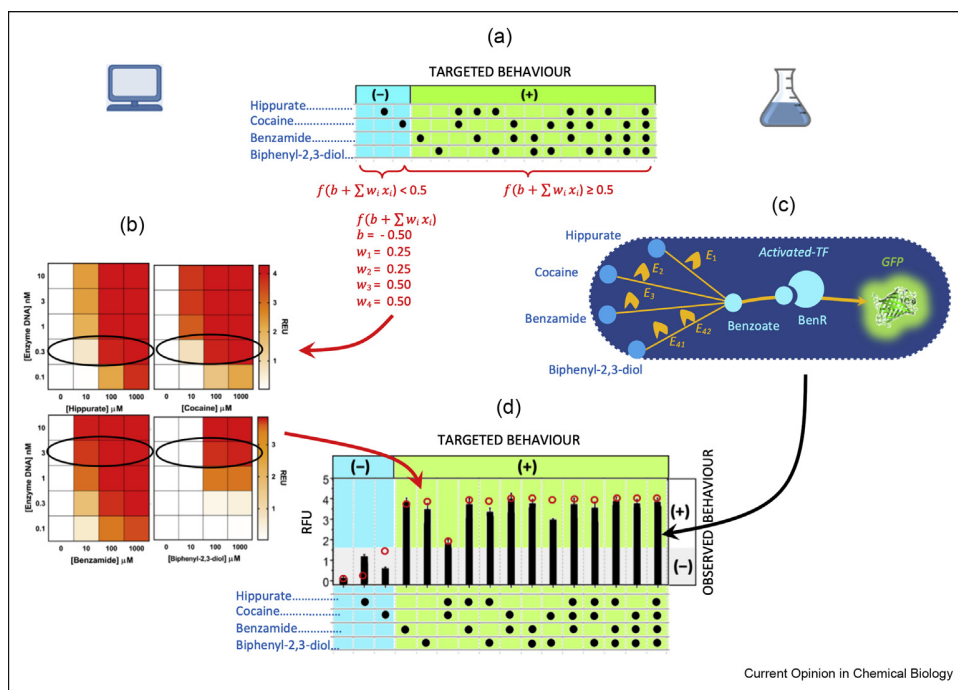
unreasonable. One strategy to overcome the complexity and timing issues is to train the circuits *in silico* and to construct *in vitro* or *in vivo* devices that reproduce the behavior of the trained circuits. That strategy has actually been followed to engineer molecular perceptrons, which are the basic units found in artificial neural networks. In a pioneer work, Qian et al. [41] built a 4-input Hopfield network (a recurrent neural network) using DNA strand displacement. This Hopfield network was trained *in silico* to remember four input patterns: 0110, 1111, 0011, and 1000. Weights were implemented changing the concentrations of the DNA strands used in the circuits.

In a more recent work, presented in Figure 4, Pandi et al. [42] trained a 4-input perceptron to classify 16 input patterns. Figure 4 shows a 16-input pattern perceptron implemented through a metabolic network expressed in cell-free systems. The input

patterns are based on the presence or absence of four input metabolites (hippurate, cocaine, benzamide, and biphenyl-2,3-diol). The observed behavior (relative fluorescent unit) matches well the targeted behavior and the kinetics model predictions (red circles). Other classifiers can be constructed using the same setup by simply changing the weights and the corresponding concentrations of enzyme DNA (cf. Pandi et al. [42] for other examples).

As the last example, a trained perceptron was constructed *in vivo* to classify 12 input patterns [43]. The trained perceptron was implemented engineering two *E. coli* strains: a sender and a receiver. The sender produced quorum molecules (acyl-homoserine lactone 3OHC14:1-HSL), and the receiver was engineered to respond on detection of these molecules by expressing a fluorescent reporter. The perceptron weights were instantiated by varying the promoter strength, affecting

Figure 4



Building a metabolic perceptron. In Figure 4a, a targeted behavior is chosen arbitrary for a classification into positives and negatives. An *in silico* perceptron is trained via simple logistic regression to determine the weights best matching the targeted behavior. Figure 4b shows enzymes metabolizing the input metabolites into benzoate, which is an activator of the transcription factor BenR. BenR is then used to express a green fluorescent protein. In Figure 4c, a kinetics model (cf. Pandi et al. [42] for details) is used to determine the enzyme DNA concentrations corresponding to the weights calculated in panel (a). Finally, the perceptron is constructed in Figure 4d using the enzyme determined in panel (b) and the concentrations of the enzyme DNA calculated in panel (c).

the production level of the quorum molecules in the sender strains.

### Conclusion and perspectives

The use of machine learning in biology will continue to grow. In fact, a search on bioRxiv with the key words 'deep learning' returns about 450 articles deposited each month for the last year and that number nearly doubled between march 2020 (370) and march 2021 (682). However, the number of published articles actually prompting design of experiments and new experimental finding is much smaller. That number will undoubtedly increase as machine learning techniques are being interfaced with robotized workstations allowing automated engineering as it is currently being carried out in chemistry with synthesis planning [44]. One area of particular interest to synthetic biology is the development of molecular devices enabling *in vitro* or *in vivo* learning like the perceptrons presented in section 4. Aside from finding practical applications with biomarker detection and decision-making for medical diagnostics [45], such devices could also be used to probe to what extent molecular and cellular networks

can handle problems currently solved *in silico* and even shed some light on how cognition could emerge from basic molecular circuits, a fundamental and long-standing question [46].

The neural networks of our brains inspired the development of artificial neural networks; perhaps artificial neural networks can now prompt the discovery and engineering of new learning molecular devices in living systems.

### Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

### Acknowledgements

J-L.F. would like to acknowledge funding provided by the ANR funding agency, grant numbers ANR-15-CE21-0008, ANR-17-CE07-0046, and ANR-18-CE44-0015. L.F. is supported by INRAE's MICA department and INRAE's metaprogram BIOLPREDICT.

### References

Papers of particular interest, published within the period of review, have been highlighted as:



\* of special interest  
 \*\* of outstanding interest

1. Carbonell P, Radivojevic T, García Martín H: **Opportunities at the intersection of synthetic biology, machine learning, and automation.** *ACS Synth Biol* Jul. 2019, **8**:1474–1477, <https://doi.org/10.1021/acssynbio.8b00540>.
2. Larranaga P, et al.: **Machine learning in bioinformatics.** *Briefings Bioinf* Mar. 2006, **7**:86–112, <https://doi.org/10.1093/bib/bbk007>.
3. Faulon J-L, Misra M, Martin S, Sale K, Sapra R: **“Genome scale enzyme–metabolite and drug–target interaction predictions using the signature molecular descriptor.”** *Bioinformatics* 2008, **24**:225–233.  
 \*  
 The study presents for the first time a tensor vector product merging biological sequences and chemical structures. The paper shows that accuracies are higher with the tensor product than when learning is performed separately on biological or chemical information.
4. Li Y, et al.: **DEEPred: sequence-based enzyme EC number prediction by deep learning.** *Bioinformatics* Mar. 2018, **34**: 760–769, <https://doi.org/10.1093/bioinformatics/btx680>.
5. Ryu JY, Kim HU, Lee SY: **Deep learning enables high-quality and high-throughput prediction of enzyme commission numbers.** *Proc Natl Acad Sci Unit States Am* Jul. 2019, **116**: 13996–14001, <https://doi.org/10.1073/pnas.1821905116>.  
 \*  
 This study is one of the first to make use of deep learning in the context of EC number prediction.
6. Sureyya Rifaioglu A, Doğan T, Jesus Martin M, Cetin-Atalay R, Atalay V: **DEEPred: automated protein function prediction with multi-task feed-forward deep neural networks.** *Sci Rep* Dec. 2019, **9**:7344, <https://doi.org/10.1038/s41598-019-43708-3>.
7. Senior AW, et al.: **Improved protein structure prediction using potentials from deep learning.** *Nature* Jan. 2020, **577**, 7792, <https://doi.org/10.1038/s41586-019-1923-7>.  
 \*\*  
 The paper presents a deep learning strategy to predict protein structures with high accuracies. The method is exemplified with results obtained for the CASP13 competition (Critical Assessment of Protein Structure Prediction).
8. Wang Y, Wang H, Liu L, Wang X: **Synthetic promoter design in *Escherichia coli* based on generative adversarial network.** *Bioinformatics* Feb. 2019, <https://doi.org/10.1101/563775>.
9. Valeri JA, et al.: **Sequence-to-function deep learning frameworks for engineered riboregulators.** *Nat Commun* Dec. 2020, **11**:5058, <https://doi.org/10.1038/s41467-020-18676-2>.
10. Angenent-Mari NM, Garruss AS, Soenksen LR, Church G, Collins JJ: **A deep learning approach to programmable RNA switches.** *Nat Commun* Dec. 2020, **11**:5057, <https://doi.org/10.1038/s41467-020-18677-1>.
11. Wang J, Cao H, Zhang JZH, Qi Y: **Computational protein design with deep learning neural networks.** *Sci Rep* Dec. 2018, **8**:6349, <https://doi.org/10.1038/s41598-018-24760-x>.
12. Zeleznik A, et al.: **Machine learning predicts the yeast metabolome from the quantitative proteome of kinase knockouts.** *Cell Syst* Sep. 2018, **7**, <https://doi.org/10.1016/j.cels.2018.08.001>. 269–283.e6.
13. Peng W, et al.: **The artificial neural network approach based on uniform design to optimize the fed-batch fermentation condition: application to the production of iturin A.** *Microb Cell Factories* Apr. 2014, **13**:54, <https://doi.org/10.1186/1475-2859-13-54>.
14. Nielsen AAK, Voigt CA: **Deep learning to predict the lab-of-origin of engineered DNA.** *Nat Commun* Aug. 2018, **9**:1, <https://doi.org/10.1038/s41467-018-05378-z>.
15. Chen P, Huang JZ, Gao X: **LigandRFs: random forest ensemble to identify ligand-binding residues from sequence information alone.** *BMC Bioinf* 2014, **15**(Suppl 15):S4, <https://doi.org/10.1186/1471-2105-15-S15-S4>.
16. Mellor J, Grigorias I, Carbonell P, Faulon J-L: **Semisupervised Gaussian process for automated enzyme search.** *ACS Synth Biol* 2016, **5**:518–528, <https://doi.org/10.1021/acssynbio.5b00294>.  
 \*\*
17. Martin S, Roe D, Faulon J-L: **Predicting protein-protein interactions using signature products.** *Bioinformatics* Jan. 2005, **21**:218–226, <https://doi.org/10.1093/bioinformatics/bth483>.
18. Yabuuchi H, et al.: **“Analysis of multiple compound–protein interactions reveals novel bioactive molecules.”** *Mol Syst Biol* Mar. 2011, **7**:472, <https://doi.org/10.1038/msb.2011.5>.
19. Carbonell P, Faulon J-L: **Molecular signatures-based prediction of enzyme promiscuity.** *Bioinformatics* Aug. 2010, **26**: 2012–2019, <https://doi.org/10.1093/bioinformatics/btq317>.
20. Käll L, Canterbury JD, Weston J, Noble WS, MacCoss MJ: **Semi-supervised learning for peptide identification from shotgun proteomics datasets.** *Nat Methods* Nov. 2007, **4**:11, <https://doi.org/10.1038/nmeth1113>.
21. Cohn DA, Ghahramani Z, Jordan MI: **Active learning with statistical models.** *Jair* Mar. 1996, **4**:129–145, <https://doi.org/10.1613/jair.295>.
22. Pertusi DA, Moura ME, Jeffries JG, Prabhu S, Walters Biggs B, Tyo KEJ: **Predicting novel substrates for enzymes with minimal experimental effort with active learning.** *Metab Eng* Nov. 2017, **44**:171–181, <https://doi.org/10.1016/j.ymben.2017.09.016>.  
 \*\*  
 This paper presents the first attempt to use active learning in the context of enzymatic activity prediction.
23. Nielsen J, Keasling JD: **Engineering cellular metabolism.** *Cell* Mar. 2016, **164**:1185–1197, <https://doi.org/10.1016/j.cell.2016.02.004>.
24. Hanrahan G, Lu K: **Application of factorial and response surface methodology in modern experimental design and optimization.** *Crit Rev Anal Chem* Dec. 2006, **36**:141–151, <https://doi.org/10.1080/10408340600969478>.
25. Jervis AJ, et al.: **Machine learning of designed translational control allows predictive pathway optimization in *Escherichia coli*.** *ACS Synth Biol* 2019, **8**:127–136, <https://doi.org/10.1021/acssynbio.8b00398>.
26. Opgenorth P, et al.: **“Lessons from two design–build–test–learn cycles of dodecanol production in *Escherichia coli* aided by machine learning.”** *ACS Synth Biol* Jun. 2019, **8**: 1337–1351, <https://doi.org/10.1021/acssynbio.9b00020>.
27. Hamedirad M, Chao R, Weisberg S, Lian J, Sinha S, Zhao H: **Towards a fully automated algorithm driven platform for biosystems design.** *Nat Commun* Dec. 2019, **10**:5150, <https://doi.org/10.1038/s41467-019-13189-z>.  
 \*\*  
 The paper presents an automated platform coupled with active learning to optimize the production of metabolic pathways.
28. Zhou Y, Li G, Dong J, Xing X, Dai J, Zhang C: **MiYA, an efficient machine-learning workflow in conjunction with the YeastFab assembly strategy for combinatorial optimization of heterologous metabolic pathways in *Saccharomyces cerevisiae*.** *Metab Eng* May 2018, **47**:294–302, <https://doi.org/10.1016/j.ymben.2018.03.020>.
29. Borkowski O, et al.: **Large scale active-learning-guided exploration for in vitro protein production optimization.** *Nat Commun* Apr. 2020, **11**:1872, <https://doi.org/10.1038/s41467-020-15798-5>.  
 This study is making use of artificial neural networks coupled with active learning to boost the productivity of cell-free systems. The optimized system has a productivity substantially higher than those obtained using standard protocols.
30. Silver D, et al.: **Mastering the game of Go with deep neural networks and tree search.** *Nature* Jan. 2016, **529**:7587, <https://doi.org/10.1038/nature16961>.
31. Koch M, Duigou T, Faulon J-L: **Reinforcement learning for bioretrosynthesis.** *ACS Synth Biol* Jan. 2020, **9**:157–168, <https://doi.org/10.1021/acssynbio.9b00447>.  
 \*  
 Reinforcement learning is used in the context of retro-biosynthesis, solutions are scored making use of enzyme availability among other criteria.

32. Wang X, *et al.*: **Towards efficient discovery of green synthetic pathways with Monte Carlo tree search and reinforcement learning.** *Chem Sci* Oct. 2020, **11**:10959–10972, <https://doi.org/10.1039/D0SC04184J>.
33. Pandian BJ, Noel MM: **Control of a bioreactor using a new partially supervised reinforcement learning algorithm.** *J Process Contr* Sep. 2018, **69**:16–29, <https://doi.org/10.1016/j.jprocont.2018.07.013>.
34. Petsagkourakis P, Sandoval IO, Bradford E, Zhang D, del Rio-Chanona EA: **Reinforcement learning for batch bioprocess optimization.** *Comput Chem Eng* Feb. 2020, **133**:106649, <https://doi.org/10.1016/j.compchemeng.2019.106649>.
35. Delépine B, Duigou T, Carbonell P, Faulon J-L: **RetroPath2.0: a retrosynthesis workflow for metabolic engineers.** *Metab Eng* 2018, **45**:158–170, <https://doi.org/10.1016/j.ymben.2017.12.002>.
36. Nielsen AAK, *et al.*: **Genetic circuit design automation.** *Science* Apr. 2016, **352**:6281, <https://doi.org/10.1126/science.aac7341>.
37. Green AA, Silver PA, Collins JJ, Yin P: **Toehold switches: de-novo-designed regulators of gene expression.** *Cell* Nov. 2014, **159**:925–939, <https://doi.org/10.1016/j.cell.2014.10.002>.
38. Bonnet J, Yin P, Ortiz ME, Subsoontorn P, Endy D: **Amplifying genetic logic gates.** *Science* May 2013, **340**:599–603, <https://doi.org/10.1126/science.1232758>.
39. Andrews LB, Nielsen AAK, Voigt CA: **Cellular checkpoint control using programmable sequential logic.** *Science* Sep. 2018, **361**:6408, <https://doi.org/10.1126/science.aap8987>.
40. Farzadfard F, Lu TK: **Synthetic biology. Genomically encoded analog memory with precise in vivo DNA writing in living cell populations.** *Science* Nov. 2014, **346**:1256272, <https://doi.org/10.1126/science.1256272>.
41. Qian L, Winfree E, Bruck J: **Neural network computation with DNA strand displacement cascades.** *Nature* Jul. 2011, **475**:7356, <https://doi.org/10.1038/nature10262>.  
 In this pioneer work the authors make use of DNA strand displacement to encode a Hopfield (recurrent) neural network enabling to distinguish input patterns.
42. Pandi A, *et al.*: **Metabolic perceptrons for neural computing in biological systems.** *Nat Commun* 2019, **10**:3880, <https://doi.org/10.1038/s41467-019-11889-0>.  
 The authors engineer the first metabolic network reproducing the behavior of a perceptron. The network is used to classify samples with different metabolic profiles.
43. Li X, Rizik L, Daniel R: “Synthetic neural-like computing in microbial consortia for pattern recognition,” in *Review, preprint*. Sep. 2020, <https://doi.org/10.21203/rs.3.rs-82365/v1>.  
 This paper presents an implementation of a perceptron *in vivo* using sender and receiver engineered strains.
44. Coley CW, *et al.*: **A robotic platform for flow synthesis of organic compounds informed by AI planning.** *Science* Aug. 2019, **365**:eaax1566, <https://doi.org/10.1126/science.aax1566>.
45. Voyvodic PL, *et al.*: **Plug-and-play metabolic transducers expand the chemical detection space of cell-free biosensors.** *Nat Commun* 2019, **10**:1697, <https://doi.org/10.1038/s41467-019-09722-9>.
46. Tagkopoulos I, Liu YC, Tavazoie S: **Predictive behavior within microbial genetic networks.** *Science* Jun. 2008, **320**:1313–1317, <https://doi.org/10.1126/science.1154456>.



## **Appendix E**

### **Co-authored publication n°2: Cell-Free Biosensors and AI integration**





# Chapter 19

## Cell-Free Biosensors and AI Integration

Paul Soudier, Léon Faure, Manish Kushwaha, and Jean-Loup Faulon

### Abstract

Cell-free biosensors hold a great potential as alternatives for traditional analytical chemistry methods providing low-cost low-resource measurement of specific chemicals. However, their large-scale use is limited by the complexity of their development.

In this chapter, we present a standard methodology based on computer-aided design (CAD) tools that enables fast development of new cell-free biosensors based on target molecule information transduction and reporting through metabolic and genetic layers, respectively. Such systems can then be repurposed to represent complex computational problems, allowing defined multiplex sensing of various inputs and integration of artificial intelligence in synthetic biological systems.

**Key words** Metabolite biosensors, Transcription factors, Machine learning, CAD, Artificial neural networks, Perceptron

---

## 1 Introduction

Metabolite biosensors emerged as a major application of synthetic biology. Repurposing biological systems naturally present in various organisms enabled the development of synthetic sensing devices [1] giving rapid and inexpensive point of care measurement of various molecules of interest levels in a broad range of samples with applications in health [2], environment [3], industries, and fundamental research [4]. The potential of these types of devices lies in the fact that they are able to deliver information on the presence and levels of certain chemicals without the need for expensive reagents, trained operators, and large-size facilities. Transcription factor-based biosensors are a subset of those sensing devices repurposing transcription factor and inducible promoter for the quick detection of molecules of interest.

---

The original version of this chapter was revised. The correction to this chapter is available at [https://doi.org/10.1007/978-1-0716-1998-8\\_26](https://doi.org/10.1007/978-1-0716-1998-8_26)

Ashty S. Karim and Michael C. Jewett (eds.), *Cell-Free Gene Expression: Methods and Protocols*, Methods in Molecular Biology, vol. 2433, [https://doi.org/10.1007/978-1-0716-1998-8\\_19](https://doi.org/10.1007/978-1-0716-1998-8_19), © The Author(s), under exclusive license to Springer Science+Business Media, LLC, part of Springer Nature 2022, corrected publication 2022

Cell-free biosensors emerged as an alternative for traditional whole-cell biosensors, solving several issues carried by these systems [5]. Cell-free biosensors are efficient, generating big signal over noise ratios [6]. They are suitable for the detection of molecules toxic for bacterial growth or molecules that are not able to cross the cell membrane. Their ability to be freeze dried on paper enables long-term storage at room temperature without loss of activity pushing their use as point-of-care devices, especially in low resource communities [7]. The absence of living organisms in it is also facilitating their industrial developments with reduced regulatory issues and low biosafety concerns. Finally, the ability to express each gene from distinct DNA fragments (plasmids or linear fragments) [8] and to fine-tune their expression levels by varying their respective concentrations enable a fast development and optimization of new biosensor candidates or more complex devices relying on them. This central property of straightforward tuning of gene expression is necessary for the development of the complex information processing systems that will be described in this chapter.

The main issue encountered in the development of transcription factor (TF)-based biosensors is the limited number of molecules for which an interacting transcription factor has been described. The list of these metabolites known to trigger transcriptional response, either in natural systems or in the context of synthetic biosensors, has been described in various databases [9, 10] and compiled in a dataset of small molecules triggering transcriptional and translational cellular responses [11].

In order to detect molecules for which associated transcription factors are not available, new methodologies have been developed relying on the use of metabolic pathways. It has indeed been shown that enzymes could be used to convert nondetectable molecules into molecules known to regulate a characterized transcription factor [12]. This framework has been formalized and showcased as the sensing-enabling metabolic pathway (SEMP) concept [13]. This shifted the problem from finding a TF binding the molecule of interest to finding any potential enzymatic route between the molecule of interest and the known set of detectable molecules. This was addressed by repurposing bioinformatic tools traditionally used for retrosynthesis [14] into a platform able to find potential SEMP for any molecule of interest. This web-service called SENSIPATH is a CAD program predicting pathways of up to two steps allowing detection for any query compounds [15]. This approach was then used for the development of a plug and play cell-free workflow where enzymes formalized as metabolic transducers were used as interchangeable modules converting various chemicals into detectable molecules called effectors [6]. Such biosensors were found to be highly efficient, the cell-free system potentialities allowing to reach optimal response for each biosensor by a systematic tuning of each component involved (reporter, transcription factor, and enzyme encoding DNA). They were also

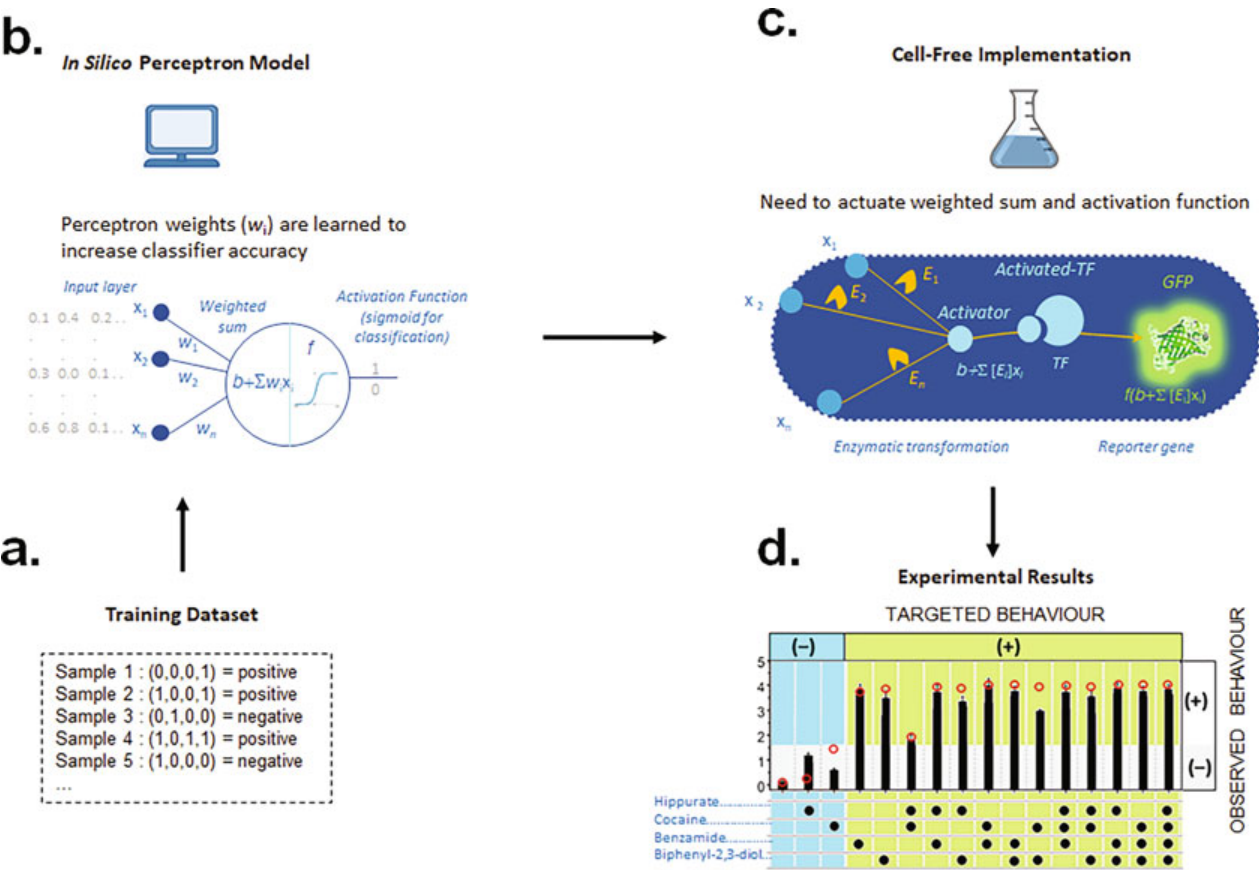
applied for the commercially relevant sensing of molecules of interest in complex real-world samples (preservative in beverage, disease biomarker and drug in clinical samples).

In addition to its biosensing potential, this framework has also then been applied to the field of information processing. Indeed, the constructed devices sensing various molecules can also be conceptualized as information processing systems, converting an input signal (the concentration of detected molecules) into an output one (the level of protein synthesized by the reporter plasmid). This processing device uses the continuous concentration of input metabolites as analog signals able to be processed by the metabolic or the genetic layer featured in the system. Multiple molecules can be continuously converted into the same effector at the same time allowing the construction of multi-input devices derived from analog adders.

Starting from this basis, we have implemented a machine learning architecture that allows integration of AI computations in cell-free systems [16]. One of the simplest architectures for signal processing is the perceptron [17]. A single-layer neural network transforms an input vector into an output value through the application of a weighted sum of the input followed by a thresholding function on the obtained result. Input data can be integrated as categorical or numerical (discrete or continuous). This signal-processing unit has been implemented in a cell-free system to enhance its multiplex sensing abilities [16]. This system is indeed capable of classifying complex samples in two categories (above or below a threshold of the reporter gene expression) based on the recognition of certain patterns of molecules present in those samples. A theoretical perceptron was first constructed and trained “in silico” to compute the optimal set of weights required to solve a defined problem of classification. The perceptron was then implemented in cell-free using a molecular model, giving concentrations of enzymes coding DNA that correspond to the weight of each input (*see* Fig. 1). The thresholding applied through the effector biosensor dose–response curve.

Different elements of the computing unit have a defined terminology: e.g., transducer, adder, actuator. A transducer, in cell-free terms, means one or more enzymes transforming one molecule into another. An adder consists of several reactions having different substrates but yielding the same compound. An actuator is simply a way for the cell-free system to report its activity, through the expression of a reporter gene, for example. Theoretically, many more parameters of a cell-free reaction could be implemented in such signal-processing manner. The diversity of elements can be increased, as well as the number of layers in the perceptron, and the number of detectable molecules.

Here we describe a straightforward methodology that details how to use CAD tools to identify potential new biosensors for a chosen molecule, how to build and test these biosensor candidates, and finally how to repurpose these cell-free biosensors into signal processing devices to implement neural computing in biological systems.



**Fig. 1** Implementation of a clustering problem in a biological system through the metabolic perceptron: to obtain a genetic device operating multiplex sensing on a set of defined clustered samples an in silico model (b) is first trained with the input dataset (a). The results give information on the best set of DNA concentration to implement the problem in a cell free environment (c). The device is evaluated for response to a set of samples with various compositions (d)

This method to this date has been applied for the construction of a single layer 4-input perceptron but is easily scalable for more inputs and adaptable for more complex designs including multi-layer perceptron and other types of computational architectures. An example of potential implementation of a multilayer perceptron using this system can be found in the supplementary figure 14 of the original metabolic perceptron paper [16].

## 2 Materials

### 2.1 Preparation of Cell-Free Extract and Buffer

1. Extract preparation equipment.  
In addition to common lab equipment, materials for the preparation of cell extract include large-volume centrifugation equipment (for 1 L bottles) and a French press.
2. Extract preparation media and buffers.  
Cells are grown in 2YTP media (31 g/L 2xYT, 40 mM potassium phosphate dibasic, 22 mM potassium phosphate monobasic).

S30A and S30B buffers are also required for the preparation of the cell extract.

They are prepared with the following components:

- (a) S30A = (14 mM Mg glutamate, 60 mM K-glutamate, 50 mM Tris, pH 7.7).

S30A is titrated using acetic acid and 2 mM DTT is added just before use.

- (b) S30B = (14 mM Mg glutamate, 60 mM K-glutamate, ~5 mM Tris, pH 8.2).

S30B is titrated using 2 M Tris and 1 mM DTT is added just before use.

### 3. Chemicals for the reaction buffer.

- (a) Solutions of the 20 canonical amino acids, 19 of them concentrated at 168 mM, except leucine that is concentrated at 140 mM.
- (b) Chemicals for the energy solution, including individual solutions of: HEPES pH 8, ATP, GTP CTP, UTP, tRNA, CoA, NAD, cAMP, folinic acid, Spermidine, and 3-PGA, all of them concentrated close to their limit of solubility.
- (c) Additional components for the reaction buffer including K-glutamate solution at 3000 mM, Mg-glutamate solution at 100 mM, PEG8000 solution at 40% w/w, and DTT solution at 1 M.
- (d) Reference plasmid pBEST-OR2-OR1-Pr-UTR1-deGFP-T500 (Addgene #40019) for reaction buffer calibration experiments

## 2.2 Cloning of DNA Parts and Production of Plasmids

Additional materials for the construction and production of plasmid encoding the elements for biosensors include:

1. High-fidelity PCR material (Q5 High-Fidelity 2× Master Mix or equivalent).
2. Template plasmid for backbone amplifications (plasmids #114597 and #114598 from Addgene) and respective primers:

**FWD primer Backbone for CDS:**

**FBC:** cccGGTCTCtGCTTactttatctgagaatagtc.

**REV primer Backbone for CDS:**

**RBC:** cccGGTCTCtCATCatatctcttcttaaagttaaac.

**FWD primer Backbone for Promoter:**

**FBP:**cccGGTCTCtATGCgtaaaggcgaagagctgttc.

**REV primer Backbone for Promoter: RBP:**  
cccGGTCTCtTAAGaatagtaatacaggatccgaatcgtttcag.

3. Thermocycler.
4. TAE buffer, 50×.
5. 1% agarose gel with SYBR™ Safe DNA Gel Stain or equivalent.
6. Monarch® DNA Gel Extraction Kit or equivalent.
7. Thermo Scientific NanoDrop 1000 for DNA concentration determination or equivalent.
8. Thermo Scientific™ Savant™ DNA SpeedVac™ Concentrator Kits to concentrate DNA if required.
9. BsaI enzyme.
10. T4 DNA ligase.
11. T4 DNA ligase buffer.
12. DH5α competent cells.
13. Lysogenic broth liquid medium for cell culture.
14. Lysogenic broth + agar solid medium and ampicillin for cell plating.
15. Monarch® Plasmid Miniprep Kit or equivalent.
16. Macherey-Nagel™ Kits NucleoBond™ Xtra Maxi for plasmid purification.
17. Sequencing plasmids seqF: gataggttaaggaacgg and seqR: ttgatgcctggcaccaac for sequencing verifications of the inserts.

### **2.3 Running Cell-Free Reactions**

1. Plate reader equipment: Typically, Synergy HTX Multi-Mode Reader from Biotek®.
2. 384-well black plate, optically clear polymer bottom.
3. Adhesive PCR plate seals.
4. Analytical-grade chemical powder or solution for any sensed effector and target molecule.

### **2.4 In Silico Materials**

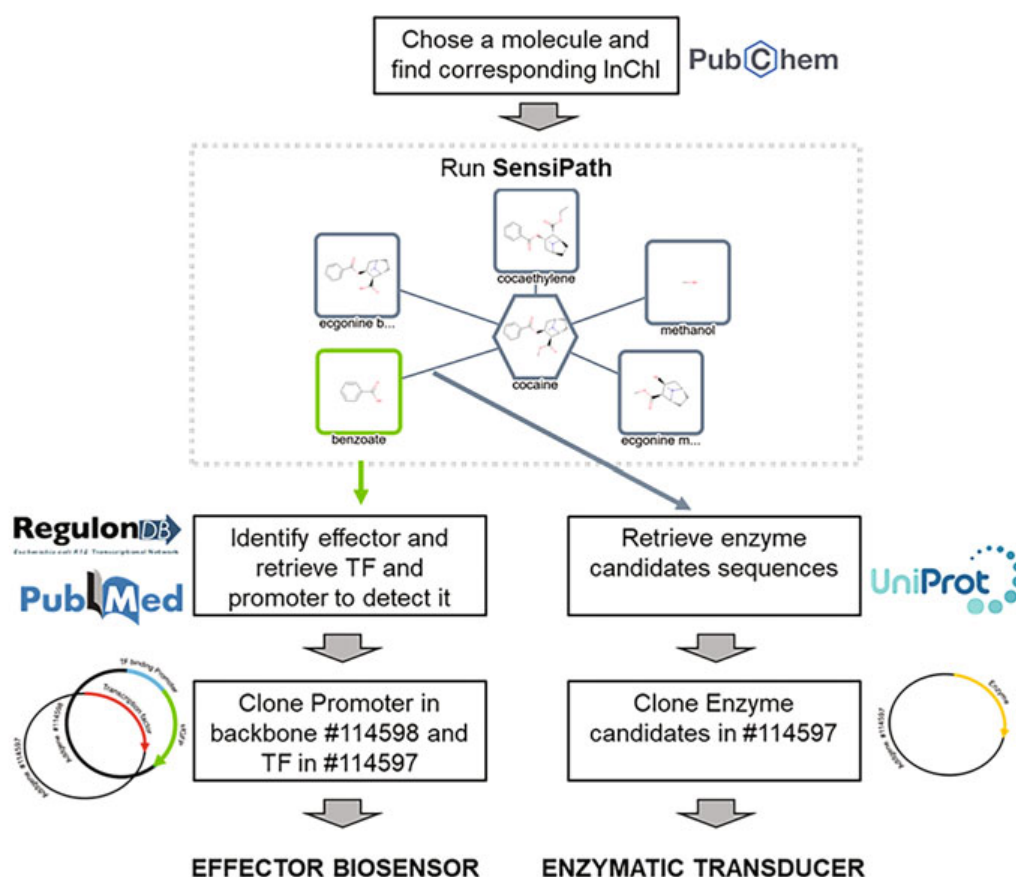
The R programming language (version 3.2.3) was used for fitting experimental data from actuator and transducers, as well as for defining and solving classification tasks. One can perform calculations described in Subheading 3.6. with regular computational power, and computational clusters are not required. Other programming languages can be used, but we recommend to use R in order to take advantage of our git repository containing all files from Pandi et al. work [16]. The repository is freely available here: [https://github.com/brsynth/metabolic\\_perceptrons/tree/master/cell\\_free](https://github.com/brsynth/metabolic_perceptrons/tree/master/cell_free).



### 3 Methods

#### 3.1 Identifying Sensing Routes for a Target Molecule (Fig. 2)

1. Choose a target molecule to sense. Check that the molecule you want to sense is not one of the components of the cell-free buffer you are using (*see Note 1*).
2. Identify the InChi identifier of your target. To do so, you can either search it by name on the database Pubchem (<https://pubchem.ncbi.nlm.nih.gov/>) or search for the InChi in the section Identifier. The other possibility is to go on Pubchem sketcher: (<https://pubchem.ncbi.nlm.nih.gov/edit3/index.html>) and to draw the molecule to retrieve its InChI by replacing SMILES by StdInChI.
3. Go to <http://sensipath.micalis.fr/> to access the online platform. Use the section “Query with a Standard InChI” and paste the InChI of your molecule.
4. Run the tool first for one step pathways. If the results are not satisfying, you can run a second attempt for two step pathways.



**Fig. 2** CAD pipeline for Cell-Free biosensor Design. Main steps of the biosensor design and construction process are presented, from the identification of the chemical identifiers for the target molecule, to the cloning of the DNA parts in plasmids. The sensipath results showed here are the output given for a query of 1 step detection potentialities for cocaine

This can be the case if no pathway to a detectable molecule (in green on the graph view) is identified or if the identified detectable molecule is unsuitable for cell-free biosensors (components of the cell-free buffer) (*see* **Note 2**).

5. Identify the promising effector molecules from the “Pathways view” section.

If the molecule you want to sense appears in green, it means it is directly detectable without need for enzymatic conversion. For developing a biosensor for it you can skip the enzyme-related steps.

6. Download the database of detectable molecules from: [https://github.com/brsynth/detectable\\_metabolites](https://github.com/brsynth/detectable_metabolites) and isolate the lines corresponding to the identified effector. First identify transcription factor that interacts with your component. You can use column E to retrieve the names of it or, if this information is not available, use the column B that contains literature reference of papers describing potential sensing mechanisms for the molecule.
7. Once you find a TF of interest, you have to find the promoters that are potentially regulated by it. To do so, you can search for it in databases such as RegulonDB (<http://regulondb.ccg.unam.mx/>) for *E. coli* and Subtiwiki (<http://subtiwiki.uni-goettingen.de/>) for *B. subtilis*. You can also try a naive bibliographic search for this regulator to identify features linked to it like regulated promoters but also mechanism of action and possible existing design of biosensors using it.
8. Identify the enzymes converting the molecule you want to sense into the effector for which you found a TF. By clicking on the edges from the SENSIPATH graph view, you can retrieve the references associated with the enzymatic reaction of interest. From this point using cross-references between databases or identifiers such as EC number, you can find enzyme candidates in UNIPROT with described catalyzed reactions matching your expectations. You can also use the computational tool Selenzyme (<http://selenzyme.synbiochem.co.uk/>) with SENSIPATH provided references to find potential hits for other enzymes candidates (*see* **Note 3**).
9. Retrieve DNA sequences for the identified parts. TF and enzymes sequences can be codon optimized using any of the available tools. Promoter sequence is often defined as the 200 nucleotides before the start codon of a regulated gene. For troubleshooting purposes, various size promoters can be synthesized to test for their response to the TF and their transcription initiation ability (*see* **Note 4**).



### 3.2 Constructing Candidate Biosensors Plasmids

1. Synthesize the previously isolated sequence of your transcription factor, its regulated promoter, and any enzyme required to convert the molecule you want to detect into the TF binding effector.
2. Design and orders primers for golden gate cloning of the synthesized parts. Genes (TF and enzymes) require overhangs in the format:

FWD: ccGGTCTCtGATG... REV: ccGGTCTCtAAGC...

Promoters require overhangs in the format:

FWD: ccGGTCTCtCTTA... REV: ccGGTCTCtGCAT...

3. Run high fidelity PCRs to amplify with the correct golden gate overhangs the vectors and the synthesized inserts. The reaction typically consists of pipetting 25  $\mu$ L of 2 $\times$  Q5<sup>®</sup> polymerase master mix, 2.5  $\mu$ L of FW and RV primer at 10  $\mu$ M, 1  $\mu$ L of template DNA (at concentration around 100 ng/ $\mu$ L), and 19  $\mu$ L of water.

This mix is then typically incubated in a thermocycler applying the following program: 30 s at 98 °C then 35 cycles with (10 s at 98 °C, 30 s at  $T_m$  +3 °C, 30 s/kb at 72 °C) then 2 min at 72 °C.  $T_m$  beings the lowest melting temperatures of the two primers and kb being the size of the amplicon in kilo bases. Use primer pairs FBP/RBP on the template plasmid 114598 to reamplify the linearized reporter backbone and the primers FBC/RBC on the template plasmid 114597 to reamplify the linearized backbone for enzyme or TF. Use the newly designed primers from **step 2** on the synthesized parts from **step 1** to reamplify the inserts.

4. Run an electrophoresis on the PCR product on a 1% agarose gel stained with SYBR safe using an appropriate DNA ladder to be able to discriminate your amplicon by its size. After an approximate time of 30 min at 100 V, identify and cut the band of the expected size by imaging on a blue-light transilluminator.
5. Recover the DNA from the Gel using a DNA Gel extraction kit, following the kit's instructions to purify your DNA fragment. Determine the titer of purified DNA using a NanoDrop spectrophotometer.
6. Prepare a golden gate reaction to insert each fragment in its respective backbone (*see Note 5*). To do so, you need to calculate the molarity of your DNA. You can use the tool available at <https://nebiocalculator.neb.com/#!/dsdnaamt> using the NanoDrop determined concentrations of each purified DNA fragment. For the golden gate reaction, incubate 100 fmol of insert with 50 fmol of linearized backbone in a tube with 1  $\mu$ L T4 DNA ligase, 1  $\mu$ L BsaI enzyme, 2  $\mu$ L T4

DNA ligase buffer, and water to adjust the volume to 20  $\mu\text{L}$ . Incubate the mix at the 37 °C for 1 h and then 16 °C for 5 min (*see* **Note 6**).

7. Transform DH5 $\alpha$  competent cells with the golden gate reaction product. Incubate 5  $\mu\text{L}$  of golden gate product with 50  $\mu\text{L}$  chemically competent cells at 4 °C for 30 min.
8. Heat-shock at 42 °C for 45 s.
9. Incubate at 4 °C for 3 min.
10. Add 300  $\mu\text{L}$  LB media and incubate at 37 °C for 1 h.
11. Finally spread 100  $\mu\text{L}$  of the final mix on LB agar + ampicillin (100  $\mu\text{L}/\text{mL}$ ) plates and incubate overnight at 37 °C.
12. Small-scale culture for screening:
  - (a) Select 4 colonies per assembly and inoculate them in 3 mL LB medium + ampicillin (100  $\mu\text{L}/\text{mL}$ ) overnight.
  - (b) Purify the plasmids from 2 mL the bacterial cultures using a plasmid miniprep kit and use the primers seqF and seqR to check the integrity of each cassette by Sanger sequencing.
  - (c) Save one correct clone per construction by freezing the rest of the liquid culture at  $-80$  °C after addition of glycerol to final concentration of 25%.
13. Large-scale culture for plasmid production. You will require big quantities ( $>100$   $\mu\text{g}$ ) of each plasmid to run the cell-free reactions for the characterization and optimization of each biosensor candidate (*see* **Note 7**). To do so, you have to realize large-scale culture and plasmid extraction for each construction.
  - (a) Inoculate 300 mL of LB + ampicillin (100  $\mu\text{L}/\text{mL}$ ) from the  $-80$  °C glycerol stock and grow the cells overnight.
  - (b) Pellet the cells by spinning them at  $6000 \times g$  for 15 min at 4 °C.
  - (c) Use the Maxiprep kit to recover plasmid DNA from the pellet.
  - (d) After the last step of your purification, resuspend the precipitated DNA in 200  $\mu\text{L}$  pure water in order to have a final solution at high concentration.
  - (e) Measure the final concentration using a nanodrop and adjust it at 1  $\mu\text{M}$  by either diluting it with water or concentrating it using a SpeedVac machine.

### 3.3 Preparing in House Cell-Free Extract and Buffer

The method briefly described here is adapted from a widely used protocol [18] of 3-PGA powered cell-free mix with minor modifications mostly concerning the lysis method and the starting strain.

3.3.1 *Extract Preparation*

1. Inoculate BL21\* cells from an overnight culture in 4 L of 2YTP medium.
2. Stop the culture at OD 2 and pellet the cells by centrifugation for 12 min at  $5000 \times g$  at 4 °C in 4-L bottles.
3. Rinse the cells twice by successive resuspension/centrifugation steps with 250 mL of S30A buffer.
4. Resuspend the pellets in 40 mL of S30A buffer and transfer the suspension to preweighed 50-mL Falcon tubes.
5. Centrifuge the tubes at  $2000 \times g$  at 4 °C during 8 min, discard the supernatant, and after weighing the pellets freeze them overnight at –80 °C.
6. Thaw the cell-pellet on ice, weigh the pellet, and resuspend them in 1 mL of S30A buffer per gram of cell pellet.
7. Lyse the cells by passing the whole flow once through a French press at 15,000 psi (*see Note 8*).
8. Centrifuge the lysate at  $12,000 \times g$  at 4 °C during 30 min.
9. Incubate the supernatant for 1 h at 37 °C with a 220 rpm shaking before a second centrifugation at  $12,000 \times g$  at 4 °C during 30 min.
10. Transfer the supernatant to a 12–14 kDa MWCO dialysis cassette and incubate the cassette overnight in 2 L S30B buffer at 4 °C (*see Note 9*).
11. After a last centrifugation at  $12,000 \times g$  30 min 4 °C, aliquot the supernatant (500 µL in 1.5-mL tubes) and flash-freeze them in liquid nitrogen before storing at –80 °C.

3.3.2 *Buffer Preparation*

1. Starting from individual solutions of chemicals dissolved in pure water (described in detail in the original paper [18]), prepare an amino acid (at 4× concentration) and an energy solution mix (at 14× concentration) that will be used for buffer preparation. The amino acid mix has to be prepared by mixing all the 20 canonical amino acids at a final concentration of 6 mM except for leucine at 5 mM. Prepare energy solution with HEPES pH 8 700 mM, ATP 21 mM, GTP 21 mM, CTP 12.6 mM, UTP 12.6 mM, tRNA 2.8 mg/mL, CoA 3.64 mM, NAD 4.62 mM, cAMP 10.5 mM, folinic Acid 0.95 mM, Spermidine 14 mM, and 3-PGA 420 mM. Store each mix at –80 °C.
2. Using previously prepared extract, amino acid mix, and the energy solution, you will have to evaluate in this order the best concentration of: (a) Mg-glutamate, (b) K-glutamate, (c) DTT, and (d) PEG8000 to add to your buffer to optimize the protein production of your extract. You will run four consecutive calibration experiments of 8 h cell-free reactions. For each calibration, prepare a master mix for 12 reactions by

mixing 88  $\mu\text{L}$  extract, 66  $\mu\text{L}$  amino acid mix, 18.86  $\mu\text{L}$  energy solution, and 13.2  $\mu\text{L}$  of the Addgene plasmid #40019 concentrated at 200 nM. Add the remaining three components that you are not calibrating for from the following four (Mg-glutamate, K-glutamate, PEG8000, and DTT) at either a starting concentration (if the optimum has not been determined yet) or the optimal concentration determined in a previous step. Finally, add water to this mix to reach a final volume of 237.6  $\mu\text{L}$ . Prepare each calibration reaction by mixing in PCR tubes 19.8  $\mu\text{L}$  of the master mix with 2.2  $\mu\text{L}$  of the tested component concentrated at 20 $\times$ . Then, take 20  $\mu\text{L}$  from each PCR tube and pipette them inside individual wells of a 384-well microplate before incubating the plate for 8 h while measuring GFP signal produced (ex: 458 nm, em: 528 nm).

The concentration that leads to the highest GFP signal at 8 h is identified as the optimal one to be used for future calibrations and run.

The starting concentrations for each component are the following ones: (6 mM for Mg-glu, 80 mM for K-glu, 0 for DTT, and 2% for PEG8000).

The tested concentrations for each component are the following:

Mg-glu: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 mM}.

K-glu: {0, 20, 40, 60, 80, 100, 120, 140, 160, 180 mM}.

DTT: {0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4 mM}.

PEG8000 {0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4%}.

3. Prepare the final buffer mix that will be used for running the cell-free reactions. The following quantities are for 1 tube of 650  $\mu\text{L}$  (you need to scale up to have two tubes of buffer for 1 mL of final extract produced): per tube of buffer, add 128.97  $\mu\text{L}$  of amino acid mix, 110.54  $\mu\text{L}$  of energy solution, PEG8000, K-glu, Mg-glu, and DTT according to the best determined concentration and pure water to adjust the volume to 650  $\mu\text{L}$ .

### **3.4 Running Cell-Free Reactions**

All the cell-free experiments in the following parts should be run according to the same methodology. This method is for the preparation of  $N$  number of 20- $\mu\text{L}$  cell-free reactions. All the following reactions have to be prepared on ice and have to be run in technical triplicate.

1. Thaw your cell-free reagents (extract and buffer) on ice.
2. Predilute your DNA and inducer stock: You will prepare 22- $\mu\text{L}$  reactions each with 7.33  $\mu\text{L}$  of extract, 9.17  $\mu\text{L}$  of buffer, and 5.5  $\mu\text{L}$  of other components (DNA plasmids, inducer(s) and water to adjust). The easiest way to proceed if you have less

than five different components to add per reaction is to make  $20\times$  solutions of the plasmids and inducers you want to use to add 1.1  $\mu\text{L}$  of those to the reaction.

3. Prepare your cell-free master mix for  $(N + 15\%)$  reactions to compensate for the pipetting loss. Add  $(7.33 * (N + 15\%)) \mu\text{L}$  of cell extract and  $(9.17 * (N + 15\%)) \mu\text{L}$  of buffer to a single 1.5-mL tube.
4. Mix by briefly vortexing.
5. Pipette 16.5  $\mu\text{L}$  of that mix in  $N$  PCR tubes (in strips).
6. Add the respective other inputs (plasmids, inducers, water...) from the  $20\times$  stocks, to each PCR tube up to 22  $\mu\text{L}$ .
7. Close these tubes and mix them by briefly vortexing and bench centrifugation.
8. Pipette 20  $\mu\text{L}$  from each tube to a well of a black 384-well plate prechilled.
9. Then cover the plate with a transparent sealing film.
10. Using a plate reader, monitor the green fluorescence (ex: 458 nm, em: 528 nm) at various gains during a 12-h kinetic run.

### **3.5 Cell-Free Biosensors Characterizations and Optimizations**

In order to obtain the best possible biosensor response for our target, we need to first develop an efficient biosensor for the TF binding effector, before optimizing the enzymatic conversion of our target molecule into our effector.

The first step is aimed to screen for any potential response of designed candidate biosensors and to answer two questions: first, is there any interaction between the TF expression and the level of expression of the reporter, and then is this interaction modulated in any way by the effector that we are trying to detect. To answer those, you need to characterize the behavior of the reporter plasmid in a cell-free reaction in presence or absence of transcription factor and in presence or absence of effector. If you have identified multiple candidates for the sensing of the same molecule, this first experiment can also be used to decide what are the TFs and the promoters most promising for the next steps.

1. Run cell-free reactions varying the quantity of added plasmids and chemicals in the following possibilities: Reporter DNA concentration at 0 or 30 nM, TF DNA concentration at 0 or 30 nM, and inducer concentration at 0, 100  $\mu\text{M}$ , or 1 mM.
2. Use end point kinetics results at 8 h to evaluate the potential mechanism of your TF (activation or repression) and a possible response of your system to the inducer.
3. Find the optimal plasmid concentration for the reporter and TF that results in a maximal response for your biosensor. With

three concentrations of inducer (0  $\mu\text{M}$ , 100  $\mu\text{M}$ , and 1 mM), test a combination of concentrations gradients for the added plasmid DNA varying reporter and TF DNA concentration on a logarithmic scale (0, 0.1, 0.3, 1, 3, 10, 30, and 100 nM).

4. Using the previously determined best pair of plasmid concentration, evaluate the inducer dose–response of the constructed sensor. Run a cell-free experiment with inducers concentration at: 0 nM, 0.5 nM, 1 nM, 2 nM, 5 nM, 10 nM, 20 nM, 50 nM, 100 nM, 200 nM, 500 nM, and 1000 nM. Dose–response curve is obtained by plotting (fluorescence at concentration  $\times$  / fluorescence at concentration 0) for each datapoint. Use the final results obtained to assess the performance of the developed sensor and its conformity with your objective of sensitivity (*see* **Note 10**).
5. Once you have a satisfying sensor for your effector, you can start screening for enzymes candidates to convert your target into this effector. The first experiment to do consists of screening for any potential activity on each of the selected enzymes candidates. Prepare a cell-free experiment with the previously identified best concentration for reporter and TF DNA, varying enzyme DNA concentration at 0 nM or 10 nM, and the molecule of interest at 0 mM or 1 mM.
6. Select the best enzyme candidate and optimize the expression of the enzyme on a dose–response curve (same concentrations as in **step 3**) by varying the DNA concentration coding for the enzyme on the scale of 0, 0.1, 0.3, 1, 3, 10, 30, and 100 nM.

### **3.6 Design, Build, and Test a Perceptron**

1. Define the architecture of the perceptron based on input molecules. There are two possibilities depending on the level of constraints you have with the input molecules. If you want to implement a computing device with a defined behavior but without any constraint on the nature of the inputs, you can reprogram the already developed benzoate-based perceptron [16] to implement your desired computational function.
2. If you are interested in multiplexed sensing for defined molecules, you need to identify a common effector they can be converted into. You can use SENSIPATH with a reaction length of 2 to increase the chance to find a product accessible from the two or more substrates. Some central molecules (lactate, acetate, hydrogen peroxide, ammonium...) may be useful for that purpose but a specific attention has to be put on the development of biosensors with high dynamic range for these molecules with potential high-noise issues related to their central positions on the metabolic networks.
3. Create and fit the cell-free model. It can be decomposed in two parts: actuators and transducers.

4. An actuator is modeled with a modified Hill function; commonly used in biochemistry for simulating the binding of ligands to proteins according to the ligand concentration.

$$\text{Actuator}(\text{total}) = \left( \frac{(\text{total})^{\text{hill}_a}}{(K_M)^{\text{hill}_a} + (\text{total})^{\text{hill}_a}} \times \text{fc} + 1 \right) \times \text{basal} + \text{lin} \\ \times 0.0001 \times \text{total}$$

Function description:

- total: Concentration of input metabolite in  $\mu\text{M}$ .
  - $K_M$ : Concentration of input metabolite yielding half of the maximum induction of the system (also called IC50).
  - $\text{hill}_a$ : Hill coefficient characterizing the cooperativity of the induction system.
  - fc: Stands for fold-change, corresponds to the dynamic range (in Arbitrary Units) of the system.
  - basal: Basal GFP fluorescence signal without input, i.e., the background noise of the system.
  - lin: Accounts for the linearity of the system when dealing with concentrations saturating the Hill transfer function.
5. A transducer is also modeled with a particular Hill function:

$$\text{Transducer}(\text{input}) = \text{range}_{\text{enzyme}} \times \left( \frac{(E)^{n_E}}{(K_E)^{n_E} + (E)^{n_E}} \right) \\ \times \left( \frac{(\text{input})^{n_{\text{input}}}}{(K_I)^{n_{\text{input}}} + (\text{input})^{n_{\text{input}}}} \right)$$

Function description:

- input: Input metabolite concentration in  $\mu\text{M}$ .
- range\_enzyme: Coefficient characterizing the capacity of the enzyme to transduce the signal (dimensionless).
- $E$ : enzyme concentration in  $\text{nM}$ .
- $K_E$ : Hill constant for enzyme concentration  $E$ .
- $n_E$ : Hill constant for enzyme concentration  $E$ .
- $K_I$ : Hill constant for input metabolite input.
- $n_{\text{input}}$ : Hill constant for input metabolite input.

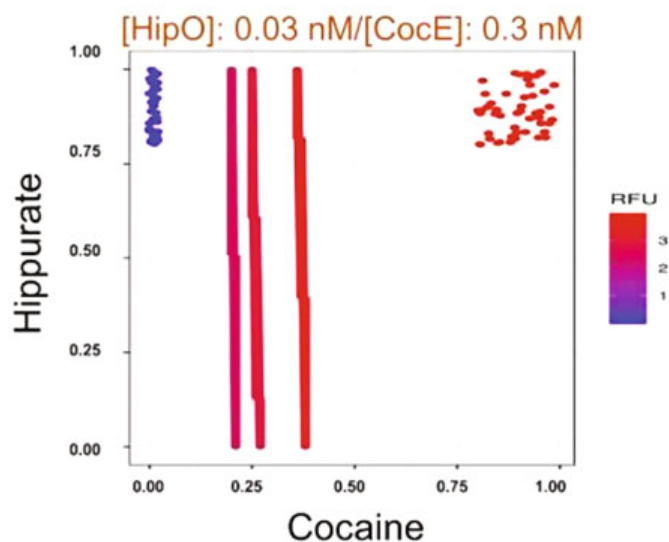
We recommend at least some inspiration from our fitting process described in the notes section (*see Note 11*) as it puts an emphasis on tackling the loss of signal of the actuator when the whole system is modeled then implemented. Otherwise, one can adapt another fitting process for a particular project's needs.

6. Measure the model's performance with different metrics: root mean square deviation (RMSD);  $R^2$ ; weighted  $R^2$ ; error percentage.

Ensuring high metrics (e.g., above 0.9 for scores between 0 and 1) on many experimental data points guarantee a robust model for predicting the weights schemes for each classification task.

7. Define your classification problem. Start by defining a set of tasks (here, classifications), which outputs either 0 (OFF) or 1 (ON); as well as a set of weights to be tested (equivalent to a range of possible enzyme concentrations in the cell-free experiment, for our work it was between 0.1 and 10 nM).
8. Continue by sampling uniformly input values for your problem, that is to be resolved by the perceptron. For example, in the case of a binary classification of solutions composed of hippurate and cocaine, sample points in a given range, either for a "low" concentration or a "high" concentration for each of the compounds. Here, let us assume we can sample between 0 and 2  $\mu\text{M}$  for low concentrations and between 80 and 100  $\mu\text{M}$  for high concentrations.
9. For each of the two clusters, choose to sample either in the low or high range for each compound.
10. Then, two clusters have been produced and a binary classifier can be easily defined on these points: the perceptron set of weights and its corresponding fluorescence threshold. Please find a visual example of the definition of a classification task on Fig. 3. Further is detailed how to find the set of weights and the fluorescence "decision threshold."
11. Predict the best set of weights for solving this problem (aka "train" the perceptron on a classification task). Using the previously fitted (part 2) and benchmarked (part 3) model, simulate all possible input combinations with all possible sets of weights.
12. Screen for performant set of weights, i.e., those enabling a sharp threshold between the output states of the system ("ON" or "OFF"). Several thresholds can be tested for considering an output value as ON or OFF, for all possible simulations.
13. Manually select the best set of weights and corresponding threshold, i.e., those that show the highest and clearest difference between ON and OFF behaviors, and in most scenarios. Also, prefer those showing low enzyme concentrations (to avoid resource competition). One can also test several possibilities (several sets of weights and/or thresholds) in the following cell-free experiments.





**Fig. 3** Visual example of a binary clustering with a metabolic perceptron. This figure has been copied from supplementary material of Pandi et al. [16] with authorization from the authors. The X-axis shows the normalized concentration of cocaine in a sample, and the Y-axis shows the normalized concentration of Hippiurate in a sample. Here, the blue points drop into the first cluster (“high Hippiurate, low cocaine”) and the red ones drop into the second cluster (“high cocaine, high hippurate”). Colors of the points correspond to the predicted RFU values by the fitted perceptron (after Subheading 3.6, part 3 is done). The three vertical lines correspond to three isofluorescence lines; equivalent to three fluorescence thresholds, each of them being associated with one set of weights of the perceptron (i.e., [HipO]: 0.03 nM and [CocE]: 0.3 nM)

14. Implement the designs predicted as best ones, to test it in a cell-free reaction. Start by drawing a test set of chemical (input) combinations from the compositions used for the training of the in silico perceptron. Be cautious to evaluate enough points from the space of possibilities to capture the behavior of your system. If your perceptron is designed to solve the particular problem of binary inputs sample classifications, you can eventually build a complete test set with all the possible combinations of inputs (*see* Fig. 1d).
15. The designed test set can be prepared at  $20\times$  concentration in PCR tube strips to be evaluated on various perceptron implementations. We advise you to prepare a single master mix with the cell-extract, the buffer, and the DNA coding for the reporter system and the various transducers at the desired concentration for each implementation of the perceptron that you designed.
16. Use this master mix to evaluate the response of the system on the chemical test set previously designed. If the perceptron does not have the expected behavior from in silico analysis, try another set of weights.

---

## 4 Notes

1. Most of the molecules present in the cell-free buffer are in high concentration masking any potential response for a biosensor designed to detect them. This list includes all the 20 amino acids, HEPES, ATP, GTP, UTP, CTP, tRNA, CoA, NAD, cAMP, folinate, spermidine, 3-PGA, magnesium, potassium, and DTT.
2. If you cannot find satisfying results with the software SENSIPATH, you can try to find other SEMP using the retrosynthesis workflow Retropath [14]. It has the advantage to allow prediction for pathways with more than two steps or pathways using promiscuous activity of enzymes to find potential new reactions. To run Retropath for biosensor design, use the list of detectable molecules [11] as Sink, the molecule you want to detect as source and the reaction rules in the forward direction. Results coming from this workflow have to be taken with more care as it relies on less-reliable predictions.
3. The Selenzyme tool is predicting potential enzymes catalyzing a defined reaction based on similarity of sequences, reactions or other features existing between enzymes from a well-annotated database and enzymes potentially catalyzing the query reaction. The predictions should be manually checked in published literature/datasets to evaluate if the identified enzyme can likely catalyze the given reaction.
4. The methodology described here is a standard to be used in the case where the identified promoter has limited annotation or features described. You are strongly encouraged to search for existing biosensing projects described in the literature that uses the same transcription factor with defined size promoters or synthetic ones (built by inserting TF-binding sequences in another promoter) as these promoters may show a better response than the natural ones.
5. Golden gate assembly was chosen over other methods like Gibson assembly as it allows reusing the same primers to reamplify backbone for every new insert cloned.
6. This thermocycler protocol for golden gate is a variation of the fast golden gate assembly protocol (1 h 37 °C, 10 min 55 °C) that is adapted to backbone plasmid containing internal *bsaI* cut sites. Removing the last 55 °C step and adding one at 16 °C for 5 min avoid cutting the final assembly containing the *BsaI* site.
7. DNA batch can have an influence on its expression level in cell-free limiting reproducibility of results from one maxiprep to another. You are advised to purify each plasmid in a big enough

quantity for running all your experiment with a single batch. If necessary, run multiple maxiprep in parallel and mix the resulting DNA to have a sufficiently large quantity of plasmid.

8. Sonication and autolysis have also been successfully tested as lysis methods on this protocol for the development of cell-free biosensors with good results. The French press method has the advantage of being easily scalable for the production of large quantity of extract but if you lack the equipment for it you can adapt the protocol to any other lysis method.
9. In our experience, the dialysis does not show a major impact on behavior of the final extract. For optimization purposes, dialyzed and nondialyzed extract can be screened to find the condition giving the best response for specific biosensors.
10. If you plan to use this effector biosensor for indirect detection (using metabolic transducers) or for multiplex sensing (through a perceptron like architecture), you may want to have it optimized for the detection of inducers present at a lower concentration than what you need for your target molecule as the transduction of the signal through the enzymatic layer often goes with a decrease of sensitivity and an increase of potential noise.
11. Once these functions are encoded in R, the actual fitting can happen.

First, fit the actuator experimental data (effector dose-response curve obtained in part 3.5 Subheading 3) to the Hill function model.

To do so, fit 100 times the actuator experimental values to the actuator model. Let all parameters be able to vary. Use ordinary least-square error or the R “optim” function (that uses the Limited-memory Broyden Fletcher Goldfarb Shanno algorithm) as the objective function for the data fitting. Keep the seed of the fitting process in memory so it ensures reproducibility. Retrieve fitted parameters for each fitting (a population of 100 sets of fitted parameters will be produced). From this population, save the mean, standard deviation, standard error, and confidence interval; for each parameter.

To account for the decrease of signal in experimental actuator data when the whole system is implemented in cell-free, one needs to fit the transducer model in a specific way.

To fit the transducers models, we need to actually fit the whole model to the whole system’s cell-free experimental data.

In order to have a coherent perceptron, we will actually constrain the actuator’s parameters, then fit all transducers *and* the actuator together.

To do so, start by constraining each of the actuator's previously fitted parameters with bounds corresponding to the 95% confidence interval; or  $\pm$  one standard deviation from the mean, in the case of following parameters: the fold-change, termed "fc," and the baseline, termed "basal."

We call "pseudo-experimental" the data used to fit the whole system, but that originates from specific components' experimental data. These individual data were aggregated as described in the whole-system model: combine transducers data by simply adding their output and feeding the sum to the actuator. Note that in the cell-free experimental system, the same process happens, e.g., benzoate concentrations are added as a result of all transducers yielding benzoate, and the actuator takes this aggregated benzoate concentration as an input. As a result, we obtain "pseudo-experimental" data for the whole cell-free system, with experimental data only available for each component separately.

Once your whole system is modeled and you have constrained your actuator's parameters, you can fit your transducer parameters. Initialize the whole model by drawing actuator parameters values according to a Gaussian distribution centered on the mean of the parameter estimations, with a standard deviation equal to the standard error of this parameter estimation. Let all transducer parameters vary to fit the whole system to the previously described "pseudo-experimental" data.

---

## Acknowledgments

PS and JLF are supported by the ANR SynBioDiag grant number ANR-18-CE33-0015. LF is supported by the French National Research Institute for Agriculture, Food, and Environment (INRAE), through the "Métaprogramme BIOLPREDICT". MK acknowledges funding support from Ile-de-France region's DIM-RFSI, INRAE's MICA department and the University of Paris-Saclay.

## References

1. van der Meer JR, Belkin S (2010) Where microbiology meets microengineering: design and applications of reporter bacteria. *Nat Rev Microbiol* 8:511–522. <https://doi.org/10.1038/nrmicro2392>
2. Chang H-J, Voyvodic PL, Zúñiga A, Bonnet J (2017) Microbially derived biosensors for diagnosis, monitoring and epidemiology. *Microb Biotechnol* 10:1031–1035. <https://doi.org/10.1111/1751-7915.12791>
3. Rodriguez-Mozaz S, de Alda MJL, Marco M-P, Barceló D (2005) Biosensors for environmental monitoring: a global perspective. *Talanta* 65:291–297. <https://doi.org/10.1016/j.talanta.2004.07.006>

4. Koch M, Pandi A, Borkowski O et al (2019) Custom-made transcriptional biosensors for metabolic engineering. *Curr Opin Biotechnol* 59:78–84. <https://doi.org/10.1016/j.copbio.2019.02.016>
5. Zhang L, Guo W, Lu Y (2020) Advances in cell-free biosensors: principle, mechanism, and applications. *Biotechnol J* 15:2000187. <https://doi.org/10.1002/biot.202000187>
6. Voyvodic PL, Pandi A, Koch M et al (2019) Plug-and-play metabolic transducers expand the chemical detection space of cell-free biosensors. *Nat Commun* 10:1697. <https://doi.org/10.1038/s41467-019-09722-9>
7. Pardee K (2018) Perspective: solidifying the impact of cell-free synthetic biology through lyophilization. *Biochem Eng J* 138:91–97. <https://doi.org/10.1016/j.bej.2018.07.008>
8. Sun ZZ, Yeung E, Hayes CA et al (2014) Linear DNA for rapid prototyping of synthetic biological circuits in an Escherichia coli based TX-TL cell-free system. *ACS Synth Biol* 3:387–397. <https://doi.org/10.1021/sb400131a>
9. Huerta AM, Salgado H, Thieffry D, Collado-Vides J (1998) RegulonDB: a database on transcriptional regulation in Escherichia coli. *Nucleic Acids Res* 26:55–59. <https://doi.org/10.1093/nar/26.1.55>
10. Cipriano MJ, Novichkov PN, Kazakov AE et al (2013) RegTransBase—a database of regulatory sequences and interactions based on literature: a resource for investigating transcriptional regulation in prokaryotes. *BMC Genomics* 14:213. <https://doi.org/10.1186/1471-2164-14-213>
11. Koch M, Pandi A, Delépine B, Faulon J-L (2018) A dataset of small molecules triggering transcriptional and translational cellular responses. *Data Brief* 17:1374–1378. <https://doi.org/10.1016/j.dib.2018.02.061>
12. Xue H, Shi H, Yu Z et al (2014) Design, construction, and characterization of a set of biosensors for aromatic compounds. *ACS Synth Biol* 3:1011–1014. <https://doi.org/10.1021/sb500023f>
13. Libis V, Delépine B, Faulon J-L (2016) Expanding biosensing abilities through computer-aided design of metabolic pathways. *ACS Synth Biol* 5:1076–1085. <https://doi.org/10.1021/acssynbio.5b00225>
14. Delépine B, Duigou T, Carbonell P, Faulon J-L (2018) RetroPath2.0: a retrosynthesis workflow for metabolic engineers. *Metab Eng* 45:158–170. <https://doi.org/10.1016/j.ymben.2017.12.002>
15. Delépine B, Libis V, Carbonell P, Faulon J-L (2016) SensiPath: computer-aided design of sensing-enabling metabolic pathways. *Nucleic Acids Res* 44:W226–W231. <https://doi.org/10.1093/nar/gkw305>
16. Pandi A, Koch M, Voyvodic PL et al (2019) Metabolic perceptrons for neural computing in biological systems. *Nat Commun* 10:3880. <https://doi.org/10.1038/s41467-019-11889-0>
17. Rosenblatt F (1957) The perceptron, a perceiving and recognizing automaton project para. Cornell Aeronautical Laboratory
18. Sun ZZ, Hayes CA, Shin J et al (2013) Protocols for implementing an Escherichia coli based TX-TL cell-free expression system for synthetic biology. *JoVE J Vis Exp*:e50762. <https://doi.org/10.3791/50762>

## **Appendix F**

**Co-authored publication n°3: A versatile active learning workflow for optimization of genetic and metabolic networks**




ARTICLE



<https://doi.org/10.1038/s41467-022-31245-z>

OPEN

# A versatile active learning workflow for optimization of genetic and metabolic networks

Amir Pandi <sup>1,9</sup>✉, Christoph Diehl<sup>1,9</sup>, Ali Yazdizadeh Kharrazi<sup>2</sup>, Scott A. Scholz <sup>1</sup>, Elizaveta Bobkova<sup>1</sup>, Léon Faure<sup>3</sup>, Maren Nattermann<sup>1</sup>, David Adam<sup>1</sup>, Nils Chapin<sup>1</sup>, Yeganeh Foroughijabbari<sup>1</sup>, Charles Moritz<sup>1</sup>, Nicole Paczia<sup>4</sup>, Niña Socorro Cortina<sup>1,5</sup>, Jean-Loup Faulon<sup>3,6,7</sup> & Tobias J. Erb <sup>1,8</sup>✉

Optimization of biological networks is often limited by wet lab labor and cost, and the lack of convenient computational tools. Here, we describe METIS, a versatile active machine learning workflow with a simple online interface for the data-driven optimization of biological targets with minimal experiments. We demonstrate our workflow for various applications, including cell-free transcription and translation, genetic circuits, and a 27-variable synthetic CO<sub>2</sub>-fixation cycle (CETCH cycle), improving these systems between one and two orders of magnitude. For the CETCH cycle, we explore 10<sup>25</sup> conditions with only 1,000 experiments to yield the most efficient CO<sub>2</sub>-fixation cascade described to date. Beyond optimization, our workflow also quantifies the relative importance of individual factors to the performance of a system identifying unknown interactions and bottlenecks. Overall, our workflow opens the way for convenient optimization and prototyping of genetic and metabolic networks with customizable adjustments according to user experience, experimental setup, and laboratory facilities.

<sup>1</sup>Department of Biochemistry & Synthetic Metabolism, Max Planck Institute for Terrestrial Microbiology, Marburg, Germany. <sup>2</sup>DataChef, Amsterdam, The Netherlands. <sup>3</sup>Micalis Institute, INRAE, AgroParisTech, University of Paris-Saclay, Jouy-en-Josas, France. <sup>4</sup>Core Facility for Metabolomics and Small Molecule Mass Spectrometry, Max Planck Institute for Terrestrial Microbiology, Marburg, Germany. <sup>5</sup>LiVeritas Biosciences, Inc., 432N Canal St.; Ste. 20, South San Francisco, CA 94080, USA. <sup>6</sup>Genomique Metabolique, Genoscope, Institut Francois Jacob, CEA, CNRS, Univ Evry, University of Paris-Saclay, Evry, France. <sup>7</sup>Manchester Institute of Biotechnology, SYNBIOCHEM center, School of Chemistry, The University of Manchester, Manchester, UK. <sup>8</sup>SYNMIKRO Center of Synthetic Microbiology, Marburg, Germany. <sup>9</sup>These authors contributed equally: Amir Pandi, Christoph Diehl. ✉email: [amir.pandi@mpi-marburg.mpg.de](mailto:amir.pandi@mpi-marburg.mpg.de); [toerb@mpi-marburg.mpg.de](mailto:toerb@mpi-marburg.mpg.de)



The understanding and engineering of biological systems require practical and efficient experimental and computational approaches<sup>1–5</sup>. Machine learning algorithms hold a big promise for the study, design, and optimization of different biological systems<sup>6–9</sup>, including genomics studies<sup>10–12</sup>, protein, enzyme and metabolic engineering<sup>4,13,14</sup>, prediction and optimization of CRISPR sequences and proteins<sup>15–18</sup>, as well as complex genetic circuits design and optimization<sup>19–21</sup>. Yet, applying machine learning is limited by the need for informatics expertise and large user-labeled datasets, which are typically time-, labor- and cost-intensive.

Active learning, sometimes called optimal experimental design<sup>22,23</sup>, is a type of machine learning that interactively suggests a next set of experiments after being trained on previous results<sup>24</sup>. This makes active learning valuable for wet-lab scientists, especially when dealing with a limited number of user-labeled data<sup>25</sup>. Active learning approaches reduce experimental time, labor and cost and have been used in cellular imaging<sup>26</sup>, systems biology<sup>27</sup>, biochemistry<sup>28–30</sup>, and synthetic biology<sup>31</sup>. Despite these examples, a challenge in applying active learning methods for experimental biologists is the lack of customizable programs and workflows.

Here, aimed at democratization and standardization, we describe METIS (Machine-learning guided Experimental Trials for Improvement of Systems, named after the ancient goddess of wisdom and crafts Μῆτις, *lit.* “wise counsel”), a modular and versatile active machine learning workflow for data-driven optimization of a biological objective function (an output/target that depends on multiple factors) with minimal datasets. Note that, active learning for optimizing a system is also known as Bayesian optimization. We created METIS for experimentalists with no experience in programming, who can use the entire process of personalized active learning, experimental setup, data analysis and visualization without any advanced computational skills. METIS runs on Google Colab, a free online platform to write and execute Python codes developed for education, data science, and machine learning purposes<sup>32</sup>. The open platform does not need any installation/registration and local computational power and can be simply used via a personal copy of the respective notebook.

To establish the workflow, we first assessed the performance of different machine learning algorithms on a minimal training dataset and experimentally validated the best performing algorithm (XGBoost) by optimization of an *in vitro* cell-free transcription-translation (TXTL) system of *Escherichia coli* that is commonly used in cell-free synthetic biology for a variety of applications<sup>33</sup>, including biosensor development<sup>34</sup>, metabolic pathway prototyping<sup>35</sup>, and gene circuit design<sup>36</sup>. We then developed the modular architecture of METIS for user-defined applications through the customization of different parameters and factors.

We showcase the versatility of METIS on various biological systems, starting with an *in vitro* gene circuit. Cell-free gene circuits have recently received attention (e.g., as biosensors), but are still limited in their applicability due to their poor performance<sup>34,37</sup>. Applying our workflow, we could improve the activity of a recently reported *LacI*-based multi-level controller<sup>38</sup> by two orders of magnitude, notably by identifying and overcoming a fundamental bottleneck (i.e., resource competition) in the design of the system. We further demonstrate ten-fold improved protein production from an optimized transcription & translation unit, demonstrating that our workflow can be used for biological sequences based on categorical factors (i.e., combinatorial variants of a T7 promoter, ribosome binding site (RBS), N- and C-terminal amino acids). Finally, we use METIS to improve a complex metabolic network, the so-called crotonyl-CoA/ethylmalonyl-CoA/hydroxybutyryl-CoA (CETCH)<sup>39</sup> cycle, a new-to-nature synthetic

CO<sub>2</sub>-fixation cycle, comprising 17 different enzymes plus 10 different cofactors and components, which was shown to be (thermodynamically) more efficient compared to natural photosynthesis. Yet, the network's full kinetic potential had not been exploited, as efficient strategies to explore its combinatorial space had been lacking so far. Using METIS allowed us to improve productivity of the CETCH cycle by ten-fold with (only) 1,000 experiments, resulting in the most efficient CO<sub>2</sub>-fixing *in vitro* system described to date. Overall, these results demonstrate the ability of our workflow for the optimization of various complex biological networks with minimal experimental efforts, providing multiple opportunities for the study and engineering of different biological systems in the future.

## Results

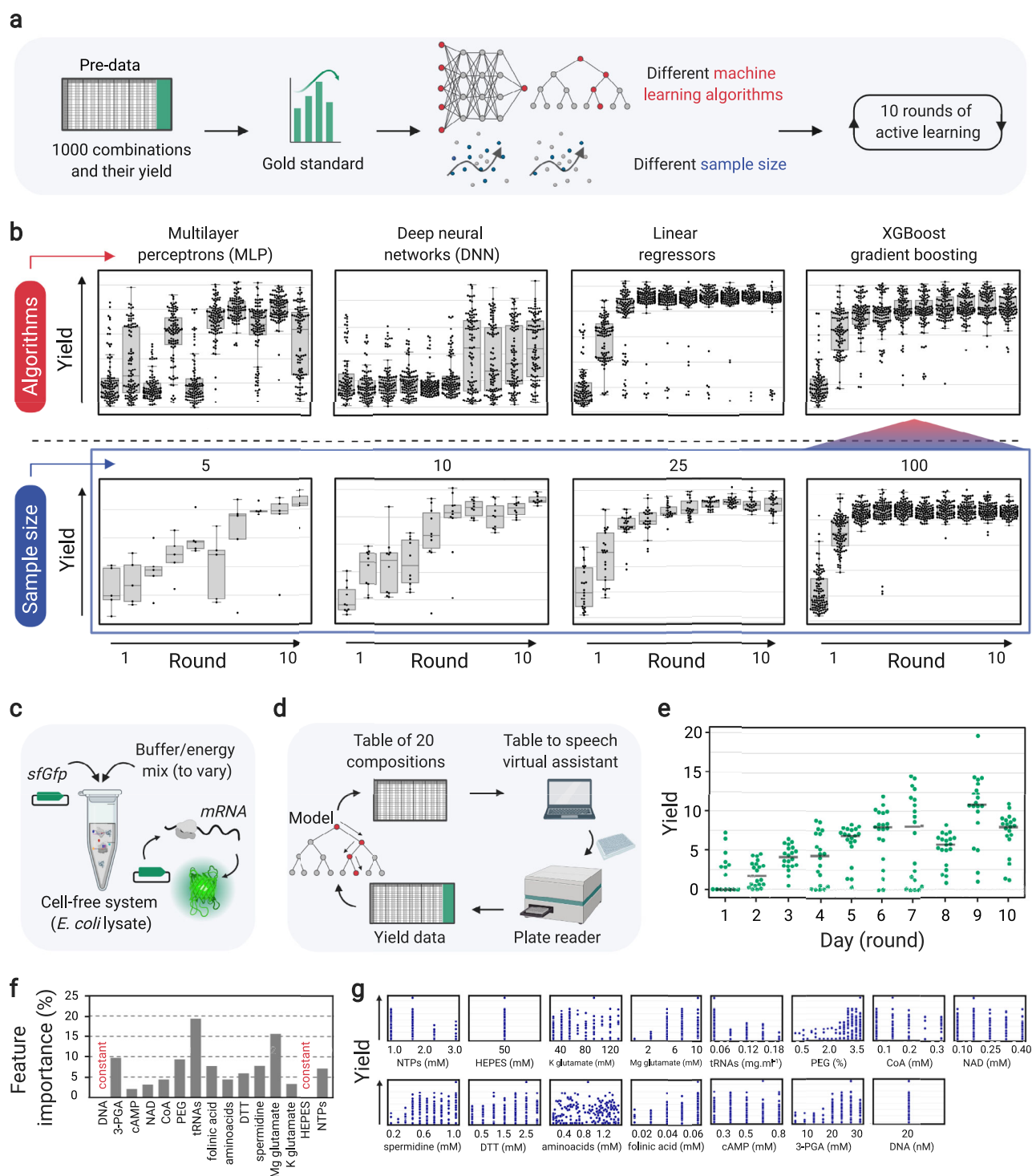
**Assessing the performance of different algorithms for our workflow.** We first tested which machine learning algorithm would perform best with a limited number of experimental data typical for a standard research lab setup. To that end, we took advantage of an existing dataset from a recent optimization of an *E. coli* extract-based *in vitro* TXTL system<sup>31</sup>. In their study, Borkowski *et al.* optimized cell-free protein production in *E. coli* lysate by varying 12 different factors including salts, energy mix, amino acids, and tRNAs, and measuring production yield of Gfp (produced by a plasmid expressing *Gfp*) as output. Altogether, the dataset encompassed around 1000 data points. We fitted the dataset to obtain a standard as a gold regressor (a reference model fitted on pre-existing experimental data to evaluate new algorithms) and divided it further into test and training sets, with 20% and 80% of data, respectively. While the latter set was used to train the model, the test set was used to validate the gold regressor (Methods).

We used the gold regressor to assess the performance of four different machine learning algorithms over 10 rounds of active learning (Fig. 1a). The tested algorithms included deep neural networks (DNN), multilayer perceptrons (MLP), linear regressors, and XGBoost gradient boosting, which all show different capabilities for a given problem set and its data sample size. Over 10 rounds of active learning with 100 data points in each round, XGBoost and linear regressors showed better performance (Fig. 1b) compared to DNN and MLP, which generally need larger datasets for training<sup>40</sup>. XGBoost outperformed linear regressors when fewer data points per round were used (Supplementary Fig. 1).

XGBoost is an improved random forest-type algorithm, working through gradient boosted decision trees<sup>41</sup> by aggregating and compiling sets of models. This algorithm is a sparsity-aware, fast, scalable as well as versatile model for handling tabular data with complex non-linear interactions<sup>41</sup>. These features make XGBoost a promising algorithm for machine learning applications on different biological systems with limited datasets. For our workflow, we therefore selected XGBoost, which has also been used for different biological applications previously<sup>18,42,43</sup>. To determine the minimum dataset required for optimization, we compared active learning rounds with 5, 10, 25, and 100 data points. Notably, a sample size as low as 10 data points still allowed sufficient yield optimization (i.e., in the scale of the original study<sup>31</sup>) within 10 learning cycles (Fig. 1b).

**Testing the workflow with minimal experimental work.** Having validated the workflow with an existing data set, we next sought to test it in a real-world experimental setup, simulating a situation in which the number of combinations that can be tested is limited by available equipment, readout and experimental cost. We chose (again) to optimize relative Gfp production (yield) in an *E. coli*





lysate TXTL system (Fig. 1c) that consists of 13 variable factors (components).

To optimize composition of the TXTL system, we defined a concentration range for each of the 13 factors (Code availability), and performed an active learning process over 10 rounds with only 20 experiments per round (Fig. 1d, see Supplementary Note 1 for details) quantifying Gfp yield (i.e., Gfp fluorescence reported from each composition normalized by the Gfp fluorescence of the standard composition<sup>33</sup>), as objective function. Over 10 rounds of active learning, the relative yield increased up to 20 and the median increased from zero to over 10 in the 9th round (Fig. 1e, see also Supplementary Fig. 2). Note

that low-yield data points (even those observed in the late learning cycles) are equally informative as high-yield ones, because they allow to explore the landscape around and beyond local maxima, as defined by the exploration to exploitation ratio of our workflow that we fully discuss in Supplementary Note 2.

Beyond the simple optimization of a given system, our workflow can also quantify the contribution of different factors during optimization. Figure 1f represents feature importance, i.e., the effect of each individual factor on the objective function. The importance is given as a relative fraction (or percentage) in the prediction of the values of the objective function by the model, with the sum of all factors set to 100%. Our analysis showed that

**Fig. 1 Assessing the performance of different algorithms and testing the active learning workflow with minimal data points.** **a** An existing dataset of cell-free gene expression compositions composed of 1000 data points was used to build a gold standard regressor and assess the performance of different machine learning algorithms in 10 rounds of active learning. **b** Top panel: performance of 4 algorithms, multilayer perceptrons (MLP), deep neural networks (DNN), linear regressors, and XGBoost gradient boosting in 10 rounds of active learning (100 data points per round). Bottom panel: performance of the XGBoost gradient boosting algorithm as the selected algorithm with different sample sizes. The boxplots with whisker length of 1.5, represent the minimum, 25th percentile (bottom bound of box), median (center of box), 75th percentile (upper bound of box), and maximum. **c** An in vitro or cell-free transcription-translation (TXTL) system (based on *E. coli* lysate) to test the workflow with 20 data points per round. A plasmid expressing *sfGfp* was added to TXTL reaction mix along with 13 components of reaction buffer and energy mix. **d** Overview of the active learning cycle. 13 components are varied starting with random compositions and over 10 rounds of results are imported to the model, which learns and suggests new compositions for improvement of the objective function. **e** The plot presenting the average of triplicates ( $n = 3$  independent experiments) of the objective function (yield) for compositions in 10 rounds (days) of active learning. The gray lines show the median. **f** Feature importance percentages show the effect of each factor on the model's decision to calculate yields for the suggested compositions. **g** Distribution of different concentrations of each factor within the measured yields. The Google Colab Python notebook and all active learning data (combinations and yields) in this figure are available at <https://github.com/amirpandi/METIS>.

tRNA mix and Mg-glutamate were the most important components in optimizing Gfp yield, while cAMP and NAD were the least important contributors. Figure 1g shows the distribution of Gfp yield at different concentrations of individual factors (see also Supplementary Fig. 3). Decreasing concentrations of tRNA and NTP mixes correlated with high yield, while PEG 8000, Mg-glutamate, 3-PGA, folinic acid, and spermidine showed similar effects at increasing concentrations. Together, these data did not only result in an optimized TXTL system but also allowed to identify the most crucial components during system optimization, providing the basis for a deeper understanding of the system itself. All combinations and yields are provided as results files for each experimental round (**Data availability**), and the Google Colab notebook with all analyses and visualization modules are also accessible (**Code availability**).

**Development of METIS, a user-friendly, versatile modular workflow.** After demonstrating that our workflow is capable of working efficiently with minimal datasets, we sought to build METIS, a modular architecture that can be easily applied for the optimization of different biological objective functions. We implemented our workflow in Google Colab Python notebooks that can be accessed by the user—without installation or registration—simply through a personal copy of the notebook from a web browser. Defining the objective function and the variable factors (Fig. 2a), the user can simply open the link of Google Colab notebook and directly use the workflow as shown in Fig. 2a, b, Supplementary Figs. 4–6.

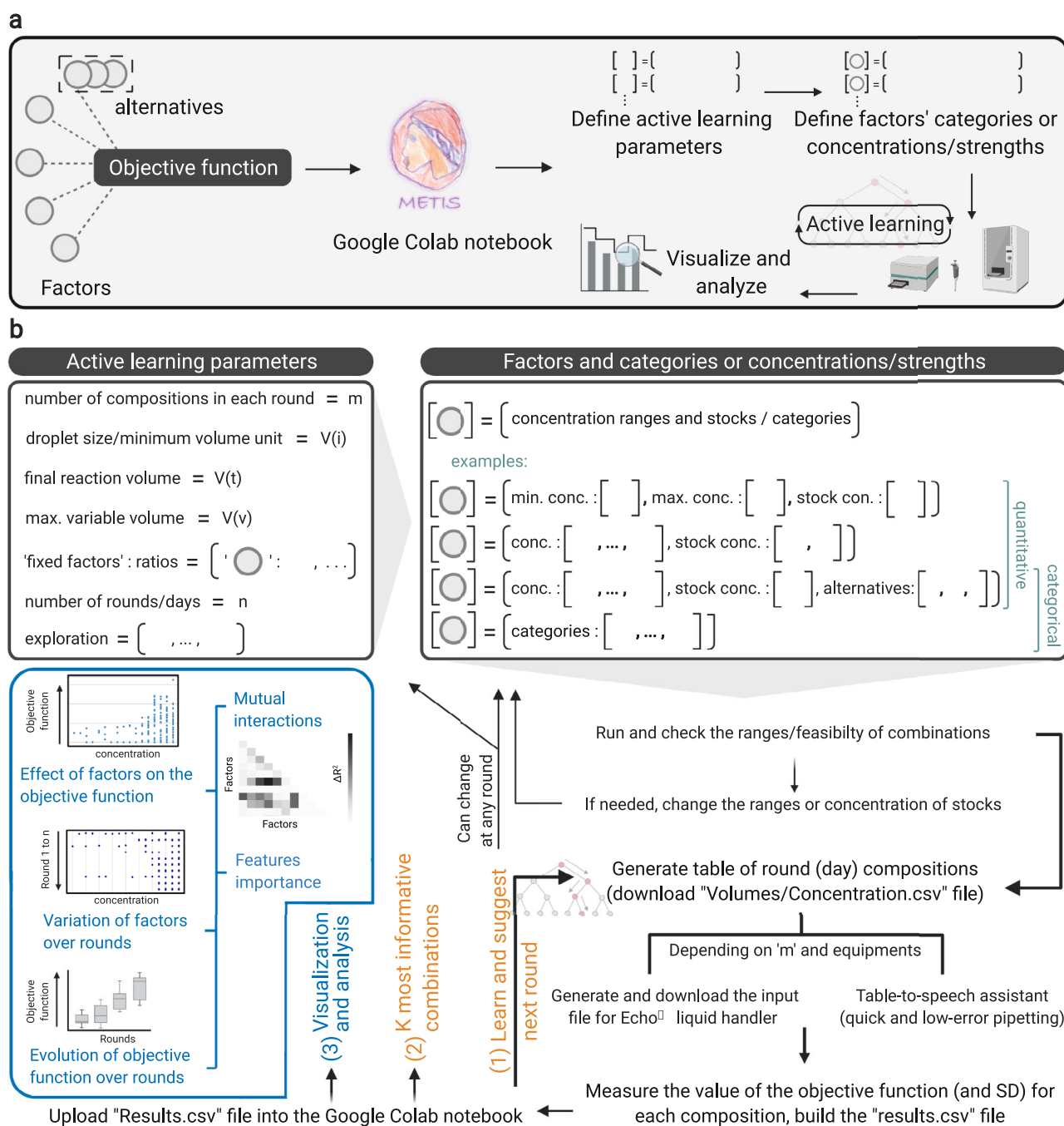
In Supplementary Note 2, we provide a detailed description of all features of METIS. The modular workflow enables the use of factors with numerical values (examples in Figs. 1, 3 and 5), categories (examples in Fig. 4, Supplementary Fig. 19), or both (example in Figs. 3 and 5). Active learning can be initialized by random combinations generated by the workflow in the first round (example in Figs. 1, 3 and 4). Alternatively, pre-existing datasets can be imported and used for optimization or simulations (examples in Supplementary Figs. 18 and 19). Although our workflow is designed as an active learning approach over iterative experimental rounds, it can be also used in a classical machine learning setup, when only using one round of experiments. Multiple data analysis and visualization modules are available that can be used in each round of active learning as shown in example applications (Fig. 2b, Supplementary Note 2). The workflow is able to generate a pipetting table output (exemplified for the experiments in Figs. 1 and 3), which alongside our table-to-speech virtual assistant tool, improves the speed and accuracy of manual pipetting (Supplementary Note 1, 2). For more complex experiments where multiple components in different volumes are required, the workflow can be interfaced

with lab automation (e.g., an Echo<sup>®</sup> acoustic liquid handling robot, see optimization of the CETCH cycle in Fig. 5).

### Application of METIS for optimization of a *LacI* gene circuit.

Next, we aimed to apply METIS for optimization of *LacI*-based gene circuits that were described recently<sup>38</sup>. Greco et al. developed a strategy for stringent gene expression by engineering transcriptional and/or translational small RNA inhibitors upstream of a *Gfp* reporter gene under the control of the pTAC promoter (Fig. 3a). Starting from a standard pTAC architecture, a so-called single-level controller (SLC), Greco et al. constructed three different multi-level controllers (MLC): pTHS (toehold switch; translational control), pSTAR (small transcription activating RNA; transcriptional control), and pDC (double controller; transcriptional and translational control)<sup>38</sup>. Notably, the authors could improve the rate of in vitro protein production by 35-fold with different MLC designs. Yet in these efforts, the fold-change in total protein production remained low (Supplementary Fig. 7), which was likely the result of leaky repressor-regulated promoters in the OFF state, as noted earlier<sup>34,37</sup>. A high fold-change in protein production, however, would be strongly desired for application of gene circuits, e.g., as diagnostic sensors, where a high signal-to-noise ratio is important. Additional to the high fold-change (FC), a desired circuit should have a high level of protein production, a feature that can be quantified by the dynamic range (DR) (Fig. 3a).

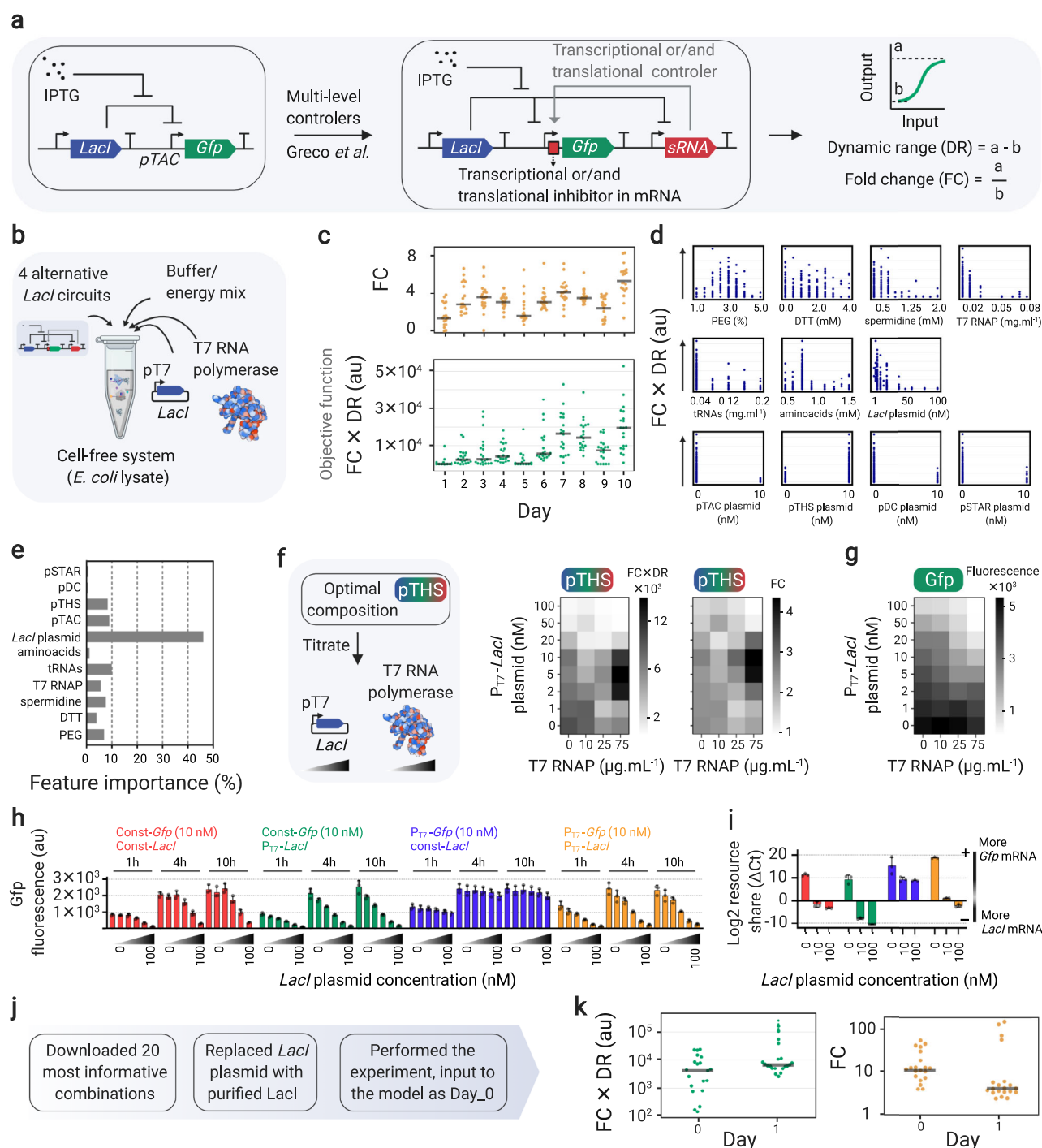
Here, we aimed at using our workflow to optimize the SLC and MLC *LacI* circuits. We performed 10 rounds of active learning with the objective function of  $FC \times DR$ , to score those compositions that result in not only high fold-changes but also total Gfp productions. The fold-change can be improved by supplying an additional plasmid expressing *LacI* (under the control of a T7 promoter transcribed by purified T7 RNA polymerase) and the dynamic range can be improved through alternative selection of SLC and MLC circuits and through tuning TXTL composition. The active learning cycle received input from several factors in the *E. coli* cell-free system; amino acids and tRNAs, which are important when extra DNA is added, DTT as reducing reagent, spermidine for DNA-protein binding, and PEG 8000 as crowding agent (Fig. 3b), and four *LacI* circuits (one SLC and three MLC) were considered as one categorical feature with four alternatives. While the objective function improved during the active learning cycle (bottom plot in Fig. 3c), we did not observe a substantial improvement in fold-change of Gfp production alone (upper plot in Fig. 3c). Feature importance analysis identified the concentration of the  $P_{T7}$ -*LacI* plasmid as strong contributor (Fig. 3d, e, Supplementary Fig. 8), indicating deleterious *LacI* protein-DNA interactions or resource limitation of the TXTL system through production of the *lacI* protein<sup>44</sup>.



**Fig. 2 A representation of METIS, a modular active machine learning workflow for biological systems.** **a** The first step is choosing an objective function (an output/target that depends on multiple factors), then continuing with the Google Colab Python notebook, performing experiments, and visualizing and analyzing results. **b** Users should define active learning parameters depending on the application, equipment, and the size of the combinatorial space. Factors' ranges/categories are conditions that are varied to explore the behavior of the objective function. In each round of active learning, while the users perform experiments and label the suggested combinations with measured objective function values (parameters and factors' conditions can be readjusted at any round), the data can be analyzed and visualized using the workflow's modules. See Supplementary Note 2 and Supplementary Figs. 4–6 for a detailed explanation and guide for each step.

By performing a titration experiment with  $P_{T7}$ -*LacI*, we could show that addition of the *LacI* plasmid has indeed a strong negative effect on the optimal *LacI* circuit (i.e., with pTHS) (Fig. 3f) and in an independent TXTL protein production (Fig. 3g) (see also Supplementary Note 3 for details of the active learning cycle and titration experiments). To further investigate this effect, we titrated the *LacI* plasmid with either T7 or a constitutive promoter against a fixed concentration of the *Gfp*

expressing plasmid under control of either T7 or a constitutive promoter. While increasing concentrations of the plasmid with constitutive *LacI* expression did only slightly affect *Gfp* expression from the T7 promoter, increasing concentrations of *LacI* plasmid under T7 control strongly affected *Gfp* production, especially when *Gfp* was expressed from the constitutive promoter (Fig. 3h). These results indicated a resource competition between the two plasmids, according to which the T7 promoter wins competition



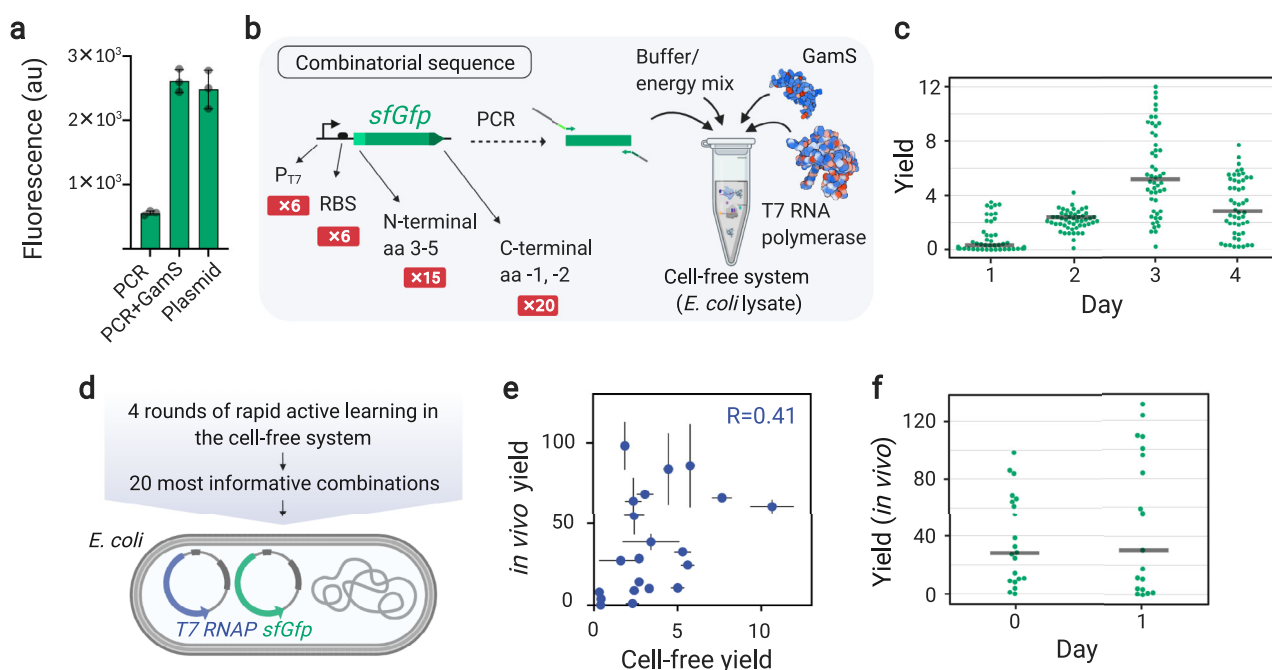
at the transcriptional and consequently the translational level. Quantifying the levels of *Gfp* and *LacI* mRNA by qPCR confirmed a direct correlation between mRNA and Gfp production levels, further supporting the resource competition hypothesis (Fig. 3i).

To overcome resource competition, we tested purified LacI protein instead of the *LacI* plasmid in the TXTL system, which resulted in improved Gfp productivity (Supplementary Fig. 9). Thus, we sought to optimize Gfp fold-change with using purified LacI protein instead of a *LacI* expressing plasmid. Using a module of METIS called “K most informative combinations” (with the number K to be defined by the user), we extracted the 20 most informative combinations of the active learning cycle and

repeated these 20 setup by replacing  $P_{T7}$ -*LacI* plasmid with purified LacI protein (Fig. 3j), resulting in a strong improvement in the objective function, and in particular Gfp fold-change (Fig. 3k). Note that among these 20 combinations were again all four SLC and MLC circuits. All of them improved upon providing external LacI, clearly demonstrating that resource competition had been limiting performance of the SLC and MLC circuits. Continuing with only one additional round of active learning using this dataset, we were able to improve the fold-change to up to 123 (Fig. 3k), which is 15-fold improvement compared to that of 10 rounds of active learning with the  $P_{T7}$ -*LacI* plasmid and 34-fold improvement compared to the initial setup.



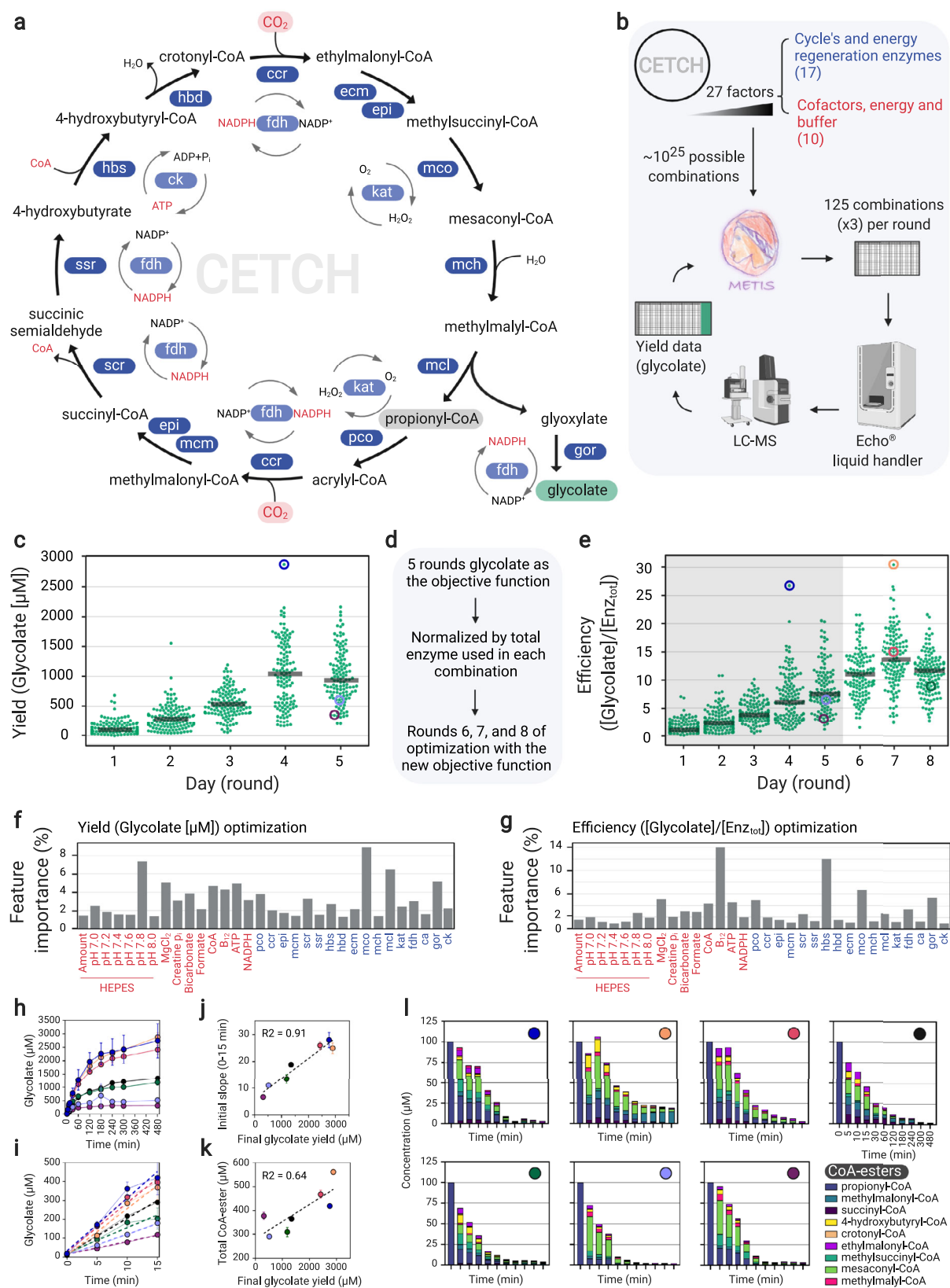
**Fig. 3 Application of METIS for optimization of a *LacI* gene circuit.** **a** *LacI* gene circuits characterized by dynamic range (DR) and fold-change (FC) of the output (Gfp fluorescence) between 0 and 10 mM IPTG. **b** Active learning by varying components of *E. coli* TXTL, 4 *lacI* circuit plasmids as alternatives, T7 RNA polymerase and a T7-*lacI* plasmid. **c** The objective function (FC × DR) and fold change (FC) values, average of triplicates ( $n = 3$  independent experiments) in 10 rounds of active learning. The gray lines show the median. **d** The distribution yield values within the range of each factor. **e** Feature importance percentages showing the effect of each factor on the objective function. **f** Titration of P<sub>T7</sub>-*LacI* plasmid and T7 RNA polymerase with the optimal composition (from active learning that achieved with pTHS circuit). The heatmaps show FC × DR (left) and FC (right) values (average of triplicates,  $n = 3$  independent experiments) of the titration. **g** Fluorescence values (average of triplicates,  $n = 3$  independent experiments) of the similar titration as in **f** but instead of the pTHS circuit, a *Gfp* expressing plasmid was used). **h** Titration of *LacI* plasmids with constitutive/T7 promoter in combination with a *Gfp* plasmid with constitutive/T7 promoter. **i** The RT-qPCR results of the relative level of *LacI* and *Gfp* mRNAs after 10 h. Relative log2 resource share between *LacI* and *Gfp* mRNA in each sample is reported to account for RNA purification efficiency variability. In **h** and **i** bars are the average of triplicates ( $n = 3$  independent experiments) and error bars are standard deviation. **j** Usage of the METIS module, K most informative combinations for further *LacI* circuit optimization. **k** Objective function FC × DR and FC (average of triplicates,  $n = 3$  independent experiments) of 20 most informative combinations with purified *LacI* (Day 0) followed by Day 1 experiments suggested by METIS. The gray lines show the median. The Google Colab Python notebook and all active learning data (combinations and yields) in this figure are available at <https://github.com/amirpandi/METIS>. Source data for **f**–**i** are provided as a Source Data file.



**Fig. 4 Application of METIS for optimization of a transcription & translation unit.** **a** The cell-free expression of *sfGfp* (super-folder *Gfp*) using plasmid, linear DNA (PCR) and linear DNA plus GamS protein, a nuclease inhibitor that protects linear DNA from degradation. The bars and the error bars are the average and standard deviation of triplicates ( $n = 3$  independent experiments), respectively. **b** Design of a transcription & translation unit controlled by variants of a T7 promoter, ribosome binding site (RBS), N-terminal amino acids 3, 4, and 5, and the last two C-terminal amino acids. The combinatorial transcription & translation units are expressed from linear DNA in the TXTL system consisting of the *E. coli* lysate, buffer and energy mix, as well as purified GamS and T7 RNA polymerase. **c** The plot representing the average of triplicates ( $n = 3$  independent experiments) as the result of 4 rounds of active learning, with 50 transcription & translation units tested per round. The yield is the Gfp fluorescence readout after 6 hours at 30 °C normalized by the same value from the reference constructs commonly used in the lab (Methods). The gray lines show the median. **d** A list of 20 most informative combinations of 4-day active learning performed in the cell-free system (**c**) was downloaded and the combinations were cloned in a vector and transformed into *E. coli* DH10 $\beta$  harboring a plasmid expressing auto-regulated T7 RNA polymerase (Methods). **e** Cell-free versus in vivo yields (average and standard deviation of triplicates,  $n = 3$  independent experiments) for the 20 most informative combinations. **f** In vivo yield results (average of triplicates,  $n = 3$  independent experiments) of Day 0 (20 most informative combinations) and Day 1 (suggested by the workflow). The gray lines show the median. The Google Colab Python notebook and all active learning data (combinations and yields) in this figure are available at <https://github.com/amirpandi/METIS>. Source data for **a**, **e** are provided as a Source Data file.

Overall, these experiments demonstrated how our workflow can be used to improve the signal-to-noise-ratio of an existing in vitro gene circuit by two orders of magnitude. Notably, the feature importance module of METIS, which identified apparent bottlenecks (i.e., resource competition by the *LacI* plasmid) and the K most informative combinations module of the workflow were crucial for success. A Google Colab notebook and all combinations and results are provided through Code and Data availability.

**Application of METIS on a transcription & translation unit.** To demonstrate that our workflow can also be used with categorical factors such as biological sequences, we tested METIS for the optimization of a transcription & translation unit. This unit is composed of six variants of a T7 promoter<sup>45</sup>, six ribosome binding sites (RBS)<sup>46</sup>, as well as 15 variations of N-terminal amino acids 3 to 5<sup>47</sup>, and 20 variations of the last two C-terminal amino acids<sup>48</sup>, which is in line with two recent studies that



reported the importance of N- and C-terminal amino acids on mRNA translation<sup>47,48</sup>. To establish a convenient cell-free screening system, we sought to use linear DNA (i.e., a PCR product)<sup>49</sup> as template in combination with GamS, a small, 136 amino acid-long nuclease inhibitor from phage λ<sup>50</sup> that binds and protects linear DNA from degradation. First, we validated that addition of linear DNA with GamS resulted in gene expression levels comparable to that of plasmid DNA (Fig. 4a), which allowed the fast and efficient assembly of DNA templates through PCR primers without extensive cloning, transformation, and plasmid preparation steps (Fig. 4b).

**Fig. 5 Application of METIS for optimization of an in vitro CO<sub>2</sub>-fixation pathway (CETCH cycle).** **a** Reaction sequence of the CETCH cycle (see Methods for enzyme names and information). **b** Active learning with 125 conditions tested in each round. ECHO<sup>®</sup> liquid handler pipetted the combinations and the reactions were started with 100  $\mu$ M propionyl-CoA and stopped after 3 h. The glycolate content was measured by LC-MS. **c** Optimization of the CETCH cycle with glycolate yield. **d** Summary of the optimization and the switch of the objective function. **e** Transformed data of **c** (glycolate yield divided by the total amount of enzymes = efficiency) for rounds 1–5, shaded region, and the data of three additional rounds of optimization with efficiency as the objective function (rounds 6–8). The yields in **c** and **e** are average of triplicates, ( $n = 3$  independent experiments) and the gray lines show the median. **f, g** Feature importance of factors for active learning in **c** and **e**, respectively. **h–l** Manually pipetted experiments for seven conditions, three highest glycolate yields (blue, orange and red), a control (black) and three randomly picked underperformed conditions (green, lavender, burgundy) color coded the same in **h–l** and circled in **c** and/or **e**. These plots show glycolate production over 8 h (**h**) and its first 15 min with slopes (**i**), initial production rate versus the final glycolate yield (**j**), total amount of measured CoA esters after 8 h versus the final glycolate yield (**k**), and quantified CoA esters over 8 h (**l**). The plotted values in **h–k** are the average of triplicates ( $n = 3$  independent experiments), and the error bars represent the standard deviation. In (**l**), bars are the average of triplicates ( $n = 3$  independent experiments), each compound is plotted with error bars in Supplementary Fig. 17. In **l** the amount of propionyl-CoA within the zero samples is the added amount (100  $\mu$ M) to start the reaction and was not measured by LC-MS. The Google Colab Python notebook and all active learning data (combinations and yields) in this figure are available at <https://github.com/amirpandi/METIS>. Source data for **h–l** are provided as a Source Data file.

We then optimized the transcription & translation unit that theoretically consists of 6 ( $P_{T7}$ )  $\times$  6 (RBS)  $\times$  15 (N-terminal)  $\times$  20 (C-terminal) = 10,800 potential conditions (i.e., combinations) through screening of only 200 combinations in 4 rounds of active learning (Fig. 4b). As the objective function, we defined the yield of the Gfp fluorescence readout of each transcription & translation unit normalized by a construct comprising wild-type T7 promoter, B0032 RBS and *sfGfp*. Yields were quantified after 6 hours of incubation of the different transcription & translation units at 30 °C in the *E. coli* cell-free system supplemented with purified GamS and T7 polymerase. Over 4 rounds of active learning, yield of the transcription & translation unit improved up to 12-fold on Day 3. Using a high exploration rate on Day 3 resulted in a wide distribution of yields, but no further improvement, indicating that an optimum had been reached (Fig. 4c). The distribution of alternative factors within the yield of 200 combinations and a representation of the feature importance are shown in Supplementary Fig. 10. Altogether, our experiments demonstrated again, how METIS can be used to improve a described genetic unit by more than an order of magnitude with minimal experimental efforts.

After having rapidly explored the combinatorial space of the sequence controlling the transcription & translation unit in a cell-free setup, we additionally investigated the effect of the 20 most informative combinations in vivo (Fig. 4d). Surprisingly, however, the cell-free and in vivo yields for the 20 combinations showed a relatively low correlation of 0.41 (Day 0, Fig. 4e, Supplementary Fig. 11). This indicated that although cell-free systems offer rapid prototyping solutions, the optimal candidates are not necessarily directly transferable in vivo. To investigate whether we can further improve the performance of the transcription & translation unit in vivo, we used the data from Day 0 and continued with one more round of experiments guided by our workflow (Day 1, Fig. 4f). This resulted in an improvement by 130% for the highest yield in vivo.

**Application of METIS for optimization of the CETCH cycle.** Finally, we aimed at assessing the performance of METIS for the optimization of complex metabolic networks. The collection of thousands of different enzymes and recent progress in enzyme engineering has opened the way for the design and construction of synthetic metabolic networks with new-to-nature properties<sup>35,51,52</sup>. One recent example is the CETCH cycle (Fig. 5a), a synthetic in vitro metabolic network consisting of 17 different enzymes that was built around a highly efficient CO<sub>2</sub>-fixing enzyme, Crotonyl-CoA carboxylase/reductase (Ccr), converting CO<sub>2</sub> into the C2-compound glyoxylate<sup>39</sup> or glycolate<sup>53</sup>. Notably, the CETCH cycle is more efficient than natural

occurring CO<sub>2</sub>-fixing pathways like the Calvin-Benson-Bassham (CBB) cycle<sup>39</sup>. However, since the enzymes used for its construction derive from different organisms and thus metabolic backgrounds, several rounds of rational optimization were needed to harmonize the enzyme reactions and cofactors used in the cycle; and even though the kinetic parameters of the individual enzymes are known, their interactions in such a complex setup are non-linear, hardly predictable and basically impossible to disentangle with pure rational approaches. Hence, we sought to use our active learning workflow to improve the CETCH cycle's productivity further.

The setup of the CETCH cycle consists of 27 components encompassing 13 core enzymes, as well as four accessory enzymes, and nine other components such as magnesium chloride, CoA, NADPH, ATP and the starting substrate propionyl-CoA (see all components in Fig. 5 and their concentration range in the Code availability). To minimize handling errors and automate the experimental setup of individual CETCH assays, we used an ECHO<sup>®</sup> 525 acoustic liquid handler with a minimal pipetting volume of 25 nL. Miniaturizing the assay to 10  $\mu$ L of total volume allowed us to work with 384-well plates and assay 125 different conditions in triplicates per active learning round (Fig. 5b). To determine the CETCH cycle's productivity (i.e., formation of glycolate from CO<sub>2</sub>), we developed an LC-MS (liquid chromatography-mass spectrometry) method using <sup>13</sup>C<sub>2</sub>-glycolic acid as an internal standard (Methods).

For the first five rounds of optimization, we used product yield (glycolate) as objective function (for a description of the used parameters see Supplementary Note 4). After four iterative rounds, we reached a final concentration of  $2.87 \pm 0.09$  mM glycolate in the best performing condition starting from 100  $\mu$ M propionyl-CoA (Fig. 5c). This yield translates into 57.4 fixed CO<sub>2</sub>-equivalents per acceptor (propionyl-CoA) and is >10 times more productive compared to the originally reported, already rationally optimized version 5.4 of the CETCH cycle<sup>39</sup>.

As we had not restricted the component resources during optimization, most of the superior conditions used more enzymes (compared to CETCH 5.4) to increase glycolate production (Supplementary Fig. 12). Next, we aimed at increasing specific productivity of the CETCH cycle. To that end, we took the data from the initial five rounds of unrestricted optimization and divided the glycolate yield values by the total concentration of enzymes used for each combination. This data was fed back to METIS and three additional rounds of active learning were performed with the new objective function, called "efficiency" (Fig. 5d). Optimization of efficiency identified one condition in round seven that is about six times more efficient than CETCH 5.4 and 14% more efficient than the best condition from the

unrestricted optimization achieved in round four (Fig. 5e, see also Supplementary Figs. 12 and 13).

To learn more about the possible bottlenecks of the CETCH cycle, we used the feature importance module of the METIS workflow along with plots visualizing the yield distribution over the range of each factor (Supplementary Figs. 14, 15). One of the most important contributors for both optimization efforts is the enzyme Methylsuccinyl-CoA dehydrogenase (Mco) (Fig. 5f, g). The enzyme's low activity of 0.1 U/mg and its unstable substrate methylsuccinyl-CoA, which is prone to spontaneous hydrolysis, likely require large amounts of Mco to preserve flux through the cycle<sup>54</sup>. During efficiency optimization, the two most important components were 4-hydroxybutyryl-CoA synthetase (Hbs) and coenzyme B<sub>12</sub> (Fig. 5g). Analysis of the top 10% best performing conditions (Supplementary Figs. 12 and 13) revealed that the concentrations of Hbs and B<sub>12</sub> were significantly lower compared to the control (CETCH 5.4). To verify that high concentrations of Hbs have a negative impact on the cycle, we tested our control assay with ten times less and with five times more of the enzyme. Indeed, increasing Hbs concentration in the original assay decreased yield by 40%, while decreasing Hbs by one order of magnitude did not lower glycolate yield (Supplementary Fig. 16). Regarding the negative impact of higher concentrations of B<sub>12</sub>, we reasoned that cobalt released from damaged cofactor could inhibit enzymes. Similar to high concentrations of Hbs, addition of cobalt to the original assay led to a decrease in glycolate yield (Supplementary Fig. 16).

To understand the dynamic behavior of the different CETCH cycle variants, we manually repeated the top three conditions (highest glycolate yields), a control (see Supplementary Note 4) and three underperforming conditions, taking time point samples for eight hours. The yield from this manual approach reflected the yield from the previous automated, miniaturized experiments, validating the results of our optimization efforts (Supplementary Table 3). Interestingly, the final glycolate yield after eight hours (Fig. 5h) and the initial glycolate formation rates of these conditions over the first 15 minutes (Fig. 5i) were highly correlated (Fig. 5j), indicating that total flux and not improved enzyme/cofactor stability (or life-time) was responsible for the observed increased productivity of the system. This trend was further confirmed by a detailed analysis of 9 CoA-ester intermediates at different time points (Fig. 5k, l). Quantification of the CoA-ester intermediates did not show accumulation of single metabolites in the underperforming conditions or the control, indicative of specific bottlenecks (Fig. 5l, Supplementary Fig. 16). Instead, the underperforming conditions showed overall a faster depletion of intermediates, in line with the hypothesis that high flux through the cycle is important to prevent the loss of intermediates towards side reactions or hydrolysis.

In summary, our optimization efforts of the CETCH cycle resulted in variants that showed more than ten-fold productivity and almost six-fold improved efficiency, representing to the best of our knowledge the most efficient in vitro CO<sub>2</sub>-fixing system described to date.

## Discussion

In this work, we describe METIS, a versatile, modular active learning workflow for the optimization of various biological objective functions, such as genetic and metabolic networks. This study democratizes machine learning applications for experimentalists without any programming skills or sophisticated lab equipment. We provide Google Colab notebooks (see Code availability) that can be adapted to different optimization applications (also known as Bayesian optimization) and even used for data-driven predictions (for use of the latter see Supplementary Table 1, Supplementary Note 5, Supplementary Fig. 18).

For tailoring the workflow, the number of rounds and experiments per round need to be defined, which should take into account the number of different factors and their conditions, complexity of the objective function, as well as experimental throughput. For applications with a larger combinatorial space, more combinations need to be tested (Fig. 5). However, if the number of experiments is limited by cost, effort, or lab equipment, performing active learning in more rounds can be used to compensate for a lower number of total combinations tested. To explore a system beyond a local optimum, it is advised to adapt the exploration to exploitation ratio for each round individually (fully discussed in Supplementary Note 2). Users should apply their knowledge on the system and implicitly check whether the value of a given factor is fixed too early, probably indicating a low exploration to exploitation ratio. On the other hand, a high exploration to exploitation ratio might push the model towards random combinations, asking for a proper balance to enable explorative as well as exploitation sampling. In our empirical experience, the exploration to exploitation ratio should gradually decrease towards the late rounds of active learning to enable more explorative combinations in early rounds and more exploitation in late rounds for efficient optimization (Supplementary Note 2).

Workflows can be started either from scratch (random combination as initialization) or using existing datasets (then performing active learning). Although our workflow is designed as an active learning approach (over multiple rounds of experiments), it can also be used as a classical machine learning with only one round of experiments. Factors of a given objective function can be numerical and/or categorical. Active learning parameters can be further customized using a detailed explanation in Supplementary Note 2.

METIS provides a variety of choices for visualization and analysis of results. Most importantly, our workflow can quantify importance of individual features and provide a number of most informative combinations, which has both proven particularly useful during *LacI* gene circuit optimization (Fig. 3). Using these features of the workflow allowed us to not only to improve the fold-change of the circuit, but also spot and, using additional experiments, verify a major bottleneck in the further optimization of the system (i.e., the *LacI* expression plasmid). After replacing the *LacI* expression plasmid with purified *LacI* protein, we were able to improve the circuit by more than two orders of magnitude compared to the original system. Notably, we did not have to re-perform active learning when switching to purified *LacI* instead of the *LacI* plasmid. The 20 most informative combinations generated through our workflow offered a short and quick path toward optimization.

Applying METIS onto different biological systems, we demonstrate that our workflow is able to optimize several complex genetic and metabolic networks of medium to large combinatorial space with minimal experimental efforts. As example, we improved the CETCH cycle a system of 27 variable factors including enzymes, cofactors, and buffer composition, spanning a theoretical combinatorial space of  $\sim 10^{25}$  different conditions. Performing only 1,000 (triplicate) assays over 8 rounds of active learning yielded a system with ten-fold improved productivity and six-fold increased efficiency, representing the most efficient in vitro CO<sub>2</sub>-fixation system described to date.

The development and application of complex genetic and metabolic networks in synthetic biology is dramatically increasing and require new tools for their data-driven analysis. Efficient explorative approaches are needed not only for the optimization of existing biological networks, but also for the design and realization of new-to-nature genetic and metabolic networks for which sampling the entire combinatorial space becomes practically impossible. Apart from network optimization with minimal experimental datasets, METIS can simultaneously help to discover so far unknown interactions and bottlenecks in these



networks, which paves the way for their hypothesis-driven improvement. In the *LacI* circuit optimization, we showed how a bottleneck (i.e., resource competition) can be identified, targeted, and finally overcome, which allowed us to improve the system by 34-fold. Similarly, during optimization of the CETCH cycle, we identified Mco, Hbs and B<sub>12</sub> as limiting factors.

Numerous applications of the METIS workflow can be envisioned in the future, including the optimization of growth media and/or biochemical assays, genetic circuits, from simple transcription & translation units to more complex designs, or the guided engineering of proteins, enzymes, and metabolic pathways in vivo and in vitro. With its convenience and easy access, METIS opens the door for the study, prototyping, (combinatorial) engineering, and optimization of these systems in an efficient, standardized, and systematic manner.

## Methods

**Gold regressor and analyzing different machine learning algorithms.** To find out which machine learning algorithm and sample size are suitable for our workflow, we conducted the following simulation:

- 1017 data points (compositions and yields) were collected from a recent study<sup>31</sup>.
- An XGBRegressor model (gold regressor) was trained on 80% of the dataset and 20% of the dataset was used for validation and to avoid overfitting via early stopping.
- 100 combinations produced randomly within the range of each factor for Day<sub>1</sub>.
- Instead of doing experiments in the laboratory to determine the yield of each combination, yield values were assigned by the gold regressor.
- Note that, in this phase the test model predicts the yields and ranks them to suggest for the experiments of the next day, and the gold regressor (trained on pre-data) is used to assign yield values by prediction instead of performing the experiments in the laboratory.
- For each machine learning model (MLP, DNN, linear regressors, XGBoost) an ensemble of 5 models with different hyperparameters was produced.
- Note that the linear regressors is a deterministic approach so we just duplicated a model 5 times for which all predictions are the same.
- Each ensemble was trained on Day<sub>1</sub> data.
- 100,000 random combinations were generated, and their corresponding yield was predicted by the ensemble of models and ranked by UCB score (see method section for the core algorithm of active learning), top 100 combinations were suggested for the next day. Yields were assigned by the gold regressor.
- The last two steps were repeated for other days, and on each day the model was trained on all the previous days' data.

Note that, in Fig. 1b for different sample sizes with XGBoost, 5, 10, 25, or 100 combinations were suggested for the next day.

Hyperparameters: MLPRegressor from Sklearn (fully connected architecture with Relu activation function) was used for MLP. In ensemble of 5 models the following number of neurons were used in the hidden layer: (10, 100, 100, 20), (20, 100, 100, 10), (20, 100, 100, 20), (10, 100, 100, 10), (20, 100, 100, 50). For DNN we used the Keras implementation of fully connected layer architecture with 100, 100, 20 neurons for each of hidden layers. For Linear Regression the default implementation of Ordinary Least Square by Sklearn was used. XGBRegressor with following parameter was used for XGBoost model: objective = 'reg:squarederror', n\_estimators = 500, learning\_rate = 0.01, max\_depth = 6, min\_child\_weight = 1, subsample = 0.8.

**General description of METIS notebook.** All scripts used in this study were written in Python 3. Our modular tool, METIS, runs on Google Colab working through web browsers with a link without users needing to install Python or any packages.

Packages used in the development of METIS:

- Data processing: pandas (1.1.4) and numpy (1.18.5)
- Data visualization: matplotlib (3.2.2) and seaborn (0.11.0)
- Machine learning and deep learning: scikit-learn (0.22.2.post1), xgboost (0.90), and Keras (2.3.1) using TensorFlow backend.

**The core algorithm of active learning.** After measuring the value of the objective function (yield) for random combinations of Day<sub>1</sub>, we continued with the following algorithm:

- RandomSearchCV is used to find the optimal 20 hyperparameters for the XGBoost model.
- The ensemble of 20 models is trained with the hyperparameters on data from all previous days (Day<sub>1</sub> to present day).

- 100,000 combinations out of possible combinations are randomly selected.
- The mean and standard deviation of ensemble predictions are calculated.
- The combinations are sorted based on Upper Confidence Bound (UCB) score:  $^{31}$  exploitation \* (average of predictions) + exploration \* (standard deviation of predictions).
- To perform experiments of the next day, the combinations with the highest UCB values are suggested.

The high standard deviation represents the uncertainty and improves the prediction power of models, whereas a high average value weighs favorable combinations leading to higher yields. Hence a coupled score taking into account these two factors ranks the most promising combinations<sup>31</sup>. Note that the active learning for optimization of objective functions is also called Bayesian optimization<sup>55</sup>. In Supplementary Fig. 21 we show optional data preprocessing and an improved XGBoost model. See Supplementary Fig. 22 for an optional scoring (can be defined when using METIS), batch UCB that can generate richer combinations for subsequent rounds.

**Finding K most informative combinations.** The K most informative combinations are calculated using the following algorithm:

- RandomSearchCV is used to find the optimal 20 hyperparameters for the XGBoost model.
- 2000 subsets of length K are selected from the tested combinations. The total number of possible subsets is represented in Eq. (1).

$$\binom{N}{K} = \frac{N!}{K! \times (N-K)!} \quad (1)$$

- Then a new XGBoost with the optimal hyperparameter is trained on each subset. The model performance is then validated on unseen combinations using the Spearman correlation coefficient.
- All subsets are sorted based on their Spearman correlation coefficient, the top 5 are then chosen. Each of these 5 could be used.

Note that increasing the number of subsets leads to a longer training time.

**Finding feature importance.** Feature importance values have been calculated with the following algorithm:

- RandomSearchCV is used to find the optimal hyperparameter for the XGBoost model.
- The model is trained using the selected hyperparameter. Using the built-in "feature\_importances\_" property of the XGBoost package, the ratio of feature importance is calculated throughout the training process for each day cumulatively.

**Finding nonlinear (mutual) interactions.** In complex systems, factors usually interact with each other and epistatically affect the output. These interactions can be among many factors, however, the most relevant is the mutual or double interaction between factors<sup>56</sup>. This analysis can be a hint to discover biological phenomena's behavior. The mutual interactions were calculated through the following algorithm:<sup>57</sup>

- A linear regression model is fitted on the dataset and its performance is evaluated based on the R squared of predicted and actual values. This performance is considered as the baseline.
- Iteratively, a new feature is added to the temporary dataset that equals  $F_i \times F_j$  for i and j in the list of factors.
- The linear regression is fitted on the temporary dataset (which now has one more feature,  $F_i \times F_j$ ) its performance is measured similarly to the baseline.
- The difference between each performance and the baseline, j, is calculated and visualized.

**METIS prediction.** In contrast to METIS optimization that tries to find the most promising combinations through maximizing the objective function, METIS prediction aims to maximize the model performance on the prediction of the objective function for unseen combinations. We modified the core active learning algorithm:

- Instead of UCB (exploitation × mean + exploration × std), combinations are sorted based on only their std value and set exploitation to zero. This enables picking the most uncertain combination for the next round.
- At the end of each round, it returns a trained model instead of promising combinations, and the R squared of prediction is improved over rounds.

**Performance analysis using cross-validation.** To evaluate the model performance of the enzyme engineering notebook, we used k-fold cross-validation. In each round, all the tested combinations are divided into k subsets (k = 5 for Supplementary Figs. 18, 19), then in five steps we trained the model on 4 and evaluated its performance (R<sup>2</sup> Pearson) on the other subset. This process was

repeated for all 5 subsets. In the end, the average performance on all subsets was reported as the model's performance. We used sklearn built-in function for cross-validation.

**Table-to-speech virtual assistant.** This tool helps molecular biologists to boost their manual liquid handling through reading volume and destination well in ascending order, therefore minimizes the need for changing the pipetting volume. We used the Google Text2Speech python package to transform the text into a voice file. There are two ways to interact with this notebook to continue with the next pipetting volume. The first is to do it manually with your keyboard (what we did), the second is using the voice assistant. For transforming voice to text (specific commands like 'next', 'repeat', etc.). We used the SpeechRecognition (3.8.1) python package. The code is available on <https://github.com/amirpandi/Liquid-Handling-Assistant>.

**Plasmid and DNA preparation.** The constitutive *Gfp* under the control of J23101 promoter and B0032 RBS was built in a recent study (pBEAST-J23101-B0032-*sGfp*)<sup>58</sup>. Using golden gate cloning (BsaI-HF<sup>®</sup>v2 NEB #R3733L, T4 DNA ligase NEB #M0202T), in this plasmid, the super folder *Gfp* gene was replaced by *LacI* for constitutive-*LacI*, then the promoter was replaced by a T7 promoter (gaattaa-tacgactactatgggaga) to construct P<sub>T7</sub>-*LacI* plasmid. Since we used T7 promoters, a T7 terminator from Temme et al.<sup>59</sup> (tactgaaccctagccgctcttctcgccggctagggtttttgt) was cloned downstream. The version of *LacI* gene is similar to those in *LacI* circuits built by Greco et al.<sup>38</sup>. Plasmids for the cell-free gene expression were purified using the Machery-Nagel NucleoBond Xtra Maxi kit. For protein purification using His tag, *sGfp* and *LacI* genes were cloned with an N-terminal His tag under IPTG-inducible T7 promoter.

For cell-free experiments for optimization of the transcription & translation unit (Fig. 4b), PCRs were performed using Q5<sup>®</sup> High-Fidelity 2X Master Mix (NEB #M0492L), *sGfp* as the template, and primers (Eurofins and Sigma-Aldrich) with overhangs harboring P<sub>T7</sub>, RBS, and N-terminal sequence (forward primer) and C-terminal (reverse primer) at the final volume of 50  $\mu$ L. After verification of PCRs using agarose gel, Monarch PCR & DNA Cleanup Kit (NEB #T1030L) was used to purify the fragments and they were all adjusted to the concentration of 100 nM to use for active learning experiments.

For in vivo experiments of the transcription & translation unit (Fig. 4d) PCRs were done similar to the cell-free experiment. Restriction sites for BsaI enzyme were designed on either side of PCR fragments enabling for goldengate assembly into a pSEVA224 vector (a low copy plasmid with kanamycin marker) from the SEVA collection<sup>60,61</sup>. Since we used T7 promoters, a T7 terminator from Temme et al.<sup>59</sup> (tactgaaccctagccgctcttctcgccggctagggtttttgt) was cloned downstream.

**Protein purification.** For all enzymes involved in the CETCH cycle, expression and purification were performed as previously described<sup>62</sup>. Other proteins, T7 RNA polymerase (addgene #124138), GamS (addgene #45833), *sGfp*, and *LacI* were His-tag purified using Protino<sup>®</sup> gravity columns (Machery-Nagel #745250) and Protino<sup>®</sup> Ni-NTA Agarose (Machery-Nagel #745400). 1 L cultures in LB media supplement with appropriate antibiotic were subcultured (1:100) from overnight precultures. Cultures were grown at 37 °C for two hours, then induced by 0.1 mM IPTG, incubated for 3 more hours at 37 °C to produce proteins. Cells were harvested at 8000 g for 10 min, pellets were resuspended with 5 mL NPI-10 buffer, and sonicated. Samples were centrifuged at 18000 g for 1 hour at 4 °C. The equilibration, wash, and elution steps were done according to the manufacturer's protocol. Next, imidazole desalting was performed using PD-10 desalting columns (GE Healthcare #17085101) according to the manufacturer's protocol. The purification was verified using the SDS page and the protein concentrations were determined using the Bradford assay. Glycerol was added to the protein samples to a final percentage of 10%, then they were aliquoted and after flash-freezing in liquid nitrogen, stored at -80 °C.

**Lysate preparation.** *E. coli* lysate was prepared using an autolysis strategy<sup>63</sup>. Briefly, freeze-thawing *E. coli* BL21-Gold (DE3) cells with a pAS-LyseR plasmid produce a high-quality extract. Overnight precultures in LB-ampicillin media at 37 °C were subcultured in 5  $\times$  2 L 2xYTPG medium supplemented with ampicillin and grown at 37 °C to the OD = 1.5. Cells were harvested (2000g, 15 min, room temperature) in 10 centrifuge bottles and 90 mL of cold S30A buffer (50 mM Tris-HCl at pH 7.7, 60 mM K-glutamate, 14 mM Mg-glutamate, to the final pH of 7.7) was added to each. After vigorous vortexing, each was divided into two preweighed 50 mL falcons and centrifuged (2000g, 15 min, room temperature). The supernatants were removed carefully and after weighing falcons with pellets, the net weights were calculated. Two volumes of cold S30A with 2 mM DTT, were used to resuspend each pellet (2.8 mL for 1.4 g pellet), which were then vortex-mixed, and stored at -80 °C. The next day, frozen cells were thawed in a water bath at room temperature, vigorously vortex-mixed, and incubated at 37 °C shaking for 45 min. The vortexing and 45 min incubation steps were repeated. Finally, the samples were centrifuged (30000 g, 60 min, 4 °C) to obtain the cell extract. The supernatants were gently pipetted out in 1.5 tubes, recentrifuged (20000 g in a tabletop centrifuge, 5 min, 4 °C) to remove all the remaining cell debris aliquoted, and after freezing in liquid nitrogen stored at -80. For the composition of the cell-free reaction buffer

and energy mix, all chemicals were used as by Sun et al.<sup>33</sup> except for amino acids (L-amino acids set, Sigma #LAA21-1KT).

**Cell-free reactions.** To perform the active learning experiments in Figs. 1 and 3, Table2Seech\_Volume.csv file of each round was downloaded from the notebook and uploaded to the table-to-speech virtual assistant notebook. Before starting the pipetting, we arranged all pipette tips with numbers written on one side of tip boxes (two boxes side by side) from 1 to 20 (for 20 data points). PCR tubes in which the compositions were going to be mixed also were numbered on racks from 1 to 20. The numbering increases the accuracy of the manual pipetting. Next, the table-to-speech assistant was run on a laptop on the bench and the space key was set in the Google Colab settings to run the code. After pipetting each factor into the corresponding destination, while the right hand was replacing the tip, the left hand pressed the space key to hear the next pipetting step in a headphone as well as to see the action appearing on the screen. The table-to-speech assistant goes line by line for each factor and ranks the pipetting values from minimum to maximum, hence, minimizes changes in the pipette volume. For fixed elements such as HEPES and lysate, a master mix was made and after finishing pipetting all combinations, the master mix was added to each. All the steps were performed on ice. At the end, samples were gently mixed (not to generate bubbles) using a multichannel pipette and 10  $\mu$ L of each was transferred into a 384-well plate (Greiner Bio-One #784076). Note that the volume of mixtures should be at least 20% in excess in PCR tubes not to face difficulties in the final pipetting step into the 384-well plate. The *Gfp* fluorescence was monitored (excitation: 485, emission: 528 nm, gain: 80) every 10 min in a plate reader (Tecan Infinite 200 PRO).

The yield (objective function) in Fig. 1e, as provided in the Data availability, is the *Gfp* fluorescence (after 6 h incubation at 30 °C) of each composition normalized by a composition in which the concentration of all variable factors is at mid-range. However, the plotted yields are those values divided by 0.33, the average ratio of *Gfp* fluorescence between the active learning reference and a commonly used composition<sup>33</sup>. The objective function of the *LacI* circuit active learning in Fig. 3c is fold-change (FC)  $\times$  dynamic range (DR) of the output (*Gfp* fluorescence) between 0 and 10 mM input (concentration of IPTG). For cell-free reactions in Fig. 4c, the final volume of 5  $\mu$ L was prepared directly in a 384-well plate, 10 nM final concentration of each linear DNA was transferred and the mix of other components of the cell-free lysate plus T7 polymerase (40  $\mu$ g.mL<sup>-1</sup>) and GamS (2  $\mu$ M) was added while gently mixing. The yield (objective function) in Fig. 4c is the *Gfp* fluorescence readout (after 6 h of incubation at 30 °C) of each transcription & translation unit normalized by the *Gfp* fluorescence of a commonly used sequence in our lab, wild-type T7 promoter, B0032 RBS, and *sGfp* sequence. For all cell-free reactions, the *Gfp* fluorescence readout of the extract with no DNA was subtracted before yield calculations.

**RT-qPCR experiment.** Total RNA was extracted from cell-free expression reactions with a kit (NEB #T2010), following the manufacturer's instructions. Initial qPCR analysis indicated that a substantial amount of plasmid DNA remained in control reactions, which did not include reverse transcriptase to synthesize cDNA. Therefore, samples were subsequently treated to an additional DNase treatment by TURBO DNA-free<sup>™</sup> Kit (Invitrogen<sup>™</sup> #AM1907) according to the manufacturer's instructions. The resulting RNA produced a substantial qPCR signal (iTaQ Universal SYBR Green Supermix Bio-Rad #1725120) when converted to cDNA by ProtoScript<sup>®</sup> II Reverse Transcriptase (NEB #M0368) using the standard protocol and random hexamer primers (ThermoFisher #SO142), but not in control reactions lacking reverse transcriptase. In order to account for potential sample-to-sample variability in extraction efficiency, all data presented herein is represented as a relative difference in cycle threshold (Ct) between *Gfp* and *LacI* cDNA within each sample. Standard curves with known concentrations of plasmid DNA were analyzed in parallel for *Gfp* and *LacI* primer sets, indicating comparable qPCR efficiencies and template specificity. No further normalization was required.

**Western blot.** Cell-free expression reactions and *LacI*-6xHis purified protein dilutions were mixed with 4  $\mu$ L of non-reducing sample loading buffer (Thermo Scientific #39001) and incubated at 90 °C for 5 minutes. The samples were then loaded into pre-cast SDS-PAGE gels (Bio-Rad #4561095) and separated by electrophoresis. The gel was then immediately placed into a Bio-Rad TransBlot<sup>®</sup> Turbo apparatus for protein transfer onto a nitrocellulose membrane (Bio-Rad #1704158). Since all samples were produced from the same batch of cell-free expression reaction mix or were of known concentration, total protein concentration was not assessed. Western blot analysis was performed using a monoclonal antibody against *LacI* clone 9A5 (Sigma-Aldrich #05-503-I) and an anti-mouse HRP-conjugated secondary antibody (Invitrogen #31430) with dilution of 1:1000 and 1:100,000, respectively. After dispensing the detection reagent as indicated by the manufacturer (Neogen #324175), the blot was immediately imaged on a Bio-Rad ChemiDoc. A single clear band corresponding to the molecular weight of *LacI* was detected in lanes containing purified *LacI* or expression from a *LacI*-containing plasmid (see inset, Supplementary Fig. 9a).

**In vivo experiment of transcription & translation units.** After cloning transcription & translation units into the pSEVA224 vector (plasmid and DNA preparation section), they were transformed into *E. coli* DH10 $\beta$  harboring an autoregulated T7 RNA polymerase circuit (addgene #71428)<sup>64</sup>. 3 colonies of each were cultured in LB with 30  $\mu\text{g.mL}^{-1}$  ampicillin + 30  $\mu\text{g.mL}^{-1}$  kanamycin in a 96 deep well plate. After 10 hours of cultivation at 37 °C, 10  $\mu\text{L}$  of each was added to 190  $\mu\text{L}$  LB with 30  $\mu\text{g.mL}^{-1}$  ampicillin + 30  $\mu\text{g.mL}^{-1}$  kanamycin in a 96-well plate (Thermo Scientific #137101). The Gfp fluorescence was monitored (excitation: 485, emission: 528 nm, gain: 80) every 30 min in a plate reader (Tecan Infinite 200 PRO) shaking at 37 °C. The in vivo yield in Fig. 4e, f is the Gfp fluorescence readout (after 6 h) of each transcription & translation unit normalized by the Gfp fluorescence of a commonly used sequence in our lab, wild-type T7 promoter, B0032 RBS, and *sfGfp* sequence. The Gfp fluorescence readout of cells with no *sfGfp* gene was subtracted before yield calculations.

**Workflow for CETCH assays in 384-well plates.** The workflow generated by the METIS script was dissected into 5 worklists: dH<sub>2</sub>O, Buffers and Cofactors, Enzymes, Carbonic Anhydrase, and Substrate (pco: propionyl-CoA oxidase, ccr: crotonyl-CoA carboxylase/reductase, epi: ethylmalonyl-CoA/methylmalonyl-CoA epimerase, mcm: methylmalonyl-CoA mutase, scr: succinyl-CoA reductase, ssr: succinic semialdehyde reductase, hbs: 4-hydroxybutyryl-CoA synthetase, hbd: 4-hydroxybutyryl-CoA dehydratase, ecm: ethylmalonyl-CoA mutase, mco: methylsuccinyl-CoA oxidase, mch: mesaconyl-CoA hydratase, mcl:  $\beta$ -methylmalonyl-CoA lyase, gor: glyoxylate reductase, kat: catalase, fdh: formate dehydrogenase, ck: creatine phosphokinase). For source of enzymes and kinetic parameters see Schwander et al.<sup>39</sup>. In cases where pipetting errors occurred, we used our Exceptions to Workflow script for correction of failed transfers (provided in Code availability). This script generates a new workflow out of the exception file generated by the ECHO<sup>®</sup> and provides a list with how much volume needs to be added into which well. Dissecting the worklists guarantees for example that all buffers are transferred before enzymes are added. Note that we used fresh enzyme stocks in each round to prevent loss of activity due to repetitive freeze-thaw cycles. As source plates we used ECHO<sup>®</sup> qualified 384-Well PP 2.0 Plus Microplates from Labcyte and used AQ\_GP as the liquid class (AQueous solution; Glycerol/Protein). This liquid class was tested previously with the stocks of our assay components.

We also added a control condition with composition derived from the published assay of CETCH 5.4 (for the composition, see [Assays for determination of new enzyme stocks after round two](#) section in Supplementary Note 4). Controls can be added in the workflow as specials. The yield of our control condition increased from round 2 to 3, where new enzyme batches of four enzymes were used (Supplementary Fig. 16b). To identify the enzyme that was the reason for that, we tested the control assay with each of the four old enzymes separately (Supplementary Fig. 16b). Despite being important in the control (~280  $\mu\text{M}$  in round 1 and 2), catalase did not seem important in each condition, since we reached yields up to 1500  $\mu\text{M}$  already in round 2 with the old stock (Fig. 5c).

After starting the assays with 100  $\mu\text{M}$  propionyl-CoA we used an Axygen<sup>®</sup> Breathable Sealing Film (BF-400-S) to cover the 384-well PCR Plate (AB-1384) to allow the transfer of oxygen. The reaction (10  $\mu\text{L}$  volume) was carried out at 30 °C and mild shaking at 160 rpm in an Infors HT Ecotron shaker. The reactions were stopped after 3 h with 1.25  $\mu\text{L}$  of 500 mM polyphosphate and 1.25  $\mu\text{L}$  of 50% formic acid. While the formic acid quenches the reaction, the polyphosphate was used for enhanced precipitation of the proteins. The plate was spun for 1 h at 2272 g and 4 °C to pellet the proteins.

For analysis by LC-MS, we used a multichannel pipette to transfer 1  $\mu\text{L}$  of the supernatant into 9  $\mu\text{L}$  of precooled dH<sub>2</sub>O in a new 384-Well Thermo-Fast<sup>®</sup> plate. Afterward, we added 10  $\mu\text{L}$  of 10  $\mu\text{M}$  <sup>13</sup>C<sub>2</sub> labeled glycolic acid as an internal standard. The plate was sealed with a Corning<sup>™</sup> Microplate Aluminum Sealing Tape (6570). The assay plate with the quenched reactions was sealed with a Corning<sup>™</sup> Microplate Aluminum Sealing Tape too and stored at -80 °C.

**Timepoint assays of 7 selected conditions.** The assays were done in triplicates containing 150  $\mu\text{L}$  volume each and were carried out in a 1.5 mL reaction tube (at 30 °C, 500 rpm). The reactions were started with 100  $\mu\text{M}$  propionyl-CoA. 12  $\mu\text{L}$  samples were taken and quenched in 1.5  $\mu\text{L}$  50% formic acid and 1.5  $\mu\text{L}$  500 mM sodium polyphosphate (emplura<sup>®</sup>) at 5, 10, 15, 30, 60, 120, 180, 240, 300 and 480 min. The samples were spun for 20 min at 4 °C and 20,000 g, before the supernatant was transferred into Thermo Scientific<sup>™</sup> Abgene 96 Well Polypropylene Storage Microplates (AB-1058) and sealed with Corning<sup>™</sup> Microplate Aluminum Sealing Tape. While 2  $\mu\text{L}$  were used to prepare a 1:10 dilution in water for the measurement via LC-MS, the remaining samples were stored at -80 °C. The concentrations for the assays are shown in the table below (Buffers and cofactors in mM, enzymes in  $\mu\text{M}$ ). See Supplementary Table 3 for the details of these conditions.

**LC-MS analysis of CoA esters.** All CoA esters were measured on a triple quadrupole mass spectrometer (Agilent Technologies 6495 Triple Quad LC-MS) equipped with a UHPLC (Agilent Technologies 1290 Infinity II) using a 150  $\times$  2.1 mm C18 column (Kinetex 1.7  $\mu\text{m}$  EVO C18 100 Å) at 25 °C. The injection volume was 2  $\mu\text{L}$  of the diluted samples (1:10 in water). The flow was set to 0.400 mL.min<sup>-1</sup> and the separation was performed using 50 mM ammonium

formate pH 8.1 (buffer A) and acetonitrile (buffer B). We quantified the CoAs using external standard curves prepared in water with formic acid at pH 3. The standard curves were measured before and after the samples. Except for methylsuccinyl-CoA, all compounds were stable. For methylsuccinyl-CoA we calculated the concentration as an average of the two standard curves at the time point the sample was measured. The parameters for the multiple reaction monitoring (MRMs) and the gradient are shown in the tables below. The data analysis was done with Agilent MassHunter Quantitative Analysis (for QQQ). See Supplementary Table 4 (Gradient for the separation of CoA esters) and Supplementary Table 5 (MRM transitions).

**LC-MS analysis of glycolate.** Glycolate was measured on a triple quadrupole mass spectrometer (Agilent Technologies 6495 Triple Quad LC-MS) equipped with a UHPLC (Agilent Technologies 1290 Infinity II) using a 150  $\times$  2.1 mm C18 column (Kinetex 1.7  $\mu\text{m}$  EVO C18 100 Å) at 25 °C. The injection volume was 0.5  $\mu\text{L}$ . The diluted samples (1:10 in water), as well as the external standard curve, were diluted 1:2 with 10  $\mu\text{M}$  <sup>13</sup>C<sub>2</sub>-labeled glycolic acid as internal standard. The flow was set to 0.100 mL.min<sup>-1</sup> and the separation was performed using dH<sub>2</sub>O with 0.1% formic acid (buffer A) and methanol with 0.1% formic acid (buffer B). The parameters for the multiple reaction monitoring (MRMs) and the gradient are displayed below. Data analysis was done using the Agilent Mass Hunter Workstation Software. See Supplementary Table 6 (Gradient for the separation of CoA esters) and Supplementary Table 7 (MRM transitions).

**Data analysis.** Data was analyzed using Microsoft Excel, GraphPad Prism, and custom Python scripts (available at <https://github.com/amirpandi/METIS>) and Agilent Mass Hunter Workstation Software (QQQ) 10.0 for LC-MS data.

**Reporting summary.** Further information on research design is available in the Nature Research Reporting Summary linked to this article.

## Data availability

The source data underlying Figs. 3f-i, 4a, e, and 5h-l and Supplementary Figures 2, 7a, b, 9b, 11, 16a, b are provided as a Source Data file. All active learning data (combinations and yields) are available along with Google Colab Python notebook of each application on GitHub, <https://github.com/amirpandi/METIS>. Primers used for transcription and translation units (Fig. 4) are provided in Supplementary Tables 8, 9. Source data are provided with this paper.

## Code availability

METIS workflows for different applications used in this study run as Google Colab Python notebooks and are free open source tools available at <https://github.com/amirpandi/METIS>. All scripts used in this study were written in Python 3. Packages used in the development of the workflow are pandas (1.1.4) and numpy (1.18.5), matplotlib (3.2.2) and seaborn (0.11.0), scikit-learn (0.22.2.post1), xgboost (0.90), and Keras (2.3.1) using TensorFlow backend.

Received: 12 February 2022; Accepted: 10 June 2022;  
Published online: 05 July 2022

## References

- Purnick, P. E. M. & Weiss, R. The second wave of synthetic biology: from modules to systems. *Nat. Rev. Mol. Cell Biol.* **10**, 410–422 (2009).
- Smanski, M. J. et al. Functional optimization of gene clusters by combinatorial design and assembly. *Nat. Biotechnol.* **32**, 1241–1249 (2014).
- Dolberg, T. B. et al. Computation-guided optimization of split protein systems. *Nat. Chem. Biol.* **17**, 531–539 (2021).
- Radivojević, T., Costello, Z., Workman, K. & García Martín, H. A machine learning automated recommendation tool for synthetic biology. *Nat. Commun.* **11**, 4879 (2020).
- Naseri, G. & Koffas, M. A. G. Application of combinatorial optimization strategies in synthetic biology. *Nat. Commun.* **11**, 2446 (2020).
- Carbonell, P., Radivojević, T. & García Martín, H. Opportunities at the intersection of synthetic biology, machine learning, and automation. *ACS Synth. Biol.* **8**, 1474–1477 (2019).
- Camacho, D. M., Collins, K. M., Powers, R. K., Costello, J. C. & Collins, J. J. Next-generation machine learning for biological networks. *Cell* **173**, 1581–1592 (2018).
- Gilliot, P.-A. & Gorochowski, T. E. Sequencing enabling design and learning in synthetic biology. *Curr. Opin. Chem. Biol.* **58**, 54–62 (2020).
- Volk, M. J. et al. Biosystems design by machine learning. *ACS Synth. Biol.* **9**, 1514–1533 (2020).



10. Eraslan, G., Avsec, Ž., Gagneur, J. & Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.* **20**, 389–403 (2019).
11. Libbrecht, M. W. & Noble, W. S. Machine learning applications in genetics and genomics. *Nat. Rev. Genet.* **16**, 321–332 (2015).
12. Liu, J., Li, J., Wang, H. & Yan, J. Application of deep learning in genomics. *Sci. China Life Sci.* **63**, 1860–1878 (2020).
13. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16**, 687–694 (2019).
14. Wittmann, B. J., Johnston, K. E., Wu, Z. & Arnold, F. H. Advances in machine learning for directed evolution. *Curr. Opin. Struct. Biol.* **69**, 11–18 (2021).
15. Gussow, A. B. et al. Machine-learning approach expands the repertoire of anti-CRISPR protein families. *Nat. Commun.* **11**, 1–12 (2020).
16. Kim, H. K. et al. Deep learning improves prediction of CRISPR–Cpf1 guide RNA activity. *Nat. Biotechnol.* **36**, 239–241 (2018).
17. Wang, D. et al. Optimized CRISPR guide RNA design for two high-fidelity Cas9 variants by deep learning. *Nat. Commun.* **10**, 1–14 (2019).
18. Eitzinger, S. et al. Machine learning predicts new anti-CRISPR proteins. *Nucleic Acids Res* **48**, 4698–4708 (2020).
19. Hiscock, T. W. Adapting machine-learning algorithms to design gene circuits. *BMC Bioinformatics* **20**, 1–13 (2019).
20. Saltepe, B., Bozkurt, E. U., Güngen, M. A., Çiçek, A. E. & Şeker, U. Ö. Ş. Genetic circuits combined with machine learning provides fast responding living sensors. *Biosens. Bioelectronics* **178**, 113028 (2021).
21. Racovita, A. & Jaramillo, A. Reinforcement learning in synthetic gene circuits. *Biochem. Soc. Trans.* **48**, 1637–1643 (2020).
22. Gazut, S., Martinez, J.-M., Dreyfus, G. & Oussar, Y. Towards the optimal design of numerical experiments. *IEEE Trans. Neural Netw.* **19**, 874–882 (2008).
23. Yu, K., Bi, J. & Tresp, V. Active learning via transductive experimental design. in *Proceedings of the 23rd international conference on Machine learning*, 1081–1088 (2006).
24. Olsson, F. A literature survey of active machine learning in the context of natural language processing. DiVa [diva2:1042586] (2009).
25. Sommer, C. & Gerlich, D. W. Machine learning in cell biology - teaching computers to recognize phenotypes. *J. Cell Sci.* **126**, 5529–5539 (2013).
26. Jones, T. R. et al. Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning. *Proc. Natl Acad. Sci. USA* **106**, 1826–1831 (2009).
27. Pournara, I. & Wernisch, L. Reconstruction of gene networks using Bayesian learning and manipulation experiments. *Bioinformatics* **20**, 2934–2942 (2004).
28. Naik, A. W., Kangas, J. D., Sullivan, D. P. & Murphy, R. F. Active machine learning-driven experimentation to determine compound effects on protein patterns. *Elife* **5**, e10047 (2016).
29. Reker, D. & Schneider, G. Active-learning strategies in computer-assisted drug discovery. *Drug Discov. Today* **20**, 458–465 (2015).
30. Osmanbeyoglu, H. U., Wehner, J. A., Carbonell, J. G. & Ganapathiraju, M. K. Active machine learning for transmembrane helix prediction. *BMC Bioinformatics* **11.1**, 1.9 (2010).
31. Borkowski, O. et al. Large scale active-learning-guided exploration for in vitro protein production optimization. *Nat. Commun.* **11**, 1–8 (2020).
32. Google Colaboratory. <https://colab.research.google.com/>.
33. Sun, Z. Z. et al. Protocols for implementing an *Escherichia coli* based TX-TL cell-free expression system for synthetic biology. *JoVE* **79**, e50762 (2013).
34. Pandi, A., Grigoras, I., Borkowski, O. & Faulon, J.-L. Optimizing cell-free biosensors to monitor enzymatic production. *ACS Synth. Biol.* **8**, 1952–1957 (2019).
35. Karim, A. S. et al. In vitro prototyping and rapid optimization of biosynthetic enzymes for cell design. *Nat. Chem. Biol.* **16**, 912–919 (2020).
36. Pandi, A. et al. Metabolic perceptrons for neural computing in biological systems. *Nat. Commun.* **10**, 3880 (2019).
37. Swank, Z., Laohakunakorn, N. & Maerkl, S. J. Cell-free gene-regulatory network engineering with synthetic transcription factors. *Proc. Natl Acad. Sci. USA* **116**, 5892–5901 (2019).
38. Greco, F. V., Pandi, A., Erb, T. J., Grierson, C. S. & Gorochowski, T. E. Harnessing the central dogma for stringent multi-level control of gene expression. *Nat. Commun.* **12**, 1738 (2021).
39. Schwander, T., Schada von Borzyskowski, L., Burgener, S., Cortina, N. S. & Erb, T. J. A synthetic pathway for the fixation of carbon dioxide in vitro. *Science* **354**, 900–904 (2016).
40. Najafabadi, M. M. et al. Deep learning applications and challenges in big data analytics. *J. Big Data* **2**, 1–21 (2015).
41. Chen, T. & Guestrin, C. XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794 (2016).
42. Li, W., Yin, Y., Quan, X. & Zhang, H. Gene expression value prediction based on XGBoost algorithm. *Front. Genet.* **10**, 1077 (2019).
43. Yu, B. et al. SubMito-XGBoost: predicting protein submitochondrial localization by fusing multiple feature information and eXtreme gradient boosting. *Bioinformatics* **36**, 1074–1081 (2019).
44. Borkowski, O. et al. Cell-free prediction of protein expression costs for growing cells. *Nat. Commun.* **9**, 1457 (2018).
45. Reyes, S. G., Kuruma, Y. & Tsuda, S. Uncovering cell-free protein expression dynamics by a promoter library with diverse strengths. <https://doi.org/10.1101/214593> (2017).
46. Ribosome Binding Sites/Prokaryotic/Constitutive/Community Collection. [https://parts.igem.org/Ribosome\\_Binding\\_Sites/Prokaryotic/Constitutive/Community\\_Collection](https://parts.igem.org/Ribosome_Binding_Sites/Prokaryotic/Constitutive/Community_Collection).
47. Verma, M. et al. A short translational ramp determines the efficiency of protein synthesis. *Nat. Commun.* **10**, 1–15 (2019).
48. Weber, M. et al. Impact of C-terminal amino acid composition on protein expression in bacteria. *Mol. Syst. Biol.* **16**, e9208 (2020).
49. Yim, S. S., Johns, N. I., Noireaux, V. & Wang, H. H. Protecting linear DNA templates in cell-free expression systems from diverse bacteria. *ACS Synth. Biol.* **9**, 2851–2855 (2020).
50. Murphy, K. C. Lambda Gam protein inhibits the helicase and chi-stimulated recombination activities of *Escherichia coli* RecBCD enzyme. *J. Bacteriol.* **173**, 5808–5821 (1991).
51. Erb, T. J., Jones, P. R. & Bar-Even, A. Synthetic metabolism: metabolic engineering meets enzyme design. *Curr. Opin. Chem. Biol.* **37**, 56–62 (2017).
52. Bowie, J. U. et al. Synthetic biochemistry: the bio-inspired cell-free approach to commodity chemical production. *Trends Biotechnol.* **38**, 766–778 (2020).
53. Miller, T. E. et al. Light-powered CO<sub>2</sub> fixation in a chloroplast mimic with natural and synthetic parts. *Science* **368**, 649–654 (2020).
54. Burgener, S., Schwander, T., Romero, E., Fraaije, M. W. & Erb, T. J. Molecular basis for converting (2S)-methylsuccinyl-CoA dehydrogenase into an oxidase. *Molecules* **23**, 68 (2017).
55. Archetti, F. & Candelieri, A. *Bayesian Optimization and Data Science*. (Springer Nature, 2019).
56. Matsuura, T., Kazuta, Y., Aita, T., Adachi, J. & Yomo, T. Quantifying epistatic interactions among the components constituting the protein translation system. *Mol. Syst. Biol.* **5**, 297 (2009).
57. James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning: with Applications in R*. (Springer Science & Business Media, 2013).
58. Voyvodic, P. L. et al. Plug-and-play metabolic transducers expand the chemical detection space of cell-free biosensors. *Nat. Commun.* **10**, 1–8 (2019).
59. Temme, K., Hill, R., Segall-Shapiro, T. H., Moser, F. & Voigt, C. A. Modular control of multiple pathways using engineered orthogonal T7 polymerases. *Nucleic Acids Res* **40**, 8773–8781 (2012).
60. Martínez-García, E. et al. SEVA 3.0: an update of the Standard European Vector Architecture for enabling portability of genetic constructs among diverse bacterial hosts. *Nucleic Acids Res* **48**, 3395 (2020).
61. Standard European Vector Architecture (SEVA). <http://seva-plasmids.com/>.
62. Sundaram, S. et al. A modular in vitro platform for the production of terpenes and polyketides from CO. *Angew. Chem. Int. Ed. Engl.* **60**, 16420–16425 (2021).
63. Didovik, A., Tonooka, T., Tsimring, L. & Hasty, J. Rapid and scalable preparation of bacterial lysates for cell-free gene expression. *ACS Synth. Biol.* **6**, 2198–2208 (2017).
64. Kushwaha, M. & Salis, H. M. A portable expression resource for engineering cross-species genetic circuits and pathways. *Nat. Commun.* **6**, 7832 (2015).

## Acknowledgements

We wish to thank V. Gureghian for stimulating discussions during conception of the work, T. E. Gorochowski (University of Bristol, UK) for providing SLC and MLC constructs, and M. Kushwaha (INRAE, Jouy en Josas, France) for *E. coli* DH10 $\beta$ , harboring the auto-regulated T7 polymerase construct. We thank S. Burgener, S. Luo, A. Sánchez-Pascuala Jerez, N. Odermatt, and A. Schrodtr for graphical, technical and experimental support, as well as fruitful discussion. We thank Jule P. M. Erb for drafting the METIS logo. This work was supported by a European Molecular Biology Organization (EMBO) long-term post-doctoral fellowship (A.P. ALTG 165-2020), the Gordon and Betty Moore Foundation, GBMF10652, grant DOI 10.37807/GBMF10652 (T.J.E.), the Max Planck Research Network in Synthetic Biology (MaxSynBio) of the Max Planck Society and the Federal Ministry of Education and Research (BMBF; T.J.E.), the UDOPIA PhD program and the French National Research Institute for Agriculture, Food, and Environment (INRAE) through the MétaProgramme BIOLPREDICT program (L.F.), BMBF Grant, MetAFor, No. 031B0850B (T.J.E.), ANR iCFree grant (ANR-20-BiopNSE) (J.-L.F.) and the Max-Planck Society (MPG-FhG project eBioCO<sub>2</sub>n). Figures were created with Biorender.com.

## Author contributions

T.J.E. and A.P. conceived the work. A.P. and C.D. designed the workflow and performed experiments. A.Y.K. wrote all Python notebooks and simulations. L.F. and J.-L.F. reviewed the workflow's code and consistency and provided constructive inputs. S.A.S. performed western blot and RT-qPCR experiments. E.B. implemented the DART algorithm and SMOGN data preprocessing. M.N. prepared the enzyme mutants dataset. D.A., N.C., Y.F., and C.M. assisted with the experimental work. N.P. developed the LC-MS methods for the analysis of glycolate and CoA esters. N.S.C. wrote the code for the

conversion of the ECHO® exceptions lists into new worklists. T.J.E. supervised the work. A.P., C.D., A.Y.K. and T.J.E. wrote the manuscript with input from all other authors.

### Funding

Open Access funding enabled and organized by Projekt DEAL.

### Competing interests

The authors declare no competing interests.

### Additional information

**Supplementary information** The online version contains supplementary material available at <https://doi.org/10.1038/s41467-022-31245-z>.

**Correspondence** and requests for materials should be addressed to Amir Pandi or Tobias J. Erb.

**Peer review information** *Nature Communications* thanks the anonymous reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available.

**Reprints and permission information** is available at <http://www.nature.com/reprints>

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2022

# Bibliography

1. Papin, J. A., Price, N. D., Wiback, S. J., Fell, D. A. & Palsson, B. O. Metabolic pathways in the post-genome era. *Trends Biochem. Sci.* **28**, 250–258 (May 2003).
2. Lazar, M. A. & Birnbaum, M. J. Physiology. De-meaning of metabolism. *Science* **336**, 1651–1652 (June 2012).
3. Bich, L., Mossio, M., Ruiz-Mirazo, K. & Moreno, A. Biological regulation: controlling the system from within. *Biol. Philos.* **31**, 237–265 (Mar. 2016).
4. Gjuvsland, A. B., Vik, J. O., Beard, D. A., Hunter, P. J. & Omholt, S. W. Bridging the genotype-phenotype gap: what does it take? *J. Physiol.* **591**, 2055–2066 (Apr. 2013).
5. Riley, M. *et al.* Escherichia coli K-12: a cooperatively developed annotation snapshot-2005. *Nucleic Acids Res.* **34**, 1–9 (Jan. 2006).
6. Jensen, K. F. The Escherichia coli K-12 “wild types” W3110 and MG1655 have an rph frameshift mutation that leads to pyrimidine starvation due to low pyrE expression levels. *J. Bacteriol.* **175**, 3401–3407 (June 1993).
7. Grant, S. G., Jessee, J., Bloom, F. R. & Hanahan, D. Differential plasmid rescue from transgenic mouse DNAs into Escherichia coli methylation-restriction mutants. *Proc. Natl. Acad. Sci. U. S. A.* **87**, 4645–4649 (June 1990).
8. Struhl, K. Fundamentally different logic of gene regulation in eukaryotes and prokaryotes. *Cell* **98**, 1–4 (July 1999).
9. Liu, J. & Nussinov, R. Allostery: An Overview of Its History, Concepts, Methods, and Applications. *PLoS Comput. Biol.* **12**, e1004966 (June 2016).
10. Hoppe, A. What mRNA Abundances Can Tell us about Metabolism. *Metabolites* **2**, 614–631 (Sept. 2012).
11. Chubukov, V. *et al.* Transcriptional regulation is insufficient to explain substrate-induced flux changes in Bacillus subtilis. *Mol. Syst. Biol.* **9**, 709 (Nov. 2013).
12. Moreno-Sánchez, R., Saavedra, E., Rodríguez-Enríquez, S. & Olín-Sandoval, V. Metabolic control analysis: a tool for designing strategies to manipulate metabolic pathways. *J. Biomed. Biotechnol.* **2008**, 597913 (2008).
13. Monod, J., Wyman, J. & Changeux, J. P. ON THE NATURE OF ALLOSTERIC TRANSITIONS: A PLAUSIBLE MODEL. *J. Mol. Biol.* **12**, 88–118 (May 1965).

14. Koshland Jr, D. E., Némethy, G. & Filmer, D. Comparison of experimental binding data and theoretical models in proteins containing subunits. *Biochemistry* **5**, 365–385 (Jan. 1966).
15. Macek, B. *et al.* Protein post-translational modifications in bacteria. *Nat. Rev. Microbiol.* **17**, 651–664 (Nov. 2019).
16. Schastnaya, E. *et al.* Extensive regulation of enzyme activity by phosphorylation in *Escherichia coli*. *Nat. Commun.* **12**, 5650 (Sept. 2021).
17. Jacob, F. & Monod, J. Genetic regulatory mechanisms in the synthesis of proteins. *J. Mol. Biol.* **3**, 318–356 (June 1961).
18. Malan, T. P., Kolb, A., Buc, H. & McClure, W. R. Mechanism of CRP-cAMP activation of lac operon transcription initiation activation of the P<sub>1</sub> promoter. *J. Mol. Biol.* **180**, 881–909 (Dec. 1984).
19. Görke, B. & Stülke, J. Carbon catabolite repression in bacteria: many ways to make the most out of nutrients. *Nat. Rev. Microbiol.* **6**, 613–624 (Aug. 2008).
20. Tierrafría, V. H. *et al.* RegulonDB 11.0: Comprehensive high-throughput datasets on transcriptional regulation in *Escherichia coli* K-12. *Microb Genom* **8** (May 2022).
21. Stock, A. M., Robinson, V. L. & Goudreau, P. N. Two-component signal transduction. *Annu. Rev. Biochem.* **69**, 183–215 (2000).
22. Miller, M. B. & Bassler, B. L. Quorum sensing in bacteria. *Annu. Rev. Microbiol.* **55**, 165–199 (2001).
23. Shimizu, K. Metabolic Regulation of a Bacterial Cell System with Emphasis on *Escherichia coli* Metabolism. *ISRN Biochem* **2013**, 645983 (Feb. 2013).
24. Haverkorn van Rijsewijk, B. R. B., Nanchen, A., Nallet, S., Kleijn, R. J. & Sauer, U. Large-scale <sup>13</sup>C-flux analysis reveals distinct transcriptional control of respiratory and fermentative metabolism in *Escherichia coli*. *Mol. Syst. Biol.* **7**, 477 (Mar. 2011).
25. Mitchell, M. *Complexity: A guided tour* (Oxford University Press, Inc., 2009).
26. Matsuoka, Y. & Shimizu, K. Metabolic regulation in *Escherichia coli* in response to culture environments via global regulators. *Biotechnol. J.* **6**, 1330–1341 (Nov. 2011).
27. Karlebach, G. & Shamir, R. Modelling and analysis of gene regulatory networks. *Nat. Rev. Mol. Cell Biol.* **9**, 770–780 (Oct. 2008).
28. Covert, M. W., Knight, E. M., Reed, J. L., Herrgard, M. J. & Palsson, B. O. Integrating high-throughput and computational data elucidates bacterial networks. *Nature* **429**, 92–96 (May 2004).
29. Shlomi, T., Eisenberg, Y., Sharan, R. & Ruppin, E. A genome-scale computational study of the interplay between transcriptional regulation and metabolism. *Mol. Syst. Biol.* **3**, 101 (Apr. 2007).
30. Covert, M. W., Xiao, N., Chen, T. J. & Karr, J. R. Integrating metabolic, transcriptional regulatory and signal transduction models in *Escherichia coli*. *Bioinformatics* **24**, 2044–2050 (Sept. 2008).

31. Heckmann, C. M. & Paradisi, F. Looking Back: A Short History of the Discovery of Enzymes and How They Became Powerful Chemical Tools. *ChemCatChem* **12**, 6082–6102 (Dec. 2020).
32. Cohen, S. N., Chang, A. C., Boyer, H. W. & Helling, R. B. Construction of biologically functional bacterial plasmids in vitro. *Proc. Natl. Acad. Sci. U. S. A.* **70**, 3240–3244 (Nov. 1973).
33. Woolston, B. M., Edgar, S. & Stephanopoulos, G. Metabolic engineering: past and future. en. *Annu. Rev. Chem. Biomol. Eng.* **4**, 259–288 (Mar. 2013).
34. Stephanopoulos, G., Aristidou, A. A. & Nielsen, J. *Metabolic Engineering: Principles and Methodologies* en (Elsevier, Oct. 1998).
35. Khalil, A. S. & Collins, J. J. Synthetic biology: applications come of age. *Nat. Rev. Genet.* **11**, 367–379 (May 2010).
36. Wiener, N. *Cybernetics Or Control and Communication in the Animal and the Machine* en (MIT Press, 1961).
37. Kitano, H. Systems biology: a brief overview. *Science* **295**, 1662–1664 (Mar. 2002).
38. Barabási, A.-L. & Oltvai, Z. N. Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.* **5**, 101–113 (Feb. 2004).
39. Noda-Garcia, L., Liebermeister, W. & Tawfik, D. S. Metabolite-Enzyme Coevolution: From Single Enzymes to Metabolic Pathways and Networks. *Annu. Rev. Biochem.* **87**, 187–216 (June 2018).
40. Grozinger, L. *et al.* Pathways to cellular supremacy in biocomputing. *Nat. Commun.* **10**, 5250 (Nov. 2019).
41. Pandi, A. *et al.* Metabolic perceptrons for neural computing in biological systems. *Nat. Commun.* **10**, 3880 (Aug. 2019).
42. Adamatzky, A. *Physarum Machines: Computers from Slime Mould* en (World Scientific, 2010).
43. Scheres, B. & van der Putten, W. H. The plant perceptron connects environment to development. *Nature* **543**, 337–345 (Mar. 2017).
44. Monk, J. M. *et al.* iML1515, a knowledgebase that computes Escherichia coli traits. *Nat. Biotechnol.* **35**, 904–908 (Oct. 2017).
45. Ramon, C., Gollub, M. G. & Stelling, J. Integrating -omics data into genome-scale metabolic network models: principles and challenges. *Essays Biochem.* **62**, 563–574 (Oct. 2018).
46. Hucka, M. *et al.* The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19**, 524–531 (Mar. 2003).
47. Courtot, M. *et al.* Controlled vocabularies and semantics in systems biology. *Mol. Syst. Biol.* **7**, 543 (Oct. 2011).



48. Klipp, E., Liebermeister, W., Wierling, C. & Kowald, A. *Systems Biology: A Textbook* en (John Wiley & Sons, Mar. 2016).
49. Sun, G., Ahn-Horst, T. A. & Covert, M. W. The E. coli Whole-Cell Modeling Project. *EcoSal Plus* **9**, eESP00012020 (Dec. 2021).
50. Choudhury, S. *et al.* Reconstructing Kinetic Models for Dynamical Studies of Metabolism using Generative Adversarial Networks. en. *Nature Machine Intelligence* **4**, 710–719 (Aug. 2022).
51. Moulin, C., Tournier, L. & Peres, S. Combining Kinetic and Constraint-Based Modelling to Better Understand Metabolism Dynamics. en. *Processes* **9**, 1701 (Sept. 2021).
52. Sahu, A., Blätke, M.-A., Szymański, J. J. & Töpfer, N. Advances in flux balance analysis by integrating machine learning and mechanism-based models. *Comput. Struct. Biotechnol. J.* **19**, 4626–4640 (Aug. 2021).
53. Colarusso, A. V., Goodchild-Michelman, I., Rayle, M. & Zomorodi, A. R. Computational modeling of metabolism in microbial communities on a genome-scale. *Current Opinion in Systems Biology* **26**, 46–57 (June 2021).
54. Fang, X., Lloyd, C. J. & Palsson, B. O. Reconstructing organisms in silico: genome-scale models and their emerging applications. *Nat. Rev. Microbiol.* **18**, 731–743 (Dec. 2020).
55. O'Brien, E. J., Monk, J. M. & Palsson, B. O. Using Genome-scale Models to Predict Biological Capabilities. *Cell* **161**, 971–987 (May 2015).
56. Szélieová, D. *et al.* What CHO is made of: Variations in the biomass composition of Chinese hamster ovary cell lines. *Metab. Eng.* **61**, 288–300 (Sept. 2020).
57. Lachance, J.-C. *et al.* BOFdat: Generating biomass objective functions for genome-scale metabolic models from experimental data. *PLoS Comput. Biol.* **15**, e1006971 (Apr. 2019).
58. Antoniewicz, M. R. Methods and advances in metabolic flux analysis: a mini-review. *J. Ind. Microbiol. Biotechnol.* **42**, 317–325 (Mar. 2015).
59. Willemsen, A. M. *et al.* MetDFBA: incorporating time-resolved metabolomics measurements into dynamic flux balance analysis. *Mol. Biosyst.* **11**, 137–145 (Jan. 2015).
60. Alghamdi, N. *et al.* A graph neural network model to estimate cell-wise metabolic flux using single-cell RNA-seq data. *Genome Res.* **31**, 1867–1884 (Oct. 2021).
61. Selvarasu, S. *et al.* Characterizing Escherichia coli DH5alpha growth and metabolism in a complex medium using genome-scale flux analysis. *Biotechnol. Bioeng.* **102**, 923–934 (Feb. 2009).
62. Kauffman, K. J., Prakash, P. & Edwards, J. S. Advances in flux balance analysis. *Curr. Opin. Biotechnol.* **14**, 491–496 (Oct. 2003).
63. Karloff, H. in *Linear Programming* 23–47 (Birkhäuser Boston, Boston, MA, 2009).
64. Karmarkar, N. A new polynomial-time algorithm for linear programming. *Combinatorica* **4**, 373–395 (Dec. 1984).
65. Chandru, V. & Rao, M. Linear Programming (Mar. 1998).

66. Inuiguchi, M. & Ramík, J. Possibilistic linear programming: a brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem. *Fuzzy Sets and Systems* **111**, 3–28 (Apr. 2000).
67. Kotary, J., Fioretto, F., Van Hentenryck, P. & Wilder, B. End-to-End Constrained Optimization Learning: A Survey. *arXiv [cs.LG]* (Mar. 2021).
68. Ebrahim, A., Lerman, J. A., Palsson, B. O. & Hyduke, D. R. COBRApy: CONstraints-Based Reconstruction and Analysis for Python. *BMC Syst. Biol.* **7**, 74 (Aug. 2013).
69. Lewis, N. E. *et al.* Omic data from evolved E. coli are consistent with computed optimal growth from genome-scale models. *Mol. Syst. Biol.* **6**, 390 (July 2010).
70. Gudmundsson, S. & Thiele, I. Computationally efficient flux variability analysis. *BMC Bioinformatics* **11**, 489 (Sept. 2010).
71. Herrmann, H. A., Dyson, B. C., Vass, L., Johnson, G. N. & Schwartz, J.-M. Flux sampling is a powerful tool to study metabolism under changing environmental conditions. *NPJ Syst Biol Appl* **5**, 32 (Sept. 2019).
72. Mahadevan, R., Edwards, J. S. & Doyle 3rd, F. J. Dynamic flux balance analysis of diauxic growth in Escherichia coli. *Biophys. J.* **83**, 1331–1340 (Sept. 2002).
73. Goelzer, A. *et al.* Quantitative prediction of genome-wide resource allocation in bacteria. *Metab. Eng.* **32**, 232–243 (Nov. 2015).
74. Hoppe, A., Hoffmann, S. & Holzhütter, H.-G. Including metabolite concentrations into flux balance analysis: thermodynamic realizability as a constraint on flux distributions in metabolic networks. *BMC Syst. Biol.* **1**, 23 (June 2007).
75. Marmiesse, L., Peyraud, R. & Cottret, L. FlexFlux: combining metabolic flux and regulatory network analyses. *BMC Syst. Biol.* **9**, 93 (Dec. 2015).
76. Müller, S., Regensburger, G. & Steuer, R. Resource allocation in metabolic networks: kinetic optimization and approximations by FBA. *Biochem. Soc. Trans.* **43**, 1195–1200 (Dec. 2015).
77. Banos, D. T., Trébulle, P. & Elati, M. Integrating transcriptional activity in genome-scale models of metabolism. *BMC Syst. Biol.* **11**, 134 (Dec. 2017).
78. Amato, S. M. *et al.* The role of metabolism in bacterial persistence. *Front. Microbiol.* **5**, 70 (Mar. 2014).
79. Rather, M. A., Gupta, K. & Mandal, M. Microbial biofilm: formation, architecture, antibiotic resistance, and control strategies. *Braz. J. Microbiol.* **52**, 1701–1718 (Dec. 2021).
80. Berkhout, J., Bruggeman, F. J. & Teusink, B. Optimality principles in the regulation of metabolic networks. *Metabolites* **2**, 529–552 (Aug. 2012).
81. Schuetz, R., Zamboni, N., Zampieri, M., Heinemann, M. & Sauer, U. Multidimensional optimality of microbial metabolism. *Science* **336**, 601–604 (May 2012).
82. Bordbar, A., Monk, J. M., King, Z. A. & Palsson, B. O. Constraint-based models predict metabolic and associated cellular functions. *Nat. Rev. Genet.* **15**, 107–120 (Feb. 2014).

83. Chang, R. L., Xie, L., Bourne, P. E. & Palsson, B. O. Antibacterial mechanisms identified through structural systems pharmacology. *BMC Syst. Biol.* **7**, 102 (Oct. 2013).
84. Chung, H. *et al.* Bio-based production of monomers and polymers by metabolically engineered microorganisms. *Curr. Opin. Biotechnol.* **36**, 73–84 (Dec. 2015).
85. Mackie, A. M., Hassan, K. A., Paulsen, I. T. & Tetu, S. G. in *Environmental Microbiology: Methods and Protocols* (eds Paulsen, I. T. & Holmes, A. J.) 123–130 (Humana Press, Totowa, NJ, 2014).
86. Box, G. E. P. Science and Statistics. *J. Am. Stat. Assoc.* **71**, 791–799 (Dec. 1976).
87. Phair, R. D. Mechanistic modeling confronts the complexity of molecular cell biology. *Mol. Biol. Cell* **25**, 3494–3496 (Nov. 2014).
88. Baker, R. E., Peña, J.-M., Jayamohan, J. & Jérusalem, A. Mechanistic models versus machine learning, a fight worth fighting for the biological community? *Biol. Lett.* **14** (May 2018).
89. Alpaydin, E. *Introduction to Machine Learning, fourth edition* en (MIT Press, Mar. 2020).
90. Bi, Q., Goodman, K. E., Kaminsky, J. & Lessler, J. What is Machine Learning? A Primer for the Epidemiologist. *Am. J. Epidemiol.* **188**, 2222–2239 (Dec. 2019).
91. Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **117**, 500–544 (Aug. 1952).
92. Michaelis, L., Menten, M. L., Johnson, K. A. & Goody, R. S. The original Michaelis constant: translation of the 1913 Michaelis-Menten paper. *Biochemistry* **50**, 8264–8269 (Oct. 2011).
93. Panjkovich, A. & Melo, F. Comparison of different melting temperature calculation methods for short DNA sequences. *Bioinformatics* **21**, 711–722 (Mar. 2005).
94. Chelliah, V. *et al.* BioModels: ten-year anniversary. *Nucleic Acids Res.* **43**, D542–8 (Jan. 2015).
95. Subramanian, I., Verma, S., Kumar, S., Jere, A. & Anamika, K. Multi-omics Data Integration, Interpretation, and Its Application. *Bioinform. Biol. Insights* **14**, 1177932219899051 (Jan. 2020).
96. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (Aug. 2021).
97. Vázquez-Ingelmo, A. & García-Peñalvo, F. J. What do we mean by GenAI? A systematic mapping of the evolution, trends, and techniques involved in generative AI. *Int. J. Interact. Multimed. Artif. Intell.* (Aug. 2023).
98. Brynjolfsson, E. & Mitchell, T. What can machine learning do? Workforce implications. *Science* **358**, 1530–1534 (Dec. 2017).
99. Bellman, R. *Dynamic Programming* en (Princeton University Press, 1957).
100. Schmidhuber, J. Annotated History of Modern AI and Deep Learning. *arXiv [cs.NE]* (Dec. 2022).

101. Maier, A., Syben, C., Lasser, T. & Riess, C. A gentle introduction to deep learning in medical image processing. *Z. Med. Phys.* **29**, 86–101 (May 2019).
102. Ruder, S. An overview of gradient descent optimization algorithms. arXiv: [1609.04747](https://arxiv.org/abs/1609.04747) [cs.LG] (Sept. 2016).
103. Jin, L., Li, S., Hu, B. & Liu, M. A survey on projection neural networks and their applications. *Appl. Soft Comput.* **76**, 533–544 (Mar. 2019).
104. Hopfield, J. J. & Tank, D. W. “Neural” computation of decisions in optimization problems. *Biol. Cybern.* **52**, 141–152 (1985).
105. Tanaka, G. *et al.* Recent advances in physical reservoir computing: A review. *Neural Netw.* **115**, 100–123 (July 2019).
106. Gauthier, D. J., Bollt, E., Griffith, A. & Barbosa, W. A. S. Next generation reservoir computing. *Nat. Commun.* **12**, 5564 (Sept. 2021).
107. Zampieri, G., Vijayakumar, S., Yaneske, E. & Angione, C. Machine and deep learning meet genome-scale metabolic modeling. *PLoS Comput. Biol.* **15**, e1007084 (July 2019).
108. Sen, P. & Orešič, M. Integrating Omics Data in Genome-Scale Metabolic Modeling: A Methodological Perspective for Precision Medicine. *Metabolites* **13** (July 2023).
109. Khaleghi, M. K., Savizi, I. S. P., Lewis, N. E. & Shojaosadati, S. A. Synergisms of machine learning and constraint-based modeling of metabolism for analysis and optimization of fermentation parameters. *Biotechnol. J.* **16**, e2100212 (Nov. 2021).
110. Antonakoudis, A., Barbosa, R., Kotidis, P. & Kontoravdi, C. The era of big data: Genome-scale modelling meets machine learning. *Comput. Struct. Biotechnol. J.* **18**, 3287–3300 (Oct. 2020).
111. Rana, P., Berry, C., Ghosh, P. & Fong, S. S. Recent advances on constraint-based models by integrating machine learning. *Curr. Opin. Biotechnol.* **64**, 85–91 (Aug. 2020).
112. Procopio, A. *et al.* Combined mechanistic modeling and machine-learning approaches in systems biology - A systematic literature review. *Comput. Methods Programs Biomed.* **240**, 107681 (June 2023).
113. Zhang, J. *et al.* Combining mechanistic and machine learning models for predictive engineering and optimization of tryptophan metabolism. *Nat. Commun.* **11**, 4880 (Sept. 2020).
114. Heckmann, D. *et al.* Machine learning applied to enzyme turnover numbers reveals protein structural correlates and improves metabolic models. *Nat. Commun.* **9**, 5252 (Dec. 2018).
115. Wu, S. G. *et al.* Rapid Prediction of Bacterial Heterotrophic Fluxomics Using Machine Learning and Constraint Programming. *PLoS Comput. Biol.* **12**, e1004838 (Apr. 2016).
116. Yang, J. H. *et al.* A White-Box Machine Learning Approach for Revealing Antibiotic Mechanisms of Action. *Cell* **177**, 1649–1661.e9 (May 2019).
117. Szappanos, B. *et al.* An integrated approach to characterize genetic interaction networks in yeast metabolism. *Nat. Genet.* **43**, 656–662 (May 2011).

118. Cuomo, S. *et al.* Scientific Machine Learning Through Physics-Informed Neural Networks: Where we are and What's Next. *J. Sci. Comput.* **92**, 88 (July 2022).
119. *SciML: Open Source Software for Scientific Machine Learning* <https://sciml.ai/>. Accessed: 2022-10-5.
120. Lagergren, J. H., Nardini, J. T., Baker, R. E., Simpson, M. J. & Flores, K. B. Biologically-informed neural networks guide mechanistic modeling from sparse experimental data. *PLoS Comput. Biol.* **16**, e1008462 (Dec. 2020).
121. Fortelny, N. & Bock, C. Knowledge-primed neural networks enable biologically interpretable deep learning on single-cell sequencing data. *Genome Biol.* **21**, 190 (Aug. 2020).
122. Nilsson, A., Peters, J. M., Meimetis, N., Bryson, B. & Lauffenburger, D. A. Artificial neural networks enable genome-scale simulations of intracellular signaling. *Nat. Commun.* **13**, 3069 (June 2022).
123. Hasibi, R., Michoel, T. & Oyarzun, D. A. Integration of graph neural networks and genome-scale metabolic models for predicting gene essentiality. *bioRxiv*, 2023.08.25.554757 (Aug. 2023).
124. Freischem, L. J., Barahona, M. & Oyarzún, D. A. Prediction of gene essentiality using machine learning and genome-scale metabolic models. *bioRxiv*, 2022.03.31.486520 (Mar. 2022).
125. Gherman, I. M. *et al.* Bridging the gap between mechanistic biological models and machine learning surrogates. *PLoS Comput. Biol.* **19**, e1010988 (Apr. 2023).
126. Thornburg, Z. R. *et al.* Fundamental behaviors emerge from simulations of a living minimal cell. *Cell* **185**, 345–360.e28 (Jan. 2022).
127. Reed, J. L. & Palsson, B. Ø. Thirteen years of building constraint-based in silico models of *Escherichia coli*. *J. Bacteriol.* **185**, 2692–2699 (May 2003).
128. Plaimas, K. *et al.* Machine learning based analyses on metabolic networks supports high-throughput knockout screens. *BMC Syst. Biol.* **2**, 67 (July 2008).
129. Schinn, S.-M., Morrison, C., Wei, W., Zhang, L. & Lewis, N. E. A genome-scale metabolic network model and machine learning predict amino acid concentrations in Chinese Hamster Ovary cell cultures. *Biotechnol. Bioeng.* **118**, 2118–2123 (May 2021).
130. Kim, M., Rai, N., Zorraquino, V. & Tagkopoulos, I. Multi-omics integration accurately predicts cellular state in unexplored conditions for *Escherichia coli*. *Nat. Commun.* **7**, 13090 (Oct. 2016).
131. Lewis, J. E. & Kemp, M. L. Integration of machine learning and genome-scale metabolic modeling identifies multi-omics biomarkers for radiation resistance. *Nat. Commun.* **12**, 2700 (May 2021).
132. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (Feb. 2019).

133. Yang, Y., Cao, J., Xu, X., Hu, M. & Gao, Y. A new neural network for solving quadratic programming problems with equality and inequality constraints. *Math. Comput. Simul.* **101**, 103–112 (July 2014).
134. Varma, A. & Palsson, B. O. Metabolic capabilities of *Escherichia coli*: I. synthesis of biosynthetic precursors and cofactors. *J. Theor. Biol.* **165**, 477–502 (Dec. 1993).
135. Orth, J. D., Fleming, R. M. T. & Palsson, B. Ø. Reconstruction and Use of Microbial Metabolic Networks: the Core *Escherichia coli* Metabolic Model as an Educational Guide. *EcoSal Plus* **4** (Sept. 2010).
136. Norsigian, C. J. *et al.* BiGG Models 2020: multi-strain genome-scale models and expansion across the phylogenetic tree. *Nucleic Acids Res.* **48**, D402–D406 (Jan. 2020).
137. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research* **12**, 2825–2830 (2011).
138. Glasner, J. D. *et al.* ASAP, a systematic annotation package for community analysis of genomes. *Nucleic Acids Res.* **31**, 147–151 (Jan. 2003).
139. Chen, T. & Guestrin, C. XGBoost: A Scalable Tree Boosting System. arXiv: [1603.02754](https://arxiv.org/abs/1603.02754) [cs.LG] (Mar. 2016).
140. Orth, J. D. *et al.* A comprehensive genome-scale reconstruction of *Escherichia coli* metabolism–2011. *Mol. Syst. Biol.* **7**, 535 (Oct. 2011).
141. Nogales, J. *et al.* High-quality genome-scale metabolic modelling of *Pseudomonas putida* highlights its broad metabolic capabilities. *Environ. Microbiol.* **22**, 255–269 (Jan. 2020).
142. Beg, Q. K. *et al.* Intracellular crowding defines the mode and sequence of substrate uptake by *Escherichia coli* and constrains its metabolic activity. *Proc. Natl. Acad. Sci. U. S. A.* **104**, 12663–12668 (July 2007).
143. Niefenführ, S., Wiechert, W. & Nöh, K. How to measure metabolic fluxes: a taxonomic guide for <sup>13</sup>C fluxomics. *Curr. Opin. Biotechnol.* **34**, 82–90 (Aug. 2015).
144. Chollet, F. *et al.* Keras <https://keras.io>. 2015.
145. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585**, 357–362 (Sept. 2020).
146. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* **17**, 261–272 (Mar. 2020).
147. McKinney, W. *Data Structures for Statistical Computing in Python* in *Proceedings of the 9th Python in Science Conference* (eds van der Walt, S. & Millman, J.) (SciPy, Austin, Texas, 2010).
148. Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv: [1603.04467](https://arxiv.org/abs/1603.04467) [cs.DC] (Mar. 2016).
149. Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **9**, 90–95 (May 2007).
150. Waskom, M. seaborn: statistical data visualization. *J. Open Source Softw.* **6**, 3021 (Apr. 2021).

151. Houle, M. E., Kriegel, H.-P., Kröger, P., Schubert, E. & Zimek, A. *Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?* in *Scientific and Statistical Database Management* (Springer Berlin Heidelberg, 2010), 482–500.
152. Marler, R. T. & Arora, J. S. The weighted sum method for multi-objective optimization: new insights. *Struct. Multidiscip. Optim.* **41**, 853–862 (June 2010).
153. Sener, O. & Koltun, V. Multi-Task Learning as Multi-Objective Optimization. arXiv: [1810.04650 \[cs.LG\]](#) (Oct. 2018).
154. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. *Optuna: A Next-generation Hyperparameter Optimization Framework* in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, Anchorage, AK, USA, July 2019), 2623–2631.
155. Bergstra, J., Bardenet, R., Bengio, Y. & Kégl, B. Algorithms for hyper-parameter optimization. *Adv. Neural Inf. Process. Syst.* **24** (2011).
156. Sadowski, M. I., Grant, C. & Fell, T. S. Harnessing QbD, Programming Languages, and Automation for Reproducible Biology. *Trends Biotechnol.* **34**, 214–227 (Mar. 2016).
157. Singh, D. & Singh, B. Investigating the impact of data normalization on classification performance. *Appl. Soft Comput.* **97**, 105524 (Dec. 2020).
158. Detassis, F., Lombardi, M. & Milano, M. Teaching the Old Dog New Tricks: Supervised Learning with Constraints. arXiv: [2002.10766 \[cs.LG\]](#) (Feb. 2020).
159. Dehkordi, M. B., Pishvaie, M. R. & Vafa, E. Model predictive control of a fermenter using dynamic flux balance analysis coupled with convolutional neural networks. *Comput. Chem. Eng.* **179**, 108444 (Nov. 2023).
160. Boer, M. D. *et al.* Improving genome-scale metabolic models of incomplete genomes with deep learning. *bioRxiv*, 2023.07.10.548314 (July 2023).
161. Chen, C., Liao, C. & Liu, Y.-Y. Teasing out missing reactions in genome-scale metabolic networks through hypergraph learning. *Nat. Commun.* **14**, 2375 (Apr. 2023).
162. Costello, Z. & Martin, H. G. A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data. *NPJ Syst Biol Appl* **4**, 19 (May 2018).
163. Pandi, A. *et al.* A versatile active learning workflow for optimization of genetic and metabolic networks. *Nat. Commun.* **13**, 3876 (July 2022).
164. Kryshchuk, A., Schwede, T., Topf, M., Fidelis, K. & Mout, J. Critical assessment of methods of protein structure prediction (CASP)-Round XIII. *Proteins* **87**, 1011–1020 (Dec. 2019).
165. McGovern, P. E. *et al.* Fermented beverages of pre- and proto-historic China. *Proc. Natl. Acad. Sci. U. S. A.* **101**, 17593–17598 (Dec. 2004).
166. Payen, A. *et al.* *Annales de chimie et de physique* fr (Masson., 1833).
167. Cech, T. R. RNA as an enzyme. *Sci. Am.* **255**, 64–75 (Nov. 1986).

168. Barnett, J. A. A history of research on yeasts 5: the fermentation pathway. *Yeast* **20**, 509–543 (Apr. 2003).
169. Blattner, F. R. *et al.* The complete genome sequence of *Escherichia coli* K-12. *Science* **277**, 1453–1462 (Sept. 1997).
170. Barrett, A. J. Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB). Enzyme nomenclature. Recommendations 1992. Supplement 2: corrections and additions (1994). *Eur. J. Biochem.* **232**, 1–6 (Aug. 1995).
171. Chang, A. *et al.* BRENDA, the ELIXIR core data resource in 2021: new developments and updates. *Nucleic Acids Res.* **49**, D498–D508 (Jan. 2021).
172. Karp, P. D. *et al.* The BioCyc collection of microbial genomes and metabolic pathways. *Brief. Bioinform.* **20**, 1085–1093 (July 2019).
173. Kanehisa, M. & Goto, S. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **28**, 27–30 (Jan. 2000).
174. Carbonell, P., Lecoindre, G. & Faulon, J.-L. Origins of specificity and promiscuity in metabolic networks. *J. Biol. Chem.* **286**, 43994–44004 (Dec. 2011).
175. Notebaart, R. A. *et al.* Network-level architecture and the evolutionary potential of underground metabolism. *Proc. Natl. Acad. Sci. U. S. A.* **111**, 11762–11767 (Aug. 2014).
176. Nobeli, I., Favia, A. D. & Thornton, J. M. Protein promiscuity and its implications for biotechnology. *Nat. Biotechnol.* **27**, 157–167 (Feb. 2009).
177. Crick, F. Central dogma of molecular biology. *en. Nature* **227**, 561–563 (Aug. 1970).
178. Tan, C. L. & Anderson, E. The New Central Dogma of Molecular Biology. *Resonance* **14**, 1–32 (2020).
179. Amin, S. A., Chavez, E., Porokhin, V., Nair, N. U. & Hassoun, S. Towards creating an extended metabolic model (EMM) for *E. coli* using enzyme promiscuity prediction and metabolomics data. *Microb. Cell Fact.* **18**, 109 (June 2019).
180. Kumar, V. S. & Maranas, C. D. GrowMatch: an automated method for reconciling in silico/in vivo growth predictions. *PLoS Comput. Biol.* **5**, e1000308 (Mar. 2009).
181. Price, N. D., Reed, J. L., Papin, J. A., Famili, I. & Palsson, B. O. Analysis of metabolic capabilities using singular value decomposition of extreme pathway matrices. *Biophys. J.* **84**, 794–804 (Feb. 2003).
182. Kell, D. B. & Oliver, S. G. Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era. *Bioessays* **26**, 99–105 (Jan. 2004).
183. Lopatkin, A. J. & Collins, J. J. Predictive biology: modelling, understanding and harnessing microbial complexity. *Nat. Rev. Microbiol.* **18**, 507–520 (Sept. 2020).



