Development of deep-learning approaches to discover metabolic interaction networks of environmental microbial communities from large metagenomic datasets, domain co-occurrence and protein coevolution

Laurent David

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Laurent DAVID

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Sujet de la thèse :

Development of deep-learning approaches to discover metabolic interaction networks of environmental microbial communities from large metagenomic datasets, domain co-occurrence and protein coevolution

soutenue le

devant le jury composé de :

Mme. Sophie SACQUIN-MORA	Rapporteur
M. Rayan CHIKHI	Rapporteur
M. Johannes SOEDING	Examinateur
M. Jean-Daniel ZUCKER	Examinateur
Mme. Alessandra CARBONE	Directeur de thèse
M. Hugues RICHARD	Co-encadrant de thèse

Contents

Résumé en français **vii**

1 Introduction **1**
1.1 The general context 1
1.2 Background Notions 2
1.2.1 Preliminaries 2
1.2.2 Sequence Assembly 6
1.2.3 Protein Partner Identification Problem 12

2 Targeted domain assembly for fast functional profiling of metagenomic datasets with S3A **25**
2.1 Context 26
2.2 The S3A approach 26
2.2.1 S3A key features 28
2.3 Methods 29
2.3.1 Domain hit 29
2.3.2 Data pre-processing 29
2.3.3 The S3A algorithm 29
2.3.4 Datasets 34
2.3.5 Evaluation procedure 35
2.3.6 Influence of thresholds to evaluate precision 36
2.3.7 Evaluation of running time 36
2.4 Results 36
2.4.1 S3A improves precision, recall and running times over other targeted assemblers 36
2.4.2 Gain over whole metagenome assembly 38
2.4.3 Time performance on real datasets and comparison with MG-RAST and Xander 40
2.4.4 Influence of default parameter on S3A's performance 40

3 IMPRINT: an augmented sample space of natural sequences and multiple convolutional filters identify protein partners **43**
3.1 Context . . . 43
3.2 Materials and Methods . . . 45
3.2.1 Dataset . . . 45
3.2.2 Design of the IMPRINT architecture . . . 45
3.2.3 Hyperparameters, initialisation parameters and training parameters . . . 49
3.2.4 Evolutionary informed data augmentation to construct a large training set 49
3.2.5 Data partition for training, validation and testing . . . 51
3.2.6 Validation of IMPRINT data augmentation schema . . . 51
3.2.7 Other CNN architectures used for comparison . . . 52
3.3 Results . . . 52
3.3.1 Evaluation on the Guo's yeast dataset . . . 52
3.3.2 Computational time . . . 53

4 Discussion and perspective **55**
4.1 S3A . . . 55
4.2 IMPRINT . . . 56
4.3 Perspective . . . 58
4.4 Work done . . . 58

5 Thanks **61**

Supplementary Material **63**

A S3A supplementary information **63**
A.1 Supplementary tables . . . 63

Bibliography **67**

List of acronyms

ANN	Artificial Neural Network
bp	Base Pair
CCM	Clade Centered Models
(C)NN	(Convolutional) Neural Network
DFS	Depth First Search
ELM	Eukaryotic Linear Motif
HMM	Hidden Markov Model
ip	Identity percentage
lmsl	Longest Matching Substring Length
NGS	Next Generation Sequencing
OLC	Overlap Layout Consensus
ORF	Open Reading Frame
PPI	Protein Protein Interaction
PR	Precision Recall
ReLU	Rectified Linear Unit
RPM	Random Projection Module
SLIM	Short Linear Interspaced Motif
T(F)P(N)	True (False) Positive (Negative)

Résumé en français

Introduction

L'analyse des séquences biologiques constitue l'un des domaines majeur de la bioinformatique. En particulier, la prédiction des interactions entre protéines est une étape importante, et joue un rôle clé dans la compréhension des environnements moléculaires existant au sein de la cellule.
La production des données brutes par le séquençage de nouvelle génération se fait en deux temps. A partir d'un échantillon biologique, une population de fragments d'ADN est d'abord extraite. Ces fragments sont ensuite séquencés à leurs extrémités pour produire des lectures. Le premier défi consiste à assembler ces lectures, soit par alignement à des génomes de référence, soit par leur assemblage génomique *de novo*. Cet assemblage/alignement est généralement suivi par une étape de prédiction des régions codantes et leur annotation fonctionnelle. Ceci permet de décrire les activités métaboliques existant dans l'organisme ou la communauté séquencé, ou d'analyser la fonction des protéines impliquées dans un processus cellulaire particulier. Au cours des deux dernières décennies, le séquençage de nouvelle génération a entraîné une augmentation rapide dans la production de données biologiques. Comprendre et interpréter cette énorme quantité de données nécessite des approches informatiques efficaces et précises permettant d'extraire les informations à partir des séquences brutes. Ces approches sont de plus en plus orientées vers l'apprentissage automatique : l'extraction d'informations à partir de grands ensembles de données nous permet d'améliorer notre compréhension des mécanismes biologiques. De nombreux travaux se sont intéressés à la prédiction de la fonction des protéines, ainsi qu'à leur interactions. Dans cette thèse, nous essayons de répondre en partie à ces questions.
Ce manuscrit est divisé en deux parties. La première partie se concentre sur le développement de S3A, un assembleur de domaines ciblés pour un profilage fonctionnel rapide des ensembles de données métagénomiques. Il vise à explorer rapidement le contenu de grands ensembles de données métagénomiques, en se basant sur un profilage fonctionnel. Dans la seconde partie, nous présentons un réseau de neurones profond, IMPRINT, dont le but est d'identifier des partenaires protéiques. Il s'appuie uniquement sur des informations de séquence et permet d'évaluer la probabilité d'interaction entre les deux protéines données en entrée.

S3A : Assemblage ciblé de domaines pour le profilage rapide des ensembles de données métagénomiques

Ce chapitre présente S3A, un assembleur ciblé sur les domaines protéiques conçu pour un profilage fonctionnel rapide. A partir de lectures issues de séquences métagénomiques, S3A reconstruit les régions codantes en s'appuyant sur plusieurs concepts-clés:

- Une annotation de domaine de haute qualité pour un assemblage efficace.
- La conception d'un indice de confiance pour une évaluation rapide des lectures chevauchantes.
- Un parcours optimisé du graphe de consensus Overlap-Layout (OLC).

Son implémentation est hautement générique et peut être utilisée quelque soit le type d'annotation. Pour des données simulées, S3A atteint un niveau de précision similaire à celui d'outils d'assemblage métagénomique classiques, tout en permettant de réaliser un profilage sur les domaines d'intérêt à la fois plus rapide et plus sensible. En se focalisant sur l'étude de quelques dizaines de domaines fonctionnels – un scénario typique - S3A est jusqu'à dix fois plus rapide que les outils métagénomiques classiques. En outre, bien que conçu initialement pour des séquences métagénomiques, S3A n'est pas limité à ce domaine d'application, et peut être appliqué à tout ensemble de séquences accompagné d'une annotation.

L'approche de S3A

S3A est un outil pour la recherche ciblée de domaines à partir d'ensembles de séquences métagénomiques. Les étapes de l'analyse avec S3A sont représentées sur la figure 1. A partir d'un ensemble de données de lectures métagénomiques, S3A effectue dans un premier temps un prétraitement permettant une réduction de la taille de l'ensemble d'entrée. Celui-ci consiste en la détection des cadres ouverts de lecture (ORF), leur assemblage en unitigs, puis la traduction des séquence nucléotidiques en séquences d'acides aminés. Nous appelons ces séquences d'acides aminés des *protigs*. Dans un deuxième temps, une annotation de domaines est réalisée sur les protigs et un graphe OLC (Overlap Layout Consensus) est construit en utilisant le chevauchement entre domaines. Un graphe OLC est un graphe orienté, dans lequel chaque noeud correspond à une séquence et chaque arête à un chevauchement entre deux séquences. Dans notre cas, le chevauchement est d'abord déduit de la correspondance de domaines puis raffiné à partir des séquences protéiques pour l'étape de parcours du graphe. Les chevauchements entre protigs dépendent de deux métriques particulières qui permettent d'évaluer la fiabilité des arêtes :

- Le pourcentage d'identité entre les deux séquences en question (appelée *ip*).

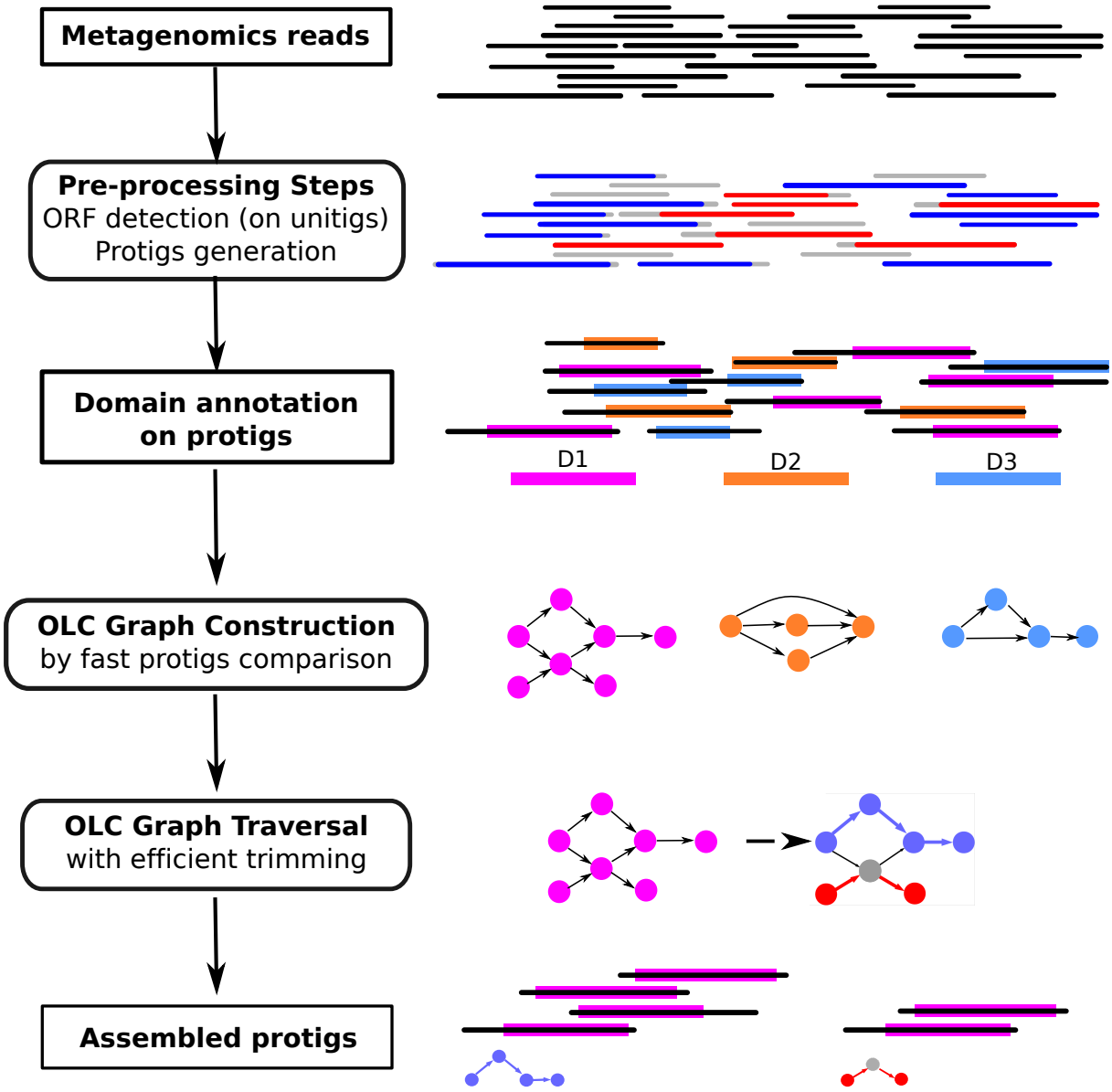

Figure 1: L'architecture de S3A

- La longueur de la plus longue sous-chaîne commune entre ces séquences (appelée *lmsl*).

Ces deux métriques permettent de filtrer efficacement les arêtes du graphe. Le parcours en profondeur du graphe OLC est ensuite l'étape permettant l'assemblage des protigs chevauchants en contigs. Ceux-ci représentent des régions consensus autour des domaines protéiques. Pour chaque domaine ciblé, S3A génère ainsi un ensemble de contigs.
La construction du graphe OLC dans S3A est différente de celle utilisée traditionnellement pour l'assemblage de séquences, où des alignements entre toutes les paires de lectures sont calculés pour en déduire les chevauchements. S3A exploite en premier l'annotation des domaines fonctionnels comme indicateur du chevauchement entre protigs, puis évalue la qualité des régions de chevauchement grâce à deux mesures rapidement calculables, *lmsl* et *ip*.

Caractéristiques principales de S3A

Une caractéristique importante de S3A est d'effectuer un assemblage se basant sur l'annotation de domaine des séquences. Ce choix permet de dériver directement l'annotation fonctionnelle de l'échantillon métagénomique étudié. Un défi fondamental pour envisager un tel assemblage ciblé réside dans la réduction du temps de calcul, tout en conservant la plus grande précision possible. S3A répond à ce défi en séparant les lectures par domaine et en les ordonnant sur le domaine à l'étape de prétraitement. Le nombre de comparaisons de lectures est ainsi fortement réduit et la performance générale de l'algorithme grandement améliorée. S3A évalue les lectures chevauchantes et partageant une annotation de domaine commune, sur la base des deux métriques *lmsl* et *ip*. Ces métriques fournissent une mesure de confiance du chevauchement qui est à la fois complémentaire et beaucoup plus rapide que de compter un nombre fixe de mismatches. Il s'agit ici d'un calcul dynamique de la distance d'édition, comme le font d'autres assembleurs ciblés comme SAT. Ces métriques permettent également un découpage personnalisé du graphe OLC qui est indépendant de la technologie de séquençage utilisée, et contribue à réduire la complexité du graphe. Elles sont également utilisées pour résoudre les cas ambigus en l'absence d'arêtes transitives, et pour sélectionner les arêtes transitives les plus fiables dans le graphe OLC. S3A peut créer des structures de graphe OLC complexes en raison des nœuds chimériques, c'est-à-dire des nœuds ayant plusieurs arêtes entrantes et sortantes. En l'absence d'arêtes transitives, les nœuds chimériques sont considérés comme peu fiables et donc supprimés. Plus important encore, la possibilité d'annoter un ensemble réduit de domaines et d'assembler uniquement les lectures impliquant ces domaines, permet une exploration rapide des ensembles de données métagénomiques, et permet par conséquent à l'utilisateur de se concentrer sur des fonctions spécifiques d'intérêt.

Résultats

S3A a été testé sur trois jeux de données synthétiques, simulés selon les technologies Illumina et 454, et avec différentes longueurs de lecture (150 et 450 paires de base) et couvertures (7x et 30x). Ces jeux contiennent un grand nombre d'espèces (55), ce qui est suffisant pour mettre en évidence la plupart des défis liés à l'assemblage à a partir de données métagénomique (régions répétées, nœuds chimériques). Travailler avec des données simulées a l'avantage que tous les chevauchements réels sont connus, ce qui nous permet une comparaison précise de S3A par rapport aux assembleurs SAT et Xander. L'assemblage ciblé améliore considérablement l'annotation des domaines, par rapport à une annotation sur les lectures brutes. En effet, le taux de faux positifs des domaines annotés par S3A diminue d'un facteur entre 2 et 10, par rapport à celui observé lors de l'annotation de lectures brutes. Dans la figure 2, nous suivons le comportement de S3A par rapport à un nombre croissant de contigs correctement assemblés (TP) et montrons à quelle vitesse la précision de S3A se détériore. Sur le jeu de données

Illu/30x, caractérisé par la couverture la plus élevée, la proportion de contigs corrects assemblés par S3A reste presque constante, autour de 94%, alors que le nombre de contigs correctement assemblés double presque. En raison de la taille de ce jeu de données, SAT n'a pas pu fonctionner. S3A est également plus sensible que SAT et Xander. En effet, à niveau de précision égal, S3A récupère 22% de contigs corrects en plus sur le jeu de données Illu/7x et 5% de plus sur le jeu de données 454/7x que SAT, démontrant ainsi sa robustesse en fonction du choix de la technologie. Sur les mêmes jeux de données, S3A est également légèrement meilleur que Xander. Il est nettement plus sensible que Xander sur le jeu de données Illu/30x, où il obtient entre 24% (paramètre par défaut) et 57% de contigs corrects annotés en plus. Il est à noter que Minia est assembleur classique, et n'est donc pas particulièrement contruit pour un assemblage ciblé. Ceci explique en partie le fait que Minia permet de récupérer plus de contigs, mais aussi que le temps d'execution est plus important pour un nombre de domaines restreint, puisque l'ensemble des lectures doivent être assemblées par Minia avant l'annotation des domaines.

IMPRINT : un espace d'échantillonnage enrichi de séquences brutes et de multiples filtres convolutifs permet d'identifier les partenaires protéiques

Dans ce chapitre, je présenterai IMPRINT, un réseau neuronal convolutif profond (CNN) siamois pour la prédiction des interactions protéiques. IMPRINT est basé sur de multiples filtres convolutifs imitant des motifs fonctionnels; et est entraîné à partir d'une description enrichie de l'espace des séquences utilisant des familles de profils. Ces familles de profils décrivent la diversité fonctionnelle et structurelle des familles de protéines au cours de l'évolution. Etant donnée une protéine en entrée, sa caractérisation sous forme de profil fournit une description de la famille de protéines au CNN lors de l'entraînement. Ce nouveau cadre d'augmentation des données pour les séquences de protéines exploite l'énorme ensemble de séquences brutes disponibles, tout en évitant de générer des séquences artificielles. Il permet à IMPRINT d'apprendre mieux que les méthodes existantes traitant la même question. IMPRINT trouve des partenaires protéiques parmi des dizaines de milliers de protéines en un temps très raisonnable. Il peut tester quelques centaines de protéines contre elles-mêmes en moins de 15 minutes.

Caractéristiques principales de IMPRINT

IMPRINT est conçu comme une architecture siamoise, telle qu'illustrée dans la figure 3. IMPRINT prend une paire de profils de protéines en entrée et fournit un score indiquant la

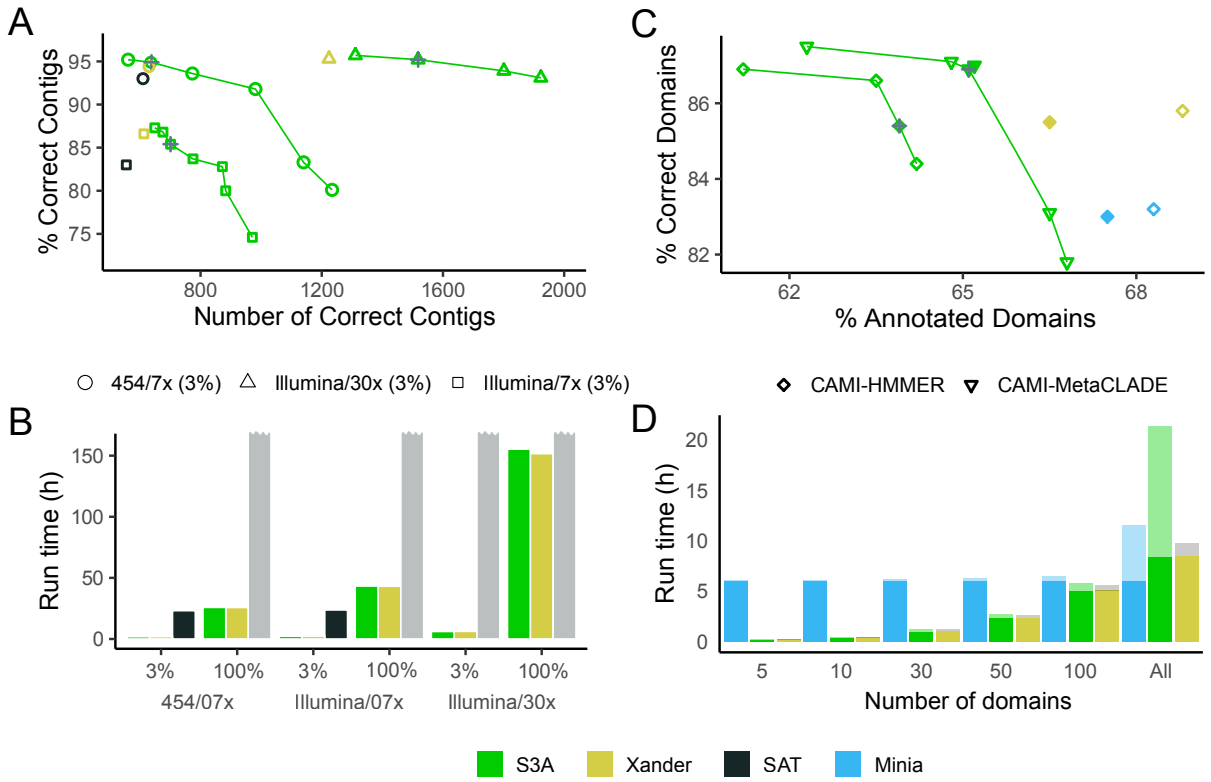

Figure 2: **Comparaison des performances de S3A, SAT et Minia.** Les analyses sont présentées en vert pour S3A, en or pour Xander, en noir pour SAT et en bleu pour Minia. **A.** La précision de S3A et SAT (rapportée en "% de contigs corrects") est calculée sur différents ensembles de données simulés (de taille réduite, 3% des lectures de l'ensemble de données complet). Les courbes vertes sont réalisées en faisant varier les valeurs du paramètre *lmsl* et en gardant le paramètre *ip* fixé à 80% (valeur par défaut). En diminuant les valeurs de *lmsl*, le nombre de contigs assemblés augmente mais la proportion de contigs corrects diminue et les courbes montrent à quelle vitesse la précision se détériore. La précision de S3A avec ses valeurs par défaut est indiquée par une croix grise. **B.** Performances en temps d'exécution de S3A, SAT et Xander calculées sur des ensembles de données simulés de deux tailles (3% ou 100% de l'échantillon), pour différentes technologies (454 ou Illumina) ou couvertures (7x ou 30x). L'exécution de SAT ne s'est pas terminée après 240h sur les plus grands jeux de données (représentés par une barre grise). **C.** Comparaison de S3A avec Minia et Xander pour la précision des domaines (exprimée en "% de domaines corrects") et le rappel des domaines (exprimé en "% de domaines annotés") pour différents seuils lmsl, sur l'ensemble "high complexity toy dataset" issu de CAMI. La précision en domaine et le rappel ont été calculés pour Minia et S3A par rapport à une annotation HMMER standard. S3A a également été analysé avec l'annotation MetaCLADE. Les valeurs par défaut sont indiquées par une croix grise. Les points remplis correspondent à la précision moyenne obtenue sur 5 sélections de 100 domaines tirés au sort. Nous constatons que la plus grande partie de la différence de rappel entre S3A et Minia/Xander peut être attribuée à un sous-ensemble de domaines qui sont plus courts que la moyenne. Avec une fraction de gènes proche de celle de Minia ou Xander, S3A peut avoir une meilleure précision de domaine pour un choix de seuil approprié. **D.** Performance en temps d'exécution de S3A, Minia et Xander sur le jeu de données CAMI de faible complexité, sur un nombre restreint de domaines, de cinq à cent, et sur le jeu de données complet. Le temps consacré à l'annotation fonctionnelle est mis en évidence en tons claires, et le temps pour la construction du modèle de domaine dans les tons foncés (pour Xander).

probabilité que la paire de protéines soient des partenaires. L'architecture siamoise est constituée de deux réseaux de neurones distincts, chacun dédié à l'apprentissage d'un profil de protéine (voir les architectures verte et bleue dans la Figure 3). Dans chaque réseaux de neurones, le module convolutionnel fait le produit de convolution d'un profil de protéine avec des filtres spécifiques (paramètres), la couche de rectification introduit la non-linéarité et l'étape de mise en commun calcule la moyenne de la réponse rectifiée de chaque filtre. Le module de convolution de notre architecture est conçu comme un ensemble de 200 ou 400 petits filtres, définis aléatoirement ou codant de petits motifs linéaires choisis aléatoirement dans une base de donnée. Chaque architecture comporte trois modules convolutifs dont le résultat sera mis en correspondance par un module de projection aléatoire. La première couche composée de nombreux petits filtres en parallèle permet d'apprendre des motifs spécifiques décrivant des interactions protéiques.

Résultats

Evaluation des performances de IMPRINT

L'augmentation des données basée sur des profils multiples a des effets postifs sur la performance d'IMPRINT. Pour le démontrer, IMPRINT a été comparé à un autre résaux de neurones (DPPI).
IMPRINT a été évalué par rapport à DPPI sur le jeu de données de Guo [25] , en l'entraînant sur un ensemble de 10 188 interactions et en le testant sur 1 000 interactions de la levure. Alors que toutes les méthodes se comportent d'une façon similaire, on peut noter une amélioration des performances sur la courbe Précision-Rappel lorsque l'on applique un schéma d'augmentation de données à trois profils, pour IMPRINT. On peut aussi remarquer une amélioration des résultats lorsque l'on considère une architecture avec plus de filtres en parallèle, comme le montre la figure 4.

Discussion

S3A

Les caractéristiques remarquables de S3A sont à la fois sa précision et sa complexité réduite, en particulier lorsqu'on se concentre sur plusieurs domaines d'intérêt. En ce sens, S3A ne cherche pas à surpasser les assembleurs de métagénome traditionnels, mais permet un compromis entre le nombre de domaines qui sont considérés et le nombre d'échantillons qui peuvent être traités dans le même temps. Les performances de S3A peuvent donc justifier son utilisation par rapport aux assembleurs classiques lorsqu'un profilage rapide et sensible d'une douzaine à quelques centaines de domaines est nécessaire. La détection de la résistance

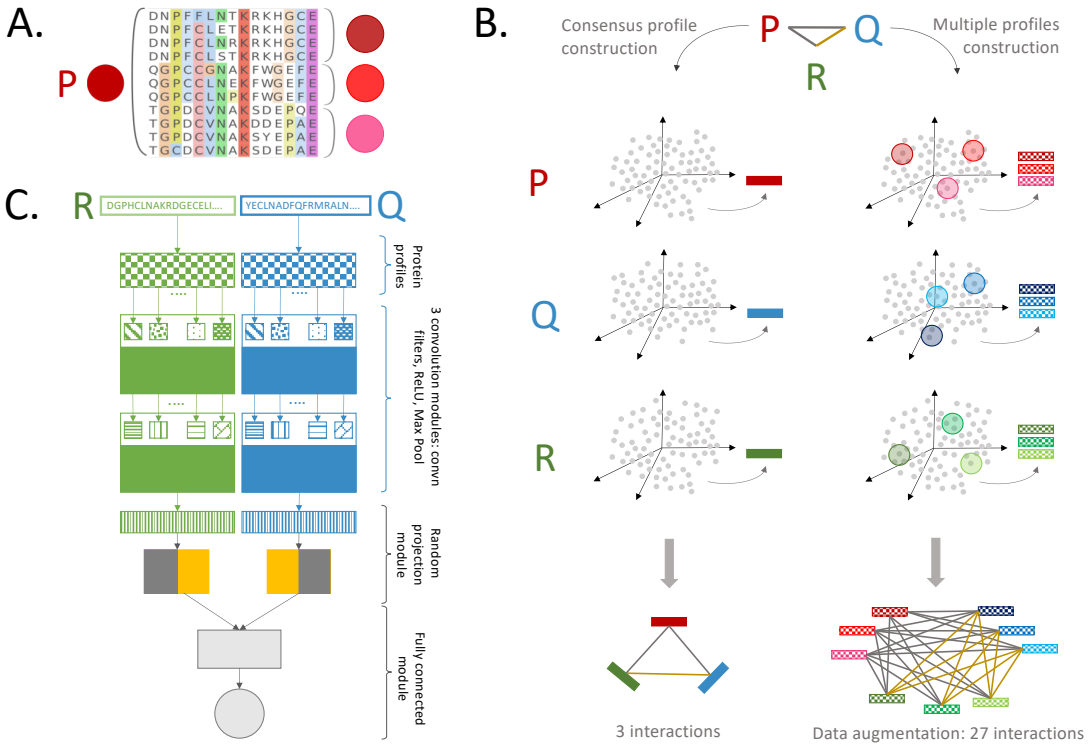

Figure 3: **Caractéristique d'IMPRINT** A. **Profiles multiples.** Les séquences homologues d'une protéine P peuvent être utilisées globalement pour construire un seul profil consensus, ou bien être divisées en sous-ensembles de séquences, qui sont proches les unes des autres, afin de construire plusieurs profils. B. **Augmentation des données pour construire de plus grands ensembles de données d'entraînement.** Considérons un ensemble d'entraînement de 3 protéines, P,Q,R, liées par des interactions considérées comme vraies (bords dorés) ou fausses (bords noirs). À gauche, les séquences homologues (points gris) dans les espaces de séquence des protéines P, Q et R, sont utilisées pour construire trois profils consensus (rectangles colorés). A droite, pour chaque protéine, 3 homologues (points colorés) suffisamment éloignés dans l'espace des séquences, sont sélectionnés ainsi que leurs séquences proches (voir voisins colorés des points colorés). Trois profils sont construits pour chacunes des familles de protéines P, Q et R. Chaque profil sera apparié avec tous les autres profils des deux autres familles, ce qui fait un total de 27 paires d'interactions à utiliser pour l'entraînement. C. **Une architecture reposant sur des motifs.** IMPRINT est conçu comme un réseau Siamois classique avec des paramètres partagés, où chaque partie du réseau est dédiée à une protéine différente (blocs bleu et vert). Il prend en entrée deux profils de protéines. Le réseau est composé de trois modules de convolution constitués de filtres de convolution parallèles de même largeur, codant des motifs protéiques. Les filtres sont regroupés et passent à travers un filtre ReLu et un filtre Max Pooling. Les sorties des deux réseaux sont ensuite fusionnées par un module de projection aléatoire combiné à une couche entièrement connectée. Une transformation linéaire est ensuite appliquée, afin d'obtenir une valeur de prédiction de l'interaction en sortie.

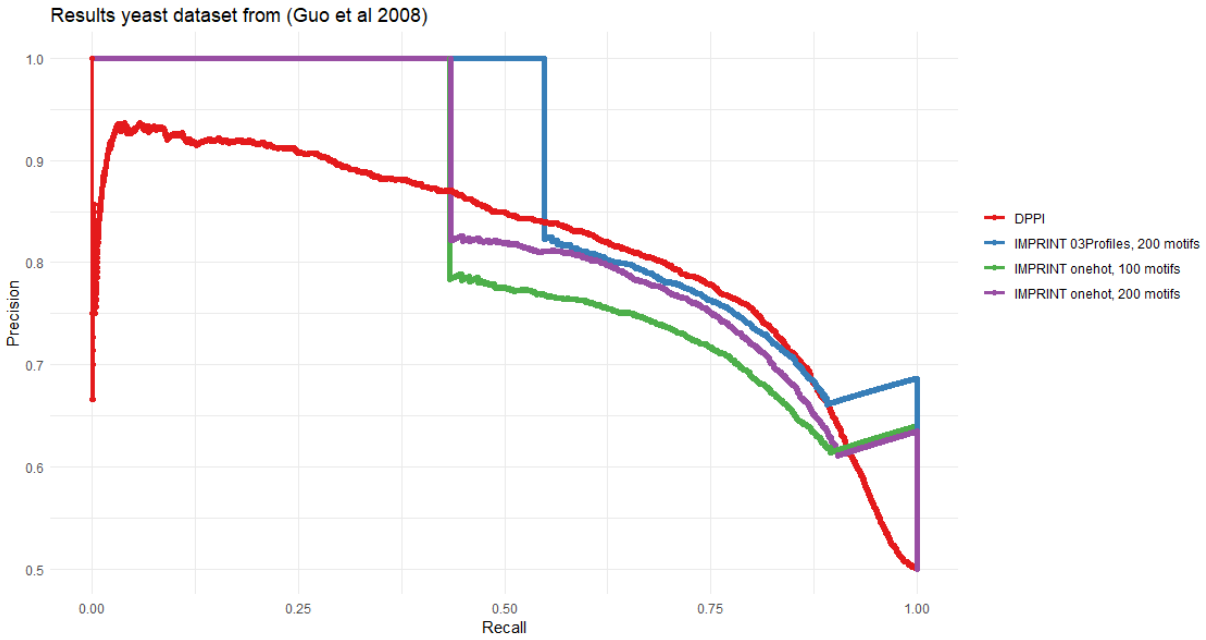

Figure 4: **Performance d'IMPRINT et DPPI sur le jeu de données de Guo. Courbe Précision-Recall pour DPPI (rouge), et IMPRINT avec hot-encoding (100 motifs en vert, et 200 motifs in violet), ainsi qu'avec une augmentation à 3 profiles (200 motifs en bleu).**

antimicrobienne, l'annotation de types spécifiques de pathogénicité et la recherche d'indicateurs d'une réaction biochimique particulière en sont quelques exemples. On peut souligner que notre stratégie de construction de graphe est très générale : n'importe quel type d'annotation de chaîne de caractères peut être fourni en entrée à notre assembleur. Cela pourrait facilement conduire à une amélioration des temps d'exécution, où l'étape "coûteuse" de l'annotation HMM (effectuée séquentiellement sur tous les protigs), pourrait être remplacée par une indexation et un regroupement efficaces des protigs. Dans un autre ordre d'idées, la manière dont le graphe OLC est construit permet d'annoter plusieurs domaines, de sorte qu'un graphe unique est construit pour l'ensemble des domaines. Une amélioration de notre méthode consisterait à gérer des annotations de domaines multiples qui se chevauchent, de sorte que chaque arête pourrait avoir des poids pour différents domaines. Nous pensons que cela améliorerait la sensibilité du poids donné aux arêtes du graphe qui sont utilisées pendant la traversée du graphe. Pour les longs protigs, cela permettrait également de reconstruire les cas de cooccurrence de domaines. Ceci est prometteur dans un contexte d'assemblage ciblé, puisque cela devrait à la fois permettre un assemblage encore plus rapide, tout en permettant la détection de fonctions plus précises, basées sur des domaines multiples. Une limitation de notre approche est sa dépendance aux poids des arêtes, qui sont utilisés à la fois pour l'élagage du graphe et pour le parcours. Tout en essayant de maintenir un seuil générique indépendant de la longueur du protig, le seuil optimal pourrait varier en fonction du type d'espèces à partir duquel les lectures ont été séquencées. L'application d'un outil de correction des erreurs avant l'annotation, une pratique courante pour l'assemblage *de novo*, pourrait améliorer la robustesse

des paramètres *lmsl* et *ip* pour S3A. Cependant, les résultats sur différents jeux de données/technologies de séquençage montrent que S3A conserve un très haut niveau de performance, même lorsque les valeurs de seuil par défaut sont modifiées.

IMPRINT

Les bases de données documentant les interactions protéiques déterminées expérimentalement sont encore loin d'être complètes à l'heure actuelle, même si on considère des organismes modèles bien étudiés comme la levure ou l'homme. Leur extension est un domaine de recherche actif. Aujourd'hui, on constate que les réseaux d'interactions physiques obtenus par des technologies à haut débit comprennent de nombreuses interactions non fonctionnelles. Au même moment, de nombreuses interactions réelles sont manquantes. C'est la raison principale pour laquelle une des méthodes de reconstruction computationnelle *ab initio* des interactions protéiques sont nécessaires. Une contribution d'IMPRINT au domaine de l'apprentissage profond appliqué à la protéomique est l'introduction d'une nouvelle façon d'augmenter l'ensemble d'entraînement. Il a été observé que la quantité de données utilisées dans l'étape d'apprentissage est essentielle, à condition qu'elle apporte une bonne représentation de l'ensemble des données. Le surapprentissage induit par des entrées similaires doit aussi être tant que possible évité. Pour aborder ce point, IMPRINT augmente l'ensemble d'apprentissage avec de multiples profils décrivant la variabilité d'une famille de protéines à travers l'espace de ses homologues. Cette idée a déjà été appliquée de manière fructueuse pour l'annotation de domaines génomiques et métagénomiques. Appliqué à la prédiction d'interaction, le schéma d'augmentation permet d'agrandir quadratiquement le nombre de paires considérées en fonction du nombre de profils et donc de créer un ensemble d'apprentissage bien plus grand. Il convient de souligner que le transfert d'informations sur les interactions par homologie n'est pas toujours correct. En effet, certains homologues de protéines en interaction peuvent être des paralogues qui ont diversifié leur fonction et leurs partenaires d'interaction. Nous nous attendons donc à ce que notre schéma d'augmentation contienne des exemples d'interaction qui ne sont pas corrects. Cependant, en multipliant la taille de l'ensemble d'apprentissage avec des protéines pertinentes, nous démontrons qu'IMPRINT permet à l'algorithme d'apprentissage d'être robuste. Notre stratégie d'augmentation pourrait être appliquée avec succès à d'autres problèmes de classification fonctionnelle en protéomique. À cet égard, nous remarquons que les paires de profils dans IMPRINT sont construites à partir de séquences homologues, indépendamment de leur origine phylogénétique. Un contrôle sur la sélection des séquences de protéines d'où provient un profil peut être ajouté à la stratégie de calcul. Le deuxième élément important de la conception d'IMPRINT est l'utilisation d'une architecture CNN basée sur des motifs ayant le potentiel de détecter des positions le long de la séquence de la protéine qui sont liées à l'identification du partenaire, en particulier les sites de liaison.

Conclusion

Mon premier projet était axé sur l'assemblage dans un contexte de métagénomique. En particulier, nous avons utilisé une annotation fonctionnelle pour guider l'assemblage, et conçu S3A de manière à ce que les lectures appartenant à différents micro-organismes puissent être séparées et assemblées indépendamment. Suite à ce travail, nous avons conçu une approche d'apprentissage profond, IMPRINT, qui peut prédire la probabilité d'interaction entre deux protéines données, en utilisant uniquement les informations de la séquence d'acides aminés. En outre, nous avons montré qu'IMPRINT pouvait également fournir des informations supplémentaires sur les positions de la séquence de la protéine qui sont importantes pour son interaction avec d'autres protéines. En ce sens, les deux projets peuvent être considérés comme complémentaires. En partant des ensembles de données métagénomiques, nous pouvons fournir un assemblage ciblé efficace pour les séquences d'acides aminés des protéines. Cela peut ensuite être utilisé pour évaluer l'affinité d'interaction entre les différentes protéines de l'ensemble de données. Dans une étape supplémentaire, les travaux futurs pourraient inclure l'utilisation de ces prédictions pour une communauté microbienne entière, afin de prédire les réseaux métaboliques présents. Pour les deux projets, nous avons utilisé de manière approfondie les informations sur l'évolution, qui nous ont permis de déduire des informations biologiques importantes. Cela inclut l'annotation des domaines pour S3A, et l'utilisation de profils protéiques comme données d'entrée pour IMPRINT.

Publications

Tout au long de ma thèse, j'ai participé aux articles suivants :

- L. David, R. Vicedomini, H. Richard, A. Carbone. "Targeted domain assembly for fast functional profiling of metagenomic datasets with S3A." *Bioinformatics*, 2020.
- E. Teppa, F. Nadalin, C. Combet, D.J. Zea, L. David, A. Carbone. "Coevolution analysis of amino-acids reveals diversified drug-resistance solutions in viral sequences: a case study of hepatitis B virus." *Virus evolution*, 2020.
  `https://academic.oup.com/ve/article/6/1/veaa006/5728782?login=true`
  Ma contribution à cet article est le développement de l'algorithme de Sankoff .
- C. Dequeker, Y.Mohseni Behbahani, L. David, E. Laine, A. Carbone. "From complete cross-docking to partners identification and binding sites predictions", *PLOS Computational Biology*, 2022.
  `https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1009825`
  Ma contribution ici a été le test du réseau DPPI sur le jeu de données PPDBv2 .

- L.David, H. Richard, and A. Carbone. "IMPRINT: an augmented sample space of natural sequences and multiple convolutional filters identify protein partners", en préparation, 2022.

Chapter 1

Introduction

1.1 The general context

The analysis of biological sequences is a main research areas in Bioinformatics. In particular, predicting protein-protein interactions is an important step in the construction of underlying protein networks, and plays a key role in understanding molecular and cellular environments. The raw data produced by next generation sequencing consists of *reads* obtained from fragments extracted from a metagenomic sample. The first challenge is to assign these reads, either by alignment to reference sequences or by their assembly. This assembly/alignment is usually followed by a functional annotation of the predicted coding regions to describe the community's metabolic activities, which can in turn be analyzed to interpret the protein function in cellular processes.

Over recent years, next generation sequencing has led to a rapid increase in the production of biological data in a wide variety of fields. To understand and interpret this huge amount of data requires efficient and accurate computational approaches to extract information from the raw sequences. These approaches are more and more directed towards machine learning, and in particular deep learning. Data mining, which consists in extracting information from large datasets for the purpose of learning patterns and models, allows us to improve our understanding of biological mechanisms. In molecular biology, much work has been done for protein function prediction and for protein-protein interaction prediction, questions we partly address in this thesis.

This manuscript is divided into two parts. The first part focuses on the development of S3A, a targeted domain assembler for fast functional profiling of metagenomic datasets. It aims at rapidly exploring the content of large metagenomic datasets with respect to specific domain targets. In the second part, we present a deep neural network, IMPRINT, whose goal is to identify protein partners. It relies solely on sequence information to assess the probability for two input proteins to interact, while also delivering information on interaction surfaces.

In summary, we focus on two biological challenges that in fact occur at two subsequent steps

of the annotation process. Sequencing is a fundamental step for the generation of large biological datasets, which in turn are useful for machine learning tools to exploit and interpret. A better quality of the sequencing data leads to a better annotation, which in turn improve the quality of the predictions made by the learning system. If the learning process is sufficiently robust, it can in turn be interpreted to enhance the annotation.
The main contributions of this thesis are two-fold. Our goal first goal concerns assembly in the metagenomics context, and led to the development of an open-source software, S3A, which relies on protein domain annotation to provide a fast and accurate targeted assembler. The idea behind our tool is to use domain annotation as further evidence for putative overlapping amino-acid sequences. Moreover, the assembler's implementation is focused on the efficiency of its traversal and on its genericness. In a second project, we conceived a deep neural network, IMPRINT, to identify protein partners. This convolutional neural network relies only on each pf the two amino acid protein sequences, and uses a novel strategy for sample space augmentation. The training dataset is greatly expanded by constructing multiple profiles for each protein sequence, which allows for improved performances. Furthermore, an analysis on the network interpretability led to interesting information concerning interaction of single residues with a specific partner.
Both project were designed to be easily accessible, openly available for use and to provide new possibilities compared to the already existing tools.

1.2 Background Notions

1.2.1 Preliminaries

Throughout this chapter, I will introduce the various terms and fundamental biological concepts that we find necessary for a clear understanding of our work and its goals. Furthermore, we will attempt to highlight how a better comprehension of the biological mechanisms and their evolution can lead to advances in Bioinformatics applications; and to what aspects of these mechanisms we focused on. We will also stress the challenges usually found, both biologically and computationally.

From DNA and RNA to proteins

Knowledge of DNA sequences is at the core of research in molecular biology. Deoxyribonucleic acid is a molecule composed of two chains constituted of four different nucleobases (adenine, cytosine, thymine and guanine). The sequence of bases in the DNA forms the genome, which carries the genetic information from one generation to the next one. This sequence can be transcripted, then translated into a protein to execute a function. From the computational point of view, the genome can be represented by a set of strings in the

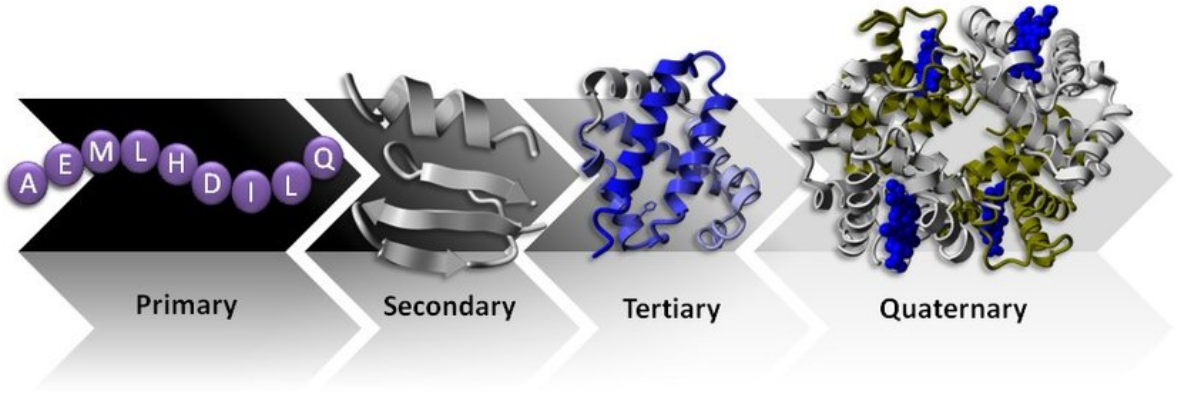

Figure 1.1: The four levels of protein structure (Source: [32]).

alphabet {A,C,G,T}. The transcription step is performed by messenger RNAs (mRNA), which are also constituted by four bases but where the thymine is replaced with another base, the uracile. The mRNA carries the information copied from a region in the DNA sequence. It is less stable than the DNA, as its purpose is only to transfer the necessary information for the ribosomes to construct a protein. In the translation step, these ribosomes read the *mRNA* sequence by steps of three bases (codons) at a time, each corresponding to an amino-acid residue. Twenty different amino-acid residues can be translated from the genetic DNA code, and their aggregation into a chain constitutes a protein. This chain folds itself into a 3D conformation determined by the protein molecular environment and its sequence. The conformation can be summarized at different levels of abstraction (see Figure 1.1):

- The *primary structure* corresponding to the residue sequence.
- The *secondary structure* describing the three-dimensional form of local segments. α-helix and β-strands, which are determined by the torsion angles of the residues' backbone, are the most common secondary structures.
- The *tertiary structure* representing the folding of the protein on itself.
- The *quaternary structure*, which is the association of several protein chains or subunits into a single functional unit called multimer.

Even though we will in this thesis mainly focus on the primary structure of the protein (the analysis of interaction being also relevant to the quaternary structure), the tertiary structure is essential in understanding its interaction with other biological objects, with other proteins among others. Predicting how a protein folds itself is a very challenging problem in its own, which is why being able to identify potential partners without having to rely on the 3D structure can be beneficial. On the other hand, in some cases the 3D structure cannot be used, such as for disordered protein's interaction, or interactions that take place after conformational

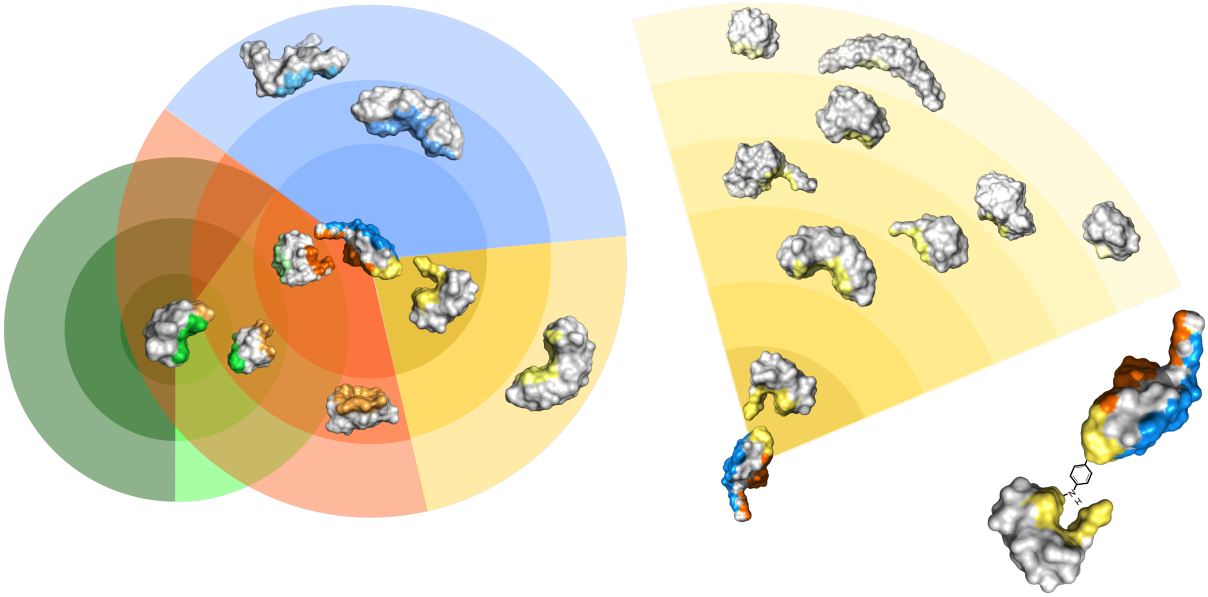

Figure 1.2: A given protein can have serveral potential partners (left), with different levels of affinity (source: courtesy of A.Carbone & E.Laine).

changes. Such tools can thus be used on a wider range of data (where the structural information may not be available), and at a larger scale.

In eukaryotes, genes are organized in coding parts (exons), interspersed by non-coding ones (introns), the latter being removed after transcription. Exons can then be joined in different combinations, leading to different mRNA strands. This process is called *alternative splicing*. This implies that a single gene can code for multiple proteins, and increases the difficulty of identifying coding regions. Transcriptonics is defined as the study of all the possible sets of RNA transcripts in an organism.

Finally, proteins accomplish their function by binding to other proteins. In that sense, identifying potential partners for a given protein helps to understand its role within its environment, for instance within a microbial community. Even with structural information, this identification is difficult, because the possibility of one protein binding another is not a binary choice. For that reason, it is usually more reasonable to describe potential partners as having an affinity of interaction. Moreover, a given protein can have one or several true partners, and its interactions can be more or less permanent (see Figure 1.2.

More specifically, protein interactions can be indicated as either *stable* or *transient*. *Stable interactions* are organized in protein complexes such as ribosome and hemoglobin, while *transient interactions* are short-time interactions that alter or transport a protein and lead to subsequent changes such as protein kinases and nuclear pore importins. Knowledge about protein-protein interactions (PPIs) can be applied to the function prediction of uncharacterized proteins, enhancement of the details about a signaling pathway, and characterization of the protein relationships that establish poly-molecular complexes.

Genomics, metagenomics.

While genomics is the study of the genomes from a single species, metagenomics focuses on genetic material recovered directly from environmental samples. The whole genomic DNA is prepared from samples, regardless of its microbial composition and is characterized by whole genome sequencing. The important advantage is to be able to explain the relationships between the different species in the microbial community. Moreover, a significant amount of microorganisms are difficult to study in isolation, as they either fail to grow independently, or depend on other organisms to function. However, the large amount of data typically generated by metagenomics opens two major challenges: data storage and computational power.

Protein sequences and functional motifs

Based on the structure or sequence similarity, proteins are grouped into categories, that mostly focus on functional characterization. Therefore, for a newly discovered protein, its function can be deduced according to the category in which it has been put. These categories can rely on common evolutionary origins (families), or independent regions of the protein's chain. Domains in particular are conserved functional and structural regions of a given protein that can evolve and exist independently in various proteins. They can form functional units, although similar domains can imply different functions. Moreover, domains can also interact, so that several domains in a protein are responsible for a specific function. Domain size can vary from 50 amino-acid to hundreds of residues. Different classifications of protein domains and families exist, among which Pfam [5] is one of the most used. It relies on multiple sequence alignments and profile hidden Markov models to provide a general and complete classification of protein domains and families, and is extensively used by researchers ([60]). Pfam entry types are as follows:

- **Family** defining that members of the family are related.
- **Domain** describing independent structural or sequential unit found in multiple proteins.
- **Repeat** for non independently stables entries required to be combined to form a domain.
- **Motif** corresponding to shorter sequence units which are stable in isolation and found outside of globular domains.

Among this last entry type, we are particularly interested in *short linear motifs* (SLIMs), that are generally located in disordered regions, because they mediate protein-protein interactions. One example of protein database that focuses on SLIMs is the Eukariotic Linear Motif (ELM) [15]. Short linear motifs range from 6 to 11 amino-acids.
There are different classifications of protein domains and families. These classifications rely on automatic or semi-automatic procedures, mostly based on HMMs [19] and multiple

sequence alignment [12]. Some automatic domain annotation tools are specifically adapted to the metagenomics context. For the purpose of our work, we used metaCLADE [57], which provides a profile-based domain annotation pipeline directly from reads, and is based on the multi-source domain annotation strategy. For each one of the approximatively 15,000 Pfam domains, hundreds of probabilistic models are generated from *homologous* sequences (i.e. that share a common DNA ancestral sequence). Domains are searched using a library of those models, and the most likely annotation for the read is selected by filtering on the domain hits produced from the models. Another very commonly used tool is HMMER [19], which uses probabilistic models, and can, given a profile database such as Pfam, and search a query protein sequence against it to perform the annotation.

1.2.2 Sequence Assembly

In the absence of a reference genome, genome assembly is usually the first step towards the processing of our sequencing data. It is usually followed by the identification of *open reading frames* (ORFs), which are the coding regions within the genome. In the transcriptonics context, this leads to additional difficulties for eukaryotic genomes, as alternative splicing allows for the production of different transcripts from the same gene. Finally, metagenomics assembly leads to the challenge of separating microorganism among a mixed population.

Sequencing technologies

Sequencing is the process of extracting information from a nucleic acid in the form of sequences of bases. The problem of reading the complete DNA sequence of an organism is a task that strongly depends on the sequencing technology used. Due to technological limits, it is impossible to read out a single, complete and exact genomic sequence from a cell. A DNA sequencer is a machine that reads stretches of DNA and reports its sequence of bases. It allows automatized and parallel sequencing experiments. Each sequence tag outputted by a sequencer is caller a read, and each read contains the following information: an ID, a sequence of bases s in the alphabet Sigma = A,C,G,T (and a quality sequence containing a value for each bases of s).

Next-generation sequencing (NGS). Next-generation sequencing is a massively parallel sequencing technology, which brings high throughput, scalability, and speed. It is used to determine the order of nucleotides in entire genomes or targeted regions of DNA or RNA. Input sample must be cleaved into short sections, whose length depends on the sequencing machinery.

Sanger, 454, Ilumina, nanopore. Among the different NGS technologies, we present:

- **Illumina**, the most used technology, relying on PCR to amplify each read and creating a spot with many copies of the same read. They are then separated into single strands to be sequenced. Its length varies from 100 to 150 base pairs, while its error rate, while depending from the platform, ranges from 0,6 to 0,08 percent.
- **454 Pyrosequencing**, that can sequence much longer reads than Illumina (typically around 450 base pairs). Like Illumina, it does this by sequencing multiple reads at once by reading optical signals as bases are added. In our context, the latter technology has the advantage of delivering longer reads, which can thus be annotated more easily and with higher chance of covering whole domains. However, Illumina technology is more widely used, and has a lower error-rate.
- **Nanopore** sequencing is based on monitoring changes to an electrical current as nucleic acids are passed through a protein pore. The resulting signal is decoded to provide the specific DNA or RNA sequence. In this case, Nanopore can cover even longer reads.

A *sequencing error* are erroneous nucleotide substitutions that can occur during the sequencing-by-synthesis. On average, error rate of this approach is reported to be 0.1% per nucleotide. Error rate assessment is an important aspect for the choice of which NGS technology to choose, as the architecture of our assembler can be more or less sensitive to them, on to their most probable positions within the read [42].

De novo assembly

De novo assembly and the role of reference genomes. DNA sequencing approaches can be grouped in two categories: *de novo* sequencing and resequencing. The latter consists in comparing the whole or part of an organism's DNA sequence to a reference genome. This can lead for example to a comparison between genes of interest of two organisms from the same species, or between two different species, in order to discover possible mutations or study evolution. It can also be used for a reference-guided assembly, where reads are mapped against the known genome to facilitate the assembly. Conversely, *de novo* assembly doesn't use any information other than the set of reads provided by *de novo* sequencing to reconstruct the complete DNA sequence of an organism. Initially, the definition of assembly was that it corresponded to finding the shortest string that explains the set of reads extracted from it, which in terms of combinatorics was thought to be a NP-hard problem. However, this complexity comes from the idea that the assembly problem requires solving the Hamiltonian cycle problem in the overlap graph (a Hamiltonian cycle in a graph is a cycle that visits every vertex at least once). In fact, it was shown that finding all possible contigs can be done in polynomial time [10]. Nevertheless, the assembly problem is further complexified by the presence of *sequencing errors* (bases within the read differing from the original sequence), as

well as the presence of repeated regions within the genome. Repeats are patterns of bases that occur multiple times throughout the genomes, and can lead to reads being mapped to an incorrect part of the genome. Metagenomics assembly further increases the difficulty, because instead of assembling a single genome, the goal is to reconstruct the whole mixture of organisms, who are more or less abundant and related with each other.

Read length and coverage. We present here several basic notions. The output of a sequencer is a set of reads obtained from a genome that is usually over-sampled. The length of a read is measured in *base pairs* (bp). Since the read length is usually much shorter than the genome length, oversampling is key for a correct assembly. Firstly, because overlap between reads are necessary in order to assess that two given reads correspond to consecutive regions in the genome. Secondly, it brings that much evidence to the correctness of the assembly. Thirdly, it can be used to overcome sequencing errors. We define *coverage* by the expected number of reads that include a given nucleotide in the genome. Given a genome g and a set of N reads of length L, the coverage c is:

$$c = \frac{NL}{|g|} \tag{1.1}$$

Higher coverage generally increases the possibility of a true reconstruction, at the cost of a higher computational need and experimental cost.

String graph, contigs and protigs. Let g be the string corresponding to the genome we wish to reconstruct, and S a finite set of substrings of g. Let r_1 and r_2 be two reads and $o(r_1, r_2)$ the length of the maximal overlap between r_1 and r_2. Given $k \in \mathbb{N}$, let $G = (V, E, w)$ be a directed labeled graph named overlap graph, where:

- $V = S$.
- E the set of directed edges between r_1 and r_2 if $o(r_1, r_2) > k$.
- w the weight assigned to each edge (r_1, r_2).

The assembly problem we wish to solve is finding a path g in G visiting every node V at least once, and for which $w(P)$ is minimal. We can reduce this overlap graph to a string graph, where each read contained within other reads are discarded, and *transitive edges* are eliminated. A *transitive edge* (as illustrated in Figure 1.3) contains a useful yet redundant information concerning the overlap between two reads. Finally, we define *contigs* (resp. *protigs*) as a series of overlapping DNA (resp. amino-acid) sequences. Bridging the gaps between contigs to reconstruct g, along with ordering and orienting contigs, is called *scaffolding*.
To overcome the difficulties linked to *de novo* assembly, several approaches have been proposed based on string graphs.

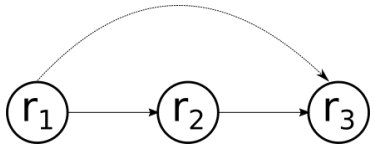

Figure 1.3: A transitive edge linking r_1 and r_3.

Overlap-Layout Consensus graphs and de Bruijn graphs. The first and most straightforward approach of de novo assembly is the Overlap-layout-consensus technique. The idea is to construct an overlap or string graph, where each node represents a read, and each edge a suffix-prefix overlap between pair of reads. Once the graph is built, usually a certain number of graph simplification techniques are performed to reduce its complexity, such as transitive edge simplifications. Then a depth first search on the graph is performed. This leads to the generation of contigs, which correspond to the solution of the assembly problem for this graph. This technique has several drawbacks, which include an usually high memory and time consumption. However it has the advantage of keeping reads as whole entities, which can be important in some cases. The second technique is based on the construction of a de Bruijn graph, where nodes are defined by k-mers and edges by $(k-1)$-length overlaps between k-mers. The de Bruijn problem then consits in finding a superstring that contains each k-mer. Both methods are illustrated in Figure 1.4. De Bruijn graphs are well suited for short reads, we will however focus on the OLC graph for our problem, because fragmenting reads into k-mer would imply not using domain annotation to its full extent. However, it is important to highlight that most assemblers, including some against which we will compare our tool, are using de Bruijn graphs.

State-of-the art assemblers. Various approaches using de Bruijn graphs are found in the literature, but in our context, OLC graphs are also used, as they are suited to targeted assembly. This is also the case for S3A. **Velvet** [70] was one of the first proposed assembler for short reads. It has the particularity of eliminating spurs from the de Bruijn graph to significantly reduce its complexity. The graph is further simplified by reducing each linear path into a single node that corresponds to a *unitig*. Then bubbles (branches that converge after few nodes) are resolved by taking the most supported path. It also uses long reads information, if available, that are used to connect nodes in the graph. **SOAPdenovo**[38] is another assembler based on a de Bruijn graph implementation. The assembler first uses k-mer frequencies to correct reads. Its design is similar to Velvet, but requires less memory. It also uses pair read information to get trusted contigs mapping all reads against the assembly and removing transitively reducible edges. S3A has been compared to the following three tools:

1. **Minia**[14], a short-read assembler based on a de Bruijn graph, which uses Bloom filters to greatly reduce the space needed for the graph encoding. Large genomes can be

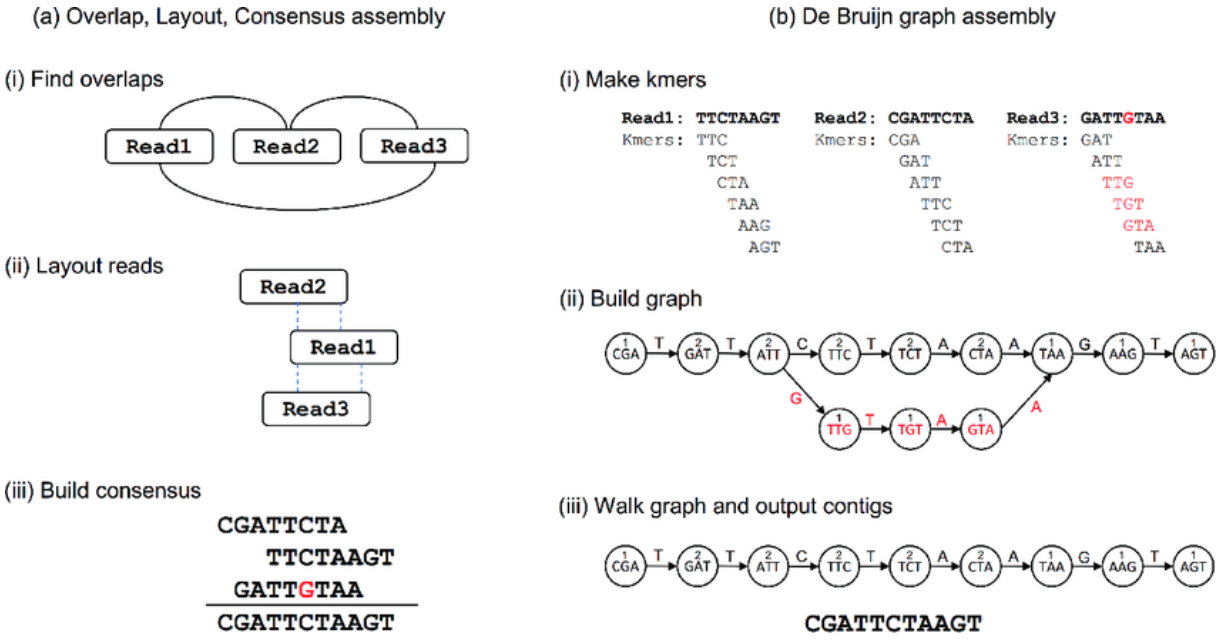

Figure 1.4: Two algorithmic approaches to de novo assembly. **left:** Overlap-Consensus assembly consist in finding overlaps between reads (i), then join then into contigs (ii) to finally build the consensus sequence (iii). **right:** de Bruijn assembly first decomposes reads into k-mers (i), connect overlapping kmers with edges (ii) and constructs contigs by a graph traversal (source: [3]).

assembled in a short time, and with similar accuracy than the previously mentioned assemblers.

2. **Xander**[65] is a metagenomics assembler targeting specific protein-coding genes. It combines a de Bruijn graph and a profile hidden Markov model (HMM) for the gene of interest, to produce a weighted assembly graph.

3. **SAT**[71] is another targeted assembler, using an OLC graph. Reads are aligned against profile HMMs, and a family-specific graph is generated for each gene family. S3A retraces the same architecture as SAT.

Sequence data preparation

To further increase a *de novo* assembly project's performance, it is in most cases very useful to introduce a preprocessing step in the pipeline.

Data cleaning. The first step is a general data cleaning procedure, which include the removal of bad quality reads, trimming, and/or error correction. In this sense, being able assess the quality of reads is important. Read clustering is also a preprocessing step that collapses redundant clusters in a single cluster. Such a method can greatly enhance time and memory performances by reducing the data complexity. Some tools provide standalone cleaning

procedures, while some assemblers include them in their pipeline. Among them, BCALM [13] generates set of unitigs from a de Bruijn graph, and is used in our pipeline.

Sequence assembly evaluation

Since sequencing is an essential step before any other subsequent analysis, being able to obtain a reliable reference genome and assess its correctness are mandatory for the community. In this sense, several research studies have addressed the difficulty of *de novo* assembly evaluation, and aimed at providing benchmark datasets and standardized metrics for a correct evaluation of assemblers. One such example is the Assemblathon [17], or more recently CAMI [49] (Critical Assessment of Metagenome Interpretation), a benchmark evaluation on highly complex and realistic data sets. The issue with many of these competitions is the difficulty of guaranteeing that a given assembler would generalise well to other datasets classes (real datasets for example, if the competition uses simulated ones). While some are based on specific genomes (that have already been sequenced), other datasets such as CAMI or Assemblathon use never-before-seen real genomes. It is however essential to give a common basis enabling comparison between the different assemblers.

Simulated sequencing approaches. Among the different options to assess a *de novo* assembler's performance is the use of simulated datasets. Read simulators are tools that generate synthetic NGS reads (Illumina for example) along with their sequencing artifacts. This is important to provide test datasets that can model different variants, and give a ground truth towards which the assembler is evaluated. Among others, MetaSim [45] is a sequencing simulator generating collections of synthetic reads that reflect the diverse taxonomical composition of typical metagenome data sets. FlowSim [4] specialises in simulated 454 reads. A drawback of using only simulated reads is the risk that the evaluation will be biased towards the potential of the read simulator to faithfully reproduce error schemes or not. In that sense, testing on real datasets is also fundamental.

Metrics (contiguity, completeness, correctness). Another difficult task when evaluating assemblers is the choice of correct and general metrics. There are many different views on what metric is better suited, and it can depend on the assembler's target. However, three parameters on which assemblers can be evaluated can be defined:

1. *contiguity*, to evaluate ability of joining sequences together.
2. *completeness*, to determine the fraction of the genome successfully rebuilt.
3. *correctness*, to give its degree of accuracy compared to a ground truth.

Other metrics, which we will use, include number of contigs, mean contig length and total assembly length. In any case, it is difficult to decide which feature describes best the quality of the assembler.

Running time, memory requirements. One last aspect which is important in the assembly context is its computational cost. In particular, running time and memory usage can be significant, which can make it less accessible. For this reason, some assemblers also focus on limiting or reducing these requirements, as is the case for our assembler.

1.2.3 Protein Partner Identification Problem

Machine learning principles.
"We are drowning in information and starving for knowledge", *John Naisbitt* (from Murphy, Machine Learning a probabilistic perspective).
The goal of machine learning is to be able to automatically **detect patterns** in data, and then to be able to use the machine built to **predict** future data points.
Generally, we start from a training set $\mathcal{D}$ which correspond to N observed variables $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. The variables $\mathbf{x}_i$ are called covariates or features. It is possible that each feature $\mathbf{x}_i$ is associated to a response variable y_i. In this case, we are in the first main category of learning which is *supervised learning*. We hypothesise that there exists a function f that provides a mapping between $\mathbf{x}_i$ and y_i. The problem of supervised learning is then to find an estimate $\hat{f}$ of f that will predict $\hat{y}$: $\hat{f}(\mathbf{x}) = \hat{y}$. In the vast majority of the cases, the features variables are real valued ($\mathbf{x}_i \in \mathbb{R}^d$). Concerning the response variable y, it can be either:

- categorical and we speak about classification,
- real valued, we are in the context of regression.

To determine $\hat{f}$, supervised learning searches for the function whose predictions $\hat{y}$ diverge the least from the real ones on $\mathcal{D}$. This divergence can be defined in different ways, depending on the problem and it is called the *loss*. For instance in the case of classification, a common definition for the loss is the misclassification error:

$$l = \sum_{i=1}^{N} \mathbb{I}_{y_i \neq \hat{y}_i}$$

.
An other category of learning is *unsupervised learning*. In this case $\mathcal{D}$ is limited to the input variables $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and the goal here is to find patterns in the data. In this case there is no obvious objective function on which the learning is based, but it relies usually on a kind of distance measure between instances. The different types of learning are detailed in the next paragraph and illustrated in Figure 1.5.

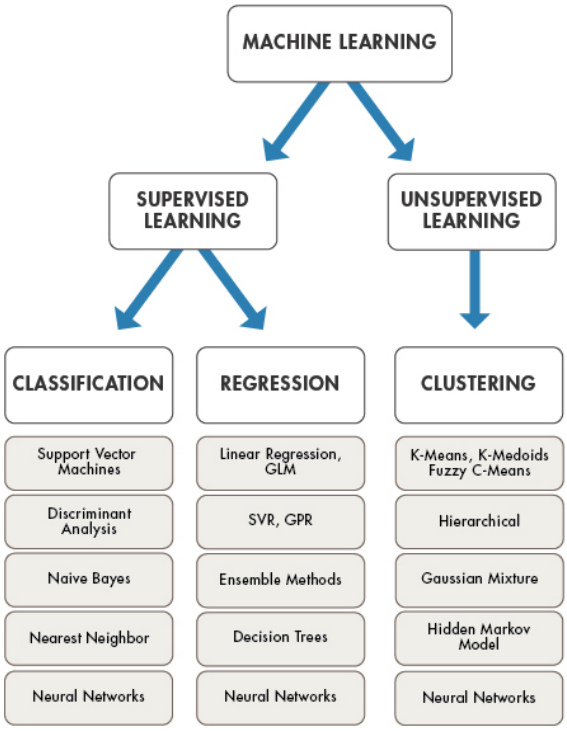

Figure 1.5: Examples of machine learning methods.

Supervised / unsupervised learning. Among different ways to discover patterns and extract information, machine learning focuses on the design and development of algorithms with which machines gain the capability to learn without being explicitly programmed. Machine learning can further be divided into three categories:

1. Supervised learning.
2. Unsupervised learning.
3. Semi-supervised learning.

In this context, supervision is defined as problems where inputs and outputs are explicit, and the goal is to map the input to the output. Learning then comes from the evaluation of the machine's output against the true output. Conversely, problems where the output is not known fall into the unsupervised learning problems. Datasets contain data points that are neither classified nor labeled, and the goal is then to model the organization or distribution of the data. Finally, semisupervised learning uses datasets containing both labeled and (usually the greatest part) unlabeled outputs. This is often the case in real world scenarios, when some information is either inaccessible, too costly, or necessitating labeling from experts. An example in biology is for example the specifying the three-dimensional structure of proteins by experiments.

Applying machine learning with Data mining. Data mining is a key step in the process of knowledge discovery. It refers to the extraction or mining knowledge from large amounts of data, and typically consists of the following steps:

1. **Data cleaning** to remove noise and inconsistent data.
2. **Data integration** to combine multiple data sources.
3. **Data selection** to retrieve the relevant data among existing sources.
4. **Data transformation** to transform the data for analysis.
5. **Data mining** to extract patterns in the data.
6. **Pattern evaluation** to measure and identify relevant patterns.
7. **Knowledge presentation** to visualize and represent mined patterns to the users.

We can consider the four first steps as pre-processing, while the last two form the post-processing procedure. The data mining step and the different techniques used to efficiently uncover new information are usually the most discussed upon. However, both the pre- and post-processing steps are crucial in data mining, especially in the current context where we have access to larger and larger amounts of data. In many cases, breakthroughs in machine learning come from a well-thought pre-processing step, where particular attention has been taken in the quality of the databases used. In the biological context, this step is essential and often challenging. As presented in the previous chapter, access to very large amounts of data is now widespread (in our case sequenced metagenomics data, protein databases). The difficulty now resides in processing this information, and adding relevant knowledge to it. Another important challenge is how reliable the data is, and to what extent it is possible to extract knowledge based on partial or sometimes contradictory data. In other words, whatever the design used for the data mining, its interpretation can vary depending on the initial assessment of the biological data.
The data mining step can greatly vary depending on the problem of interest, such as characterization, associations, classification, clustering or prediction, and come from different fields like statistics, big data technologies, or deep learning. Post-processing is of course another important step, because it impacts the final interpretation. The challenges are in a sense similar to that of pre-processing, because of the intrinsic nature of the biology field.

General notation. For a purpose of clarity, I will introduce several general notations that will be used throughout the presentation of neural networks in this introduction. Given a neural network, we define:

- $\mathbf{w} = [w_1 ... w_n]$ the vector of n weights.

- b a distinguished weight called the bias.
- x the vector of inputs.
- f the output function, corresponding to the class attribution by the network.
- σ the loss function.

Neural network principles. Artificial neural networks (ANNs) are a subset of machine learning, stemming from a biologically-inspired programming paradigm which enables a computer to learn from observational data. ANNs are comprised of nodes forming an input layer, one or more hidden layers, and an output layer. Each node n_i is connected to one or several others, and has an associated weight w_i and a threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. Two classes of neural networks exist, differing in the way information is transmitted from layer to layer.

Feed-forward neural networks allow a signal to travel in one and only one direction, meaning that the output of any layer does not affect nodes of the same layer. In general, machine learning for these types of networks follow the following steps. In a first step, training data is fed to the input layer. It then passes through the succeeding layers until the information reaches the output layer of the network. The training process consists in adjusting the weights and thresholds until the output layer yields a stable and accurate output according to specific measures of performance. In the context of supervised learning, the weight and threshold adjustments are set in order to have a consistency between the network's output and the output labels. A typical example of fast-forward network is presented in Figure 1.6.

Feedback or recurrent neural networks do not follow that rule. Instead, signal can travel both ways through the network, which introduces loop if we see the network as a directed graph (see Figure 1.7). Training these types of networks can be more challenging, as the nodes' state continues to be adjusted until an equilibrium is found. In that sense, feedback networks are dynamic, and can be seen as bringing some kind of memory since weights and thresholds can be updated using information from earlier inputs.

A simple example: the perceptron. Neural networks can be classified into different groups, depending on the purpose they are used for. The simplest neural network is the perceptron, which consists only in a single neuron, a vector of inputs and single output. This corresponds to a linear classifier, since it is the dot product of a weight vector with a feature vector, and more specifically a binary classifier because there is a single output. The perceptron predicts whether the input belongs to a certain class or not:

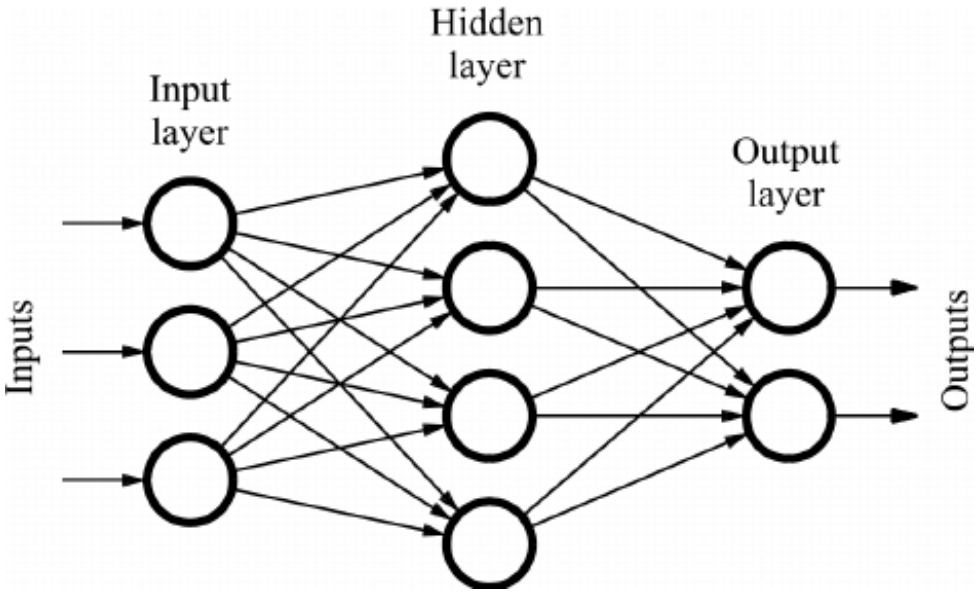

Figure 1.6: Example of a feed-forward neural network.

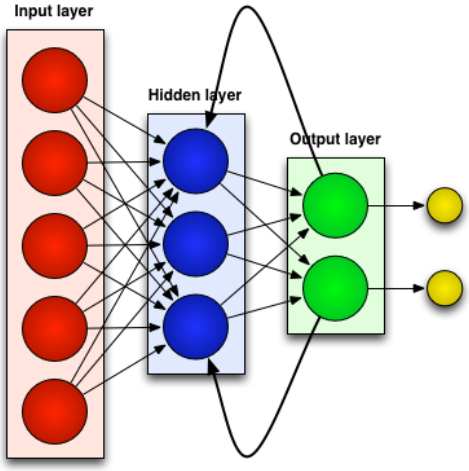

Figure 1.7: Example of a recurrent neural network. The data from the output layer nodes are sent to the hidden layer.

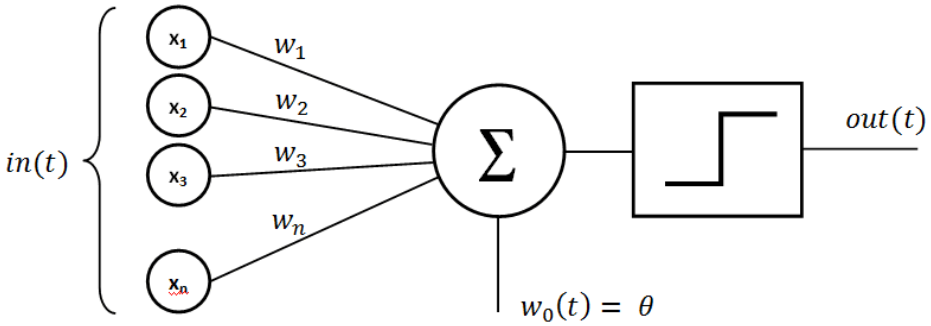

Figure 1.8: $x_1, x_2, .., x_n$ are the n input data of the perceptron. The only neuron in the hidden layer combines the weights $w_1, w_2, .., w_n$ to output a binary value. This output is the prediction of the network for the input data to belong to a certain class, or not.

More specifically, the function f computed by the neuron and corresponding to the binary class attribution is the following:

$$f(x) = \begin{cases} 1 & \text{if } b + \mathbf{w} \cdot \mathbf{x} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

We define *linear units* $\sum$ as the elements computing the dot product $b + \mathbf{w} \cdot \mathbf{x}$. The perceptron is the simplest example of binary classification, and we will use it as illustration for further definitions. Changing the classification into a multi-class decision problem can be done by adding m independent neurons as outputs instead of one and choosing the class with the highest output value. For example, classifying numbers from one to ten would imply using 10 perceptrons, with the ouput of the classifier corresponding to the sum of all the perceptrons outputs.

Training, validation and test sets

All machine-learning problems require at least, and preferably three sample sets to work with:

1. **The training set**, which contains the examples that are going to be used to adjust the parameters of the network.
2. **The validation set** which is used to test the model, and modify if needed the the network's architecture.
3. **The test set** to measure the final performance of the network, once the model is fixed.

Separating the examples into these three sets is important for several reasons. Firstly, performance in machine learning is achieved via minimization of a cost function. The evaluation of that performance, assessed via the test set, can then be biased if the machine learns from these examples. In that sense, in having distinct training and test sets, the model is prevented to learn from test examples. Secondly, another bias can occur if the model is modified to improve the results, because its architecture may become fitted to the specific examples in the test set. Avoiding this issue implies a further separation into validation and test sets. Thirdly, another data bias can stem from an unrepresentative selection of the training examples. Overcoming this problem is especially challenging in the biological context, because biological data comes from experiences that are usually biased towards a small set of model organisms. Another aspect of this bias is the representativeness of the examples into the classes they belong to. This is especially challenging in our protein-protein interaction context, because negative and positive interactions don't have the same value. Proof of interaction between two proteins is a challenging problem that is the subject of many experiences. However, it is often even more difficult to prove that two given proteins cannot interact. This leads to either unbalanced sets of positive and negative examples in the training set, or weaker

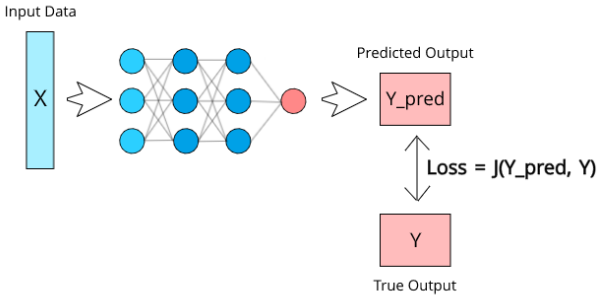

Figure 1.9: Given an input data and a network, the loss is computed by calculating the difference between the predicted output in the true ouput (that is given by the labeled examples).

assumptions for negative examples. In general, avoiding these biases is one of the more complex aspects of machine learning, which is sometimes overlooked. *Overfitting* corresponds to the possibility that the model learns the details of the learning examples rather than its underlying structure. This negatively impacts the model's ability to fit to any new data, and thus its reliability for predicting future observations.

Learning parameters, hyper-parameters

Since the goal of the network is to assign a class to the input examples, the difference between the actual attribution and the expected result must be quantified in order to adjust the weights of the network accordingly. A *loss function* is a function from an outcome to how it differs from the expected result (see Figure 1.9). The goal is then to minimize this loss. A straighforward way to do this is with *gradient descent learning* which is done by modifying the parameters according to the rule:

$$\Delta_{w_i} = -L\frac{\delta l}{\delta w_i} \tag{1.2}$$

where l is the loss, and L the learning parameter that scales how much we want a parameter to change at a given time. The most popular loss function is called the *cross-entropy loss* function. For this, a generalization from the logistic output function to a probability distribution is needed. Hence the definition of the *softmax* function:

$$\sigma(\mathbf{x})_j = \frac{e^{x_j}}{\sum_i e^{x_i}} \tag{1.3}$$

Using the softmax function, the cross-entropy function X is then the negative log probability that the class assigned to x is a:

$$X(\mathbf{x}, x) = -lnp_{\mathbf{w}}(a_x) \tag{1.4}$$

The gradient descent algorithm would then have all the weights adjusted with the following rule:

$$\Delta_{w_{i,j}} = -Lx_i\frac{\delta\Phi}{\delta_j} \tag{1.5}$$

Batch learning, epochs

In this manner, all the parameters are adjusted at the same time, when a complete pass through all the training examples has been made. In practice, doing so would be slow, even more so if the dataset is large. For that reason, batch learning breaks the training set into smaller chunks of m examples (denoted as *batch size*, after each of which the weights are updated. This allows for a faster and more flexible training, and is called *stochastic gradient descent* Moreover, the training set is usually passed through the network several times, in a random order so that weights are learned differently. Each pass through the whole dataset is called an *epoch*. Increasing the number of epochs progressively improves the model's performance, until a certain point. In practice, selecting batch sizes, learning rate and other types of learning parameters can depend on the problem to be solved, and leads to other testing options. For example, a lower learning rate brings smaller changes to the weights, so that a better local loss reached, but at the cost of a slower training.

Filters as Matrices. In practice, we represent layers by using matrices. We then represent a filter W of size (l, m) by the following matrix:

$$\mathbf{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,m} \\ w_{2,1} & w_{2,2} & \dots & w_{2,m} \\ & & \dots & \\ w_{l,1} & w_{l,2} & \dots & w_{l,m} \end{pmatrix} \tag{1.6}$$

where $w_{i,j}$ are the weights in that layer.

Convolutional Neural Networks. We can consider two categories of neural networks:

1. *fully connected networks*, where all linear units in a layer are connected to all the linear units in the next layer.
2. *partially connected networks* that are not.

Belonging to the latter category, a special case is **convolutional neural networks (CNNs)**, which have been first applied to computer vision, but are now extensively used everywhere. In our deep learning context, a convolution corresponds to the the dot product between a filter K and an equal sized input patch I, and is defined as the function:

$$V(x, y) = (I \cdot K)(x, y) = \sum_m \sum_n I(x + m, y + n)K(m, n) \tag{1.7}$$

This implies a partial connection between layers in CNNs. In the computer vision environment, this can be seen as a local descriptions of the input image. If we transpose this idea to a CNN where inputs are biological sequences, this expresses the idea that some feature

can represent substrings of the input sequence. Local description is the main aspect of CNNs, and where they differ from fully-connected networks. They are usually composed of:

- Convolutional layers, as described previously.
- *Pooling layers* that reduce the dimensionality of the input layers by downsampling. A particular example is the max-pooling layer, which calculates the maximum value for each patch of the input.
- Nonlinearity, generally a *Rectified Linear Unit* (ReLU). Convolutional layers being linear functions, their purpose is to add non linearity to the model to make it possible to reprensent non-linear data. The corresponding function f_{ReLU} is defined by:

$$f_{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise.} \end{cases}$$

- Fully-connected layers, usually placed as last layers, as they are suited to classification and regression tasks.

Figure 1.10 summarizes the different typical components of a CNN.

Data representation

Inputs and outputs for Neural Networks. In Bioinformatics, an increasing number of deep learning architectures has been proposed in the last years. This is due to the fact that deep learning is well adapted to handling large datasets, finding hidden structure within them, and for making accurate predictions [2]. Among the various field where this approach has been used, we focus on *protein-protein interaction* (PPI) prediction. In these studies, the basic inputs are the two protein sequences, but architectures are not limited to only amino-acid sequences. In fact, any type of additional information can be added as input, provided it is relevant, and preferably not redundant with already existing feature. Such additional information includes:

- k-mers [28].
- structural information [54].
- hydrophobicity of the amino acids [16].
- multiple sequence alignment or protein profiles [30].

Deep learning approaches for the PPI problem based on sequences. Among the variety of deep learning approaches for PPI prediction, we present and compare our tool to :

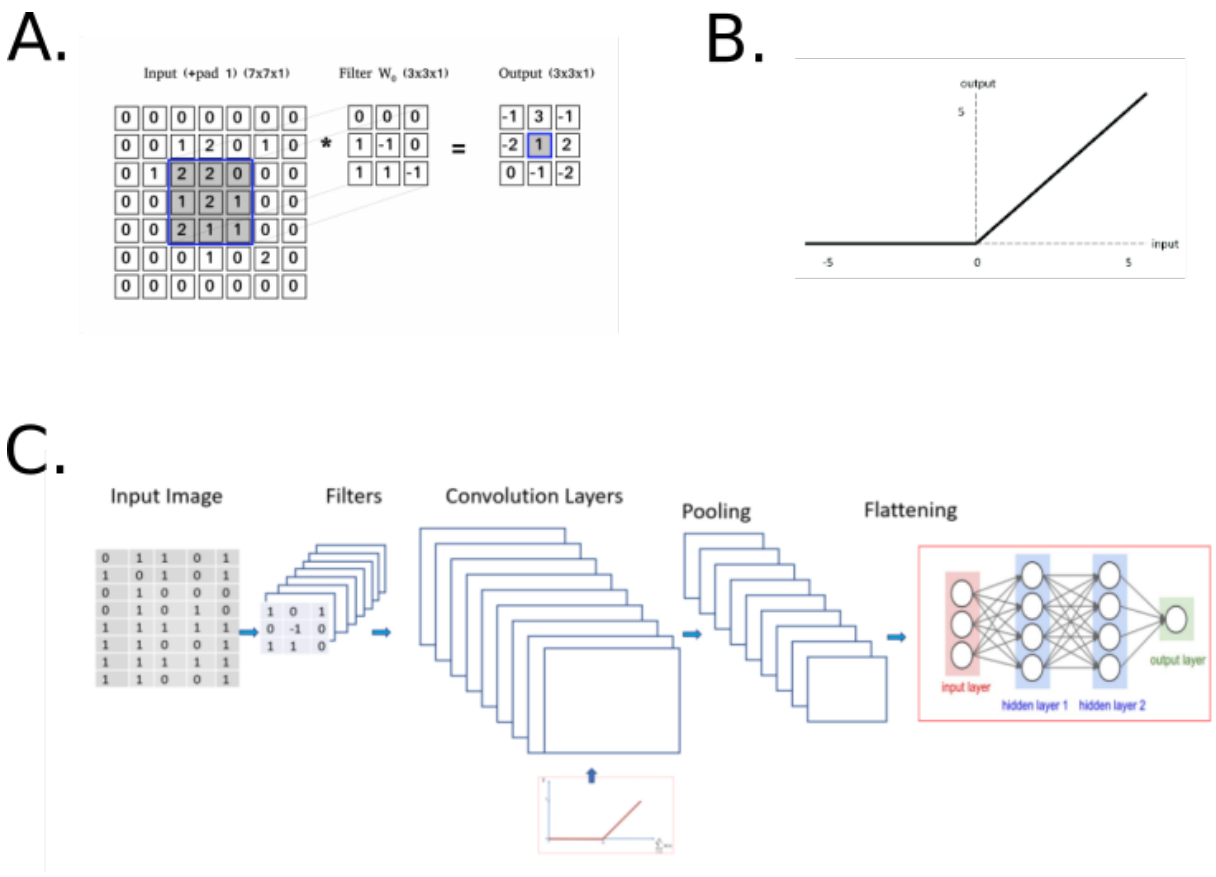

Figure 1.10: **A. A convolutional filter of size** $(3, 3)$**.** The result of the convolution between the blue input patch and the filter W is the blue square in the output. **B. The ReLU function. C. An example of CNN architecture.** This architecture used several convolutional layers followed by ReLU and pooling layers, and ends with a fully-connected layer to give a binary classification from an input image.

- **DPPI** [30], a CNN using protein profiles to predict PPI. It uses a Siamese neural network, which is is a class of neural network architectures that contain two or more identical subnetworks that shares weights. This is especially well-suited for our problem, as the input is composed of two proteins.
- **PIPR** [11], an efficient and robust Siamese Residual CNN taking only the protein amino-acid sequences as inputs.

Protein protein interaction databases. PPI databases can be sorted into two categories:

1. Canonical databases that have been entirely made by the authors
2. Complex databases, regrouping multiple canonical databases.

Among the most used databases, we describe the following ones:

- **Intact**, a protein interaction software and database which houses PPI models and their analysis [35]. Data are accumulated from peer-reviewed journals and are manually annotated by expert curators.
- **The Database of Interacting Proteins** (DIP) [69], is a biological database that aims to document protein-protein interactions which are experimentally determined. It contains more than 80,000 proteins (28,850 interactions) from human and bacteria.
- **The Human Protein Reference Database** (HRPD) [68] regrouping 30,000 proteins entries, and has been manually extracted from the literature by experts.
- **the Human Integrated Protein-Protein Interaction Reference** (HIPPIE) [48]. A core component of HIPPIE is the confidence scoring of interactions based on the amount and reliability of evidence supporting them. It integrates data from more than 10 source databases (among which BioGrid, DIP, HRPD), and contains 273,000 interactions.
- **the Negatome** [52], a collection of protein and domain pairs unlikely engaged in direct physical interactions.

Positive/negatives examples, Redundancy reduction

These databases regroup what constitutes positive examples, meaning evidence of interaction. However, machine learning networks also need negative examples for classification. This is even more pressing in the case of PPI prediction, where the total number of possible interactions is quadratic in the proteome size. This would mean examples of proteins pairs where there is evidence that they do not interact. As said previously, this criteria is difficult to match, because some proteins can interact up to a certain level. This raises another challenge, which is either to:

1. rely on few true negative examples (where there is a strong evidence of non-interaction). This is for example provided by the Negatome, but the limited size of its database implies a strongly unbalanced training set.
2. broaden the spectrum of what can be considered as negative examples to have a more balanced set, the downside being the quality drop of the learning dataset. A commonly used strategies for gathering negative examples is to consider that two proteins do not interact if they do not belong in the same sub-cellular unit.

Statistical Validation

The cause of poor performance in machine learning is either overfitting or underfitting the data. By definition, generalization refers to how well the patterns and regularities learned by a

model apply to new specific examples. Overfitting corresponds to a scenario where the model is adjusted to closely to the training data.
In general, several several strategies are used to avoid overfitting, or to evaluate how well a network can generalize new data, after it has been trained. We will present the strategies we have applied to our network.

Redundancy reduction. A widespread strategy to prevent the network to be too specific is adding a pre-processing step to the data preparation. Input sequences from the training set are clustered, so that representative from each cluster do not share more than a certain level of similarity/identity (for example 40%).

Hyper-parameter validation. As described previously, it is important to keep a distinct dataset for validation. This means that this validation set is used to optimize hyper-parameters. Hyper-parameters are parameters whose value are used to control the learning process. This can be the size of filters, their number, etc. A risk of not using validation is to use the test set to modify the network's structure, implying that not only the training but also the test set have been used to optimize the network.

Classes of interaction. In the PPI context, one way to control whether a network is generalizable is to use 3 classes of interactions. They are defined by [41] in the following manner. C1 denotes that both proteins involved in the interaction were used in training, C2 that only one protein was used in training, and C3 that none of the proteins has been used in training (see Figure 1.11). Note that a significant drop in performance for class C3 would indicate either over-fitting or an insufficient propensity of the model to predict new interactions. If the network performs well for the C1 class examples, but poorly for the C3 class in particular, then there is a risk that it will perform equally poorly if examples unknown to the network are presented for prediction.

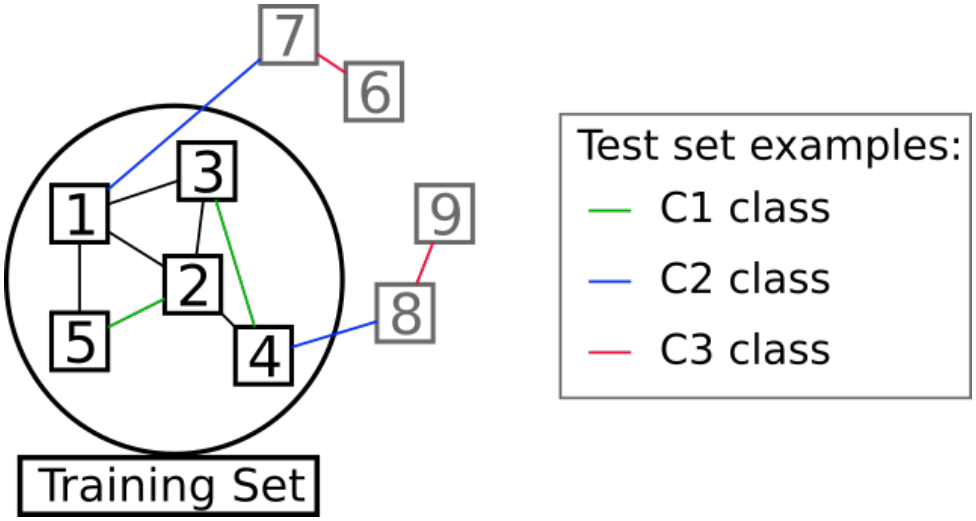

Figure 1.11: An example of class division. Within the test dataset, the interaction between proteins 3 and 4 belong to class C1 because both proteins were in the training set. On the contrary, proteins 6 and 7 were not, meaning that the interaction between both proteins will be categorized in the C3 class.

Chapter 2

Targeted domain assembly for fast functional profiling of metagenomic datasets with S3A

In this chapter, I will present S3A, a fast and accurate domain-targeted assembler designed for a rapid functional profiling. It is based on a novel construction and a fast traversal of the Overlap-Layout Consensus graph, designed to reconstruct coding regions from domain annotated metagenomic sequence reads. S3A relies on high quality domain annotation to efficiently assemble metagenomic sequences and on the design of a new confidence measure for a fast evaluation of overlapping reads. Its implementation is highly generic and can be applied to any arbitrary type of annotation. On simulated data, S3A achieves a level of accuracy similar to that of classical metagenomics assembly tools while permitting to conduct a faster and sensitive profiling on domains of interest. When studying a few dozens of functional domains - a typical scenario - S3A is up to an order of magnitude faster than general purpose metagenomic assemblers, thus enabling the analysis of a larger number of datasets in the same amount of time. S3A opens new avenues to the fast exploration of the rapidly increasing number of metagenomic datasets displaying an ever-increasing size. In this context, S3A provides a fast exploration of large datasets. While it was designed for metagenomic sequences, its usage is not limited to them, as long as some kind of read annotation is included.

Availability `http://stackoverflow.com/` The development of S3A has led to the publication of an article in Bioinformatics (`https://academic.oup.com/bioinformatics/article/36/13/3975/5824791`). The source code for S3A is openly accessible on the LCQB website dedicated page, as well as prerequisites and code examples (`http://www.lcqb.upmc.fr/S3A_ASSEMBLER/`).

2.1 Context

In metagenomics, annotation is hampered for shorter sequences of 100-150 bp in length - common with current technologies - thus making sequence assembly a prerequisite for any improvement. In this context, a good-quality assembler is necessary, as it increases the length of assembled coding regions. The sheer size of metagenomic datasets typically requires huge time and memory resources when doing *de novo* metagenome assembly [39]. Thus, several strategies have been proposed to perform a targeted assembly [71, 65], based on a preliminary protein domain annotation followed by a domain-guided assembly.
Domain targeted assembly has a second major advantage. Indeed, it can be restrained to a limited number of domains, from a few 10's to the 100's, providing a fast way to "explore" many large metagenome datasets with a given hypothesis in mind. Metagenomics studies are usually interested in understanding one given function or biochemical pathway across multiple conditions or samples, and, in practice, only a limited number of domains (a few dozens) needs to be annotated when profiling. Examples range from the annotation of RNA transcripts in extreme environments [9], to a particular biochemical reaction in the gut microbiota [64, 55], to the detection of antimicrobial resistance [31]. Various targeted assemblers were proposed for performing this task. They can either perform an assembly around an identified domain [71] or annotate domains after reads' clustering [67, 33]. On very large datasets, the first are unable to scale and the latter are excessively slow. To overcome this limitation, S3A combines a step of fast reads clustering (using BCALM 2 [13]) with an efficient assembly performed from domain annotation. S3A is in practice as accurate, more sensitive and up to one order of magnitude faster than existing targeted domain assembly tools like the SAT assembler [71]. It is slightly more precise than the Xander assembler [65] showing the same computational efficiency on up to 100 domains. It is on par with traditional assemblers, such as Minia [14], when considering up to 100 domains. When considering realistic metagenomic dataset analyses on a few dozens of domains, S3A can, in the same running time and final accuracy as a metagenomic assembler, annotate 6 to 8 times more samples.

2.2 The S3A approach

S3A is a tool for targeted domain search in metagenomic datasets. It is designed as an assembly algorithm of annotated reads addressing the problem of reducing the time complexity of the OLC graph construction step, the bottleneck of targeted assembly. The S3A flowchart is depicted in Figure 2.1. It starts from a dataset of metagenomic reads, performs a preprocessing of the reads to reduce the size of the set by constructing protigs, through the detection of Open Reading Frames (ORF), their assembly in unitigs and a mapping of the nucleotide sequence into an amino acid sequence. Then, it parses protigs for a protein domain

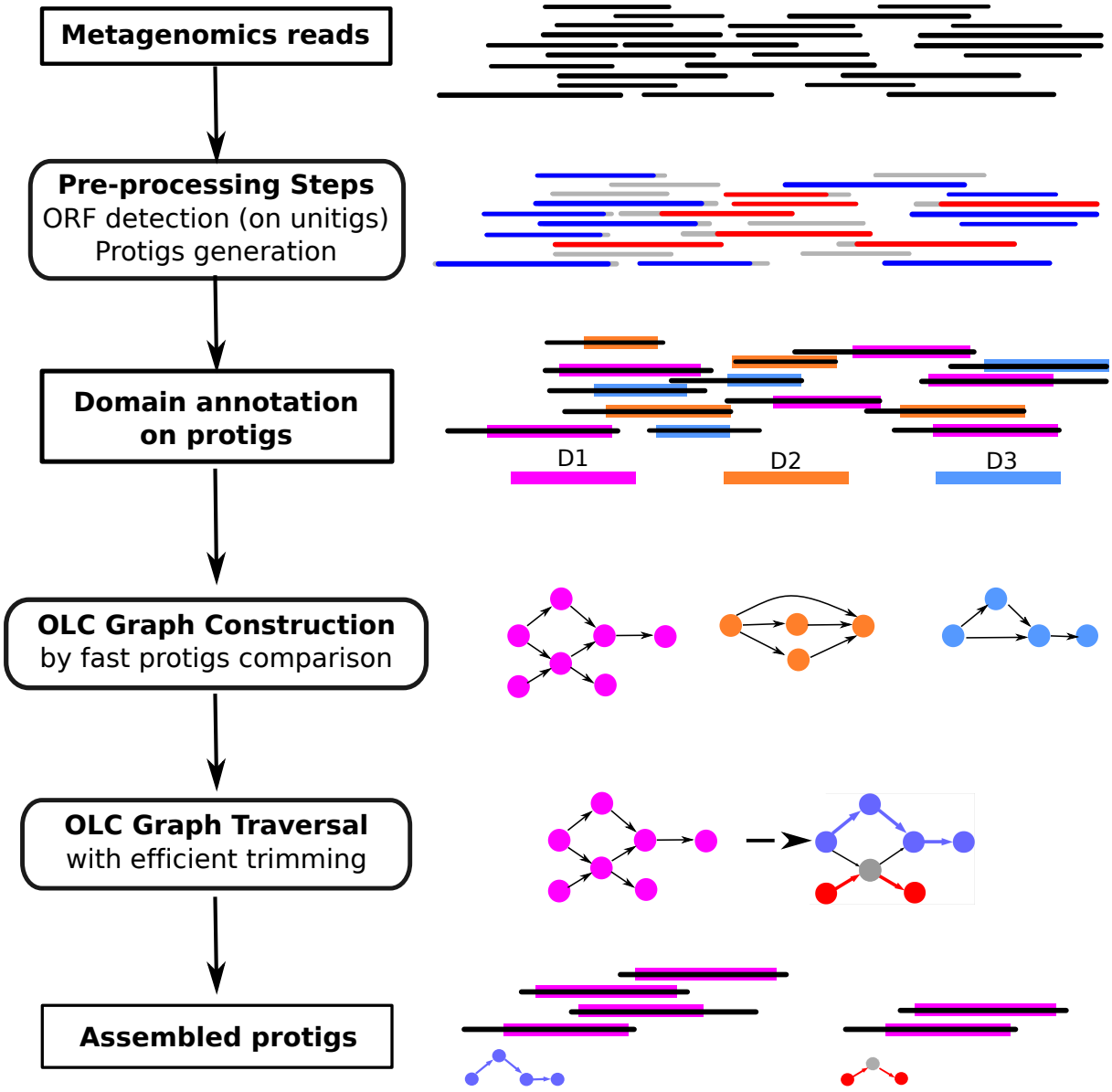

Figure 2.1: S3A flowchart. From domain annotated protigs, obtained by reads' pre-processing (ORF identification, unitig generation and mapping into amino acid sequence), S3A performs an efficient protigs comparison and builds an OLC graph. A graph traversal approach based on an efficient resolution of alternative paths combining two measures of sequence overlap, allows to assemble protigs.

annotation and it constructs the OLC graph, in the amino acid sequence space, based on the overlap of domain annotations: an OLC graph is a directed graph where each node corresponds to a protig, and each edge to an overlap between two protigs. Based on two metrics, used to identify unreliable edges and prune the graph (the longest matching substring length, $lmsl$, and the percentage of identity, ip; see Methods), this step performs a depth-first search of the graph, called "graph traversal", to identify and assemble a set of overlapping protigs representing consensus DNA regions surrounding protein domains, referred to as contigs. S3A outputs a set of contigs for each targeted domain.
S3A exploits functional domain annotation as a first indicator for protig overlap, making OLC graphs a central choice for targeted assembly. The construction of the OLC graph in S3A is different from the traditional one used for sequence assembly, where read pairwise alignments are evaluated by the Hamming distance of the overlapping region. Instead, in S3A, the quality of overlapping regions is evaluated by the two fast computable measures $lmsl$ and ip. An OLC graph is also different from a de Bruijn graph, used in (non targeted) assembly algorithms, which replaces every read with the corresponding set of k-mers [70].

2.2.1 S3A key features

The basic choice of S3A to assemble domain annotated sequences is important for directly deriving the functional annotation of the metagenomic sample. The second main motivation to considering domain target assembly is the reduction of the time complexity while retaining the highest accuracy possible. Indeed, by separating and ordering reads by domain in the pre-processing step, the number of read comparisons is highly decreased and the general algorithm performance is greatly improved.
S3A evaluates overlapping reads sharing a common domain annotation, on the basis of two metrics, $lmsl$ and ip. These metrics provide an overlapping confidence measure that is both complementary and much faster than counting a fixed number of mismatches by dynamically computing an edit distance, as done by other targeted assemblers like SAT [71]. They also allow for a tailored OLC graph trimming which is independent on the sequencing technology used and helps reducing the graph complexity. Moreover, $lmsl$ is used to resolve ambiguous cases in the absence of transitive edges, and to select the most reliable transitive edges in the OLC graph (Figure 2.3).
S3A might create complex OLC graph structures due to *multi-branching nodes*, that is nodes with multiple entry and exit edges. In the absence of transitive edges, multi-branching nodes are considered unreliable and therefore removed, the goal of S3A being to be as accurate as possible.
Most importantly, the possibility to annotate a reduced set of domains and assemble only reads involving these domains, allows for a fast exploration of metagenomic datasets allowing the user to concentrate on specific functional targets.

2.3 Methods

2.3.1 Domain hit

Given a sequence r annotated with a given domain d, a portion or all of r will match to the domain. We define the domain hit region for r as the start and end positions of the sequence matching interval, relative to the whole domain (denoted s and e). Domains annotation is realized on protigs, that is amino acid sequences generated by a preprocessing step that identifies ORF regions in metagenomic data (see Fig. 2.1 and section "Data pre-processing").

2.3.2 Data pre-processing

Metagenomic sequences are prepared before assembly using three main steps. First, an ORF prediction is realized with FragGeneScan [44], checking both the forward and the reverse strands of a read. Second, predicted ORF sequences are assembled into unitigs obtained with BCALM 2 [13], where a unitig is a local sequence assembly whose overlap are not disputed by any other data. This step reduces greatly the time needed for domain annotation. Third, the translation of the nucleotide sequence in amino acid sequence is followed by a functional domain annotation realized with HMMER [19] or MetaCLADE [57]. Any type of annotation can be used. Sequences remaining without domain annotation or annotated with more than one domain are discarded.
As a result, S3A performs the assembly of a set of amino acid sequences coming from coding regions and annotated with functional domains. They originate either directly from reads corresponding to ORF sequences or from unitigs constructed from ORF sequences, which we called protigs.

2.3.3 The S3A algorithm

S3A consists of two main steps, retracing the architecture of SAT [71]: the OLC graph construction and the OLC graph traversal. However, the corresponding algorithms are significantly different. The OLC graph construction combines domain evidence with a fast estimation of the protigs overlap, and the OLC graph traversal uses an efficient dynamic programming algorithm based on edge weights to guide the traversal more efficiently. These two steps are described in details below.

OLC graph construction

Let M be the number of domains involved in the annotation process. We consider each protig to have a single domain annotation, protigs not having a single domain annotation are discarded. Protigs are first grouped in a set of M hashtables $\{\mathcal{H}_1, \ldots, \mathcal{H}_M\}$, one per domain.

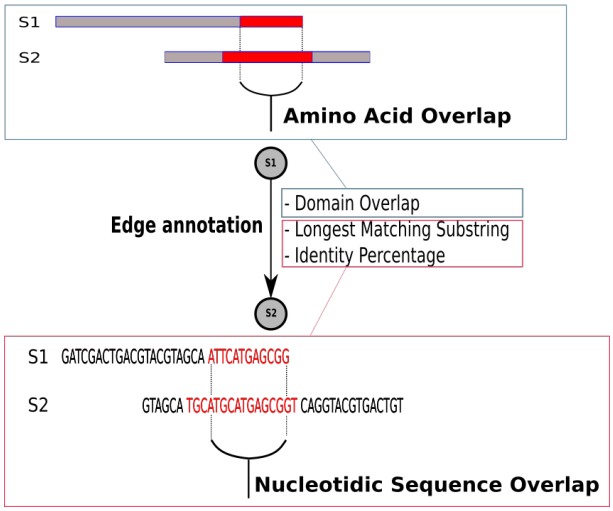

Figure 2.2: Reads Comparison. Reads belonging to the same hashtable H i are aligned on the corresponding domain. In this example, S1 and S2 have been aligned to the same domain. The likelihood of an overlap is estimated by the ip and lmsl values..

For each hashtable $\mathcal{H}_i$, protigs are sorted according to their s (start) and e (end) position on domain i. This data organisation, sorting protigs by their matching position on the domain within domain specific hashtables, breaks down the OLC graph construction to a simple interval traversal algorithm that avoids comparing each possible pair of protigs. Namely, no comparison between pairs of protigs 1. annotated by different domains, nor 2. having no domain overlapping (where two protigs have a *domain overlap* if their respective domain hits overlap; Figure 2.2, top) is needed.

The OLC graph is constructed by creating an edge between each pair of protigs (r_i, r_j) that overlap by more than c amino acids on the same domain (Figure 2.2). To each edge (r_i, r_j), we add two metric values computed from r_i and r_j nucleotidic sequences: the length of the longest matching substring ($lmsl_{i,j}$) and the percentage of identity between r_i and r_j on the domain overlap region ($ip_{i,j}$). We prefer the use of $lmsl$ and ip over more precise ones (*e.g.* edit distance), as they estimate sequence similarity much faster (the implemenation used for computing the edit distance is based on the Ukkonen algorithm).

By construction, the OLC graph is directed, but not necessarily acyclic. Ideally, each occurrence of a domain in a gene should give rise to a path in the graph. Thus, a graph pruning step will make the graph acyclic and a graph traversal step will identify contigs by traversing the graph from each *source* node (nodes without predecessors), and by using scoring paths according to the $lmsl$ values stored on the edges.

However, sequencing errors and sequence similarity between genes and species can create ambiguities in the traversal and lead to multi-branching nodes (nodes with at least 2 predecessors and 2 successors).

To help solve those ambiguities, we identify *transitive edges*, edges that connect two nodes which have an alternative path joining them. These edges can be removed without losing

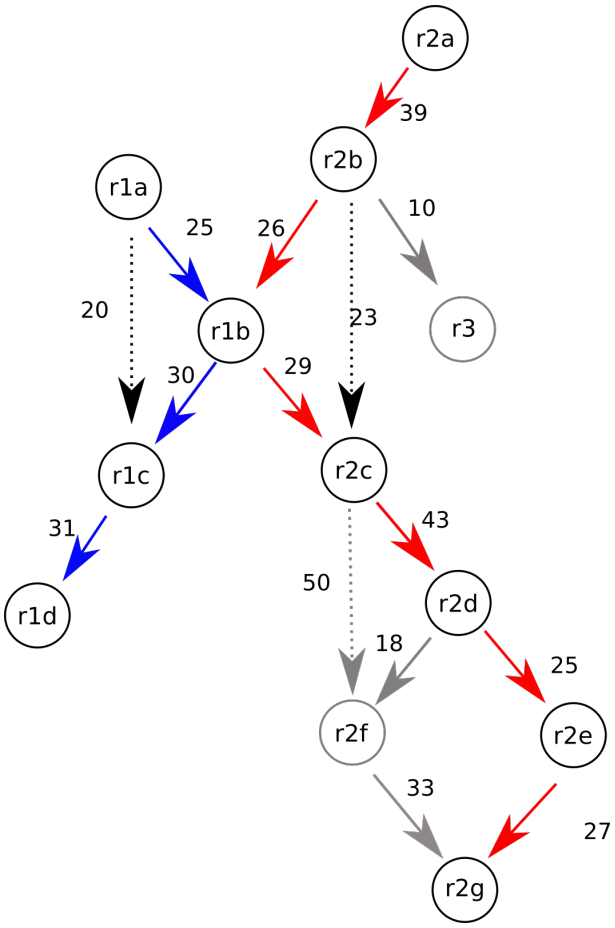

Figure 2.3: A toy model illustrating graph pruning and graph traversal for the generation of two contigs by resolving a multi-branching node. Each edge is weighted with the $lmsl$ of its nodes. We fix a threshold of 15 for acceptable $lmsl$. Note that, for sake of simplicity, the weights are not normalized in this toy example. Graph pruning step: the node r3 is removed due to the $lmsl$ threshold (see grey edge and grey node). Edges (r1a,r1c) and (r2b,r2c) are identified by transitive reduction and placed in the transitive dictionary, since their $lmsl$ weight is not stronger than the $lmsl$ weight of any edge in the corresponding path (see edges in dotted lines). Note that the weight of (r2c,r2f) is too high to place this edge in the transitive dictionary and consider it further (grey arrow). Graph traversal step: the two source nodes r1a and r2a are considered for traversal, one after the other (their order is chosen arbitrarily). First, the contig (r1a, r1b, r1c, r1d) (blue path) is extracted, the path ambiguity at node r1b being solved thanks to the (r1a,r1c) transitive edge. The same consideration applies to (r2a, r2c) where the transitive edge (r2b,r2c) helps to resolve the ambiguity at node r1b. r2d is a branching node with no transitive edge ((r2c,r2f) has been removed in the graph pruning step), hence the path with highest $lmsl$ weight is chosen. The second contig is thus (r2a,r2b,r1b,r2c,r2d,r2e,r2g) (red path). Note that r1b is a *multi-branching node*, indicating that the corresponding protig should not be part of two contigs; the presence of transitive edges allows for an unambiguous traversal.

information for the traversal, but are kept in a separate data structure $\mathcal{T}$, as they can help resolve ambiguities raised by multi-branching nodes (see Figure 2.4).

Graph pruning and traversal

Once the OLC graph is built, contigs can be generated from its traversal. The graph traversal is preceded by a pruning phase that removes unreliable edges based on $lmsl$ and ip values (step 1), and enforce an acyclic graph (step 2). The graph is then simplified by merging linear paths (step 3) and transitive edges that have no impact on the traversal are efficiently removed (step 4).

Transitive reduction The transitive reduction (step 4) is based on Depth First Search. For each node u in the graph, a DFS is performed on its successors. If a node w from the visit has an edge linking it to u, it is a transitive edge, and can be added to $\mathcal{T}$. The idea is to determine whether the transitive edges will correctly guide the traversal. The lms weights are used to estimate the likelihood of an edge to be correct: the lms of a potential transitive edge (in red) is compared to the smallest lms along its alternative path. If the latter is smaller than the potential transitive edge lms, we do not consider it to be reliable enough, and discard it without adding it to $\mathcal{T}$, as shown in Figure 2.4. The rationale of considering the lowest score along the path is illustrated in the given toy model. The edges (u, v) and (v, x) and (x, w) should all bring a greater lms than (u, w). If it is not the case, the transitive edge (u, w) is necessary to guide the traversal, and node v probably doesn't belong to the current assembly. In this case, it would mean that the transitive edge should not be removed, as it gives a higher evidence of overlap than all the edges forming the original path.

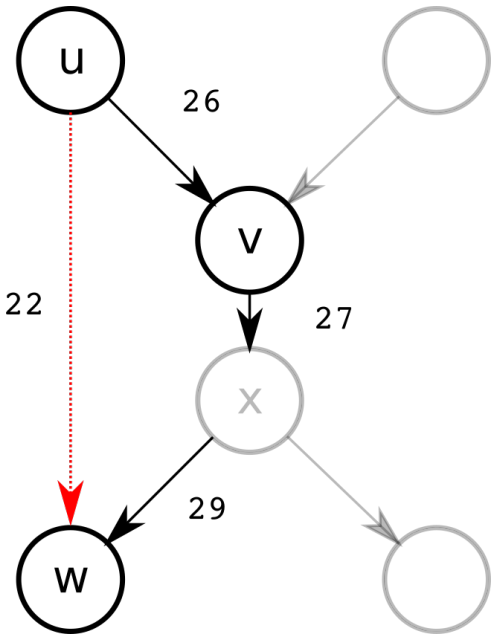

Figure 2.4: **Transitive reduction procedure.** In this toy model, the transitive edge has a smaller lms than any other edge of its alternative path, meaning that it will effectively be added to $\mathcal{T}$.

The resulting structure is a directed acyclic graph with N sources nodes. N traversals are

finally performed in order to build output contigs.
The graph is then visited in a depth-first manner, using transitive edges to resolve branching during the graph traversal. During the traversal, when a *multi-branching node* v is visited, $\mathcal{T}$ is queried to look for transitive edges linking a predecessor u to one of its successors w. If such an edge exists, the traversal will be guided to the path containing both v and w. If not, v is removed, which implies that edges joining any such node to the rest of the graph are discarded as well. As a result, its successors are therefore considered as additional source nodes of the graph.
Bubbles are other kinds of topologies that can lead to errors during the traversal. Bubbles exist in the graph when, given two nodes u and v, two alternative paths starting from u and ending in v exist. In that case, if no transitive edge exists, the ambiguity is resolved by removing the path containing the edge with the smallest $lmsl$ value.

Parameters' default values

As default values, we used a minimal domain hit length of 20 amino acids for the protigs, an ip threshold of 80% and an $lmsl$ threshold 0.2.
The minimal domain hit length has been chosen to be the same as in the SAT assembler. Since we annotate protigs, which are longer in average than reads, the threshold of 20aa is usually satisfied by domain annotation. A lower value would increment the number of false positives.
To compute the $lmsl$ between two overlapping protigs, we normalise the longest matching substring length by the mean length of the protigs obtained for the metagenomic dataset under analysis, allowing the default threshold to be used with reads coming from whatever sequencing technology (Illumina, 454). Figure 2.5AC (see highlighted nodes in the green curves) shows that for our datasets, a threshold of normalized $lmsl$ at 0.2 gives high precision and acceptable sensitivity.
The ip threshold has been set to 80% because a significantly different percentage proves to be either too lax or too strict (Figure 2.6). Figure 2.5AC and Figure 2.6 demonstrate that S3A's best performance is achieved around the default values.

Analysis of the time complexity

Let R be the number of annotated protigs, and M be the number of domains. If we consider R_M the number of protigs for the same domain, then only protigs annotated from the same domain are compared. Moreover, the protigs are sorted according to their starting position on the domain which gives an overall $O(R \log R_M)$ time complexity for the graph construction.
The graph traversal is based on the DFS algorithm, whose complexity is $O(|V| + |E|)$, where $|V|$ is the number of nodes and $|E|$ the number of edges (hence overlaps). Moreover, our transitive reduction step is quadratic in $|V|^2$ in the worst case, since a DFS is performed for

Name	Technology	Annotation	N. genomes	Read bases	Read length	Read count
454/7X	454	MetaCLADE	55	15.5Gbp	450bp	3.5M
Illu/7X	Illumina	HMMER	55	15.7Gbp	150bp	10.5M
Illu/30X	Illumina	HMMER	55	67.5Gbp	150bp	45M
CAMI/Low	Illumina	HMMER	30	15Gbp	2x100bp	15M
CAMI/High	Illumina	HMMER	450	75Gbp	2x100bp	75M
Arid soil - McMurdo Valley	Illumina	MG-RAST	NA	12.5Gbp	2x100bp	12.5M
Butyrate-producing community	Illumina	Xander	NA	9Gbp	2x100bp	9M

Table 2.1: Summary of the characteristics of the datasets used for evaluation. As opposed to the other tools, S3A doesn't use pair-end information.

every node in the graph. However, the depth of each search is bounded by the number of occurrences of the considered domain.

2.3.4 Datasets

In order to evaluate S3A, we considered a total of seven datasets of metagenomic sequences whose properties are summarized in Table 2.1. For datasets with pair-end information, which is not used in S3A, pairs of reads have been used as independent reads.
Five of the datasets are synthetic datasets. Three of them were simulated according to two technologies sampling 55 equally abundant species (11 archeal and 44 bacterial). To simulate reads we used MetaSim [45], based on a read length which is characteristic for 454 and Illumina sequencing, and different coverages (7x and 30x). FlowSim [4] was then applied to obtain insertion and deletion sequencing error patterns corresponding to the respective DNA sequencing technologies. The three datasets have been used to compare S3A and SAT. Two more datasets were taken from the CAMI challenge [49], in order to compare S3A to the classical metagenomic short-read assembler Minia [14]. CAMI is a worldwide benchmarking challenge aiming at the thorough evaluation of metagenome assembler performance. We selected two types of complexity: low (30 genomes) and high (450 genomes).
A real dataset was taken from a microbial community analysis of the Arid soil of McMurdo Valley in Antartica [9] and a second one from the butyrate-producing community in the gut microbiota [64]. They have been assembled and annotated by S3A. The McMurdo Valley dataset was analysed according to the 24 domains, related to soil communities in an extreme desert environment, reported in [9]. This datasets totals 12.5Gbp of sequence for an average 2x100bp read length. The gut microbiota dataset was analysed according to 3 domains reported in [64]. The data were originally organised in several datasets and we considered (with a random choice) 3 of them. The properties of the two real datasets are reported in Table 2.1.

2.3.5 Evaluation procedure

In order to compare S3A with other assemblers on synthetic data, we need to rely on a ground truth that takes into account the fact that S3A is restricted to domain annotated regions. To build this ground truth, we perform a domain annotation (with HMMER or MetaCLADE) and analyse the performance of the tools on the same domain annotated ground truth. Gene fraction is defined on the same ground truth.
We evaluate S3A either with respect to contigs correctly assembled around a protein domain, or with respect to correctly annotated domains in the metagenomic dataset. For this, we say that "an assembly is correct" when the two protigs are correctly localized on the reference genome and "an assembly is incorrect" when at least one of the protigs involved in the assembly is not correctly placed on the reference genome.
To evaluate S3A on contigs, we rely on the following three quantities: the number of contigs correctly assembled (true positives, TP), the number of contigs incorrectly assembled (false positives, FP) and the number of correct contigs that have not been assembled (false negatives, FN). We compute S3A precision (positive predictive value) as TP/TP+FP and S3A recall (sensitivity) as TP/TP+FN.
To evaluate S3A on domain annotated contigs, we rely on the following quantities: the number of contigs with correct or incorrect assembly that are correctly annotated (true positives, TP), the number of contigs with correct or incorrect assembly that are incorrectly annotated (false positive, FP), the number of domain occurrences in the ground truth annotation that are missed. Precision and recall are computed as above, where TP, FP and FN are defined with respect to domain annotated contigs.
The rationale behind our evaluation procedure choices is that we wish to prioritize the discovery of domains that are truly present in the dataset, whatever the composition of the contig. Since this evaluation procedure is different than those used by the other tools, we ran SAT, Minia, Xander in order to compute them.

Domain Accuracy Since S3A's goal is to uncover domains, we present a second criterion to evaluate it, the *domain accuracy*. Let $\{r_1, \ldots, r_n\}$ be a set of reads annotated with domain D and assembled with S3A, and $\{D_1, \ldots, D_n\}$ the respective domain by which they have been annotated for the ground truth. The accuracy is determined with the following definitions :

- If the assembly of $\{r_1, \ldots, r_n\}$ is correct, and $D \subset \{D_1, \ldots, D_n\}$, we consider the assembly as a **True Positive**.
- If the assembly of $\{r_1, \ldots, r_n\}$ is correct, and $D \notin \{D_1, \ldots, D_n\}$, we consider the assembly as a **False Positive**.
- If the assembly of $\{r_1, \ldots, r_n\}$ is not correct, and $D \subset \{D_1, \ldots, D_n\}$, we consider the assembly as a **True Positive**.

- If the assembly of $\{r_1, \ldots, r_n\}$ is not correct, and $D \notin \{D_1, \ldots, D_n\}$, we consider the assembly as a **True Negative**.

This evaluation does not demand assemblies to be correctly placed on reference genomes and this is especially important when functionally profiling metagenomic sequences. Indeed, metagenomic datasets are often comprised of sequences coming from very close species and an assembly of protigs from close origins appears reasonable for the functional annotation of a community.

2.3.6 Influence of thresholds to evaluate precision

Along with limiting the complexity of the graph, different threshold values for $lmsl$ (at fixed ip, see Figure 2.6) can be used to improve precision while recovering a sufficient portion of true overlaps. In this way, S3A could be used as a step to assemble longer genes, which will subsequently be annotated with better accuracy. Note that the $lmsl$ metric is less sensitive to false positive matches than other measures we tested.

2.3.7 Evaluation of running time

Running time evaluations of S3A, SAT, Xander and Minia have been realized from simulated and real datasets of metagenomic reads. For S3A running time calculation, we considered the entire S3A pipeline (Figure 2.1), including preprocessing steps and domain annotation. All the evaluations have been run on the same machine with the following configuration: Intel Xeon CPU E5-2670 (2.6 GHz), 128 Gb of RAM, using 16 threads.

2.4 Results

2.4.1 S3A improves precision, recall and running times over other targeted assemblers

S3A has been tested on three synthetic datasets simulated according to Illumina and 454 technologies, with different read length (150 and 450bp) and coverage (7x and 30x) (see Methods). They are based on a large number of species (55) which is enough to capture most of the challenges for metagenome assembly (repeated regions, multi-branching nodes). Working with simulated data has the advantage that all real overlaps are known and, as a consequence, we could precisely compare S3A with the targeted assembler SAT [71] and Xander [65].
Performing targeted assembly improves significantly domain annotation in comparison with annotation on raw reads.Indeed, the false positive rate of S3A annotated domains decreases

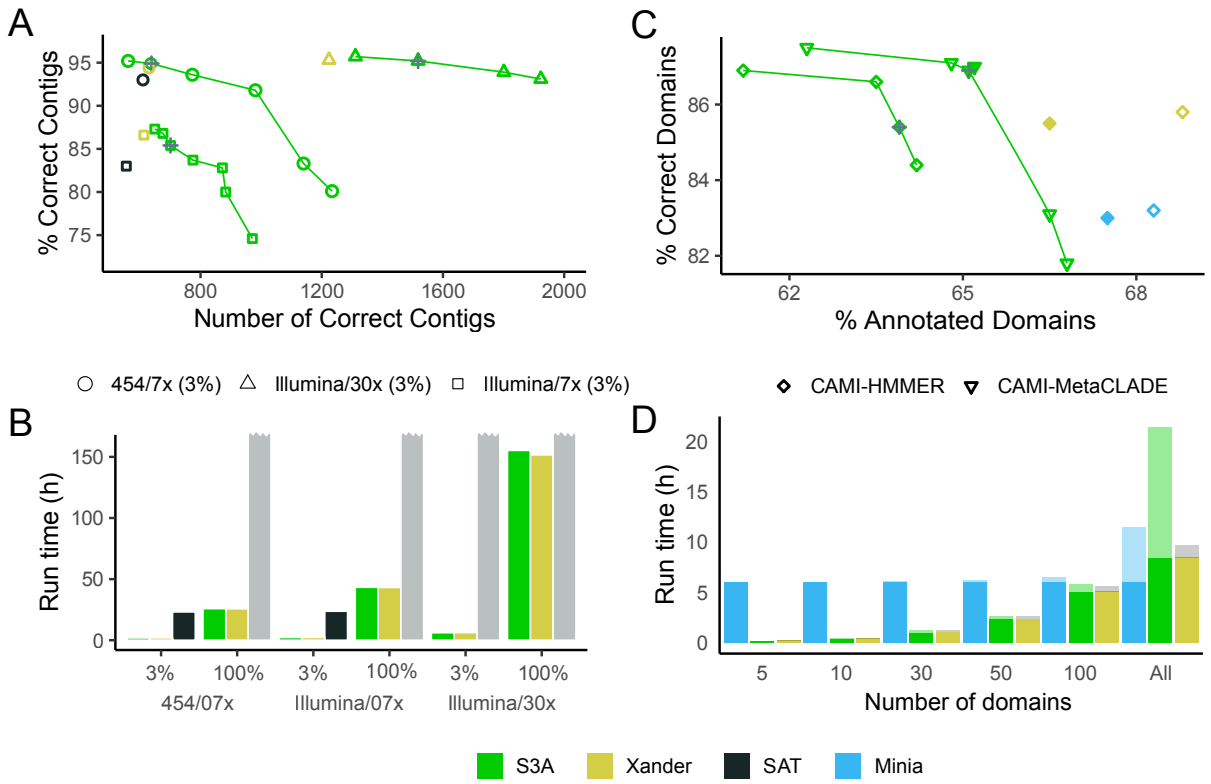

Figure 2.5: Performance comparison between S3A, SAT and Minia. The analyses are reported in green for S3A, gold for Xander, black for SAT and blue for Minia. **A.** S3A and SAT precision (reported as "% Correct Contigs") is computed on different simulated datasets (of reduced size, 3% of the reads in the full dataset). The green curves are realized by varying the $lmsl$ parameter values and keeping ip fixed at 80% (default value). By decreasing the $lmsl$ values, the number of assembled contigs augments but the proportion of correct ones decreases and the curves show how fast precision deteriorates. S3A precision at default values is reported with a grey cross. **B.** Run time performance of S3A, SAT and Xander computed on simulated datasets of two sizes (3% or 100% of the sample), for varying technologies (454 or Illumina) or coverage (7x or 30x). SAT execution did not complete after 240h on the larger datasets (represented as a grey bar). **C.** Comparison of S3A with Minia and Xander for domain precision (reported as "% Correct Domains") and domain recall (reported as "% Annotated Domain") for different $lmsl$ thresholds, on the whole high complexity CAMI toy dataset. *Domain precision* and Domain recall has been computed for Minia and S3A with respect to a standard HMMER annotation. S3A was also analysed with respect to the MetaCLADE annotation. Default values are shown with a grey cross. Filled points correspond to the average accuracy obtained over 5 selections of 100 domains drawn at random. We see that most of the difference in recall between S3A and Minia/Xander can be attributed to a subset of domains that are shorter than average (see Results). With a gene fraction close to that of Minia or Xander, S3A can have a better *domain precision* for a suitable threshold choice. **D.** Run time performance of S3A, Minia, and Xander on the low complexity CAMI dataset on restricted number of domains, from five up to one hundred, and on the full dataset. Time devoted to functional annotation is highlighted in light colors, and time for domain model construction in dark tones (for Xander).

between 2 and 10 folds with respect to the one observed while annotating raw reads. Note that constructing protigs yields an improvement up to two folds (Table A.1, middle) and that contig assembly consistently decreases the rate of incorrect domain annotation by producing longer sequences (with a median length increase of 50-70 amino acids over raw reads; Table A.1).
We also subsampled the analysis to datasets containing a small fraction of the reads (3%), and evaluated S3A, SAT and Xander performance by monitoring the run time and the precision of the predicted contigs. The algorithmic design, based on a efficient sorting of reads aligned to a domain (using the domain as reference to order reads and only comparing reads that have a chance of overlapping) and a fast sequence overlap approximation, makes S3A around 10 times faster than SAT and on par with Xander (see Figure 2.5B, columns "3%" and Table A.2). In practice, this means that S3A is capable of performing a targeted assembly for more than a million reads while this task remains impossible for SAT. S3A shows a slight but clear improvement in precision over SAT (Figure 2.5A). In Figure 2.5A, we monitor S3A behaviour with respect to an increasing number of correctly assembled contigs (TP) and show how fast S3A precision deteriorates. On the Illu/30x dataset, characterized by the highest coverage, the proportion of correct contigs assembled by S3A remains almost constant, around 94%, while the number of correctly assembled contigs almost doubles. Due to the size of this dataset, SAT could not run.
S3A is also more sensitive than SAT and Xander. Indeed, at equal precision level, S3A recovers 22% more correct contigs in the Illu/7x dataset and 5% more on the 454/7x dataset (Figure 2.5A; see Methods) than SAT demonstrating its robustness according to the technology choice (see Methods for threshold's robustness). On the same datasets, S3A is also slightly better than Xander (Figure 2.5A). It is definitely more sensitive than Xander on the Illu/30x dataset, where it achieves between 24% (default parameter) and 57% more annotated correct contigs.
As reported in Table A.3, when precision is evaluated on correct domain annotation, S3A shows to annotate consistently more domains than SAT and Xander for all three restricted datasets.
Compared to Xander, S3A is more precise at an equivalent running time (Figure 2.5B). Note that Xander cannot be run on the models of the MetaCLADE library because it generates its own HMMs as part of the assembly step. Also, Xander runs on each domain separately while S3A annotates several domains at once. In this respect, S3A design is more flexible in the treatment of multiple domains and independent from model construction.

2.4.2 Gain over whole metagenome assembly

To assess how accurate S3A is in domain annotation, we considered two datasets of low and high (depending on the number of species) complexity from the Critical Assessment of Metagenome Interpretation (CAMI) challenge (see Methods). We compared S3A to Xander

[65] and to the Minia assembler [14]. We chose Minia as it was evaluated among the best tools in the CAMI benchmark [49]. All the tools were run on the default parameters, which could imply that with optimzed parameters, some assemblers could perform better, and also knowing the Minia for example is not designed as a targeted assembler.
Given a list of domains, we wished to test whether they are present in the sample and, possibly, in which proportion.
We first restricted our evaluation to the coding regions where a domain was annotated. We further limited the evaluation at the domain level, that is, we counted the number of domains that are correctly recovered in the sample.
Figure 2.5C shows that, on the high complexity CAMI dataset, S3A global accuracy is on par with Minia and Xander, and that S3A can show a higher precision in domain annotation. On the other hand, Minia and Xander recover a larger fractions of domains, which is to be expected since some sequences assembled with Minia and Xander can in some case be too short to be annotated with a domain and thus will be missed by S3A, an effect that will be even more pronounced on the shorter domains due to their lower read coverage. We compared the median length of domains detected by Xander and not by S3A with the length of the domains detected by all of the three methods and verified that the length of the former is indeed much lower (80 a.a. vs 120 a.a.), thus supporting our interpretation. However, only a small fraction of the domains are short, and we would not expect this to impact the functional profiling of a sample in practice. To construct something more in line with the general use case, we also compared the performance on a randomly chosen set of 100 domains (Figure 2.5C, filled points). While the precision of the method does not change much, the sensitivity of S3A is now on par with Minia and Xander.
On the low complexity CAMI dataset, a much simpler assembly challenge, S3A, Minia and Xander precision is comparable and reaches 98% over more than 75% of annotated genes, as reported in Table A.4 (see Supplementary tables). S3A total running time is relatively longer (22 hours) than Xander (10 hours) or Minia (11 hours) when tested on the whole collection of domains in PFAM v30. Indeed the domain annotation hampers the total running time for the targeted assembly as it is performed before domain assembly. It is less the case for Minia or Xander, where annotation is performed either during of before assembly. However, S3A was not designed to perform a full domain annotation, it is expected to be used when only 5 to 100 domains needs scrutiny. Restricting the number of domains reduces the running time of S3A such that it becomes faster than Minia by a factor of 6-10 and slightly faster than Xander (Figure 2.5D). In practice, it permits to handle many more samples than a general purpose metagenomic assembler like Minia in the same amount of time. This is a significant practical gain when computing resources are limited and the user is studying dozens of domains [9].

2.4.3 Time performance on real datasets and comparison with MG-RAST and Xander

To assess the performance of S3A in a typical analysis workflow, we considered the microbial community of the Arid soil of McMurdo Valley in Antartica [9] and the butyrate-producing community in the a microbiota [64]. The McMurdo Valley dataset relies on read clustering and targeted HMMER annotation to uncover 24 domains. S3A correctly detects all domain occurrences much faster than MG-RAST [67, 33] (2 hrs versus 8 hrs). It reconstructs around 7% more domain sequences than MG-RAST with a HMMER annotation, and 9% more when MetaCLADE is used for annotation. S3A hence provides more information for analysis and quantification (Table A.5).
A second performance analysis was realized with a gut microbiota dataset [65] used to test the Xander assembler. It relied on a targeted HMMER annotation to uncover 3 domains involved in butyrate-production. All domain occurrences have been detected by S3A in 1h versus 1h10 with Xander. On the other hand, S3A reconstructs around 3% more domain sequences than Xander with a HMMER annotation, and 5% more when MetaCLADE is used for annotation (Table A.5).

2.4.4 Influence of default parameter on S3A's performance

Additionally, we analyzed (Figure 2.6) the influence of the parameters chosen on the simulated Illumina/7x dataset. In practice, small modifications of parameters around the default value brings relatively small changes on S3A's performance, which higlights its stability regarding the feature it uses.

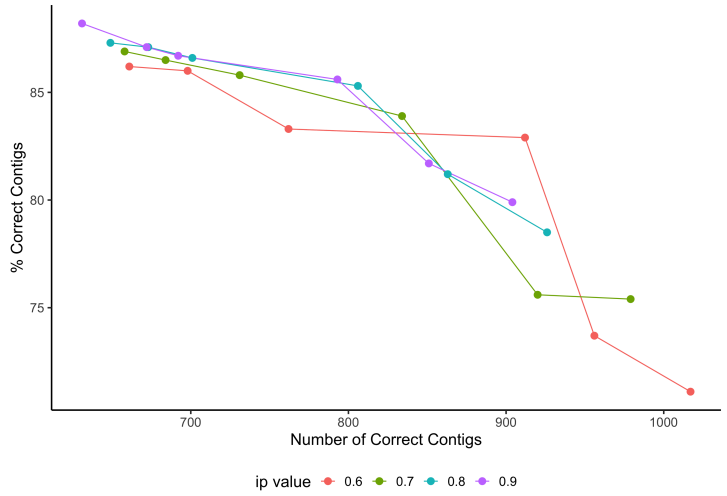

Figure 2.6: Analysis of the parameters $lmsl$ and ip on the Illumina/7x simulated dataset. The curves were drawn by varying the $lmsl$ value at different fixed values of ip: 0.6 (red), 0.7 (green), 0.8 (cyan), 0.9 (purple). Compare to the Illumina curve in Figure 2.5A drawn with the default value $ip = 0.8$ (cyan).

Chapter 3

IMPRINT: an augmented sample space of natural sequences and multiple convolutional filters identify protein partners

In this chapter, I will present IMPRINT, a relatively simple Siamese Deep Convolutional Neural Network (CNN) based on multiple convolutional filters mimicking functional motifs and trained from an enriched description of sequence space. Multiple protein profiles, describing the functional and structural diversity of protein families through evolution, provide the description of the protein family to the CNN in training. This novel data augmentation framework for protein sequences exploits the huge dataset of natural sequences available and avoids generating artificial ones. It allows IMPRINT to learn better than existing methods addressing the same question. IMPRINT finds protein partners among tens of thousands of proteins in very reasonable time. It can test a few hundred proteins against themselves in less than 15 minutes. Furthermore, it delivers information on the interaction of single residues with a specific partner.

3.1 Context

PPI networks should be understood within biological frameworks including transcriptomic and epigenetic data. Here, we address the construction of the "underlying protein network" which will serve as the basis of more sophisticated and realistic reconstructions. Because of experimental difficulties intrinsic to experimental data, *ab initio* computational reconstruction of PPIs are supposed to bring invaluable information leading to the discovery of potential molecular interactions. In this context IMPRINT couples a Deep Neural Network and protein

evolution to achieve an optimal identification of protein partners spanning a large spectrum of biological functions and leading to the reconstruction of PPIs.

The most important design element of IMPRINT concerns data augmentation. Indeed, a core limitation in a learning process is the limited amount of available training data. Gigantic improvements have been realised in the image processing field with the recent increase in the size of annotated datasets [46] and the availability of high performance dedicated hardware. Data augmentation, generating multiple "modified" training examples while preserving their label, is today a key component to overcome limited sample size and to decrease overfitting [50]. For protein partner prediction, data augmentation is far from being a trivial issue since protein sequences and protein structures cannot be simply modified randomly. since A single amino acid mutation can have tremendous deleterious impact for the protein or change its conformation.

In the past, machine learning methods represented protein sequences by considering features of various kinds, (such as using reduced alphabets, physico-chemical properties for example) [40, 53, 25, 29] and showed that sequence profiles make training of classifiers more robust improving performance [29]. More recently, the usage of multiple profiles demonstrated to be powerful in the context of domain annotation [6, 58] of genomes and metagenomes allowing for the discovery of new homologous sequences enriching protein families [22, 1], and in functional annotation [62], to capture the variety of functional motifs characterizing a protein family. Their construction demands a relatively small number (a few tens) of sequences [62]. IMPRINT original approach to augment the sample space for training the learning architecture is to extract information directly from the huge space of sequences by using multiple profiles representing the variability of natural sequences.

The second design element of IMPRINT architecture concerns the initialisation step which integrates prior knowledge on motifs interaction sites. Indeed, IMPRINT is inspired by two deep neural network architectures, DPPI [30] and PIPR [11], that have been recently proposed and that outperformed the state-of-art in *ab initio* PPI prediction. They are Siamese networks combining multiple layers of convolutional neural networks (CNN). They consider a protein sequence as a whole, without taking into account that protein physical interactions are mediated by the binding regions of the proteins that often host specific small linear motifs [24, 20]. In contrast, the first layer of IMPRINT architecture explicitly integrates the existence of interaction motifs in the design of an array of convolutional filters which are learned in the training step and partly initialised from a pool of known linear interaction patterns.

In conclusion, IMPRINT proposes a novel biologically relevant data augmentation schema applied to protein space where natural sequence variability is represented as a diverse collection of profiles. Moreover, we show with IMPRINT that the integration of prior biological knowledge in the filter initialisation could significantly enhance its performances. That strategy has already shown great success for functional prediction and characterisation of

genomic regulationory signals [34].

3.2 Materials and Methods

3.2.1 Dataset

Guo's yeast dataset

To benchmark IMPRINT against state-of-the-art deep learning methods, we used the *Guo's yeast dataset* [25]. It contains 2 497 proteins and 11 188 interactions, with a balanced number of positive and negative interactions. Positives are selected from the database of interacting proteins DIP_20070219 [47], where proteins with fewer than 50aa or $\geq 40\%$ sequence identity are excluded. Negatives are generated by randomly pairing the proteins without evidence of interaction, and filtered by their sub-cellular locations, that is, non-interactive pairs residing in the same location are excluded.

3.2.2 Design of the IMPRINT architecture

IMPRINT is designed as a Siamese architecture as illustrated in Fig. 3.1A. It is inspired by DPPI [30] and by Basset [34]. As DPPI, IMPRINT takes a pair of protein profiles as input and provides a score indicating the probability that the pair of proteins are partners. The Siamese architecture is constituted by two NNs, each dedicated to learning a protein in the pair (see green and blue architectures in Figure 3.1A), sharing the same configuration, parameters and weights. Within each NN, the convolutional module convolves a protein profile with specific filters (parameters), the rectification layer introduces non-linearity and the pooling stage computes the average of each filter's rectified response across the profile. As Basset, the convolution module of our architecture is designed as a set of either 200 or 400 small filters, either randomly defined or encoding randomly chosen small linear motifs coming from the Short Linear Interspersed Motifs (SLIMs; `http://elm.eu.org/`)[66] and the iPFAM [21] databases. Each NN architecture carries three convolutional modules whose result will be mapped by a random projection module into a representation useful for modelling paired sequences. The first layer composed of many small filters helps to learn specific motifs describing protein interactions. Compared to IMPRINT, the number of linear convolutional filters used by DPPI are much fewer (5 filters are set in the DPPI source code), but the DPPI network is deeper (5 convolutional modules).

Two inputs and one output

IMPRINT takes as input a pair of protein profiles. For a given protein sequence s with length ℓ, a protein profile S is represented as a matrix $\ell \times 20$, where $S_{i,j}$ denotes the entry of the

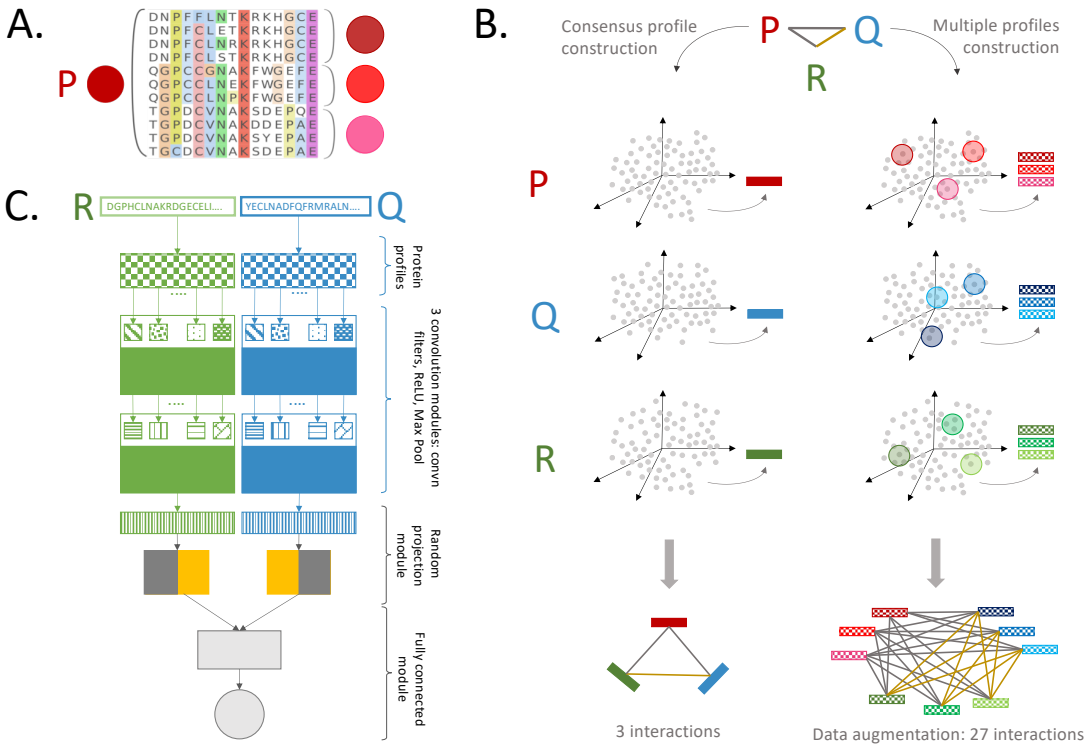

Figure 3.1: **IMPRINT key ingredients.** A. **Multiple profiles.** Homologous sequences for a protein P can be used in bulk to construct a single consensus profile or be divided in subsets of sequences, which are close to each other, to constructs several profiles. B. **Data augmentation to construct larger training datasets.** Consider a training set of 3 proteins, P, Q, R, related by true (gold edges) or false (black edges) interactions. On the left, homologous sequences (grey dots) in the sequence spaces of proteins P, Q and R are used to construct three consensus profiles (colored rectangles) used for training. On the right, for each protein, 3 homologs (colored dots) which are sufficiently far apart in sequence space are selected together with their close sequences (see colored neighbors of colored dots). Three profiles are constructed for the protein families P, Q and R from sequences in the respective neighbor. Each profile will be paired with all other profiles in the other two families, making a total of 27 pairs of interactions to be used for training. Note that all the interactions between profiles of proteins Q and R are labelled as true (gold edges) because the interaction between proteins Q, R is true. All other interactions between profiles of P, R and P, Q are labelled as false (grey edges) because the protein pairs are considered not to interact. C. **An architecture relying on motifs.** IMPRINT is designed as a classical Siamese NN with shared parameters, where each part of the network is dedicated to a different protein (blue and green blocks). It takes as input two protein profiles, each encoded in a $\ell \times 20$ matrix. The network comprises three convolution modules made of parallel convolution filters with the same width, encoding for protein motifs. Filters are grouped together and passed through a ReLu and a Max Pooling filters. The outputs from the two NNs are then merged by a Random Projection Module combined with a fully connected layer. A linear transformation is then applied, in order to get a prediction value of the interaction as output.

matrix and corresponds to the probability of the j^{th} amino acid to occupy the i^{th} position of the sequence s, where $i \in \{1 \dots \ell\}$ and $j \in \{1 \dots 20\}$. The method used for constructing protein profiles is detailed in section "Single profile generation". IMPRINT encodes the profile as an array of fixed size, 1166×20, with zero-padding if $\ell < 1166$. If $\ell > 1166$ then only the prefix of the profile will be encoded. Note that all proteins analyzed in this article have length $\ell < 1166$aa.
IMPRINT output is a score between 0 and 1 representing the probability of two proteins to interact.

A Siamese network

The prediction of the interaction between two proteins is symmetric. This is due to the symmetric design of the Siamese architecture made of two independent CNNs, with the same configuration and initialisation. The number of convolutional modules is relatively small, set at three, and each module comprises the same parameters.

A convolution module made of hundreds of parallel filters

Small linear motifs are often hosted in protein binding regions and facilitate physical interactions among proteins. At the initialisation step of the network, IMPRINT models linear motifs in order to improve its learning capacity. For this, its hyperparameters include a high number of small convolutional filters (up to 400 filters) which represent the motifs. Their initialisation can be random or based on motifs which are known to mediate protein interaction. Note that filters' initialisation in IMPRINT's three convolutional modules is identical and that, by default, our analyses based on 400 (200) filters set 100 (50) filters with SLIM motifs, 100 (50) with iPfam motifs and 200 (100) randomly (see section "Motif initialisation").
As in [34], our first convolutional layer consists of n parallel filters of size w ($w = 20$ by default) (Figure 3.1C). Given a sequence profile array SP, the convolution between each filter f_i and SP is computed with output $O_i = Conv(SP, f_i)$, for $i \in [1, n]$. Each output is then concatenated in a n-dimensional matrix, where each line O_i corresponds to the convolution through the filter f_i, and each column corresponds to a position in the profile sequence. Finally, a rectified Linear Unit (ReLU) and a Max pooling (Pool) layer are applied, so that the output of the first layer is $R = Pool(ReLU(O))$. These three layers (Conv, ReLU and MaxPool) constitute a "module" in our architecture.
In practice, for a sequence profile of length ℓ encoded in an array of dimensions 1166×20, the array resulting after execution of the first module will be of size 285×4000, computed with 200 filters and max pooling layer of size 4 as default. The value 4000 corresponds to piling up 200 arrays (the filters) of dimension 20, and 285 corresponds to the dimensionality reduction obtained from the max pooling size which is set at 4. Each concatenated i-th array of size

285×20 is the result of the convolution of the input array with the i-th filter. The Rectified Linear Unit operator does not change the dimensionality of the output array. After executing three modules, the resulting array has dimension 285×4000. The output of the three modules for the two Siamese CNN is then joined.

Filters' initialisation

The first layer of parallel filters is initialised with a set of known sequence motifs. We used two datasets of motifs known to play a role in PPI, the Short Linear Interspersed Motifs (SLIMs; `http://elm.eu.org/`) [66] and the iPFAM [21]. This latter is a subset of the PFAM database [5] specialised on domains involved in PPI; iPFAM motifs are 9500 of a length varying from 10 to 30aa. For SLIM, we used the subset of linear motifs occurring in WW-domains, which are modular protein domains, may be repeated up to four times in some proteins, that mediate specific interactions with protein ligands and are found in a number of unrelated signalling and structural proteins. The SLIM motifs taken for our initialisation contain 326 elements of length varying from 3 to 10aa. The user can use different datasets of motifs as input.
For the initialisation of the network, we selected known motifs from the two databases which occur at least once in some protein of the training set. Namely, to initialise i motifs from SLIM, j from iPFAM and l random ones ($i + j + l = n$ the total number of filters), we proceed as follows:

1. i and j motifs are randomly chosen among SLIM and IPFAM motifs, respectively. The l remaining filters are initialised randomly with a Gaussian distribution, with a 0.1 standard deviation.

2. for each selected SLIM motif m, if m is not present in some protein of the training set, m is replaced by another, randomly chosen, SLIM motif; for iPfam motifs, the procedure is the same.

3. step 2 is re-iterated until the i/j filters are initialized by SLIM/iPFAM motifs present in at least one sequence in the training set.

Motifs are encoded in filters of length 20 depending on their length: if the motif is ≤ 20aa long, it is centered in the filter and padded with 0s on the left and the right of the filter, otherwise only the first 20 positions of the motif are encoded.

Random Projection Module

As in [30], the Random Projection Module (RPM) is made of two separate, fully-connected sub-networks that join the two separate networks of the Siamese architecture. Each

sub-network computes a d-dimensional vector $o = Net_W(R)$ where:
$o_{S1} = ReLU(Batch([W^1W^2]R_{S1}))$ and $o_{S2} = ReLU(Batch([W^2W^1]R_{S2}))$, and:

- R_{S1} and R_{S2} are the output of each separate Siamese network.
- W_1 and W_2 are two $d \times (d/2)$ arrays denoting the weights of the first and second subnetworks of the RPM.

The weights of each subnetwork are initialised following a standard Gaussian distribution, and are not trained, which reduces the total number of parameters. As noted in [30], the RPM helps investigate combinations of patterns from the two input profiles, while reducing the possibility of overfitting, as the order of the inputs profiles can be ignored. RMP outputs the final representation of each input protein, after which a fully-connected layer is added to obtain a normalised value, that is the IMPRINT interaction score.

3.2.3 Hyperparameters, initialisation parameters and training parameters

IMPRINT allows the user to set a number of hyperparameters: number of filters (200 by default), filter size (20aa by default), the number of convolutional modules (3 by default). The user can run IMPRINT on pairs of protein sequences or protein profiles (by default) by choosing the corresponding input option.
By default, IMPRINT initialises filters randomly, with a Gaussian distribution with a 0.1 standard deviation. The user can specify specific linear motifs by providing a list. The iPFAM and SLIM lists, used in the analyses reported in this article, are provided to the user.
IMPRINT was trained using error backpropagation with stochastic gradient descent algorithm with momentum. Our default batch size has been set to 64, with a learning rate of 0.001.

3.2.4 Evolutionary informed data augmentation to construct a large training set

Sequence profiles improve the representation power of proteins [29]. They account for the variability occurring along positions in a sequence by highlighting those positions that are important for the stability of the protein structure and in maintaining or diversifying a protein function [56, 36, 18]. However, due to the divergence of homologous sequences, sequence profiles might degenerate and profile representations might tamper down their discriminative power in learning datasets. Following [7, 58, 62], we consider multiple profiles for a sequence, in contrast to one (Figure 3.1A). Intuitively, multiple profiles encode different pathways of protein evolution, satisfying the structural and functional constraints that a protein undergoes. In IMPRINT, we exploit this observation to control profile degeneracy by devising a data

augmentation scheme for each protein in the training set and applying it to the procedure training the NNs. Multiple profiles have a dramatic impact on the size of the training sets for PPI prediction, since the number of training examples increases quadratically with the number of profiles (see example on three sequences illustrated in Figure 3.1B). We stress that our augmented training set is "imperfect" as neither all homologs of interacting proteins are expected to interact nor they exist in the same organism. It has been previously shown in other areas of learning that large quantities of imprecise data are more efficient in training a classifier than too small ones [27].

Multiple profile data augmentation. IMPRINT constructs a small number of profiles, say k ($k \leq 20$), for each protein sequence in the training set. The data augmentation pipeline constructing k profiles per sequence is illustrated in Figure 3.1B and detailed as follows:

1. it searches for homologous sequences in UniProtKB, possibly partially overlapping. The dataset of sequences is clustered with MMseq2 (at https://github.com/soedinglab/MMseq2) by guaranteeing sequences within each cluster to have $> 40\%$ sequence similarity.

2. for each cluster, we select a representative sequence s_j and we re-iterate this selection process k times ($k = 1, 3, 5, 10, 20$ in the analysis of this article). Representative sequences from a cluster are selected with the longest hit. If k is larger than the number of clusters, all clusters are reconsidered and randomly chosen to select new representative sequences. These latter are chosen randomly.

3. Once k representative sequences are selected, a profile is generated from each of them. Due to the clustering, the profiles are expected to span sequence space and represent sequence diversity.

This step enhances the training dataset, so that instead of N inputs, we now have $N \times k$ profiles as input. The filtering of diverse models ensure that the representative models are spread along the possible functionally preserved representation of the sequence. Each representation gives a non-overlaping description of the input sequence.

Single profile generation. Given a representative sequence, a profile is generated by searching for a set of homologs with a 50% sequence identity from the sequence and by applying the construction described in [62, 59]. We ask for a minimum of 20 sequences to construct the profile. The homologs' search is done using HMMsearch with default parameters against the UniProtKB database, with an E-value threshold of 10^{-3}. A multiple sequence alignment is built using the command hhblits of the HH-suite [43] (with parameters -qid 60

-cov 70 -id 98 -e 1e-10 and database uniclust30_2017_10) and subsequently converted into a pHMM with HMMER.
A profile displays features that are characteristic of a representative sequence s_j and that might be very different for other sequences s_k in S. The more divergent the homologous sequences s_j and s_k are, the more profiles constructed from these sequences are expected to display different features. It is therefore important for a sequence to be represented by several profiles that can characterise its different pathways of evolution. These profiles have been inspired by Clade Centered Models, CCM, in [6, 58, 61].
By construction, our profiles span regions of the protein sequence space that are usually not well represented in a consensus model. These regions might highlight motifs, structural or physico-chemical properties characteristic of divergent homologous sequences. Thus, if a protein sequence is associated with many divergent homologs, the profiles are expected to describe properties that might not be detected by their consensus.
The thresholds set for sequence similarity among sequences in a cluster and for sequence identity among homologs from which profiles are constructed allow to capture features of protein interaction sites and/or determinants of functional specificity for protein families [62].

3.2.5 Data partition for training, validation and testing

In order to avoid overfitting as much as possible, we guarantee a strict partition of the data by dividing our datasets into three disjoint sets devoted to training, validation and testing. In this way, tuning our architecture's hyperparameters is done without biasing the final performance. Specifically, the validation set can be used to optimise the model's architecture, thus avoiding fitting it with the test set.
For the Guo's yeast dataset, we evaluated IMPRINT based on a 5-fold cross validation.

3.2.6 Validation of IMPRINT data augmentation schema

To validate the IMPRINT data augmentation schema, we analysed the behaviour of the architecture with respect to data augmentation. The basic representation of the input is defined by a sequence "hot-encoding", also called `IMPRINT`-0, transforming a protein sequence of length ℓ into a 0/1 array of size $\ell \times 20$ encoding the chain of amino acids as follows: $(i, j) = 1$ if and only if the amino acid j occupies position i in the sequence. `IMPRINT`-1 uses the consensus profile as entry, where the probability p that the amino acid j occupies position i in the sequence is used to encode the profile as a $\ell \times 20$ array with $(i, j) = p$. For $i \geq 2$, `IMPRINT`-i uses i profiles. Profiles are generated by the procedure described in "Data augmentation framework". As illustrated in Figure 3.1B, a training based on i profiles makes `IMPRINT`-i to learn from i^2 interactions and this implies an increase of the computational time needed for training.

3.2.7 Other CNN architectures used for comparison

We compared IMPRINT against several existing CNN architectures: DPPI [30], PIPR [11] and the simplified architectures SRGRU and SCNN discussed in [11].
DPPI is a Siamese-like convolutional neural network combined with random projection and data augmentation to predict PPIs. DPPI takes as input a high dimensional position-specific profile representation of the sequences instead of the raw sequence. Furthermore, DPPI operates on sequences of variable lengths by training on all combinations of patches, augmenting in this way the number of combinations between a pair of proteins even though they may not be involved in direct interactions. This form of data augmentation improves model robustness by injecting and training with random noise [51, 63]. IMPRINT was inspired by `DPPI`: the two approaches rely on a siamese architecture and they use the same random projection module.
PIPR is an end-to-end Siamese architecture incorporating a deep residual recurrent convolutional neural network, which integrates multiple occurrences of convolution layers and residual gated recurrent units. PIPR represents each amino acid though an encoding of their contextual and physico-chemical relatedness in a sequence, and provides a multi-granular feature aggregation process to effectively leverage both sequential and robust local information of the protein sequences.
The Siamese Residual GRU SRGRU architecture [11] is a simplification of PIPR, where all intermediary convolution layers are discarded and only the bidirectional residual GRU is kept. The Siamese CNN SCNN is obtained from PIPR by removing the residual GRU. This degenerates the PIPR framework to an architecture similar to DPPI, but differs in that SCNN directly conducts an end-to-end training on raw sequences instead of requiring the protein profiles constructed by PSI-BLAST. IMPRINT performance was evaluated with a 5-fold cross validation.

3.3 Results

3.3.1 Evaluation on the Guo's yeast dataset

IMPRINT has been compared to DPPI on the Guo's benchmark dataset [25]. It was trained on 10 188 yeast interactions and tested on 1 000 yeast interactions. While all methods perform in a similar manner, some slight improvement on the PR-curve can be noticed when training IMPRINT with a data augmentation of 3 profiles. Having a higher number of parallel filter can also improve the performance, as shown by Figure 3.2.

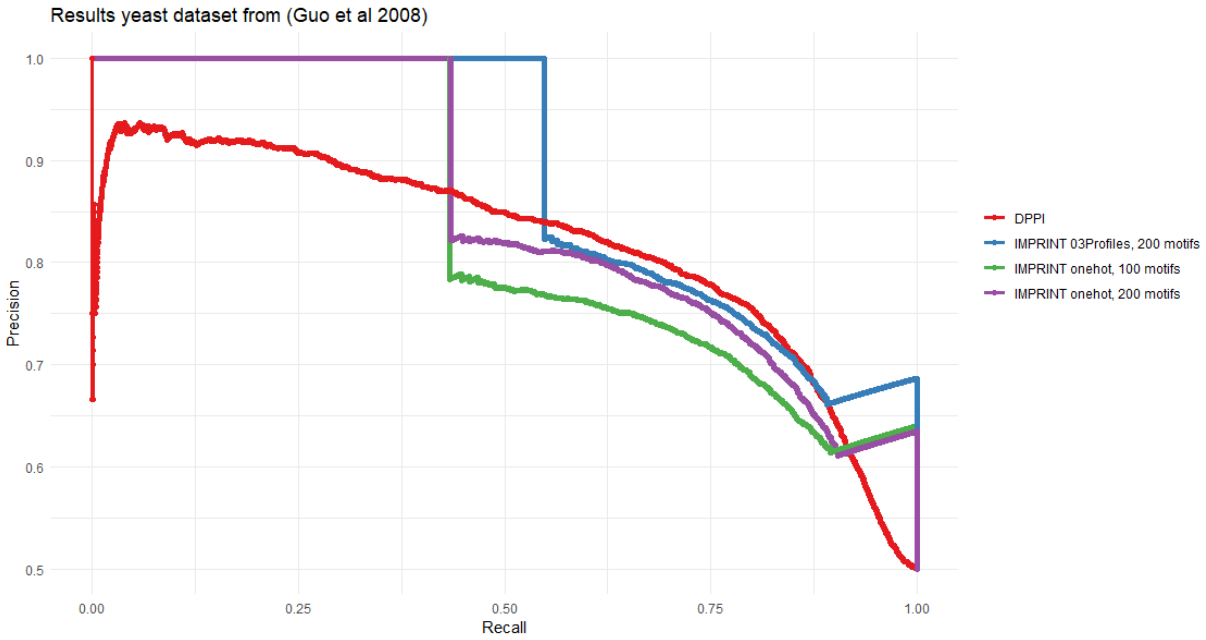

Figure 3.2: **Performance of IMPRINT and DPPI on Guo's dataset. PR curves for DPPI (red), and IMPRINT with hot-encoding (100 motifs in green, and 200 motifs in purple), and 3 profiles (200 motifs in blue).**

3.3.2 Computational time

The training time of IMPRINT is the most costly step. On the Guo dataset, with 10 188 interactions, IMPRINT training time varies between 4 hours and several days, greatly depending on the data augmentation scheme applied. Time for testing of IMPRINT is negligible: for 28 224 interactions, the run time lasted 15min. The production of multiple profiles is costly, even though, once the models are produced, they can be used to retrain the machine.

Chapter 4

Discussion and perspective

4.1 S3A

The noticeable features of S3A are both its precision and reduced time complexity, especially when focusing on several domains of interest. In this sense, S3A does not try to outcompete traditional metagenome assemblers, but enables a tradeoff between the number of domains that are profiled and the number of samples that can be considered in the same amount of time. S3A performance can thus justify its use over classical assembler when a fast and sensible profiling of a dozen up to a few hundred domains is needed. Detection of antimicrobial resistance [31], annotation of specific types of pathogenicity [26], and searching for indicators of a particular biochemical reaction [55] are a few examples.
Xander, the other targeted assembly tool we assessed, shows a reduction in running time which is comparable to the one achieved with S3A when up to a hundred domains are considered. However, in our evaluations, S3A showed better performances. Additionally one of Xander limitation is the specification of the profile models, which has to be done by the user, individually for each domain. This impairs the use of richer domain libraries, such as the one from MetaCLADE, which can greatly improve the annotation.
We should highlight that our strategy of graph construction is very general: any kind of string annotation can be provided as an input to our assembler. This could easily results in further improved running times, where the "costly" step of HMM annotation (performed sequentially on all protigs), could be replaced by an efficient indexing and clustering of the protigs.
In a different path, the manner by which the Overlap Layout graph is built allows multiple domain annotations, such that a unique graph is built for the whole range of domains. An improvement of our method would be to handle multiple overlapping domain annotations, such that every edge could hold weights for different domains. We believe that this would improve the sensitivity of the weight given to the graph edges and that are used during the graph traversal. For long protigs, it will also allow to reconstruct cases of domain co-occurrence. This is promising in a targeted assembly context, since it should both allow an even faster

assembly, while permitting the detection of more precise functions, based on multiple domains. A limitation of our approach is its dependency on the edge weights, which are used both for the graph pruning and for the traversal. While trying to hold a generic threshold independent on the protig length, the optimal threshold could vary depending on the type of species from which the reads were sequenced. In particular, the longest matching substring length metric is sensible to sequencing errors in the read tips. Applying an error correction tool before annotation, a practice common for de novo assembly, could improve the robustness of the $lmsl$ and ip parameters for S3A. However, results on different datasets/sequencing technologies show that S3A keeps a very high level of performance, even when default threshold values are changed.

4.2 IMPRINT

Current PPI experimental datasets are still far away from having reached completion, even on well-studied model organisms such as yeast or human , and their curation is an active area of research Today, physical interaction networks obtained by high-throughput techniques are found to include numerous non-functional PPI [72] and at the same time many missing true interactions. This is the main reason that an *ab initio* computational reconstruction of PPIs would bring an invaluable information.
The performances shown in this manuscript are part of a larger series of tests that have been done for IMPRINT. These tests include training with a larger data augmentation (up to 20), and with larger datasets ([29]). Moreover, the idea behind our architecture design was to be able to detect positions along the protein sequence that are related to the partner's identification, in particular binding sites. However, at the moment of this manuscript's writing, results concerning these analysis can not yet be reproduced. The main perspective concerning IMPRINT is to compare it with other state-of-the-art neural networks for protein-protein interaction prediction and with various datasets, so as to prove the gain brought by our data augmentation scheme, and IMPRINT's ability to detect interaction positions.
While yet to be proven, IMPRINT's accuracy and specificity could open up new exciting directions for the reconstruction of PPI networks for specific organisms. Yet, computational time could remain a bottleneck as it was for docking approaches [37]. The advantage of deep learning architectures is that training concentrates the full computational weight while the combinatorics of the problem (that is the quadratic number of potential pairwise interactions among the proteins) is handled efficiently by the trained architecture. As expected, IMPRINT shows that the computational issue can be overcome and that PPIs of tens of thousands of proteins can nowadays be handled. Indeed, to identify protein partners within a dataset comprised of 168 proteins, IMPRINT used $\approx$ 15min. This allows to estimate, based on the quadratic factor, about 25hrs to predict partners within a set of 1000 proteins and about 3

months for 10 000.
Another important advantage of IMPRINT over docking approaches dedicated to the complete *ab initio* PPI reconstruction is that IMPRINT is based only on sequences opening new directions of development in protein interactions. The difficulties linked to intrinsic disorder, protein instability or transient interactions, which are inherent to protein structures, will likely be avoided by sequence based methods in years to come providing a real opportunity to create novel applications in proteomics. In particular, by applying simple method from eXplanable AI (XAI, [23]) onto the IMPRINT network, we can provide interpretations of the protein sequences that inform on their potential interacting residues. XAI in bioinformatics calls for the development of new hypothesis generating representations, which could overcome the opacity of deep learning prediction while maintaining their high accuracy.
An important contribution of IMPRINT for the field of deep learning applied to proteomics is the introduction of a novel way to augment the training dataset of sequences. It has been observed that the amount of data used in the learning step is essential, provided it brings a good representation of the possible inputs, and that over-fitting induced by similar information should be avoided [8]. To address this point, IMPRINT augments the training dataset with multiple profiles describing the variability of a protein family across the space of its homologs, a notion whose impact was proved in genomic and metagenomic domain annotation [7, 59] and in functional annotation [62]. Such a scheme scales quadratically with the number of profiles considered and creates a much bigger dataset for learning.
It should be stressed that transferring interaction information by homology is not always correct. Indeed, some homologs of interacting proteins can be paralogs that diversified their function and interacting partners. Thus we expect our augmentation scheme to contain examples of interaction that are not correct. However by increasing manyfold the size of the training set with relevant proteins, we demonstrate that IMPRINT enables the learning algorithm to be robust.
Our augmentation strategy could be fruitfully applied to other functional classification problems in proteomics. In this respect, we notice that *pairs* of profiles in IMPRINT are constructed from homologous sequences irrespectively of phylogenetic origin. A control on the selection of protein sequences from which a profile originates can be added to the computational strategy.
However, we cannot alleviate that we may as well detect surrogate information about the characteristics of the two proteins, such as their sub-cellular location preference or their function, and that this information is leveraged by the classifier to predict partners. Deeper analyses of the results could lead to unexpected information.

4.3 Perspective

The title of this thesis is "Development of deep-learning approaches to discover metabolic interaction networks of environmental microbial communities from large metagenomic datasets, domain co-occurrence and protein coevolution". My first project was focused on assembly in a metagenomics context. In particular, we used a functional annotation to guide the assembly, and designed S3A so that reads belonging to different microorganims could be separated and assembled independently. Following that work, we designed a deep learning approach, IMPRINT, that can predict the probability of interaction between two given proteins, using only the amino-acid sequence information. In that sense, both projects can be seen as complementary. having high quality functional annotation is important for good protein partner identification. Starting from metagenomics datasets, we can provide an efficient targeted assembly for the protein amino acid sequences. This can then be used to assess the affinity of interaction between the different proteins in the dataset. As an additional step, future work could include using these predictions for whole microbial community, to build metabolic networks. For both projects, we have thouroughly used evolution information, that enabled us to deduce important biological information. This includes the domain annotation in the assembly part, and the usage of multiple profiles as input in IMPRINT.

4.4 Work done

Throughout my thesis, I have participated in the following articles:

- L. David, R. Vicedomini, H. Richard, A. Carbone. "Targeted domain assembly for fast functional profiling of metagenomic datasets with S3A." *Bioinformatics*, 2020.
- E. Teppa, F. Nadalin, C. Combet, D.J. Zea, L. David, A. Carbone. "Coevolution analysis of amino-acids reveals diversified drug-resistance solutions in viral sequences: a case study of hepatitis B virus." *Virus evolution*, 2020.
  `https://academic.oup.com/ve/article/6/1/veaa006/5728782?login=true`
  My contribution to this article was the participation in the development of the Sankoff algorithm.
- C. Dequeker, Y.Mohseni Behbahani, L. David, E. Laine, A. Carbone. "From complete cross-docking to partners identification and binding sites predictions", *PLOS Computational Biology*, 2022.
  `https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1009825`
  My contribution here was the test of DPPI done on the PPDBv2 dataset.

- L.David, H. Richard, and A. Carbone. "IMPRINT: an augmented sample space of natural sequences and multiple convolutional filters identify protein partners", in preparation, 2022.

Chapter 5

Thanks

I would like first to thank the members of my jury. In particular, special thanks to Sophie Sacquin-Mora and Rayan Chikhi for accepting to read my manuscript. I hope its reading will be interesting. Many thanks for Hugues and Alessandra, who have guided me throughout my thesis, always given me good advice and been patient with me. Thanks to Elodie Laine and Riccardo Vicedomini for their help and participation. I would like to mention all the people from the LCQB with which I have worked and spent great times, and of course my family and other friends who were always there when i needed it.

Appendix A

S3A supplementary information

A.1 Supplementary tables

# Dataset	Annotation	Raw Reads			Protigs			Contigs		
		Read count	Median Annotation Length (a.a.)	False Positive Domain Hits	Protig count	Median Annotation Length (a.a.)	False Positive Domain Hits	Contig count	Median Annotation Length (a.a.)	False Positive Domain Hits
3% of 454/7x	HMMER	110K	38	25%	90K	52	13%	65K	79	3%
	MetaCLADE		43	22%		53	11%	65K	92	2%
3% of Ill/7x	HMMER	320K	18	31%	210K	44	19%	150K	75	13.9%
	MetaCLADE		23	27%		44	16%	150K	90	12.2%
3% of Ill/30x	HMMER	1350K	18	22%	1100K	61	20%	750K	78	14.2%
	MetaCLADE		23	27%		62	19%	750K	94	12.8%

Table A.1: **Comparative analysis of false positive domain hits at different steps of the S3A pipeline (raw reads, protigs, contigs).** At each step, the number of reads/protigs/contigs, the median length of the domain annotated sequence and the proportion of false positive domain hits are reported. The amount of false positives is at least divided by two, thanks to the generation of protigs but also to the contig assembly step. Using MetaCLADE for domain annotation also reduces the number of false positive hits. Note that the "Protig count" column counts assembled reads together with those reads that cannot be assembled. In the same way, the "Contig count" column counts assembled protigs together with protigs and reads that cannot be assembled.

S3A runtime performances on the full and restricted synthetic datasets					
Dataset	Tool	Time	Precision	Correct contigs	Number of reads
3% of 454/7x	SAT	23h30	93.3%	627	0.11M
	Xander	2h21	94.4%	632	
	S3A	**2h18**	**94.9%**	**638**	
3% of Illu/7x	SAT	25h	83.9%	581	0.32M
	Xander	2h	86.6%	613	
	S3A	**1h55**	**87.3%**	**649**	
3% of Illu/30x	SAT	**>240h**	-	-	1.35M
	Xander	6h	95.2%	1295	
	S3A	**6h10**	**95.7%**	**1311**	
Full 454/7x	SAT	**>240h**	-	-	3.5M
	Xander	43h11	94.8%	25401	
	S3A	**43h22**	**95.1%**	**25413**	
Full Illu/7x	SAT	**>240h**	-	-	10.5M
	Xander	25h47	82.7%	21298	
	S3A	**25h53**	**83.6%**	**21378**	
Full Illu/30x	SAT	**>240h**	-	-	45M
	Xander	151h40	95.1%	41524	
	S3A	**155h30**	**95.7%**	**41649**	

Table A.2: **Performance comparison between S3A, Xander and SAT.** Precision of S3A, Xander and SAT is reported on three simulated datasets and on a portion (3%) of them selected at random. Datasets simulate the 454 and Illumina technologies. On the three full datasets and on the 3% of the Illu/30x dataset, SAT was ran but it was interrupted after 240h of runtime. Note that these dataset are 10 and 100 times larger in size. SAT precision corresponds to the best value (with respect to SAT and Xander performance) in the curves of Figure 2.5.

S3A domain annotation performances on the restricted synthetic datasets						
	3% of 454/7x		**3% of Illu/7x**		**3% of Illu/30x**	
	HMMER	MetaCLADE	HMMER	MetaCLADE	HMMER	MetaCLADE
S3A vs SAT	>2%	>3%	>10%	>15%	-	-
S3A vs Xander	<1%	<1%	<1%	<1%	>1%	>2%

Table A.3: **Comparative analysis of S3A versus SAT and Xander precision in domain annotation.** Precision of S3A is reported on restricted simulated datasets and compared with the one obtained by SAT and Xander. The table reports the percentage of domains that have been correctly annotated by S3A and were missed by the other methods. Note that Xander cannot run on a multiple domain library since it constructs its own domain models.

Assembler	Minimum Domain Coverage (# of reads within the contig)	Domain Precision	Fraction of Annotated Domain
Minia	–	97.8 %	76.2 %
Xander	–	97.9 %	75.6 %
S3A	3	98.1 %	75.3 %
S3A	10	98.3 %	74.1 %
S3A	20	98.5 %	71.1 %

Table A.4: **Analysis of S3A, Minia and Xander precision in domain annotation of the low complexity CAMI dataset.** Evolution of the accuracy of S3A when filtering contigs by their domain coverage. S3A annotation was realised with MetaCLADE. Minia and Xander have been ran with default parameters. The symbol "–" indicates that no threshold has been used for computing domain precision.

S3A annotation performances on real datasets			
	Annotation		Datasets
	HMMER	MetaCLADE	
S3A vs Xander	>3%	>5%	Butyrate metabolism
S3A vs MG-RAST	>7%	>9%	McMurdo Valley

Table A.5: **Comparative analysis of S3A versus Xander and MG-RAST precision in domain annotation.** Precision of S3A is compared to MG-RAST on the real metagenomic dataset of McMurdo Valley, and to Xander on the real metagenomic dataset of Butyrate metabolism. The table reports the percentage of domains that have been correctly annotated by S3A and were missed by other methods.

[C]Bibliography

Bibliography

[1] Alberto Amato, Gianluca Dell'Aquila, Francesco Musacchia, Rossella Annunziata, Ari Ugarte, Nicolas Maillet, Alessandra Carbone, Maurizio Ribera d'Alcalà, Remo Sanges, Daniele Iudicone, et al. Marine diatoms change their gene expression profile when exposed to microscale turbulence under nutrient replete conditions. *Scientific Reports*, 7, 2017.

[2] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7):878, 2016.

[3] Martin Ayling, Matthew D Clark, and Richard M Leggett. New approaches for metagenome assembly with short reads. *Briefings in bioinformatics*, 21(2):584–594, 2020.

[4] Susanne Balzer, Ketil Malde, Anders Lanzén, Animesh Sharma, and Inge Jonassen. Characteristics of 454 pyrosequencing data—enabling realistic simulation with flowsim. *Bioinformatics*, 26(18):i420–i425, 2010.

[5] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik LL Sonnhammer, et al. The pfam protein families database. *Nucleic acids research*, 32(suppl_1):D138–D141, 2004.

[6] Juliana Bernardes, Gerson Zaverucha, Catherine Vaquero, and Alessandra Carbone. Improvement in protein domain identification is reached by breaking consensus, with the agreement of many profiles and domain co-occurrence. *PLoS computational biology*, 12(7):e1005038, 2016.

[7] Juliana Bernardes, Gerson Zaverucha, Catherine Vaquero, and Alessandra Carbone. Improvement in protein domain identification is reached by breaking consensus, with the agreement of many profiles and domain co-occurrence. *PLoS Comput Biol*, 12(7):e1005038, 07 2016.

[8] Christopher M Bishop. Model-based machine learning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20120222, 2013.

[9] Heather N Buelow, Ara S Winter, David J Van Horn, John E Barrett, Michael N Gooseff, Egbert Schwartz, and Cristina D Takacs-Vesbach. Microbial community responses to increased water and organic matter in the arid soils of the mcmurdo dry valleys, antarctica. *Front Microbiol*, 7:1040, 2016.

[10] Massimo Cairo, Paul Medvedev, Nidia Obscura Acosta, Romeo Rizzi, and Alexandru I Tomescu. Optimal omnitig listing for safe and complete contig assembly. In *28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[11] Muhao Chen, Chelsea J T Ju, Guangyu Zhou, Xuelu Chen, Tianran Zhang, Kai-Wei Chang, Carlo Zaniolo, and Wei Wang. Multifaceted protein–protein interaction prediction based on Siamese residual RCNN. *Bioinformatics*, 35(14):i305–i314, 07 2019.

[12] Ramu Chenna, Hideaki Sugawara, Tadashi Koike, Rodrigo Lopez, Toby J Gibson, Desmond G Higgins, and Julie D Thompson. Multiple sequence alignment with the clustal series of programs. *Nucleic acids research*, 31(13):3497–3500, 2003.

[13] Rayan Chikhi, Antoine Limasset, and Paul Medvedev. Compacting de bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics*, 32(12):i201–i208, Jun 2016.

[14] Rayan Chikhi and Guillaume Rizk. Space-efficient and exact de bruijn graph representation based on a bloom filter. In *WABI*, volume 7534 of *Lecture Notes in Computer Science*, pages 236–248. Springer, 2012.

[15] Holger Dinkel, Sushama Michael, Robert J Weatheritt, Norman E Davey, Kim Van Roey, Brigitte Altenberg, Grischa Toedt, Bora Uyar, Markus Seiler, Aidan Budd, et al. Elm—the database of eukaryotic linear motifs. *Nucleic acids research*, 40(D1):D242–D251, 2012.

[16] Xiuquan Du, Shiwei Sun, Changlin Hu, Yu Yao, Yuanting Yan, and Yanping Zhang. Deepppi: boosting prediction of protein–protein interactions with deep neural networks. *Journal of chemical information and modeling*, 57(6):1499–1510, 2017.

[17] Dent Earl, Keith Bradnam, John St John, Aaron Darling, Dawei Lin, Joseph Fass, Hung On Ken Yu, Vince Buffalo, Daniel R Zerbino, Mark Diekhans, et al. Assemblathon 1: a competitive assessment of de novo short read assembly methods. *Genome research*, 21(12):2224–2241, 2011.

[18] Sean R Eddy, Graeme Mitchison, and Richard Durbin. Maximum discrimination hidden markov models of sequence consensus. *Journal of Computational Biology*, 2(1):9–23, 1995.

[19] Robert D Finn, Jody Clements, and Sean R Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic Acids Res*, 39(Web Server issue):W29–37, Jul 2011.

[20] Robert D Finn, Benjamin L Miller, Jody Clements, and Alex Bateman. ipfam: a database of protein family and domain interactions found in the protein data bank. *Nucleic acids research*, 42(D1):D364–D373, 2014.

[21] Robert D Finn, Benjamin L Miller, Jody Clements, and Alex Bateman. ipfam: a database of protein family and domain interactions found in the protein data bank. *Nucleic Acids Res*, 42(Database issue):D364–73, Jan 2014.

[22] Antonio Emidio Fortunato, Marianne Jaubert, Gen Enomoto, Jean-Pierre Bouly, Raffaella Raniello, Michael Thaler, Shruti Malviya, Juliana Silva Bernardes, Fabrice Rappaport, Bernard Gentili, et al. Diatom phytochromes reveal the existence of far-red light based sensing in the ocean. *The Plant Cell*, pages tpc–00928, 2016.

[23] Randy Goebel, Ajay Chander, Katharina Holzinger, Freddy Lecue, Zeynep Akata, Simone Stumpf, Peter Kieseberg, and Andreas Holzinger. Explainable ai: the new 42? In *International cross-domain conference for machine learning and knowledge extraction*, pages 295–303. Springer, 2018.

[24] Marc Gouw, Sushama Michael, Hugo Sámano-Sánchez, Manjeet Kumar, András Zeke, Benjamin Lang, Benoit Bely, Lucía B Chemes, Norman E Davey, Ziqi Deng, et al. The eukaryotic linear motif resource–2018 update. *Nucleic acids research*, 46(D1):D428–D434, 2018.

[25] Yanzhi Guo, Lezheng Yu, Zhining Wen, and Menglong Li. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic Acids Research*, 36(9):3025–3030, 04 2008.

[26] Ayal B Gussow, Slavé Petrovski, Quanli Wang, Andrew S Allen, and David B Goldstein. The intolerance to functional genetic variation of protein domains predicts the localization of pathogenic mutations within genes. *Genome Biol*, 17:9, Jan 2016.

[27] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[28] Tobias Hamp and Burkhard Rost. Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics*, 31(12):1945–1950, 2015.

[29] Tobias Hamp and Burkhard Rost. Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics*, 31(12):1945–1950, 2015.

[30] Somaye Hashemifar, Behnam Neyshabur, Aly A Khan, and Jinbo Xu. Predicting protein–protein interactions through sequence-based deep learning. *Bioinformatics*, 34(17):i802–i810, 2018.

[31] Baofeng Jia, Amogelang R Raphenya, Brian Alcock, Nicholas Waglechner, Peiyao Guo, Kara K Tsang, Briony A Lago, Biren M Dave, Sheldon Pereira, Arjun N Sharma, Sachin Doshi, Mélanie Courtot, Raymond Lo, Laura E Williams, Jonathan G Frye, Tariq Elsayegh, Daim Sardar, Erin L Westman, Andrew C Pawlowski, Timothy A Johnson, Fiona S L Brinkman, Gerard D Wright, and Andrew G McArthur. CARD 2017: expansion and model-centric curation of the comprehensive antibiotic resistance database. *Nucleic Acids Res*, 45(D1):D566–D573, 01 2017.

[32] Robbie P Joosten, Thomas Womack, Gert Vriend, and Gérard Bricogne. Re-refinement from deposited x-ray data can deliver improved models for most pdb entries. *Acta Crystallographica Section D: Biological Crystallography*, 65(2):176–185, 2009.

[33] Kevin P Keegan, Elizabeth M Glass, and Folker Meyer. Mg-rast, a metagenomics service for analysis of microbial community structure and function. In *Microbial Environmental Genomics (MEG)*, pages 207–233. Springer, 2016.

[34] David R Kelley, Jasper Snoek, and John L Rinn. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res*, 26(7):990–9, 07 2016.

[35] Samuel Kerrien, Bruno Aranda, Lionel Breuza, Alan Bridge, Fiona Broackes-Carter, Carol Chen, Margaret Duesbury, Marine Dumousseau, Marc Feuermann, Ursula Hinz, et al. The intact molecular interaction database in 2012. *Nucleic acids research*, 40(D1):D841–D846, 2012.

[36] Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of molecular biology*, 235(5):1501–1531, 1994.

[37] Anne Lopes, Sophie Sacquin-Mora, Viktoriya Dimitrova, Elodie Laine, Yann Ponty, and Alessandra Carbone. Protein-protein interactions in a crowded environment: an analysis via cross-docking simulations and evolutionary information. *PLoS Comput Biol*, 9(12):e1003369, 2013.

[38] Ruibang Luo, Binghang Liu, Yinlong Xie, Zhenyu Li, Weihua Huang, Jianying Yuan, Guangzhu He, Yanxiang Chen, Qi Pan, Yunjie Liu, et al. Soapdenovo2: an empirically

improved memory-efficient short-read de novo assembler. *Gigascience*, 1(1):2047–217X, 2012.

[39] Niranjan Nagarajan and Mihai Pop. Sequence assembly demystified. *Nat Rev Genet*, 14(3):157–67, Mar 2013.

[40] J Park, M Lappe, and S A Teichmann. Mapping protein family interactions: intramolecular and intermolecular protein family interaction repertoires in the pdb and yeast. *J Mol Biol*, 307(3):929–38, Mar 2001.

[41] Yungki Park and Edward M Marcotte. Flaws in evaluation schemes for pair-input computational predictions. *Nature methods*, 9(12):1134, 2012.

[42] Franziska Pfeiffer, Carsten Gröber, Michael Blank, Kristian Händler, Marc Beyer, Joachim L Schultze, and Günter Mayer. Systematic evaluation of error rates and causes in short samples in next-generation sequencing. *Scientific reports*, 8(1):1–14, 2018.

[43] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature Methods*, 9:173–, December 2011.

[44] Mina Rho, Haixu Tang, and Yuzhen Ye. Fraggenescan: predicting genes in short and error-prone reads. *Nucleic Acids Res*, 38(20):e191, Nov 2010.

[45] Daniel C Richter, Felix Ott, Alexander F Auch, Ramona Schmid, and Daniel H Huson. Metasim: a sequencing simulator for genomics and metagenomics. *PLoS One*, 3(10):e3373, Oct 2008.

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[47] Lukasz Salwinski, Christopher S Miller, Adam J Smith, Frank K Pettit, James U Bowie, and David Eisenberg. The database of interacting proteins: 2004 update. *Nucleic acids research*, 32(suppl_1):D449–D451, 2004.

[48] M. H. Schaefer, J. F. Fontaine, A. Vinayagam, P. Porras, E. E. Wanker, and M. A. Andrade-Navarro. Hippie: Integrating protein interaction networks with experiment based quality scores. *PLoS One*, 7:e31826, 2012.

[49] Alexander Sczyrba, Peter Hofmann, Peter Belmann, David Koslicki, Stefan Janssen, Johannes Dröge, Ivan Gregor, Stephan Majda, Jessika Fiedler, Eik Dahms, Andreas

Bremges, Adrian Fritz, Ruben Garrido-Oter, Tue Sparholt Jørgensen, Nicole Shapiro, Philip D Blood, Alexey Gurevich, Yang Bai, Dmitrij Turaev, Matthew Z DeMaere, Rayan Chikhi, Niranjan Nagarajan, Christopher Quince, Fernando Meyer, Monika Balvočiūtė, Lars Hestbjerg Hansen, Søren J Sørensen, Burton K H Chia, Bertrand Denis, Jeff L Froula, Zhong Wang, Robert Egan, Dongwan Don Kang, Jeffrey J Cook, Charles Deltel, Michael Beckstette, Claire Lemaitre, Pierre Peterlongo, Guillaume Rizk, Dominique Lavenier, Yu-Wei Wu, Steven W Singer, Chirag Jain, Marc Strous, Heiner Klingenberg, Peter Meinicke, Michael D Barton, Thomas Lingner, Hsin-Hung Lin, Yu-Chieh Liao, Genivaldo Gueiros Z Silva, Daniel A Cuevas, Robert A Edwards, Surya Saha, Vitor C Piro, Bernhard Y Renard, Mihai Pop, Hans-Peter Klenk, Markus Göker, Nikos C Kyrpides, Tanja Woyke, Julia A Vorholt, Paul Schulze-Lefert, Edward M Rubin, Aaron E Darling, Thomas Rattei, and Alice C McHardy. Critical assessment of metagenome interpretation-a benchmark of metagenomics software. *Nat Methods*, 14(11):1063–1071, Nov 2017.

[50] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *J. Big Data*, 6:60, 2019.

[51] Jocelyn Sietsma and Robert JF Dow. Creating artificial neural networks that generalize. *Neural networks*, 4(1):67–79, 1991.

[52] Pawel Smialowski, Philipp Pagel, Philip Wong, Barbara Brauner, Irmtraud Dunger, Gisela Fobo, Goar Frishman, Corinna Montrone, Thomas Rattei, Dmitrij Frishman, et al. The negatome database: a reference set of non-interacting protein pairs. *Nucleic acids research*, 38(suppl_1):D540–D544, 2010.

[53] E Sprinzak and H Margalit. Correlated sequence-signatures as markers of protein-protein interaction. *J Mol Biol*, 311(4):681–92, Aug 2001.

[54] Tanlin Sun, Bo Zhou, Luhua Lai, and Jianfeng Pei. Sequence-based prediction of protein protein interaction using a deep-learning algorithm. *BMC bioinformatics*, 18(1):1–8, 2017.

[55] Alessandro Tagliabue, Andrew R Bowie, Philip W Boyd, Kristen N Buck, Kenneth S Johnson, and Mak A Saito. The integral role of iron in ocean biogeochemistry. *Nature*, 543(7643):51–59, 03 2017.

[56] Roman L Tatusov, Stephen F Altschul, and Eugen V Koonin. Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks. *Proceedings of the National Academy of Sciences*, 91(25):12091–12095, 1994.

[57] Ari Ugarte, Riccardo Vicedomini, Juliana Bernardes, and Alessandra Carbone. A multi-source domain annotation pipeline for quantitative metagenomic and metatranscriptomic functional profiling. *Microbiome*, 6(1):149, 08 2018.

[58] Ari Ugarte, Riccardo Vicedomini, Juliana Bernardes, and Alessandra Carbone. A multi-source domain annotation pipeline for quantitative metagenomic and metatranscriptomic functional profiling. *Microbiome*, 6(1):149, Aug 2018.

[59] Ari Ugarte, Riccardo Vicedomini, Juliana Bernardes, and Alessandra Carbone. A multi-source domain annotation pipeline for quantitative metagenomic and metatranscriptomic functional profiling. *Microbiome*, 6(1):149, 08 2018.

[60] J Craig Venter, Mark D Adams, Eugene W Myers, Peter W Li, Richard J Mural, Granger G Sutton, Hamilton O Smith, Mark Yandell, Cheryl A Evans, Robert A Holt, et al. The sequence of the human genome. *science*, 291(5507):1304–1351, 2001.

[61] Riccardo Vicedomini, Clémence Blachon, Francesco Oteri, and Alessandra Carbone. Myclade: a multi-source domain annotation server for sequence functional exploration. *Nucleic Acids Research*, 2021.

[62] Riccardo Vicedomini, Jean Pierre Bouly, Elodie Laine, Angela Falciatore, and Alessandra Carbone. Profileview: multiple probabilistic models resolve protein families functional diversity. *bioRxiv*, page 717249, 2019.

[63] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[64] Marius Vital, André Karch, and Dietmar H Pieper. Colonic butyrate-producing communities in humans: an overview using omics data. *mSystems*, 2(6), 2017.

[65] Qiong Wang, Jordan A Fish, Mariah Gilman, Yanni Sun, C Titus Brown, James M Tiedje, and James R Cole. Xander: employing a novel method for efficient gene-targeted metagenomic assembly. *Microbiome*, 3:32, 2015.

[66] Robert J Weatheritt, Katja Luck, Evangelia Petsalaki, Norman E Davey, and Toby J Gibson. The identification of short linear motif-mediated interfaces within the human interactome. *Bioinformatics*, 28(7):976–982, 2012.

[67] Andreas Wilke, Jared Bischof, Wolfgang Gerlach, Elizabeth Glass, Travis Harrison, Kevin P Keegan, Tobias Paczian, William L Trimble, Saurabh Bagchi, Ananth Grama, et al. The MG-RAST metagenomics database and portal in 2015. *Nucleic Acids Research*, 44(D1):D590–D594, 2015.

[68] Wei-Sheng Wu, Yu-Xuan Jiang, Jer-Wei Chang, Yu-Han Chu, Yi-Hao Chiu, Yi-Hong Tsao, Torbjörn EM Nordling, Yan-Yuan Tseng, and Joseph T Tseng. Hrpdviewer: human ribosome profiling data viewer. *Database*, 2018, 2018.

[69] Ioannis Xenarios, Danny W Rice, Lukasz Salwinski, Marisa K Baron, Edward M Marcotte, and David Eisenberg. Dip: the database of interacting proteins. *Nucleic acids research*, 28(1):289–291, 2000.

[70] Daniel R Zerbino. Using the velvet de novo assembler for short-read sequencing technologies. *Current protocols in bioinformatics*, 31(1):11–5, 2010.

[71] Yuan Zhang, Yanni Sun, and James R Cole. A scalable and accurate targeted gene assembly tool (SAT-assembler) for next-generation sequencing data. *PLoS Comput Biol*, 10(8):e1003737, Aug 2014.