



HAL
open science

Graph-based neural networks for generation of synthetically accessible molecular structures

Tagir Akhmetshin

► **To cite this version:**

Tagir Akhmetshin. Graph-based neural networks for generation of synthetically accessible molecular structures. Cheminformatics. Université de Strasbourg; Kazanskiy gosudarstvennyy universitet im. V. I. Ul'anova (Kazan), 2023. English. NNT : 2023STRAF010 . tel-04208499

HAL Id: tel-04208499

<https://theses.hal.science/tel-04208499>

Submitted on 15 Sep 2023

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

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

Université

de Strasbourg

UNIVERSITÉ DE STRASBOURG

EDSC
École Doctorale des
Sciences Chimiques



Kazan Federal
UNIVERSITY

ÉCOLE DOCTORALE DES SCIENCES CHIMIQUES

Chimie de la matière complexe – UMR 7140

THÈSE présentée par :

Tagir AKHMETSHIN

soutenue le : **17 mars 2023**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/ Spécialité : **Chimie / Chémoinformatique**

**Réseaux neuronaux à base de graphes
pour la génération de structures
moléculaires synthétiquement accessibles**

THÈSE dirigée par :

M. VARNEK Alexandre

Professeur, Université de Strasbourg

M. MADZHIDOV Timur

Docteur, Elsevier Ltd.

RAPPORTEURS :

Mme. STOVEN Véronique

Professeur, Mines-ParisTech

M. LEPAILLEUR Alban

Maitre de Conférences, HDR, Université de Caen
Normandie

AUTRES MEMBRES DU JURY :

Mme. KELLENBERGER Esther

Professeur, Université de Strasbourg

Acknowledgements

From childhood, I remember one of the most important phrases I learned almost immediately. The closest translation would be "Better a hundred friends than a hundred rubles". In a way, it became my credo, and I was lucky enough to make many friends while studying, travelling and working. And all my friends and mentors have helped me along the difficult path to becoming a scientist. If I were to list them all, I'm afraid there would not be enough space for my thesis.

Nevertheless, I can still single out people without whom I would not have been able to reach such heights. First, I am very grateful to Dr Timur Madzhidov and Professor Alexandre Varnek. You have guided me and given me a vocation as a chemoinformatician. Even though I was always attracted by the idea of chemoinformatics, without you, I would not stand where I am now. In addition, I would like to thank my mentors, Ramil Nugmanov and Arkadii Lin, who taught me survival skills in the chemoinformatics jungles and were always helping with both simple and complex questions.

As a child of Kazan and Strasbourg chemoinformatics labs, I would like to thank all my teachers, colleagues and students I worked with. It is important to mention people without whom this work would be impossible - Evgenii Ziaikin, Dmitrii Babadeev, Anna Pinigina and Almaz Gilmullin. You are brilliant students, and I hope to see your success in the future. Also, I would like to thank my collaborators Timur Gimadiev, Valentina Afonina, Asima Rakhimbekova and Daniyar Mazitov for their help in the joint projects. Regarding the Strasbourg side, everyone here became to me not only the best colleagues but also a family with one nationality - chemoinformaticians. I'm happy that I was able to be a part of this family and grateful to Yuliana Zabolotna, Olga Klimchuck, Fanny Bonachera, Dragos Horvath, Gilles Marcou, Karina Pikalyova, Shamkhal Baybekov, Regina Pikalyova, Maxim Shevelev, Helena Perez Pena, Sai Prashanth Santhapuri, Polina Oleneva, Dmitrii Zankov, Pierre Llompart, Farah Askerhanova and others.

I want to thank my friends from the master who are excellent researchers and the most supportive and amazing people. To Masha Avstrikova, Ivan Reveguk, Julia Reville, Louis Plyer, Roman Lambert, Pablo Roseiro, Marie Meylacq, Antoine Danvin, Marie Gebelin, Alexandra Hllx and many others.

I am deeply grateful to my family, mum and dad, who always support me, regardless of where I am and where I plan to go.

And the special thanks go to my love - Marie, for her constant support even when I was "impossible" and for our happy moments together.

Table of contents

1	Résumé en français	1
1.1	Introduction	1
1.2	Resultats et discussions	1
1.3	Conclusion generale	16
1.4	Liste des presentation	17
1.5	Liste des publication	18
2	Introduction	19
3	Generation of molecular structures	23
3.1	Background of generative deep learning for inverse QSAR	24
3.2	Development of novel graph-based architectures	35
4	Self-learning-based synthesis planning	73
4.1	Reaction data curation	75
4.2	Retrosynthetic planning	94
4.3	Validation techniques for prediction of reaction rate constant in different conditions	131
5	Conclusions and perspectives	149
	References	153
	List of figures	167
	List of tables	169

Chapter 1

Résumé en français

1.1 Introduction

Des réseaux neuronaux profonds (DNN) peuvent être utilisés pour générer des structures moléculaires possédant des propriétés ciblées. Toutefois, ils sont souvent basés sur un codage linéaire des structures chimiques, c'est-à-dire par des chaînes de caractères (par exemple SMILES), qui induisent des problèmes techniques et conceptuels. En s'affranchissant de ces codes, les architectures basées sur les graphes semblent être plus prometteuses. Cependant, les réseaux de neurones artificiels basés sur les graphes (GNN) ne sont pas encore très utilisés car ils restent chers en termes de ressources informatiques et ils sont difficiles à optimiser. L'objectif de ce projet doctoral est d'améliorer des GNN existants pour (i) générer des molécules d'intérêt et (ii) proposer des voies de synthèses chimiques.

1.2 Resultats et discussions

1.2.1 Optimisation moléculaire

Grâce à une puissance de calcul accrue et au développement de la théorie de l'apprentissage profond, il est désormais possible non seulement de prédire les propriétés des composés, mais aussi de générer des analogues de molécules à partir de leur représentation vectorielle. A cette fin, les autoencodeurs (AE) se montrent particulièrement intéressants. Il s'agit de deux réseaux de neurones artificiels empilés : l'encodeur et le décodeur. L'encodeur transforme les caractéristiques structurelles des molécules en une représentation vectorielle : un espace chimique latent. Le décodeur reconstruit à partir des vecteurs de cet espace latent, les structures chimiques correspondantes. Ces machines permettent ainsi de générer de nouvelles structures chimiques.

Les premiers AE génératifs étaient basés sur la représentation de structures moléculaires par de chaînes de caractères (SMILES). Bien que ces approches aient montré de bonnes performances, la position des molécules dans l'espace latent dépendait de l'ordre d'apparition des atomes dans ces chaînes. La notation SMILES n'étant pas unique, la représentation dans l'espace latent ne l'est pas non plus.[1] En conséquence, les performances des modèles prédictifs basés sur ces vecteurs latents étaient réduites. Les AE à base de GNN ne souffrent pas de tels biais.

Autoencodeur basé sur des graphes annotés par comptes d'hydrogène (HyFactor)

Comme mentionné ci-dessus, l'optimisation de GNNs nécessite de grandes ressources informatiques. Normalement, ils exigent que chaque type de liaison soit traitée séparément ce qui conduit à 4 graphes spécifiques pour les liaisons simples, doubles, triples et aromatiques. Mais les architectures des encodeurs et décodeurs peuvent prendre en compte des annotations sur les atomes et les liaisons. Par analogie avec la notation InChi, nous avons identifié que le nombre d'hydrogènes sur chaque atome lourd de la structure améliore considérablement l'optimisation d'autoencodeurs basés sur des graphes. Cette stratégie nous a conduit à une nouvelle architecture baptisée HyFactor (Hydrogen-count labeled graph-based defactorization) (Figure 1.2a). Celle-ci utilise efficacement les comptes d'atomes d'hydrogènes associés à chaque nœud du graphe moléculaire - Hydrogen-count Labelled Graph (HLG) (Figure 1.1). Avec la connectivité, cette information est suffisante pour reproduire la structure moléculaire : la nature des liaisons chimiques est implicite. Les performances du nouveau modèle ont été comparées à celles de l'architecture ReFactor(Figure 1.2b) qui représente un AE-GNN conventionnel. Les deux architectures - HyFactor et ReFactor – sont comparables car elles ne diffèrent que sur la gestion de la représentation des structures moléculaires.

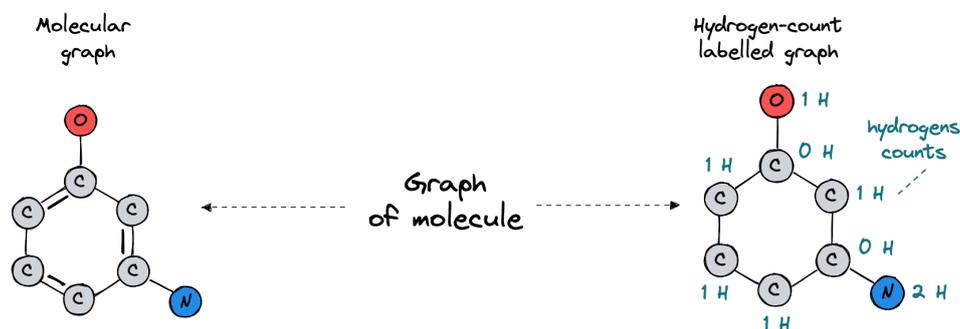


Fig. 1.1 Représentation de 3-aminophenol par un graphe moléculaire conventionnel (gauche) et par un graphe annoté par le compte d'hydrogènes (droite)

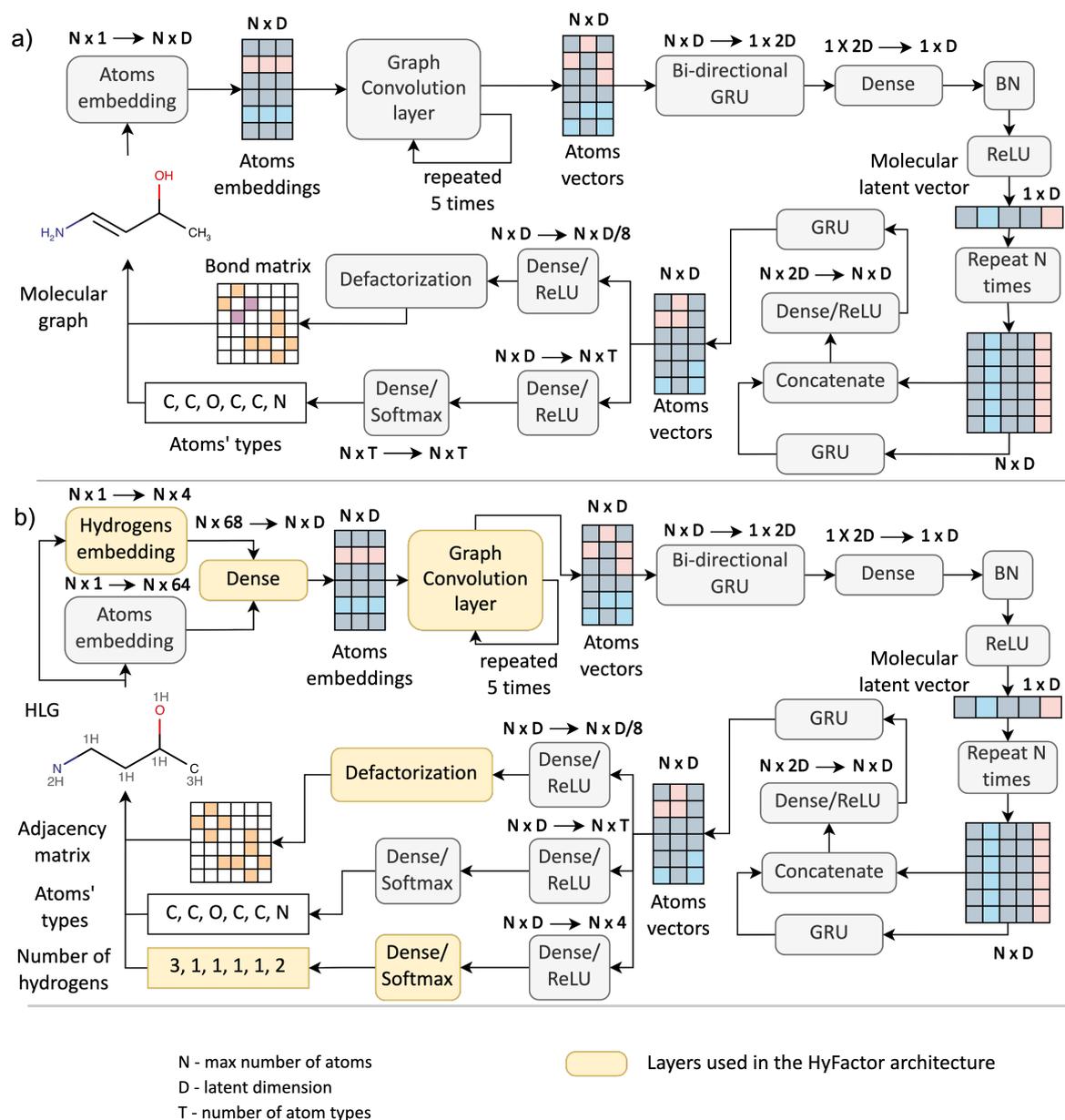


Fig. 1.2 Une représentation schématique des architectures HyFactor (a) et ReFactor (b) où les blocs en orange dépendent de la représentation des structures moléculaires.

Les HyFactor et ReFactor ont été comparés sur le jeu de données de référence ZINC 250K avec des AE-GNN (TSGCD, DEFactor et JTVAE) publiés dans la littérature. La métrique utilisée pour la comparaison était le taux de reconstruction qui mesure la fraction des structures présentées à l'encodeur qui ont été reconstruites avec succès par le décodeur. Les résultats montrent (Table 1.1) que HyFactor et ReFactor atteignent tous les deux une grande précision

dans la tâche de reconstruction et, par conséquent, qu'ils intègrent bien les caractéristiques structurelles des molécules.

Table 1.1 Résultats du benchmarking des autoencodeurs HyFactor et ReFactor sur le jeu de données ZINC 250K.

Architecture	TSGCD	DEFactor	JTVAE	ReFactor	HyFactor
Taux de reconstruction, %	90.5	89.8	76.7	90.7	89.0

Afin d'évaluer la performance de l'architecture basée sur la représentation HLG (HyFactor) par rapport à la représentation conventionnelle (ReFactor) les deux outils ont été entraînés sur la base de données ChEMBL v.27 contenant 1,6 million de molécules composées de 10 à 50 atomes lourds. Le jeu de données a été divisé en un jeu d'entraînement et un jeu de validation dans un rapport de 4 pour 1, respectivement. Les résultats résumés dans la Table 1.2 montrent que les deux outils ont des taux de reconstruction similaires mais HyFactor utilise moins de paramètres à ajuster et par conséquent est environ 1.5 fois plus rapide que ReFactor. Les graphes annotés HLG capturent donc les mêmes informations mais sont beaucoup plus économiques que le graphe moléculaire non annotés où tous les types de liaisons chimiques doivent être considérés explicitement.

Table 1.2 Résultats du benchmarking des autoencodeurs HyFactor et ReFactor sur le jeu de données ZINC 250K.

Architecture	Nombre de paramètres entraîna- bles, M	Temps par époque, min	Jeu d'entraî- nement	Jeu de vali- dation
ReFactor	50	24.3	99.8	95.2
HyFactor	40	16.5	99.7	95.0

Autoencodeur de graphes quantifiés par vecteur

Une analyse poussée a révélé que HyFactor peut générer différents vecteurs latents si on change la numérotation des atomes dans un graphe moléculaire. Cette propriété est indésirable car elle dégrade la performance de modèles prédictifs basés sur les vecteurs latents.

Inspiré par les récentes avancées dans le domaine de la génération d'images à l'aide de modèles d'apprentissage profond, un autoencodeur de graphes quantifiés par vecteur (Vector Quantized Graph AutoEncoder - VQGAE) a été développé, dont la propriété est qu'il est invariant vis-à-vis de la numérotation des atomes dans une structure chimique. Cette architecture

repose sur une opération de quantification par vecteur qui consiste à compresser l'information moléculaire (Figure 1.3). Pendant l'encodage, cette opération stocke dans des vecteurs des informations sur chaque atome et leurs voisinages respectifs. L'architecture développée est la première de son genre, capable d'apprendre l'espace des fragments moléculaires de manière non supervisée.

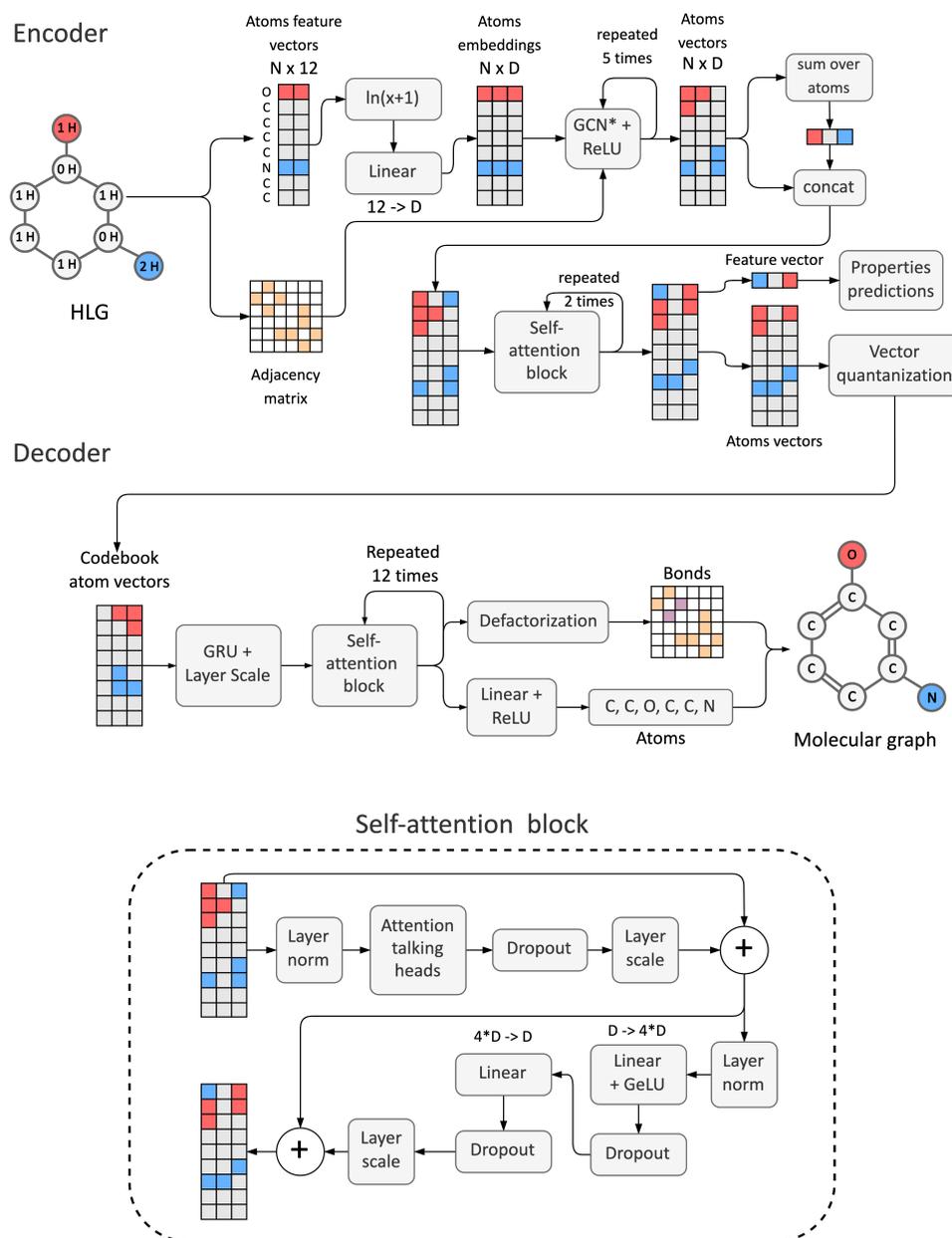


Fig. 1.3 Schéma de l'autoencodeur de graphes quantifiés par vecteur (VQAE).

Notons que le décodeur a quand même besoin d'une numérotation standard des nœuds du graphe pour une reconstruction réussie. Comme celle-ci ne peut être dérivée directement du

vecteur latent, nous avons développé un réseau dit "d'ordonnement" qui place les fragments dans un ordre canonique. Ce réseau est composé de 8 couches d'attention multi-têtes (MHA). La performance est mesurée par le taux de reconstruction de l'ordre, qui estime le pourcentage d'ordres de fragments correctement récupérés.

L'architecture VQGAE a été entraînée sur les données ChEMBL v.27 précédemment normalisé, avec la même répartition entre un jeu d'entraînement et un jeu de validation en proportion 4 :1, respectivement. Un taux de reconstruction élevé (94,6 %) sur le jeu de validation a été atteint. Pour une analyse supplémentaire de l'espace latent, les vecteurs de comptage de fragments VQGAE ont été utilisés comme descripteurs dans les modèles prédictifs Random Forest entraînés sous 532 jeux de données extraits de ChEMBL-27, dont chacun correspond à une activité biologique. À des fins de comparaison, les fingerprints ECFP, les descripteurs ISIDA, ainsi que les vecteurs latents des autoencoders HyFactor et LatentGAN ont été utilisés.

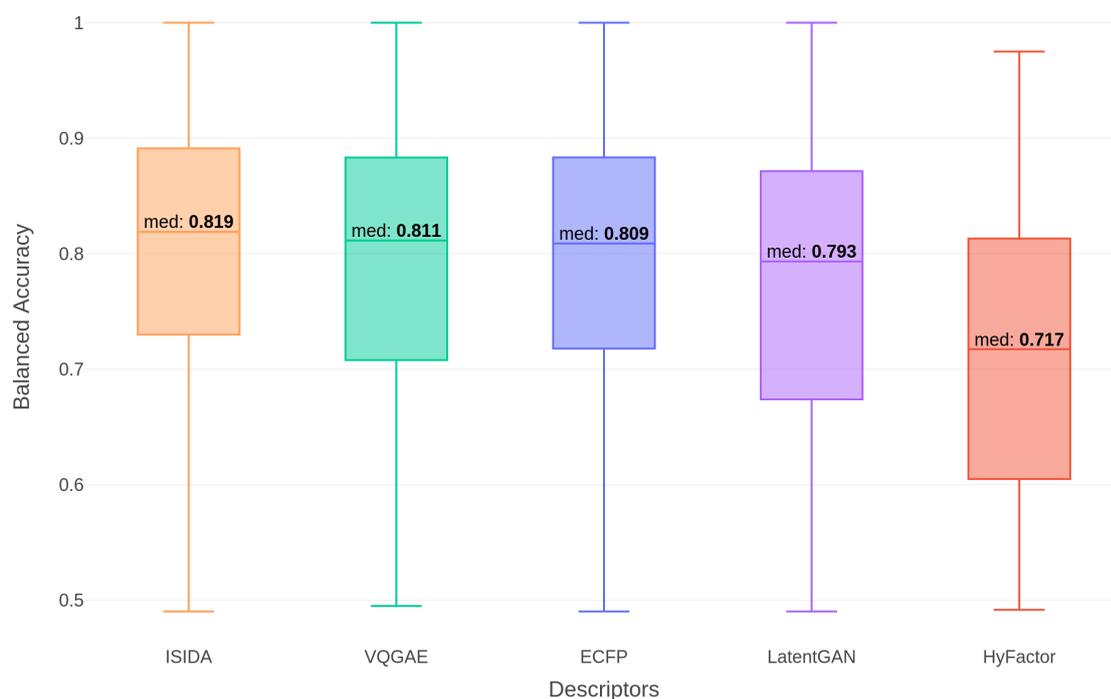


Fig. 1.4 (a) Résultats du benchmarking QSAR. Ici, "med" fait référence à la précision médiane équilibrée sur 532 cibles biologiques issues de ChEMBLv27.

Les résultats montrent (Figure 1.4) clairement que les modèles QSAR construits sur les vecteurs latents VQGAE montrent de meilleures performances par rapport au modèle HyFactor.

En même temps, ses performances sont équivalentes à celles des descripteurs ISIDA bien connus et qui sont une référence dans les approches QSAR.

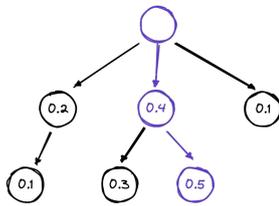
1.2.2 Analyse de la faisabilité de la synthèse des molécules

Les modèles génératifs sont capables d'optimiser la structure en fonction des propriétés physico-chimiques d'intérêt. Cependant, ils ne peuvent pas optimiser directement l'accessibilité synthétique d'une molécule. Une façon d'estimer l'accessibilité synthétique consiste à effectuer une recherche rétrosynthétique. Étant donné un ensemble de transformations chimiques, l'algorithme divise les molécules cibles en précurseurs jusqu'au niveau de « blocs de construction » (building blocks), les unités de base pour démarrer une synthèse. Cependant, le nombre de voies de synthèse possibles augmente exponentiellement avec la taille de la structure générée. Par conséquent, le but de ce projet est de créer un nouvel outil de rétro-synthèse pour évaluer l'accessibilité synthétique des structures générées à l'aide d'un AE.

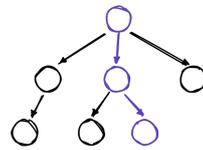
Rétrosynthèse par auto-apprentissage

Récemment, les algorithmes de rétro-synthèse par recherche arborescente de Monte-Carlo (Monte-Carlo Tree Search, MCTS) ont montré des gains de performance considérables par rapport aux approches heuristiques précédentes. Les MCTS sont un processus itératif, où à chaque itération, un nouveau nœud de l'arbre de recherche est créé (Figure 1.5). Ce nœud est constitué des précurseurs nécessaires pour synthétiser la molécule du nœud parent. L'ordre dans lequel les nœuds sont créés est déterminé par le "cœur" de l'algorithme MCTS qui définit des fonctions de politique et d'évaluation. En bref, une fonction de politique doit sélectionner et classer les transformations rétro-synthétiques (RST) qui permettent d'accéder à une molécule (un nœud de l'arbre) à partir de précurseurs (les descendants dans l'arbre). Ensuite, la fonction évaluation doit estimer la vraisemblance de la rétro-synthèse obtenue. L'algorithme va concentrer son exploration de l'arbre sur les branches qui obtiennent les meilleurs scores produits par la fonction évaluation. En suivant jusqu'un bout les branches disposant des meilleurs scores, on obtient les propositions les plus prometteuses de voie de synthèse.

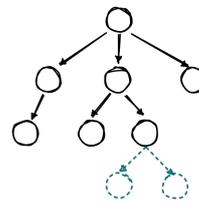
At the stage of selection the algorithm descends to the nodes with the maximum value of upper confidence bound (UCB)



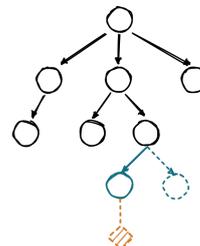
1) Selection



2) Expansion



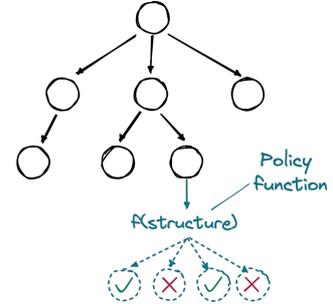
3) Evaluation



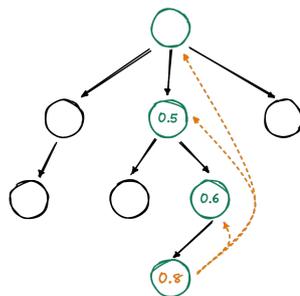
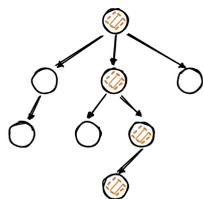
$f(\text{structure})$

The value function assigns a score to the newly created node

During expansion a policy function chooses and ranks the most potent rules



4) Update



In the end of iteration the value of the new node is used to update all parent nodes

Fig. 1.5 Les principales étapes de la recherche arborescente de Monte-Carlo. Ici, à chaque itération, un nœud est créé et ajouté à l'arbre.

Parmi les différentes approches publiées, la plus populaire consiste à utiliser une stratégie « rollout » (Figure 1.6a). L'idée du rollout est la notation rapide du nœud actuel par la création séquentielle de nœuds enfants sur la base des RST ordonnés par la fonction politique. Le processus est répété jusqu'à ce qu'un nœud ne soit constitué que de blocs de construction (fin de la rétro-synthèse) ou qu'une profondeur maximale de l'arbre soit atteinte. La principale limite de cette approche est sa forte dépendance à la fonction politique. Ainsi, une fonction de politique imparfaite ne peut pas toujours prioriser la meilleure RST pour une structure donnée, ce qui conduit à une exploration inutile de l'arbre de recherche. Une autre idée consiste à remplacer la stratégie rollout par une intelligence artificielle chargée d'explorer l'arbre rétro-synthétique dans toute son ampleur (Figure 1.6b). Le principal défi est qu'il n'existe pas de moyen évident d'estimer la proximité (en termes de transformations) entre une molécule et les blocs de construction. Il a donc été proposé d'utiliser une technique d'auto-apprentissage (self-learning) pour construire un réseau de neurones artificiels capable de prendre ces décisions (Figure 1.7). L'idée de l'auto-apprentissage est d'entraîner le modèle sur des arbres de recherche de rétro-synthèse précédemment réalisés. L'auto-apprentissage a été appliqué[2] à la MCTS pour la rétro-synthèse, mais il n'a pas été comparé aux méthodes précédentes.

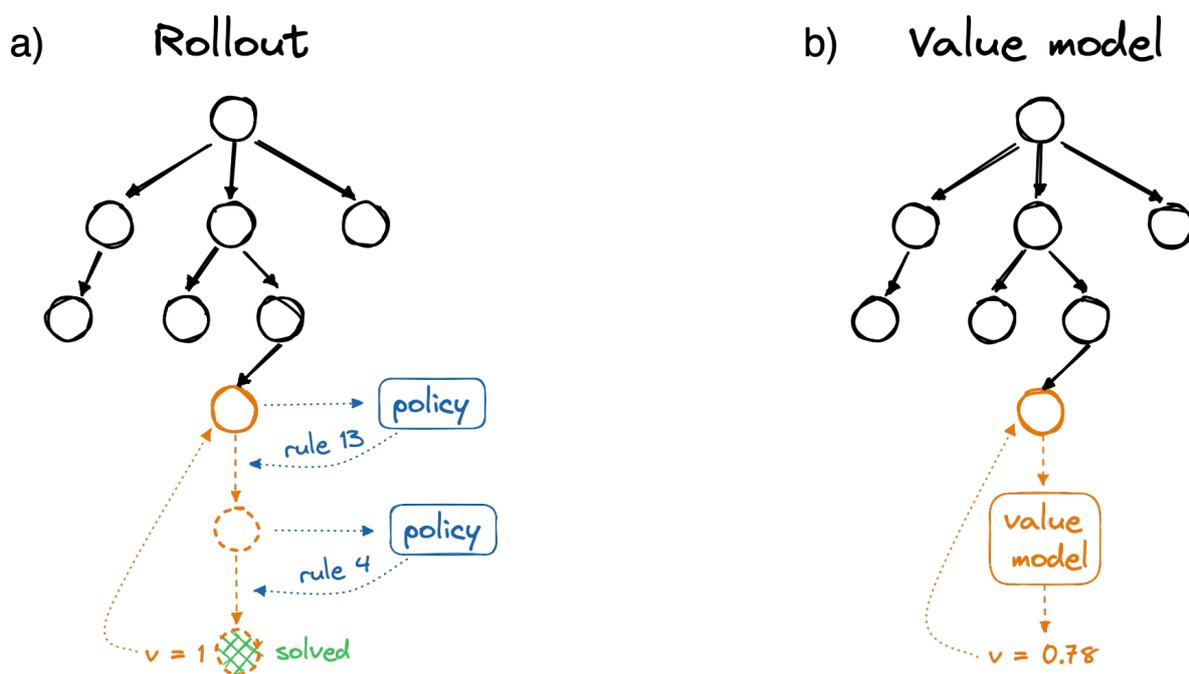


Fig. 1.6 a) Une fonction de rollout descend l'arbre en utilisant les règles top-1 les plus probables selon le réseau de politiques. Si rollout a trouvé un nœud constitué uniquement de blocs de construction, il renvoie un score de 1, sinon un score de 0. b) Un modèle de "valeur" qui prédit le score du nœud en fonction des structures du nœud évalué.

Dans ce travail, nous étendons les approches actuelles avec une nouvelle variante de la recherche de rétro-synthèses par MCTS appelée GSLRetro (Graph-based Self-Learning Retrosynthesis), où les fonctions politique et valeur sont composées de réseaux de neurones de graphes basés sur des VQGAE. Avec cet outil, nous proposons une nouvelle stratégie d'auto-apprentissage avec une technique « epsilon-greedy ». Cette technique force l'algorithme MCTS à explorer des voies de synthèses diverses au lieu de n'exploiter que la première proposée par la fonction politique. Cette architecture a été comparée à d'autres implémentant la stratégie rollout, en particulier la référence en la matière : AiZynthFinder[3]

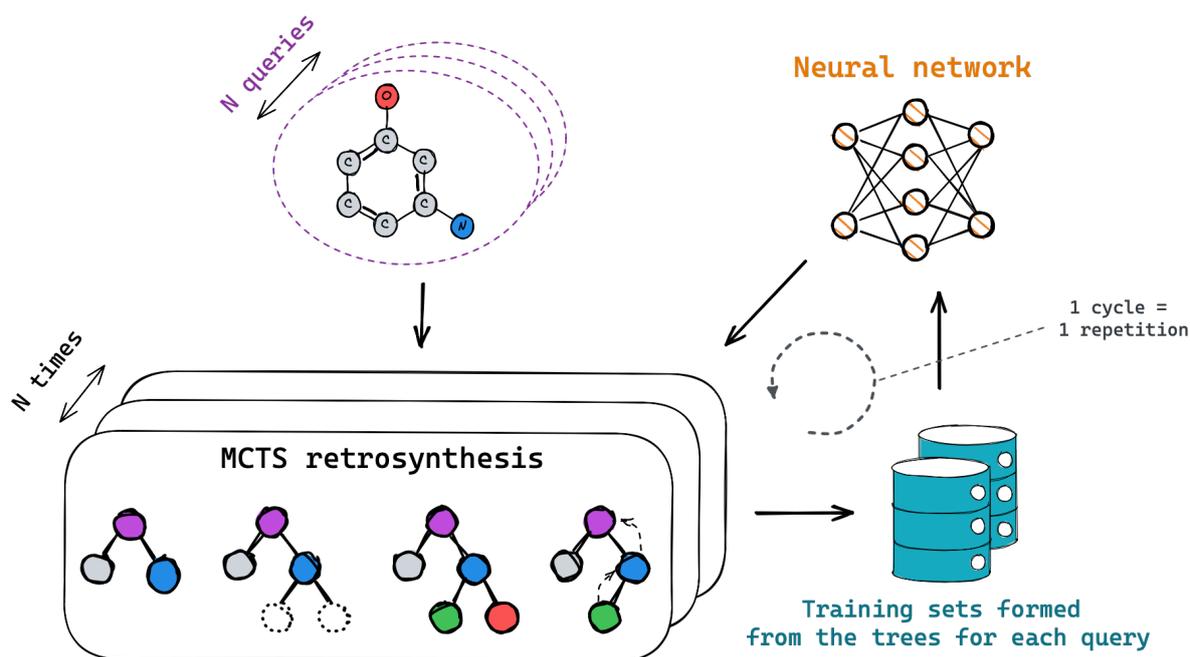


Fig. 1.7 Schéma du concept d'auto-apprentissage.

Le MCTS nécessite trois éléments : des règles de rétro-synthèse (RST), un ensemble de blocs de construction et une structure moléculaire à synthétiser. Les RST ont été extraites du jeu de données open-source de l'USPTO. Ce jeu de données contient une grande quantité de données de réactions incorrectes, qui ont été corrigées selon le protocole proposé dans notre étude[4]. Au total, 1 million de réactions ont été traitées et 12 000 RST ont été extraites. Un ensemble de blocs de construction a été extrait des catalogues des sociétés E-molécules et Sigma-Aldrich. Nous avons également constitué un ensemble de molécules cibles pour une évaluation comparative, réparties de manière égale entre des molécules "faciles à synthétiser" et des molécules "complexes" (semblables à des produits naturels). Les données ont été collectées à partir des bases de données ChEMBL et Coconut, puis divisées en 7 catégories selon le score d'accessibilité synthétique (SAScore)[5] (Figure 1.8). Ces jeux de données ont ensuite

été échantillonnés pour en maximiser la diversité selon un algorithme MaxMin. Le modèle entraîné sur 28K molécules a été testé sur un jeu de test de 700 molécules. Les résultats sont présentés sur la Figure 1.9.



Fig. 1.8 Préparation du flux de travail pour les molécules d'entraînement et de test à partir des bases de données ChEMBL et COCONUT.

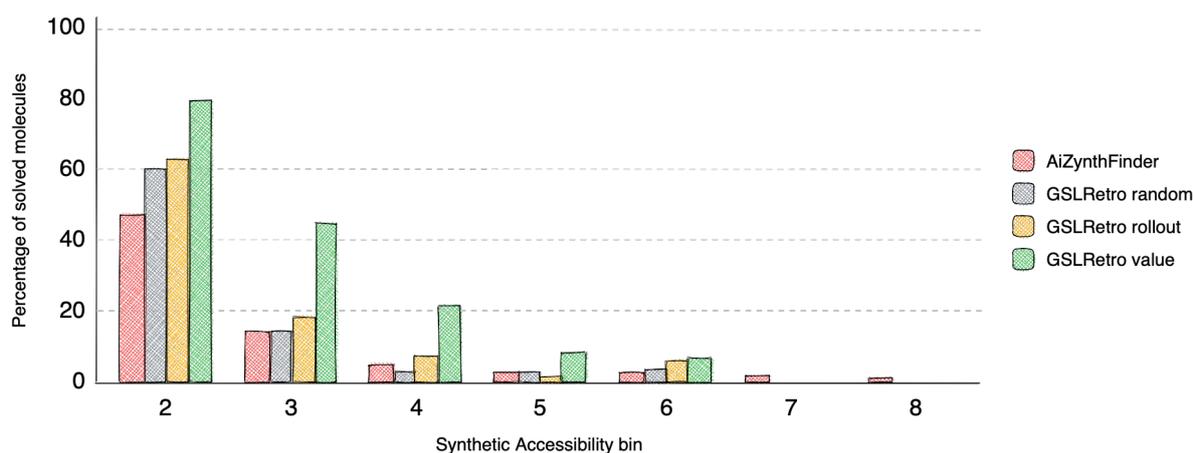


Fig. 1.9 Résultats sur des jeux d'essai d'évaluation comparative pour différentes approches rétro-synthétiques. AiZynthFinder[3] est une approche représentative de l'état de l'art avec une stratégie « rollout ». Pour comparaison, GSLRetro avec différentes fonctions d'évaluation telles que : aléatoire, « rollout » et le modèle auto-entraîné.

Les résultats montrent que GSLRetro surpasse l'algorithme de référence (AiZynthFinder[3]) dans une configuration similaire. En même temps, la fonction de valeur proposée permet de trouver des voies rétro-synthétiques pour des molécules complexes avec des performances nettement améliorées. La seule limite est la recherche de voies de synthèse pour les produits naturels avec un SAScore élevé. Ce problème pourrait être résolu par l'extraction de RST à partir de bases de données de réactions plus importantes et plus propres, telles que Reaxys, qui feront le sujet de travaux futurs. Des exemples de voies de synthèse trouvées par GSLRetro et AiZynthFinder sont donnés dans les figures 1.10, 1.11 et 1.12.

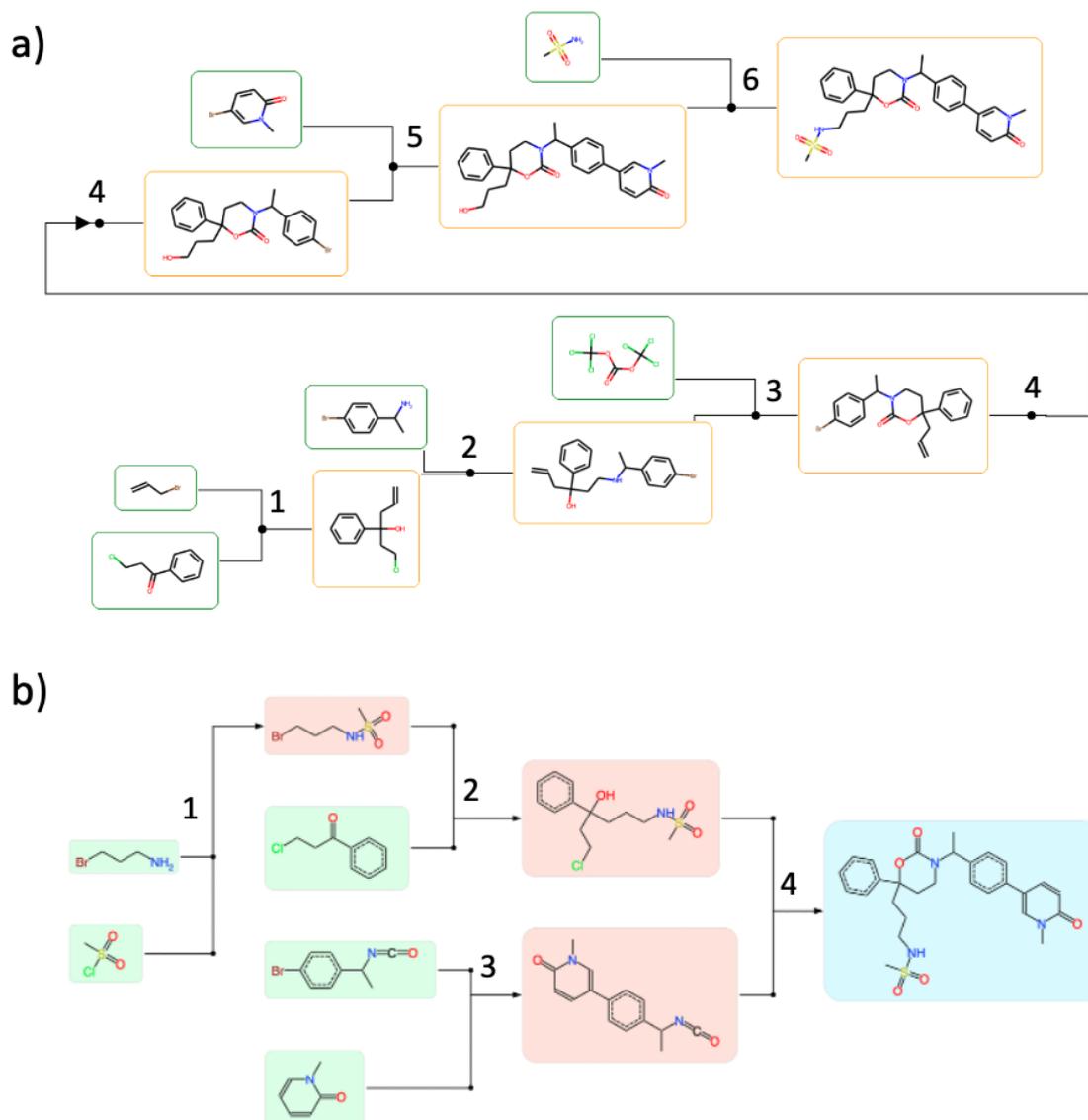


Fig. 1.10 Exemples de différentes voies de synthèse pour la même cible résolues par AiZynthFinder original (a) et GSLRetro avec réseau de valeurs (b). Ici, chaque chiffre correspond à une transformation de la réaction.

1.2.3 Exploration de l'espace chimique du récepteur de l'adénosine

Dans la dernière partie de la thèse, les méthodologies développées de conception moléculaire et d'analyse rétro-synthétique ont été testées pour générer des molécules actives contre le récepteur A2A de l'adénosine qui appartient à la famille des récepteurs couplés aux protéines G (GCPR). Un modèle Random Forest (RF) a été entraîné sur le jeu de données de 3300 molécules extraites de la base de données ChEMBL en utilisant les vecteurs latents de l'encodeur VQGAE comme descripteurs. Le modèle entraîné a été utilisé comme fonction de score pour un algorithme génétique (GA). Les vecteurs optimisés par le GA ont été décodés par le décodeur VQGAE, ce qui a mené à 4 nouvelles structures avec les valeurs prédites de $K_i < 10$ nM (Figure 1.13). Les voies de synthèse de ces structures ont été retrouvées à l'aide de GSLRetro, comme illustré sur la Figure 1.14.

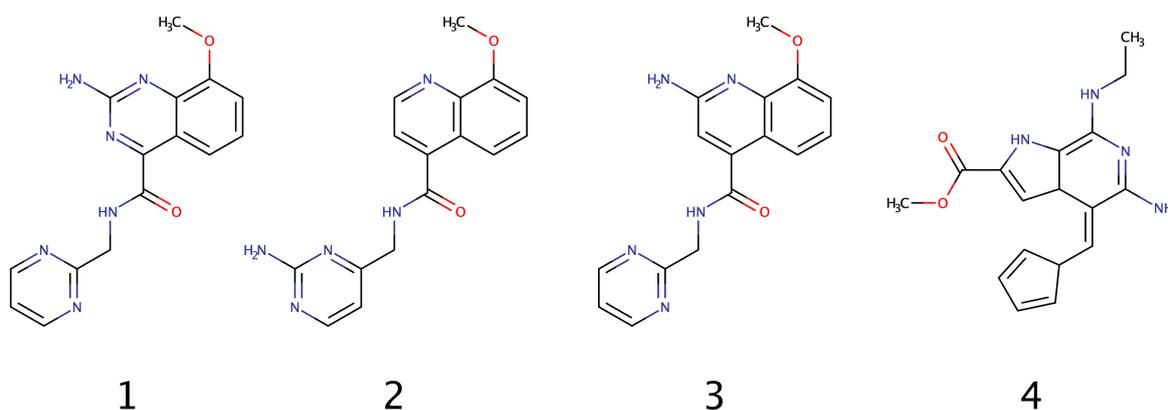


Fig. 1.13 Structures moléculaires générées avec VQGAE.

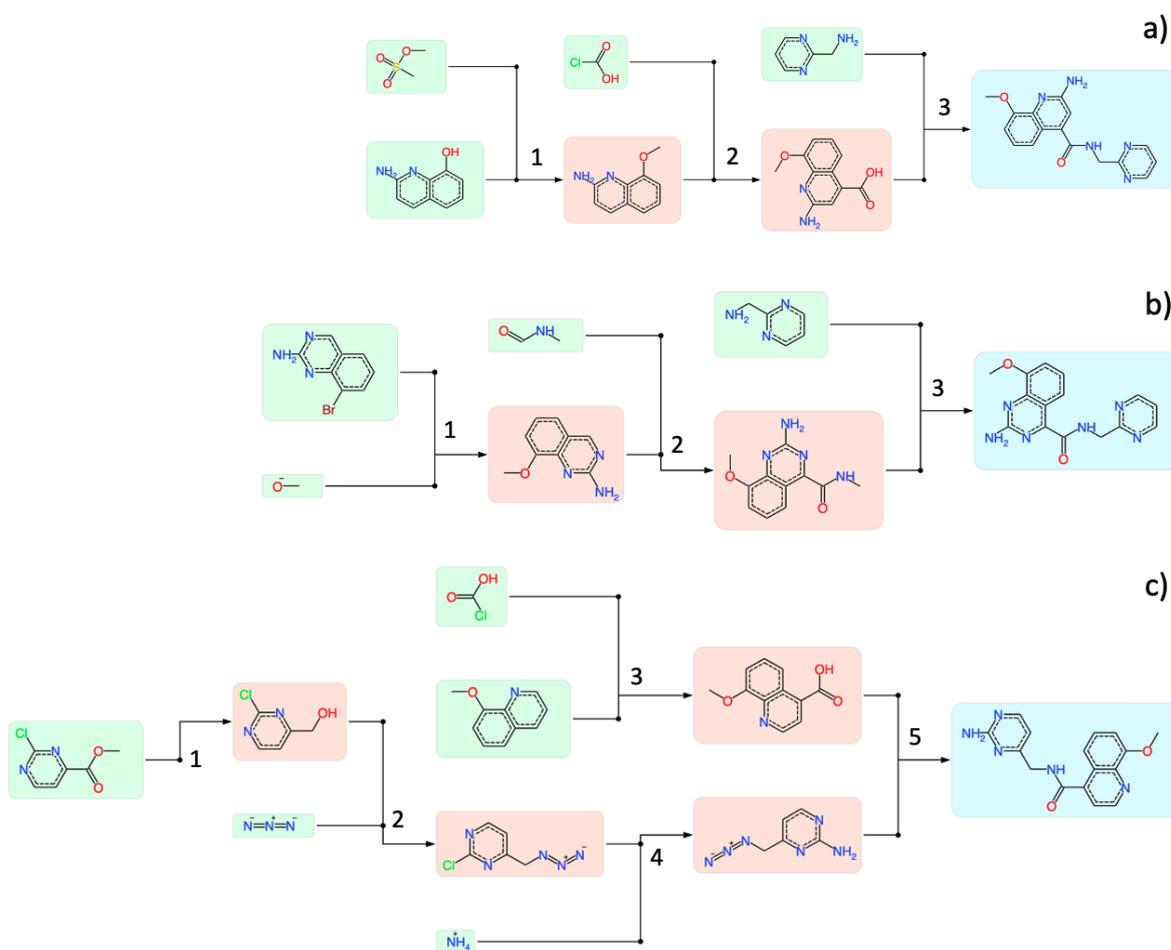


Fig. 1.14 Exemple de voie de synthèse pour le ligand généré pour le récepteur A2A de l'adénosine. En bleu une molécule générée par VQGAE, en rouge les molécules intermédiaires et en vert les blocs de construction.

Nous avons retrouvé que l'une de structures générée (1ère dans la Figure 1.13) est déjà enregistrée dans la base de données de brevets SureChEMBL en tant qu'antagoniste A2A. Ceci démontre le potentiel de notre méthodologie pour générer les molécules possédant des activités désirées et de vérifier leur faisabilité synthétique en fournissant une voie de synthèse.

1.3 Conclusion generale

Trois nouvelles méthodologies basées sur des réseaux de neurones de graphe ont été développées pour trouver de nouvelles structures chimiques et des voies de synthèse. La première méthodologie, HyFactor, s'est attachée à réduire les paramètres des autoencodeurs en utilisant une nouvelle représentation de la molécule, les graphes labellisés par les nombres d'hydrogène (HLG). Il a été démontré que HyFactor a 20% de paramètres en moins et est 1.5 fois plus rapide pendant l'entraînement que ReFactor, qui est basé sur un graphe moléculaire non annoté. En même temps, HyFactor montre des performances de reconstruction élevées sur les jeux de données ZINC 250K et ChEMBL, similaires à celles de ReFactor et des autoencodeurs de la littérature. Cela prouve que la représentation HLG de la molécule est utile pour réduire les paramètres du réseau de neurones et peut refléter efficacement la structure de la molécule.

Cependant, il a été constaté que les vecteurs latents générés par HyFactor dépendent de la numérotation des atomes. Ce problème a été résolu par une nouvelle architecture appelée autoencodeur de graphes quantifiés vectoriels (VQGAE). Cet autoencodeur a montré une haute performance de reconstruction (comparable à HyFactor et ReFactor) et ses vecteurs latents se sont avérés efficaces dans les simulations QSAR. VQGAE a été utilisé pour générer des molécules actives contre les récepteurs A2A en utilisant un algorithme génétique. Bien que le nombre de structures générées soit assez faible, il a été constaté que certaines d'entre elles existent déjà dans d'autres bases de données et sont connues pour être actives contre cette cible.

Dans la deuxième partie de la thèse, un nouvel outil de recherche rétro-synthétique basé sur le MCTS et les réseaux de neurones de graphes a été développé. Cet outil est capable d'apprendre et de s'améliorer à partir des recherches précédentes et a atteint une performance rivalisant avec les outils de référence sur un ensemble de données extraits des bases de données ChEMBL et Coconut. L'outil a été utilisé pour trouver des voies de synthèse pour les structures générées et prédites comme étant actives sur le récepteurs A2A et a trouvé avec succès des voies pour chacune d'entre elles. En conclusion, il a été démontré que toutes les méthodologies peuvent former un système de recommandation efficace pour la tâche de conception moléculaire de novo.

1.4 Liste des presentation

- **T. Akhmetshin**, A. Lin, D. Mazitov, Y. Zabolotna, E. Ziaikin, T. Madzhidov, A. Varnek. HyFactor: Hydrogen-count labelled graph-based defactorization autoencoder, Strasbourg Summer School in Chemoinformatics, Strasbourg, France, 28 July 2022 (poster).
- **T. Akhmetshin**, A. Lin, T. Madzhidov, A. Varnek, Presentation on PhD project at UMR scientific day, Strasbourg, France, 10 May 2022 (oral).
- **T. Akhmetshin**, A. Lin, T. Madzhidov, A. Varnek. Order-independent graph-based AutoEncoder for exploration of chemical space, 4th Artificial Intelligence in Chemistry Symposium, Cambridge, United Kingdom, 28 September 2021 (poster).
- **T. Akhmetshin**, T. Gimadiev, R. Nugmanov, T. Madzhidov, A. Varnek. Reactions cleaning: Big data challenge, IV Kazan Summer School on Chemoinformatics, Kazan, Russia, 7 June 2019 (oral).
- **T. Akhmetshin**, R. Nugmanov, T. Madzhidov, A. Varnek. Transformation-out and solvent-out strategies for reaction model's predictive ability assessment, III International School-Seminar From Empirical to Predictive chemistry, Kazan, Russia, 6 April 2018 (poster).
- **T. Akhmetshin**, R. Nugmanov, T. Madzhidov, A. Varnek. Development of a prediction's performance metric for chemical reactions' models, III Kazan Summer School on Chemoinformatics, Kazan, Russia, 6 July 2017 (poster).

1.5 Liste des publication

- **Akhmetshin, T.**; Lin, A.; Mazitov, D.; Zabolotna, Y.; Ziaikin, E.; Madzhidov, T.; Varnek, A. HyFactor: A Novel Open-Source, Graph-Based Architecture for Chemical Structure Generation. *J. Chem. Inf. Model.* 2022, 62 (15), 3524–3534. DOI: 10.1021/acs.jcim.2c00744
- Gimadiev, T. R.; Lin, A.; Afonina, V. A.; Batyrshin, D.; Nugmanov, R. I.; **Akhmetshin, T.**; Sidorov, P.; Duybankova, N.; Verhoeven, J.; Wegner, J.; Ceulemans, H.; Gedich, A.; Madzhidov, T. I.; Varnek, A. Reaction Data CurationI: Chemical Structures and Transformations Standardization. *Mol.Inform.* 2021, 40 (12), 1–16. DOI:10.1002/minf.202100119
- Rakhimbekova, A.; **Akhmetshin, T. N.**; Minibaeva, G. I.; Nugmanov, R. I.; Gimadiev, T. R.; Madzhidov, T. I.; Baskin, I. I.; Varnek, A. Cross-Validation Strategies in QSPR Modelling of Chemical Reactions. *SAR QSAR Environ. Res.* 2021, 32 (3), 207–219. DOI: 10.1080/1062936X.2021.1883107
- Nugmanov, R. I.; Mukhametgaleev, R. N.; **Akhmetshin, T.**; Gimadiev, T. R.; Afonina, V. A.; Madzhidov, T. I.; Varnek, A. CGRtools: Python Library for Molecule, Reaction, and Condensed Graph of Reaction Processing. *J. Chem. Inf. Model.* 2019, 59, 2516–2521. DOI: 10.1021/acs.jcim.9b00102

Chapter 2

Introduction

The concept of fully automated molecular design holds the potential to enable the rapid production of drugs for various diseases and the discovery of new materials in a matter of days without human intervention. Once considered science fiction, recent developments have brought the dream of autonomous design closer to reality.

The main challenge towards automated molecular design is the exploration of a chemical space [6, 7]. Chemoinformatics defines chemical space as a universe of all possible molecules. While the chemical space is infinite by definition, in practice, it refers to the subspace of drug-like compounds, estimated to consist of 10^{33} molecules [8]. The positions of molecules in the chemical space are encoded by their vector representations – numerical descriptions of their physicochemical and structural properties

One of the most important computational methods used to explore chemical space is virtual screening [9, 10]. The idea behind virtual screening is to search or “screen” chemical databases of synthesised or virtually created compounds and identify those with desirable properties. This is the method of choice when molecule availability is a key constraint. The researcher need only virtually browse the list of available compounds to ensure that the virtual hits are guaranteed to be on the shelf and ready for testing or at least highly likely to be possible products of carefully selected virtual combinatorial libraries. The screening uses simple statistical filters such as Lipinski’s rule of five [11] and techniques such as docking and quantitative structure-activity relationships (QSAR) modelling. For its high efficiency, virtual screening is still actively used today. However, it is limited in chemical space exploration, as generating and screening all possible drug-like compounds is impossible.

One concept developed to explore the unknown chemical space without its complete enumeration is “de novo” molecular design [12]. The de novo design aims to identify an optimal chemical structure that meets a specified set of physicochemical properties. However, this search process assumes that the visited points in the chemical space can be related to the

desired properties, despite the absence of a universally applicable relationship between chemical space and measurable properties. Therefore, such relationships (Quantitative Structure-Activity Relationship, QSAR) need to be learned, associating properties of known molecules to the chemical space zones in which they reside, and then fitting property predictor functions on top of this data. Methods for de novo design generate molecules from scratch using iterative search methods and the above property predictor models, acting as objective functions to optimise. Previously, these search methods were based on heuristics modifying the actual structures, where compounds were constructed using atoms or pre-defined fragments [13]. While these methods have efficiently explored chemical space, some have been limited in their ability to operate in large spaces or have relied on human expert rules for assembling structures. [14]

In the past decade, chemical space exploration has significantly transformed due to the emergence of deep learning (DL) techniques [15]. Deep learning is a subclass of machine learning based on deep neural networks. While neural networks existed long before, deep neural networks gained popularity much after. Starting from the outstanding performance of the “AlexNet” neural network[16] in computer vision in 2012, the potential of DL to outperform other machine learning approaches led to its adoption in various fields, including natural language processing[17], speech recognition[18], and protein folding[19]. In chemoinformatics, the capacity of neural networks to accept a massive amount of raw data input without pre-processing into a vector format has been advantageous, leading to the development of the first DL models for molecular structure generation[14, 20] and chemical synthesis planning[21].

In recent years, we have just begun to tap into the limitless potential of neural networks, and there is still so much more to discover and achieve. However, while various DL architectures have been developed, not all innovative approaches have been made publicly available, and our understanding of what they learn is limited [22, 23]. Moreover, like any other machine learning method, neural networks can learn biased knowledge and make incorrect associations due to inaccurate input data representation. This issue has also been observed in DL models for chemistry, where a phenomenon known as atom ordering bias has been identified and has affected many models for encoding chemical structures, particularly those based on alphanumerical string representations of molecules such as Simplified Molecular-Input Line-Entry Systems (SMILES) [24]. One way to mitigate this issue is to use graph-based representations of molecules as inputs to neural networks. In chemistry, molecules are often represented as graphs because this 2D representation captures most of the molecule’s structural properties and graph neural networks (GNNs) are invariant to the ordering of atoms. However, when it comes to generating new molecular structures, models cannot be fully invariant and require sequential operations [25]. As a result, string-based neural networks are still the most widely used method for structure generation.

The design of novel molecules using generative deep learning models presents several challenges, one of the most significant of which is the synthetic feasibility of the generated structures.[26] The synthetic feasibility of a structure refers to its ability to be chemically synthesised, which is a non-trivial property to measure. Indeed, to be synthetically feasible, a compound needs to be thermodynamically stable at common temperatures – a physicochemical property already challenging to predict. However, this is not sufficient – there must also be a reaction path leading to it specifically rather than to other concurrent products. Furthermore, the most challenging part is that this path must imply available and cheap building blocks and eco-friendly reaction conditions. Herewith, feasibility is not only a physicochemical but also a logistical and economic issue – a practical problem with forcibly empirical and never perfect answers. Previous works[14] used empirically calibrated functions of the structural features in the molecule, such as Synthetic Accessibility score (SAScore)[5] as scoring functions or filters. Even if a score can predict whether a compound is principally feasible and stable, it cannot predict whether it can be practically produced because it does not include logistical and economic constraints: availability and cost of reactants, catalysts, reactors, etc. Another approach used is based on the combination of generation by forward synthesis and selection by some property[27], where neural networks are trained on existing reaction data to predict possible products given a set of available reactants. These models can ensure the availability of starting materials; however, goal-directed optimisation can be challenging as it is unclear which choice of starting molecules and products will lead to the maximisation of desired properties. In addition, the search space of available molecules and possible reactions can be vast, which only makes the task more difficult. In contrast, with the development of DL-based retrosynthetic search, the combination of generative neural networks with retrosynthetic tools became a promising direction. Retrosynthetic planning algorithms are designed to find synthetic pathways for molecules, which can ensure their synthetic feasibility. Thus, retrosynthetic tools can not only filter out those molecules that cannot be synthesised (from available reactants) but also provides synthetic pathways for others.

The integration of generative neural networks with deep learning-enabled retrosynthesis may represent the ultimate solution for the automated design of novel molecules, an endeavour that is notoriously challenging. In light of this, this thesis attempts to address the problems associated with the use of graph generative neural networks to find new molecular structures and optimise retrosynthesis algorithms.

In this work we focused on the following topics:

1. **Neural Network architecture based on hydrogen count labeled graph.** Compared to conventional graph-based NN, the new architecture has significantly lower number of parameters
2. **New architecture of generative GNN:** we present a new graph-based generative model that learns order-independent vector representations of molecules. Order independence was achieved through the use of permutation invariant operations. In addition, the architecture of GNN is based on a vector quantisation technique that enables the demonstration of a new concept called unsupervised fragment learning.
3. **Methodology for reaction data curation:** a unified protocol for standardisation of reaction data, including standardisation of chemical structures and reaction transformations.
4. **New retrosynthetic planning tool based on neural networks:** the new instrument for retrosynthetic search that can learn new synthesis strategies by training from results of previous searches.
5. **New cross-validation techniques for reaction data:** with these techniques, it is possible to evaluate the unbiased performance of ML and DL models in predicting physicochemical parameters of reactions in new transformations and conditions.

The manuscript contains two parts devoted to graph-based deep learning architectures for structure generation and synthesis planning, respectively.

Chapter 3

Generation of molecular structures

The de novo molecular design field was typically formulated through the goal-directed optimisation paradigm. Within this framework, generated molecules were evaluated using a scoring function, which evaluates whether they possess the desired properties. The scoring function can be based on molecular simulation, docking, or structure-activity relationships (QSAR) models. The latter approach, which utilises machine learning models to predict molecular properties, has become increasingly popular due to its computational efficiency and precision of predictions.[28, 29]

The performance of QSAR models depends to a large extent on the choice of vector representation, or “descriptors,” used to represent the molecular structures. Descriptors can be based on experimental measurements of physicochemical properties (e.g., molecular weight, lipophilicity, and refractivity) or structural features (e.g., molecular fragments and topological indices). Over time, a set of requirements for “good” molecular descriptors has been established, including that they should have a structural interpretation, correlate with at least one molecular property, and be invariant to the numbering of atoms in a molecule. [30]

While QSAR models can be used as scoring functions and filters of generated molecules, they can also provide a vector that might correspond to the structure with the highest properties of interest. Searching for the “optimal” vector and corresponding molecular structure is called the inverse QSAR task[31, 32]. Compared to other de novo molecular design approaches, inverse QSAR algorithms can be fast methods for creating new molecular structures with a high probability of possessing the desired properties. This task can be divided into two main subtasks: searching for a molecular descriptor that maximises the expectation of having the most suitable properties and searching for corresponding molecular structures to a given vector representation. Although the first task is conceptually straightforward, the second is computationally challenging to solve [33]. As a result, previous approaches to the inverse

QSAR task have not been widely used, as existing generators based on fragment descriptors often yielded a large number of solutions.[34, 35]

In recent years, the development of modern DL algorithms has revolutionised the field of “de novo” molecular design[32]. The neural networks, characterised by their ability to learn patterns from large datasets[36], have enabled the creation of models suitable for inverse QSAR tasks. This chapter will discuss new approaches for inverse QSAR using GNNs. The discussion is structured as follows:

- Section 3.1 introduces existing concepts in deep learning for inverse QSAR of molecules. Initially, two main molecular representations, SMILES and molecular graphs, are discussed. Then, several classifications of generative neural networks are provided with examples of known architectures. Next, a particular class of the most promising architectures, graph-based autoencoders, is described in more detail. After, the applications of generative architectures are presented. Lastly, the current challenges for generative models are outlined.
- Section 3.2 defines the main goal of this work and presents our main developments:
 - Section 3.2.1 presents a new generative neural network based on a graph representation of molecules called the Hydrogen-count Labelled Graph (HLG). The new architecture was tested in the reconstruction task and the generation of analogues of molecules. For benchmarking, a new autoencoder based on molecular graphs, called ReFactor, was also developed and compared in the same tasks.
 - Section 3.2.2 describes the new order-independent graph-based autoencoder. This autoencoder can learn structural fragments in an unsupervised manner, making it the first of its kind. The latent vectors of this autoencoder are compared to other generative architectures in similarity search and QSAR tasks.

3.1 Background of generative deep learning for inverse QSAR

3.1.1 Molecular representations

One of the key benefits of deep learning methods is the capability to extract knowledge directly from “raw” molecular representations[36], leading to the question: “which representation would be the most suitable for the efficient generative architectures?”. The typical choice is string- and graph- based representations.

The common string-based approaches use SMILES[24] or InChI[37, 38] representations of molecules. The SMILES notation, in particular, has gained widespread usage due to its

compatibility with natural language processing (NLP) deep learning methods, which have been shown to achieve high performance in the generation of new text. However, it should be noted that the SMILES representation of a molecule is not unique due to the linear traversal of the molecular graph used in its construction. Depending on the starting point and path taken, this can result in different SMILES strings for the same molecular structure. This issue was addressed with the concept of canonical SMILES notation, which formalises the traversal of the molecular graph with the Morgan algorithm. Despite this, it has been observed that generative models trained on canonical SMILES are affected by atom order bias. Studies have shown that augmenting training data with non-canonical SMILES can improve the performance of generative models.[39] However, models that encode molecular structures in their vector representations are more affected by atoms order bias and require more sophisticated solutions to mitigate this issue.[1] Alternative representations of SMILES, such as DeepSMILES[40] and SELFIES[41], have been proposed as well, which are better suited for use with deep learning algorithms.

Another form of representation is graph-based, specifically molecular graphs. In comparison to strings, molecular graphs do not require special symbols to encode the arrangement of atoms and bonds. One benefit of this representation is the absence of atoms ordering bias. Graphs, by nature, are order-invariant objects and are widely used in chemistry. However, graph-based generative architectures are complex in optimisation, especially those generating graphs atom by atom, also known as atom-based generators. As an alternative, some approaches focus on the reconstruction of molecules through the combination of predefined molecular fragments, known as fragment-based approaches[42, 43]. The effectiveness of such generators is contingent on the diversity and specificity of the extracted fragments. These methods are becoming more popular due to the reduced complexity of the generation process. An exception to this classification is the work of Maziarz et al.[44], where both atom-based and fragment-based approaches were combined, allowing for the decoder to select from individual atoms or predefined fragments

3.1.2 Types of neural networks for inverse QSAR

Generative neural networks suitable for inverse QSAR must generate molecules from their vector representations. As such, generative neural networks can be broadly classified into two categories: those that generate molecules from existing descriptors, such as physicochemical properties or fragment descriptors, and those that formulate new “reversible” vector representations from “raw” molecular representations (both structure \rightarrow descriptor vector and descriptor \rightarrow structure transformation rules being learned in parallel).

The first category of generative neural networks used in inverse QSAR is known as conditional generative models. These models use a calculated descriptor as a starting point to

generate corresponding molecules. A popular class of conditional generators is the iterative generators. These models originated in natural language processing, where they are used to predict the next symbol in a string. Due to its simplicity of training and speed of generation, the combination of iterative generators with SMILES strings has gained significant popularity in molecular design. One subclass of iterative approaches is based on recurrent neural networks (RNNs), which predict the next symbol based on a hidden vector that retains information about what should be generated in the following steps (Figure 3.1a)[45, 20]. Another subclass of iterative generators is based on autoregressive modelling. In contrast with RNNs, autoregressive generators predict the following symbols from the vectors of previous symbols given to the network explicitly (Figure 3.1b)[46, 47].

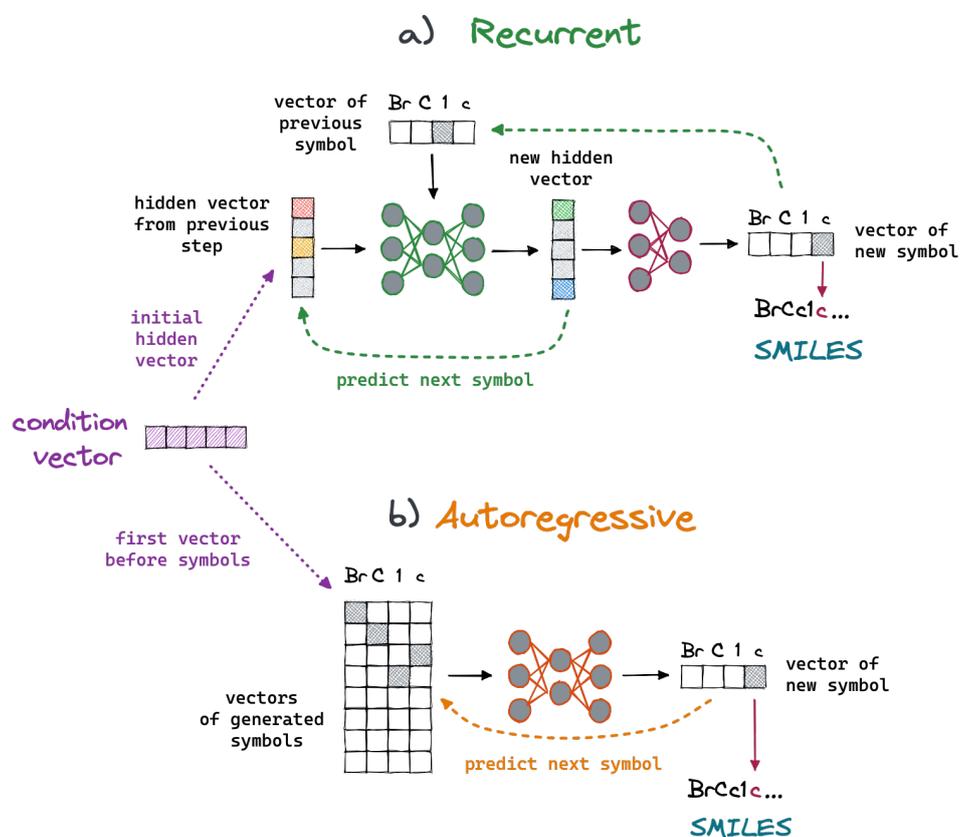


Fig. 3.1 Scheme of conditional iterative generators. In both cases, one symbol is predicted on each iteration until a maximum size is reached or a special “STOP” symbol is generated. a) Recurrent generators that iteratively predict the following symbols based on the hidden vector and the vector of generated symbols obtained in the previous step. In a conditional setup, the condition vector is the initial hidden vector from which the first symbol is predicted. b) Autoregressive generators that re-use vectors of previously generated symbols to predict new ones. In the conditional setup, the first vector is the condition vector, which is later concatenated with vectors of generated symbols

The second category of generative architectures that create vector representations of molecules from the input is based on the autoencoder framework (Figure 3.2). Autoencoders consist of two interconnected networks: an encoder and a decoder. The training process is formulated as an unsupervised learning problem in which the encoder converts a raw input of a molecular structure into a vector representation, and the decoder subsequently reconstructs the original structure. The primary metric used to evaluate the model's performance is the reconstruction rate, which measures the percentage of fully reconstructed molecular structures. A reconstruction rate of 100% indicates that the autoencoder has been trained perfectly. The architectures based on the autoencoding concept can employ one (flow-based models[48]) or two networks (discrete[49] and variational autoencoders[50]).

Discrete autoencoders[49] (Figure 3.2a) are autoencoders without constraints imposed on their latent space. Thus, the trained decoder can be employed as a generative architecture; however, it should be noted that the "random walking" in the learned chemical (latent) space will result in a low number of chemically valid (without valence mistakes) and energetically stable structures. This effect is related to latent space learned by the autoencoder, which is intractable, meaning it cannot be analytically expressed. As no function describes the latent space, there is no way to ensure that structures resulting from random latent vectors correspond to valid and realistic molecular structures. Therefore, for efficient generation in terms of the number of valid molecules, the discrete autoencoders can be combined with search methods or other generative approaches.

Variational Autoencoders (VAEs)[50] (Figure 3.2b) are another subclass of autoencoders. These models can be seen as a regularised version of discrete autoencoders, where intractable posterior distribution is shifted towards known probability distribution, for example, normal distribution. This "shift" is done by calculating the evidence lower bound, which is a difference between reconstruction and regularisation terms. The regularisation term is a distance metric between two probability distributions, for example, Kullback–Leibler divergence [50, 51] or Wasserstein distance [52]. Once trained, the known probability distribution can be utilised by the decoder to generate new structures. Although VAEs are of significant interest, their training can be challenging as the reconstruction and regularisation loss interfere with each other. Most known VAEs for the inverse QSAR design are based on SMILES[14, 53–56] and graph representations[57–59].

The next generation of models based on the autoencoding concept is flow-based models (Figure 3.2c) [48]. These models differ from traditional autoencoders in that they use a single network for encoding and decoding instead of two separate networks. This is achieved through sequences of invertible layers so that the decoder is the inverse operation of the encoder. However, these approaches are limited by choice of operations, as only for some neural

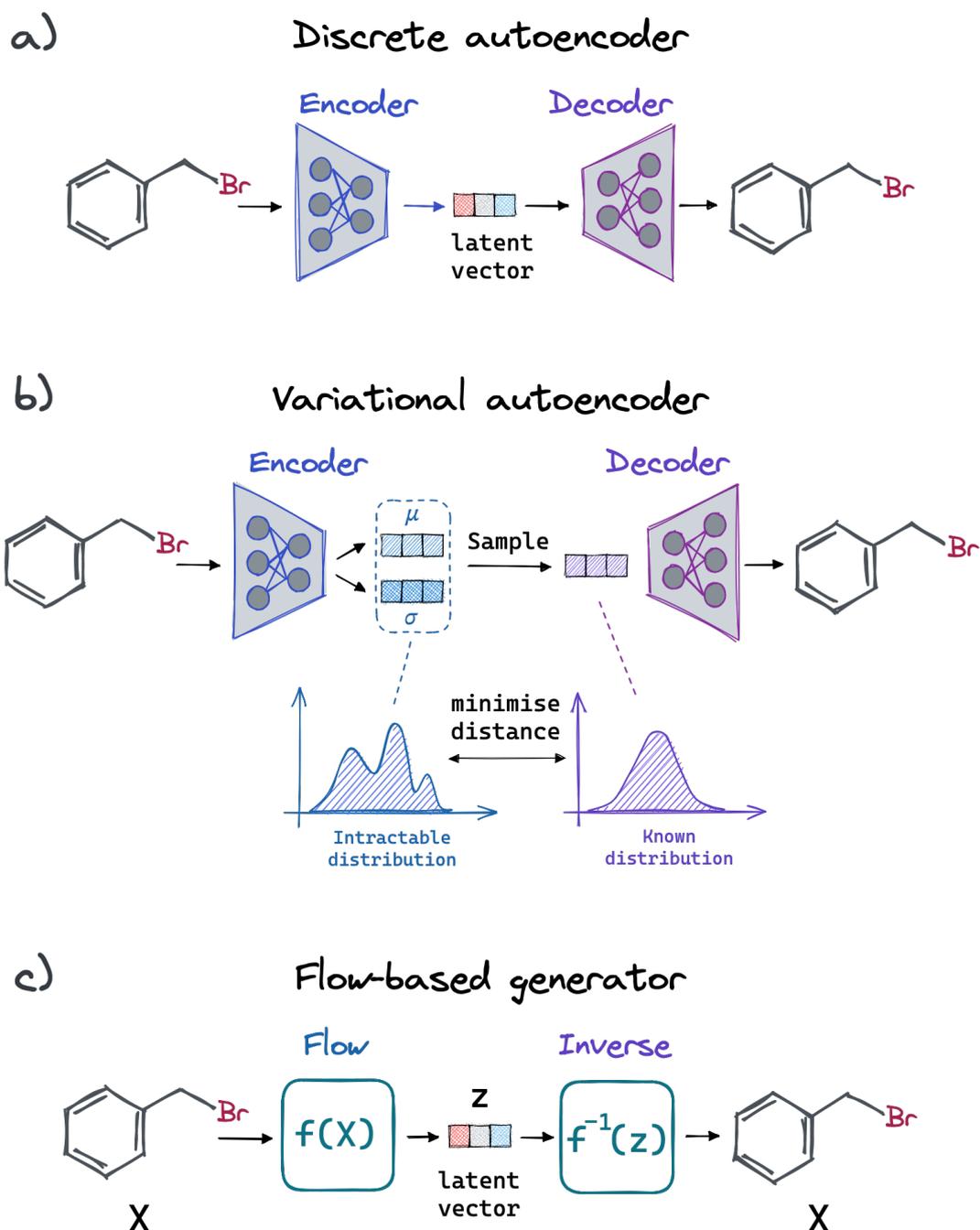


Fig. 3.2 Scheme of different types of neural networks that are based on the autoencoding concept. a) Discrete autoencoder, which has no constraints applied to the latent vectors. b) Variational autoencoder, where the encoder returns vectors of means (μ) and standard deviations (σ), which in this case define normal distribution that should be close to the standard normal distribution with $\mu = 0$ and $\sigma = 1$. These vectors are used to sample (also known as the reparameterisation trick) a latent vector that goes to the decoder. c) Flow-based generator, where “decoder” is the inverse operation of “encoder”.

networks it is possible to compute gradients for the inverse operation. The most dominant representation used for the generation of molecules with normalising flows is graph-based. [60–64]

An exceptional type of architecture that can formulate a molecular vector representation and conditionally generate a molecule from an existing one is the conditional autoencoder. In this architecture, the conditional vectors are added to the latent vector from the encoder as input to the decoder. This approach can also be combined with a variational autoencoder to obtain a generative model from the decoder, generating molecules with specified properties or fragments.

Graph-based autoencoders One promising direction in de novo design and inverse QSAR is the usage of graph-based autoencoders.[26, 65] As previously noted, the use of graph representation allows for the elimination of atom ordering bias and can explicitly incorporate 2D information about a molecule.

Most graph-based autoencoders share the encoding part, which is based on graph neural networks (GNNs). Similar to the convolution layers used in image processing, for each node, graph neural networks aggregate information from neighbouring nodes. The literature distinguishes three classes of graph neural networks: convolutional, attentional, and message-passing. However, the latter category can serve as a generalisation of the first two.[66] For simplicity, we will express GNN through a message passing (MP) scheme[67]. The central idea behind message passing is the exchange of information between the nodes of a graph or, in the context of a molecular graph, the atoms within a molecule. Conceptually, each atom in a molecular graph is represented by an embedding vector that captures all atom’s relevant features. In the message-passing network, this vector is updated via “messages” from neighbouring atoms, which are computed by multiplying the vectors of neighbours by message coefficients. These message coefficients can be either pre-determined based on the atom’s neighbourhood or learned during the training process. (Figure 3.3)

Here we describe the general equation applicable for most graph neural networks with message passing which operate on graphs G , including molecular graphs. In this graph, the properties of node (atom) v are encoded in a vector representation called a node (atom) embedding h_v^0 . Edge properties, or bonds, between atom v and its neighbour w can be represented by the vector e_{vw} . The forward pass of the message-passing neural networks (MPNN) has two phases: a message-passing phase and a readout (or pooling) phase. During the first phase, the atom embeddings h_v^0 pass through T message passing layers, where at each layer t the atoms’ hidden states h_v^t , until final atoms vectors h_v^T are obtained. The message passing is defined in terms of message functions M_t (equation 3.1) and vertex update functions U_t (equation 3.2):

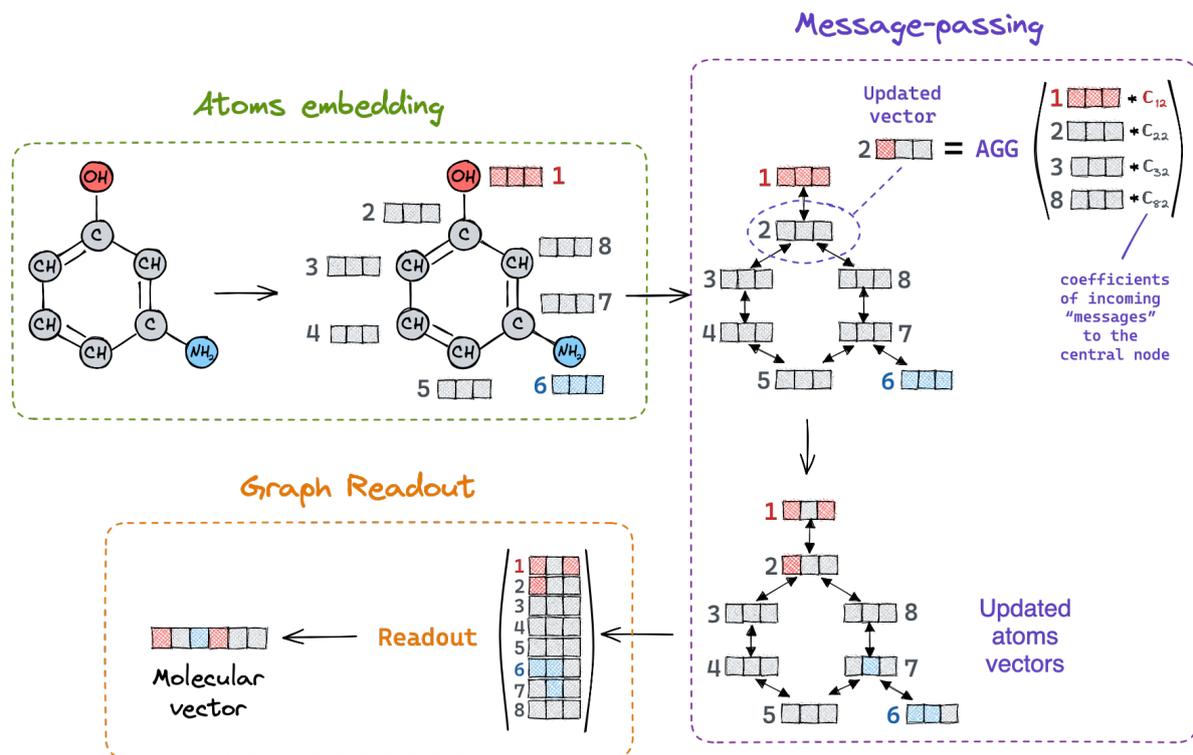


Fig. 3.3 The general scheme of message-passing neural networks for molecular graphs. In the atom embedding step, each atom is assigned the vector representation. Then, in the message-passing step, each atom vector is updated depending on the neighbours around the atom and the bond types. The messages are computed by multiplying the neighbour vectors by the message coefficients, and the "agg" refers to the aggregation function of the messages. The atom vectors are compressed to a molecular vector by a readout operation in the last step. The indices around the atoms are given for simplicity but are not considered at any stage.

$$m_v^{t+1} = \text{agg}_{w \in N(v)}(M_t(h_v^t, h_w^t, e_{vw})) \quad (3.1)$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \quad (3.2)$$

where *agg* is the message aggregation function, so that $w \in N(v)$, in which $N(v)$ denotes the neighbours of atom v in graph G . The aggregation function is usually represented by *sum*, *average*, *max*, *min* or *standard deviation* [68]. Once atoms vectors h_v^t are created, they are passed to the readout function R (can be both learnable or static), which reduces the atoms vectors into a single graph vector:

$$\hat{y} = R(h_v^T \mid v \in G) \quad (3.3)$$

All functions M_t , U_t and R can be parametrised by neural network or static differential functions, for example, summation or average.

One of the most popular GNNs is the Graph Convolution Network (GCN), proposed by Kipf and Welling [69]. The graph convolution can be described by the following equation:

$$H^{(t+1)} = \text{ReLU}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}W^t) \quad (3.4)$$

where $\tilde{A} = I + A$ in which A is the adjacency matrix and I is the identity matrix, \tilde{D} is a degree matrix (diagonal matrix of number of atom's neighbours) of \tilde{A} , W^t is a matrix of weights for the current layer t , and ReLU is the rectifier activation function. In this way, the GCN layer updates the central atom using messages calculated from the hidden states of its neighbours. The message is computed by dividing the hidden state of the neighbour by the product of the central atom's degree and the neighbouring atom's degree. All messages are then aggregated by summation. The original GCN implementation, designed for node classification, does not include a pooling operation; however, in other work using GCNs, summation has been the most popular choice.[70]

One limitation of GCN is that it needs to be adapted to process graphs with different types of edges (multigraphs). One solution proposed in GCN modifications[71, 57] is to split multigraphs on edge-specific graphs that pass through separate graph convolution layers (Figure 4a). Another approach is based on the concatenation of three vectors corresponding to the central atom, its neighbour and the bond between them [72] (Figure 4b). Unfortunately, both solutions create additional computational overhead for molecular multigraphs with different bond types.

Due to the efficiency and flexibility of GNNs, they became essential both in "forward" and "inverse" QSAR. While many other approaches exist, I refer the reader to the reviews [66, 73, 74] that discuss them in detail.

In comparison to encoders, the architectures of decoders are highly varied. A broad classification of decoders can be made by grouping them into two categories: iterative and one-shot decoders. Iterative decoders, which are closely related to iterative generators discussed earlier, create molecules by repetitively reusing one layer and adding structural units of the molecule at each step. These iterative decoders are typically based on recurrent neural networks[42, 70], autoregressive generators[43, 61, 64], and reinforcement learning-based generators[44, 75]. They currently achieve high-quality generation and are widely used; however, they are complex in terms of optimisation and typically include a large number of parameters. In contrast, one-shot generators create atoms and bonds in a single pass through a neural network.[76, 58, 77, 78] These approaches allow for high speed during the training and prediction stages; however, the number of valid structures during generation is usually lower.

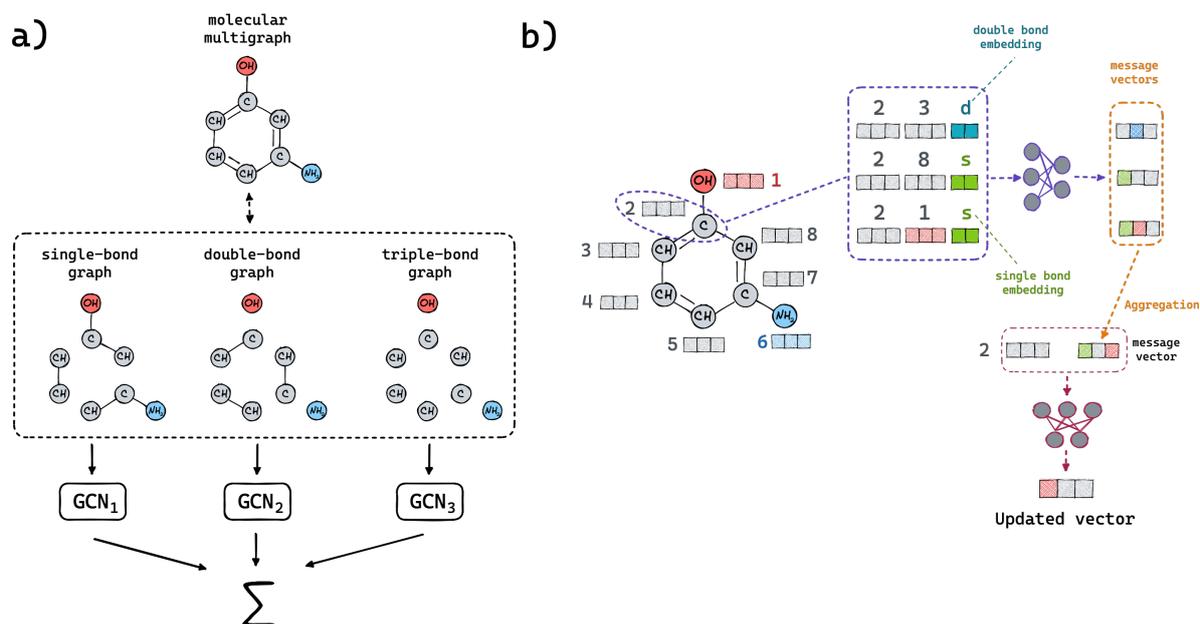


Fig. 3.4 Schematic illustration of different techniques for graph convolution on multigraphs. a) Multigraph is split into several edge-specific graphs (in case molecular graphs are bond-specific). Each graph is passed through separate graph convolution, and resulting atom vectors are summed between edge-specific graphs. b) Pair vectors of a central atom and its neighbour with the bond vector are concatenated and passed through a neural network that predicts “message” vectors. These vectors are then aggregated into one vector, concatenated with the central atom vector and passed through another neural network to obtain an updated vector for the central atom.

3.1.3 Application of neural networks for inverse QSAR

The conditional generators were found to be helpful in the task of inverse QSAR for drug design. For example, Kotsias et al.[79] compared physics-based and fingerprint-based condition vectors for the cRNN model in generating novel ligands for the dopamine receptor D_2 . The authors showed that fingerprint-based cRNN generated structures more similar to known ligands and retained more scaffolds than physchem-based cRNN. Thus, the fingerprint-based cRNN could be useful in the task of analogue generation, whereas the physchem-based cRNN can help in the discovery of novel structures. In another work [80], three-dimensional structural information of the protein binding pocket was used for the conditional generation of molecules. It was shown that a cRNN with pocket information was able to generate molecules with lower docking scores than an unconstrained RNN generator. Li et al. [81] applied cRNN to generate a virtual library of receptor-interacting protein kinase 1 (RIPK1) inhibitors. After a virtual screening of the generated library, the authors found a molecule with a previously unreported scaffold that showed activity *in vitro* and *in vivo*.

Autoencoders were also actively used in inverse QSAR, starting from the pioneering work of Gomez-Bombarelli et al. [14], where authors first applied SMILES-based VAE for the generation of new molecules. After this work, many variants of autoencoders were developed, including conditional variational autoencoders [33, 82]. The application of autoencoders to medicinal chemistry is a rapidly evolving field, with a few studies proving the activity of generated molecules in vitro. One such example is the GENTRL[83] approach, which employs a SMILES-based variational autoencoder to facilitate the rapid discovery of potent inhibitors of discoidin domain receptor 1 (DDR1), a kinase target linked to fibrosis and other diseases. Within the generated structures, one molecule exhibited favourable pharmacokinetics in mice. Another study employed a graph-based variational autoencoder to design active molecules for Chagas parasitic disease[84]. The authors successfully identified a molecule that showed activity in an in vitro study. These studies demonstrate the potential of deep learning architectures based on the autoencoder concept for the efficient and cost-effective development of drugs, which can be applied in large pharmaceutical companies and the treatment of rare and tropical diseases

3.1.4 Challenges of DL for inverse QSAR

The generative neural networks should be flexible and reliable and answer the need of human experts. However, the reliability of neural networks for inverse QSAR is concerning, as these models may exhibit undesirable biases that can negatively impact performance. One such bias identified is the atoms order bias[1], which can affect architectures that utilise string-based representations. As previously mentioned, this bias can affect not only the generative component of these models but also the encoding part in the case of autoencoders. However, there may be other biases that have yet to be identified. Therefore, it is important to increase the interpretability of these architectures to understand them better and address such biases. Another solution is to develop a set of guidelines or “cookbook” to design task-specific generative approaches, as has been done in the field of QSAR[85].

Furthermore, the problem of the computational efficiency of generative architectures is yet to be solved. Thus, with the development of computational libraries for the design of DL architectures, it became easier to develop generative neural networks. However, the question of the scalability of these architectures to handle larger datasets (greater than 10^7 structures) that include more complex compounds (consisting of structures larger than 35 atoms and similar to the natural-product molecules) remains unresolved. The generation of big molecules is particularly challenging for graph-based architectures[86]. Among the various concepts, fragment-based approaches are a promising solution to this challenge.[26]

With the development of generative architectures for inverse QSAR, the problem of the synthetic feasibility of generated structures still needs to be solved [26, 87, 88]. This problem has been addressed by new generative methods trained on reaction data, known as forward synthesis generators [27]. The idea behind these models is to predict possible products given a set of input molecules or reactants. However, the search space over both reactants and reactions is vast and current methods need to indicate whether a synthesis is likely successful. Alternative approaches have therefore been proposed, including the combination of generators with retrosynthetic algorithms. These strategies are discussed in more detail in the following chapter. For a more detailed discussion of forward synthesis generators, I refer the reader to the following works [89, 90].

These examples are among the many other challenges[88] facing scientists to create better generative neural networks.

3.2 Development of novel graph-based architectures

As previously discussed in the literature review, there are still numerous challenges associated with the development of powerful deep learning approaches for de novo drug design. Among the various concepts and architectures, graph-based architectures appear to possess a greater degree of flexibility and potential, as they are able to incorporate both 2D and 3D information of molecules and are not influenced by atoms order bias. However, the generation of graphs remains a significant challenge due to their discrete nature and non-linear structures, which are more complex than images or text. Consequently, previous works have proposed architectures that are sophisticated and computationally expensive[91, 92] or are tested only on small and simple structures[76, 70, 77]. At the same time, only a few of them published the source code[44, 92], making others irreproducible.

The primary objective of this work is to design a deep learning model for inverse QSAR. Among the various candidates of generative architectures, we posit that discrete autoencoders are the most appropriate for graph-based architectures, given that graphs are discrete in nature. Additionally, their combination with generative approaches and combinatorial optimisation techniques can make the architecture modular and flexible, enabling the evaluation of each component separately. Hence, our goal can be formulated as follows:

- The autoencoder model should be computationally efficient;
- It should achieve high reconstruction rates, including for molecules larger than 35 atoms;
- It should be devoid of undesired biases, such as atoms order bias;
- It should be easily interpretable, requiring no complex methods or heuristics to understand what the model has learned.

3.2.1 Hydrogen-count labeled defactorisation graph-based autoencoder

As was mentioned above, autoencoders are a promising architecture for inverse QSAR task. Their ability to learn vector representations of molecules and generate their structures from them makes them flexible and efficient in goal-directed optimisation tasks. However, designing computationally efficient graph-based generative networks poses significant challenges. One such challenge concerns graph convolutions, which do not natively support the multigraphs and require separate convolution layers for each edge type. However, molecules are often represented as multigraphs, with edge types corresponding to the chemical bonds between atoms.

This section introduces a new graph-based autoencoder architecture named Hydrogen-count labelled defactorisation graph-based autoencoder (HyFactor). This autoencoder is based on a representation of molecules known as a Hydrogen-count Labelled Graph (HLG). In this graph, bond types are replaced by the connectivity between atoms and the number of hydrogens attached to heavy atoms. HyFactor was compared with the implementation of its “clone” version, called ReFactor, which is based on the molecular multigraph. Both HyFactor and ReFactor were evaluated on the ZINC 250k benchmark set and the ChEMBL v. 27 datasets in a reconstruction task and on the MOSES dataset in an analogue generation task.

HyFactor: A Novel Open-Source, Graph-Based Architecture for Chemical Structure Generation

Tagir Akhmetshin, Arkadii Lin, Daniyar Mazitov, Yuliana Zabolotna, Evgenii Ziaikin, Timur Madzhidov,* and Alexandre Varnek*



Cite This: *J. Chem. Inf. Model.* 2022, 62, 3524–3534



Read Online

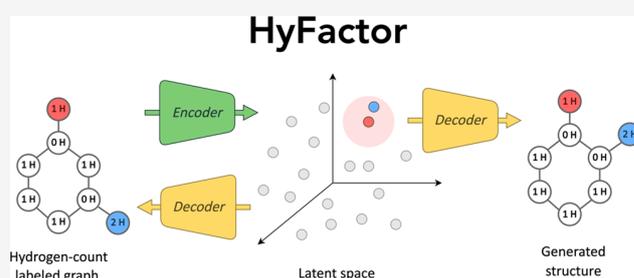
ACCESS |

Metrics & More

Article Recommendations

Supporting Information

ABSTRACT: Graph-based architectures are becoming increasingly popular as a tool for structure generation. Here, we introduce novel open-source architecture HyFactor in which, similar to the InChI linear notation, the number of hydrogens attached to the heavy atoms was considered instead of the bond types. HyFactor was benchmarked on the ZINC 250K, MOSES, and ChEMBL data sets against conventional graph-based architecture ReFactor, representing our implementation of the reported DEFactor architecture in the literature. On average, HyFactor models contain some 20% less fitting parameters than those of ReFactor. The two architectures display similar validity, uniqueness, and reconstruction rates. Compared to the training set compounds, HyFactor generates more similar structures than ReFactor. This could be explained by the fact that the latter generates many open-chain analogues of cyclic structures in the training set. It has been demonstrated that the reconstruction error of heavy molecules can be significantly reduced using the data augmentation technique. The codes of HyFactor and ReFactor as well as all models obtained in this study are publicly available from our GitHub repository: <https://github.com/Laboratoire-de-Chemoinformatique/HyFactor>.



INTRODUCTION

Nowadays, deep neural networks (DNNs) play a significant role in drug and material discovery, being used for property prediction,¹ de novo design,² and computer-aided retrosynthesis.³ One of the most widely used DNN architectures is the autoencoder (AE).⁴ It is able not only to encode chemical structures in their latent representation but also to generate new compounds by decoding sampled latent vectors using a decoder subnetwork.

To generate new molecular structures, the majority of AEs use SMILES strings⁵ as an input, which allows one to employ the power of natural language processing (NLP) techniques. Although SMILES seems suitable for de novo design tasks, the latent representation of text strings may not reflect chemical similarity relationships between considered structures.

Graph-based AE (GAE) architectures⁶ serve as a valuable alternative to the SMILES-based ones. They present a chemical structure as a graph in which nodes and edges encode atoms and chemical bonds, respectively. GAEs have three fundamental advantages over SMILES-based autoencoders. First, no specific order of graph traversal is required, which solves the problem of fixing the canonical ordering of atoms or training on random ordering. Second, a graph object does not need to follow specific grammar rules, such as opening and closing brackets, cycle numbering, etc., which seriously limits the generation ability of neural networks.

Notice that different non-canonical SMILESs describing the same structure may be embedded to different latent vectors. Finally, GAEs always generate graph objects, which, in turn, allows for a meaningful chemical analysis of errors including detection of graph disconnectivity and erroneous valence.

A molecular graph can be represented by an ensemble of atom vectors and bond matrices, which, in turn, can be transformed into its vector representation using graph convolutional networks (GCN).⁷ Once a latent vector is obtained, it can be decoded using either single-shot or iterative decoders. Single-shot decoders generate atoms and bonds in a graph in a single pass.^{6,8} Their training is fast, but simultaneous generation of atom vectors and bond matrices is technically challenging.⁹ In contrast, iterative decoders create atoms and bonds sequentially one by one until a molecule is reconstructed.¹⁰ Iterative decoders can be categorized into two classes. The first class generates atom vectors one by one until vectors of all atoms are sampled. These vectors are then used to extract atom and bond types. For example, in the

Received: June 12, 2022

Published: July 25, 2022



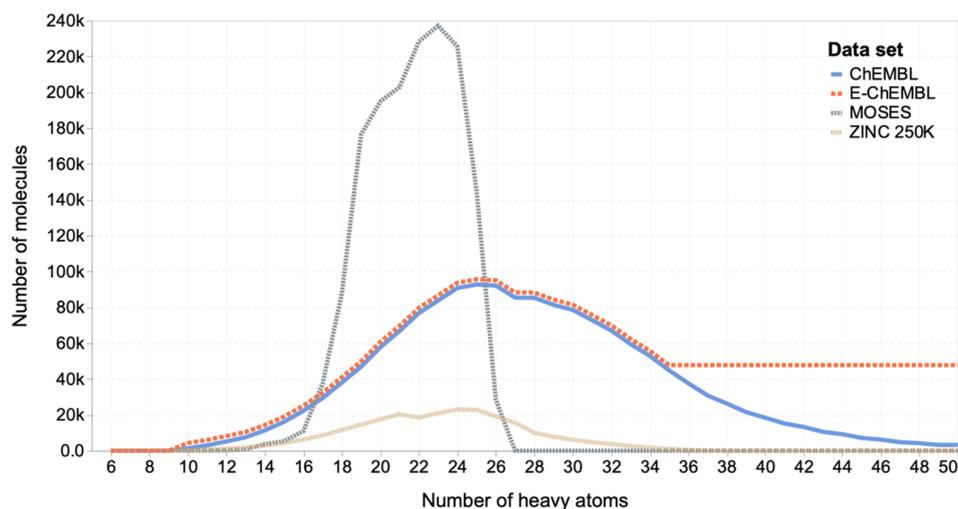


Figure 1. Heavy-atom count distribution in studied data sets.

autoregressive method, the generation of the next vector is based on previously created atom vectors.¹¹ Another popular approach employs a recurrence-based generation where the next atom vector is generated from a hidden (or difference) vector updated at every step.¹² The second class of decoders uses a Markov decision process. In contrast with previous methods, they require the explicit creation of the molecule's substructure at each step of generation. To achieve that, they perform several actions such as "creation of atoms" and "creation of bonds" until the molecule is generated.¹⁰ However, iterative generation requires a much more complex and slow network architecture compared to single-shot autoencoders.⁸ One of the most efficient recurrence-based iterative decoder architectures was recently implemented in the DEFactor tool reported by Assouel et al. in the arXiv e-print.¹¹ The encoder in DEFactor is a multi-layer GCN, whereas the decoder combines the long short-term memory (LSTM)¹³ cell for atomic vector generation with a new adjacency matrix defactorization procedure.

Typically, the GCN employed by the encoder subnetwork in GAEs computes the neighbors' messages within each bond-type specific channel. For this reason, it is necessary to store up to four bond-type-specific adjacency matrices and specific trainable weight matrices. This takes a lot of memory and requires numerous mathematical operations with the corresponding computational graph. A complex iterative process of atom and bond "creation" and related high memory and time costs prevent GAE architectures from becoming widely used.

In this paper, we propose an alternative to conventional structure encoding, which may help reduce both the required GPU memory and the model training time. Instead of considering different bond types, we propose to use the number of hydrogens attached to each heavy atom, similar to the InChI linear notation.¹⁴ Together with an adjacency matrix, this information is sufficient to reproduce molecular structures. Also, it solves the problem of a standard representation of functional groups and aromaticity. In this case, a molecular graph can be represented by three objects: (1) a vector of atoms, (2) a vector of hydrogen counts, and (3) a binary adjacency matrix. The above approach was implemented in a new hydrogen-count labeled graph-based defactorization (HyFactor) GAE architecture. In HyFactor, a DNN is combined with the algorithm needed to convert a

regular molecular graph to a hydrogen-count labeled graph (a graph where a certain number of hydrogens is assigned to each heavy atom) and back.

In order to assess the efficiency of the new architecture compared to conventional GAE, we have decided to compare HyFactor with DEFactor. However, neither DEFactor codes nor neural network hyperparameters (e.g., the number of convolutional layers in the encoder or the batch size, etc.) needed for re-implementation of this tool were provided in the original publication.¹¹ Therefore, we attempted to re-implement and further improve the DEFactor architecture in its advanced version referred here as ReFactor. Here, we describe the HyFactor and ReFactor architectures and report the benchmarking results on ZINC250K, ChEMBL v. 27, and MOSES data sets in the reconstruction and generation tasks.

METHODS

Data and Curation. Three data sources were used: ZINC250K benchmarking data set extracted from the ZINC database⁴ by Kusner et al.,¹⁵ MOSES data set from the MOSES package (v. 1.0),¹⁶ and ChEMBL database (v. 27).¹⁷ All sets were standardized with ChemAxon JChem's utilities¹⁸ using the following procedures: (1) dearomatization, (2) isotope removal, (3) stereo mark removal, (4) explicit hydrogen removal, (5) small fragment removal, (6) solvent removal, (7) salt strip, (8) neutralization of charges, (9) functional group transformation, (10) selection of the canonical tautomer form of the molecule, (11) aromatization, (12) duplicate removal, and (13) dearomatization. The order of atoms in standardized structures was defined by the canonical SMILES string produced by ChemAxon JChem.

Some 1.7K structures were removed from the ZINC250K data set as a result of the cleaning procedure. The cleaned set was split into training, validation (tuning), and test sets as reported by Kusner et al.¹⁵ The validation set was used for early stopping.¹⁹ The test set consisted of 5K predefined molecules, and the remaining structures were randomly split into training and validation sets in a ratio of 9:1. Note that several duplicates were found in ZINC250K (see Table S1) due to the presence of stereoisomers in the data set. This may cause some overestimation of the performance of the earlier reported models.

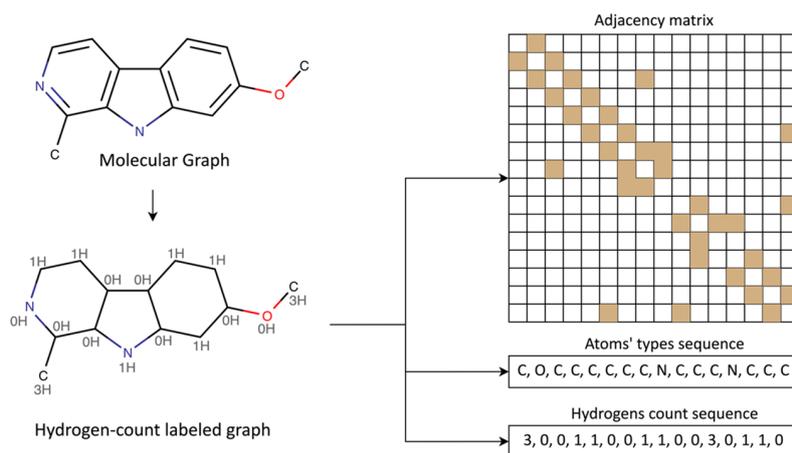


Figure 2. Hydrogen-count labeled graph representation. Here, the molecular graph (hydrogens are hidden) is converted into a graph with no edge features, while the nodes have two features, namely, the type of atoms and number of hydrogens.

The MOSES data set is a benchmarking set for generative models (details are given in the [Supporting Information](#)). It was analyzed with the proposed standardization procedure; however, no mistakes were found. Therefore, it was used as it is. The original “training” set was split into training and validation sets in a 4:1 ratio.

The ChEMBL database was standardized by the same workflow as the ZINC250K data set. The initial database consisted of 1.9M molecules, and it was reduced to 1.6M of standardized structures. The prepared data set was additionally analyzed in terms of the frequency of atom types ([Figure S1](#)). Sixty unique atomic types (including information on atomic charges) were found in ChEMBL, and molecules containing only 15 atomic types (C, O, N, S, F, Cl, N⁺, O⁻, Br, P, I, N⁻, B, Si, and Se) have been retained according to the threshold of 1000. The compounds containing less than 5 and more than 50 heavy atoms have been discarded due to their under-representation. The filtrated ChEMBL data set was then split into a training set (80% of data or 1.3M molecules) and a test set (20% of data or 327K molecules). We did not use a validation set for early stopping since, starting from 100 epochs, the model performance parameters (the loss and reconstruction rate) practically did not vary. The modeling was stopped at 150 epochs.

All chemical structures from each data set have been kekulized in order to avoid a need to introduce an additional aromatic bond type for the ReFactor architecture. The latter would increase the size of the graph-based architecture, slow down the calculations, and decrease the number of valid structures in sampling.

Both the ChEMBL and ZINC250K data sets have a similar distribution of heavy atom counts ([Figure 1](#)). However, the MOSES data set differs from both, and most of the structures lie in the range from 16 to 26 heavy atoms. For some computational tests, the ChEMBL database was enriched by 460K virtual structures bearing more than 35 heavy atoms. The heavy-atom distribution of the enriched ChEMBL (E_ChEMBL) is shown in [Figure 1](#).

Hydrogen-Count Labeled Graph. In the hydrogen-count labeled graph (HLG), only the connections between atoms and the count of hydrogens connected to the atoms are taken into account (see [Figure 2](#)). The formal charge of an atom is used as a vertex label. Such a representation has already been tested in the development of structure–property models with

graph convolution networks.²⁰ The implemented workflow first transformed a molecular graph to HLG and then to three complementary representations: adjacency matrix, atomic types, which include the atom symbol and charge, and hydrogen-count vectors ([Figure 2](#)). The conversion from a molecular graph to HLG and back was performed with the help of the CGRtools toolkit.²¹

Autoencoders Based on Conventional (DEFactor and ReFactor) and Hydrogen-Count (HyFactor) Representations of the Molecular Graph. All three autoencoder architectures used in this work, namely, DEFactor (reported earlier)¹¹ and ReFactor and HyFactor (both developed in this work), are depicted in [Figure 3](#). Their detailed description is given below.

DEFactor Architecture. The encoder in DEFactor uses one-hot embedding to represent atoms in the molecular graph and consists of several layers of edge-specific graph convolution networks^{7,22} that can be expressed as

$$\mathbf{H}^{l+1} = \text{ReLU} \left[\sum_b (\mathbf{D}_b^{-1/2} \mathbf{E}_b \mathbf{D}_b^{-1/2} \mathbf{H}^l \mathbf{W}_b^l) + \mathbf{H}^l \mathbf{W}_{\text{self}}^l \right] \quad (1)$$

where \mathbf{H}^l is the atoms' vectors after the l th graph convolution layer, \mathbf{E}_b is a bond-type specific adjacency matrix, \mathbf{D}_b is the corresponding bond-type specific diagonal degree matrix, \mathbf{W}_b and \mathbf{W}_{self} are trainable matrices of weights for every bond type b and weights for self-channel, respectively, and ReLU is the rectifier activation function. The aggregation of atomic vectors is performed with the help of a long short-term memory (LSTM)¹³ unit followed by a one-layer perceptron (see [Figure 3a](#)), giving a molecular latent vector.

In the decoder, the molecular latent vector is unpacked into a set of atomic embeddings, $\tilde{\mathbf{H}}$, where each h_i is predicted using the LSTM layer. Thus, the entire matrix of atoms' embedding is restored. Next, it passes through two subunits in parallel where the first one is represented by a multilayer perceptron (MLP) with the *softmax* activation function that returns predictions of atom types ($\tilde{\mathbf{A}}$). The second subunit realizes the multichannel defactorization procedure²³ needed to reconstruct the bond matrix according to [eq 2](#)

$$\tilde{\mathbf{E}}_b = \sigma(\tilde{\mathbf{H}} \mathbf{W}_b \tilde{\mathbf{H}}^T + \text{bias}) \quad (2)$$

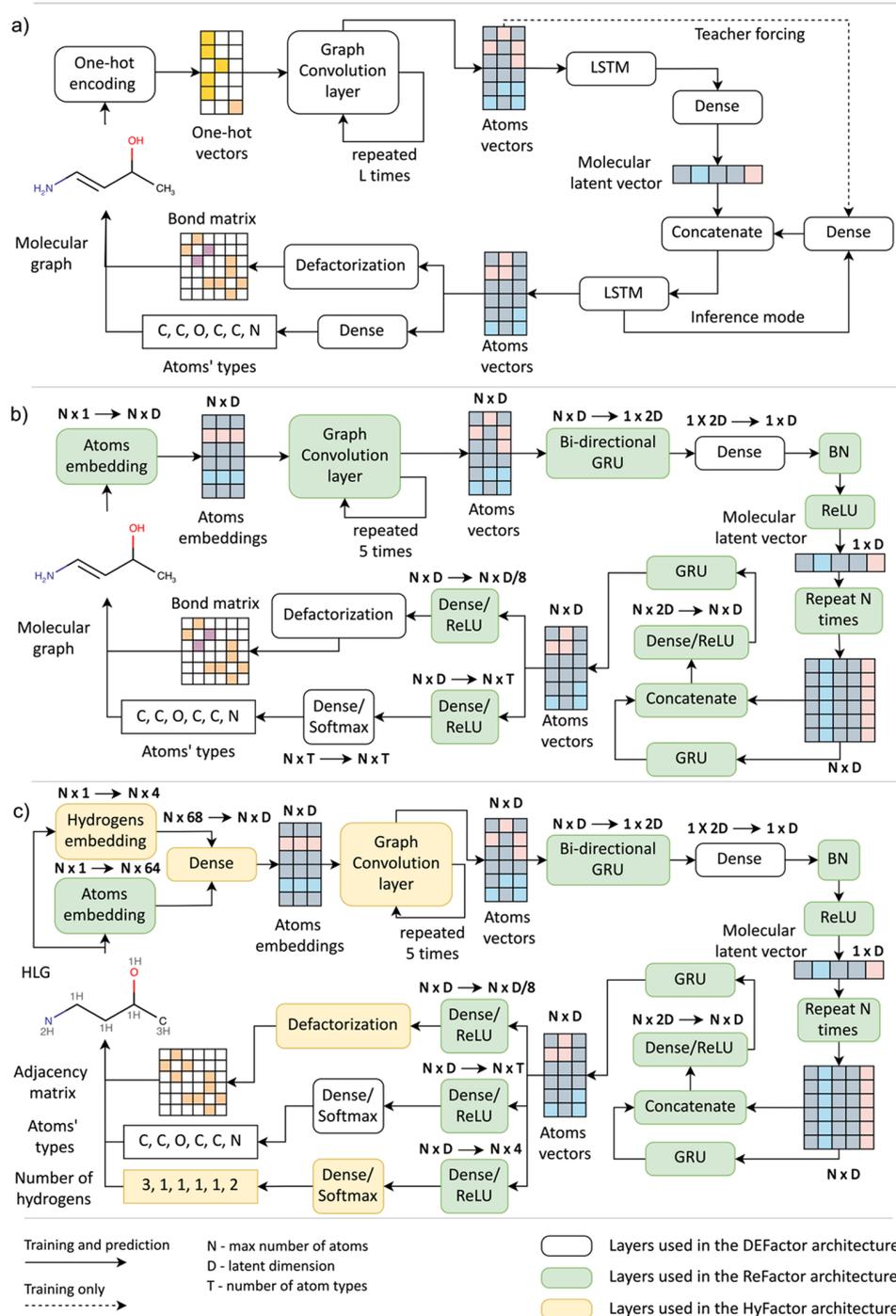


Figure 3. Architectures of different autoencoders considered in this work: (a) DEFactor,¹¹ (b) ReFactor, and (c) HyFactor. BN refers to the batch normalization layer, and GRU refers to the gated recurrent unit. Parameters for each experiment are specified in Table S2.

where \tilde{E}_b is the reconstructed adjacency matrix for a bond type b , \tilde{H} is the matrix of the recovered atoms' embeddings h_i , W_b is a diagonal matrix of weights for the bond type b , and σ is the sigmoid activation function. A certain probability is returned for each bond type between each pair of atoms, and the bond type with the highest probability is selected for the reconstruction. A three-step procedure including teacher forcing was used to speed up the DEFactor training. Within each step, trainable weights of a certain part of the network were frozen and then relaxed at the next step.

The loss function is a sum of categorical cross-entropy for atom predictions (eq 3) and binary cross-entropy for bond predictions (eq 4)

$$L_{\text{atoms}} = -\frac{1}{n} \sum A \times \log(\tilde{A}) \quad (3)$$

$$L_{\text{bonds}} = -\frac{1}{n^2} \sum_b^4 [E_b \times \log(\tilde{E}_b) + (1 - E_b) \times \log(\tilde{E}_b)] \quad (4)$$

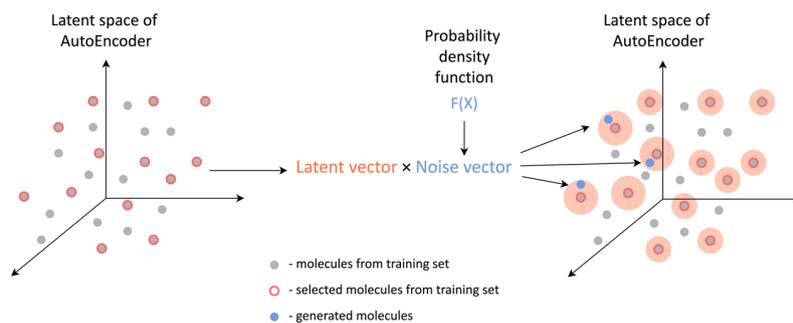


Figure 4. Sampling of new structures from the autoencoder latent space.

where A is the one-hot matrix for atom types, \tilde{A} is the predicted atom-type probability matrix, n is the number of atoms in a molecule, and E_b is the real adjacency matrix for bond type b .

It should be noted that some important information like the number of GCN layers as well as the dimensionality of the atoms' embedding matrix was missed in the original publication by Assouel et al.¹¹ Therefore, we reimplemented the DEFactor architecture with some modifications that improved its performance, at least, for large molecules (see below).

ReFactor Architecture. The ReFactor architecture keeps the main ideas of the DEFactor model. Some parts of DEFactor that were explained well were kept and reimplemented in the Tensorflow package²⁴ v. 2.6. Others were modified or replaced by new layers. Thus, it was decided that token embedding is a more powerful and flexible technique than a simple one-hot embedding. Hence, the latter was replaced by a token embedding layer.

To stabilize the learning process, the GCN from the DEFactor was completed by a layer normalization (LN)²⁵ layer among the atoms' features (parameter "axis = -2") and masking of imaginary atoms (padding):

$$\mathbf{H}^{l+1} = \text{mask} \times \text{ReLU} \left[\sum_b (\mathbf{D}_b^{-1/2} \mathbf{E}_b \mathbf{D}_b^{-1/2} \times \text{LN}(\mathbf{H}^l \mathbf{W}_b^l)) + \text{LN}(\mathbf{H}^l \mathbf{W}_{\text{self}}^l) \right] \quad (5)$$

In each experiment, the number of GCN layers was fixed at five. The dimensionality of the input and output vectors did not change across the layers.

For atomic vector aggregation, LSTM units in DEFactor were replaced with bidirectional gated recurrent units²⁶ (GRU; see Figure 3b). The output of GRUs was passed to the dense, batch normalization (BN), and ReLU activation layers to obtain the molecular latent vector. A teacher-forcing technique applied in the original article was skipped since no predictive performance improvement was detected in our experiments.

In the decoder, the atoms' vectors were generated by two sequentially connected GRU layers and a dense layer in between headed by a RepeatVector layer (see Figure 3b). In such a case, the input molecular latent vector was repeated N times (i.e., according to the maximal molecular graph size) and passed through the first GRU, and intermediate atom vectors were returned. These intermediate vectors were then concatenated with the repeated molecular vectors. They passed first through a perceptron layer with a ReLU activation

function and then through the second GRU layer. Further, the hidden vectors of the second GRU were used as the retrieved atoms' embeddings. For the atom reconstruction, the retrieved atoms' embeddings were passed through two dense layers with the output dimensionality equal to the number of atom types. Activation of the first layer was ReLU, and activation of the second was the *softmax* function. During the bond reconstruction step, the atoms' embeddings were forwarded to a dense layer with the output dimensionality of the latent vector divided by 8 and ReLU activation and then to the defactorization layer.

These and other minor changes allowed us to handle molecules containing up to 50 heavy atoms (see Results and Discussion). Unless specified, the dimensionality of the atom embedding vectors as well as all internal and latent vectors was the same. Parameters for each experiment are specified in Table S2. All layers were taken with standard parameters if not specially mentioned.

HyFactor Architecture. The main changes compared to ReFactor concern the steps of graph convolution and graph reconstruction from the atoms' vectors (see Figure 3c). First, the HLG was transformed to the atoms' (dimension of 64) and hydrogens' (dimension of 4) embeddings. These embeddings were concatenated and passed through the dense layer with the ReLU activation function. The graph convolution network was similar to that in ReFactor

$$\mathbf{H}^{l+1} = \text{mask} \times \text{ReLU} [\mathbf{D}^{-1/2} \mathbf{E} \mathbf{D}^{-1/2} \times \text{LN}(\mathbf{H}^l \mathbf{W}_{\text{neighbors}}^l) + \text{LN}(\mathbf{H}^l \mathbf{W}_{\text{self}}^l)] \quad (6)$$

where \mathbf{E} and \mathbf{D} are adjacency and degree matrices of HLG, and the other designations are the same as in eqs 1 and 2. Here, the number of the training parameters is twice less than that for ReFactor GCN.

At the graph reconstruction stage, the number of hydrogens and atom types were predicted similar to using dense layers with the *softmax* activation function. The maximal number of hydrogens attached to an atom was equal to 3. The adjacency matrix was reconstructed using the defactorization procedure (eq 2), ignoring bond types; only one trainable diagonal \mathbf{W}_b matrix was used.

The initialization of weights for each layer was performed with HE normal initialization.²⁷ Training of the architecture was performed using the AdaBelief optimizer²⁸ with default parameters. Exponential learning decay was applied in order to maintain the training stability.

Sampling Procedure. New molecular structures were generated by sampling latent vectors in the vicinity of the known molecules.²⁹ Here, the latent vectors of selected

molecules from the training set were used as seeds (Figure 4). During generation, these latent vectors were multiplied by noise vectors composed of random numbers generated from a probability density function of a log-normal distribution followed by their decoding to a molecular graph.

In order to effectively explore the chemical space around the given seed, the “mean” parameter was set to 0, whereas the “standard deviation” was systematically varied. Generally, the probability of generating more dissimilar structures increases with the “standard deviation” value.

RESULTS AND DISCUSSION

Reconstruction Rate of Different Graph Autoencoders. The performance of the autoencoders’ model is measured by the reconstruction rate representing a percentage of correctly reconstructed structures in the considered data set. Reconstruction rate values for several SMILES-based and graph-based autoencoders on the ZINC 250K set are reported in Table 1.

Table 1. ZINC 250K Reconstruction Benchmarking Results^a

architecture name	molecular representation	reconstruction rate (%)		
		training set	validation set	test set
TSGCD ⁹	molecular graph			90.5
DEFactor ¹¹	molecular graph			89.8
JTVAE ¹²	molecular graph ^b			76.7
rebalanced VAE ³⁰	SMILES			92.7
all SMILES ³¹	SMILES			87.6
SDVAE ³²	SMILES			76.2
ReFactor	molecular graph	99.5	90.8	90.7
ReFactor ^c	molecular graph	99.7	90.0	90.0
HyFactor	HLG	99.3	89.3	89.0
HyFactor ^c	HLG	99.2	89.8	88.4

^aReconstruction rate values for other architectures are taken from the original publications. ^bJTVAE uses hierarchical fragments instead of atoms to reconstruct the molecule. ^cAdditional standardization and duplicate data removal have been applied to the initial ZINC 250K data set.¹⁵

One can see that the performances of graph-based and SMILES-based architectures are similar. The leading graph-based architecture (TSGCD) has a reconstruction rate that is only 2% lower than the best SMILES-based autoencoder (rebalanced VAE). The DEFactor architecture also demonstrates high performance, which is only 1% lower than the leading graph-based autoencoder. Notice that the data standardization issue was not sufficiently discussed in the publications on all the architectures mentioned in Table 2. Therefore, results for the ReFactor and HyFactor architectures are given for both initial¹⁵ (non-standardized) and standardized data sets. In addition, results for training and validation sets for model overfitting analysis are also reported.

The reconstruction rate for ReFactor obtained on the initial data set is slightly better than that of the standardized data set. This fact supports our assumption that the performance of autoencoder may be overestimated since training and test sets partially overlap in the non-standardized data. As it follows from Table 1, ReFactor outperforms DEFactor on the initial data set.

Table 2. MOSES Metrics^a Calculated for the 100 K Structures Generated with Different Standard Deviations (STDs)^d

STD	validity		uniqueness		novelty	
	original ^b	modified ^c	original ^b	modified ^c	original ^b	modified ^c
	ReFactor					
0.2	0.997	0.996	0.656	0.105	0.074	0.042
0.4	0.921	0.886	0.748	0.265	0.634	0.578
0.6	0.698	0.503	0.948	0.730	0.875	0.814
0.8	0.540	0.149	0.998	0.971	0.978	0.855
1	0.516	0.022	0.961	1.000	0.999	0.983
	HyFactor					
0.2	1.000	0.9880	0.661	0.193	0.117	0.006
0.4	0.989	0.7290	0.795	0.267	0.729	0.129
0.6	0.940	0.1820	0.980	0.634	0.923	0.288
0.8	0.886	0.0120	1.000	0.966	0.993	0.491
1	0.822	0.0003	0.981	1.000	1.000	0.912

^aValidity is a fraction of valid molecules compared to generated ones. Uniqueness (or Unique 10K) is defined as a fraction of the first 10K unique molecules among the valid ones. Novelty is a fraction of the novel generated molecules among unique ones. ^bMetrics calculated by the MOSES package. ^cMetrics calculate by CGRtools after removal of structures with valence errors and disconnected graphs. ^dResults for STD = 0.4 selected for further tests are shown in italics.

The HyFactor architecture has almost the same reconstruction rate as ReFactor but has a smaller number of neural network parameters (10M compared to 12M in ReFactor). The training time for HyFactor was 2 h and 45 min. (with 5 GB of GPU RAM allocated), while for ReFactor, it was 4 h and 10 min. (with 5.5 GB of GPU RAM allocated).

Structure Generation Performance of Graph-Based Autoencoders. The efficiency of the proposed graph-based autoencoders to generate valid chemical structures has been investigated on the MOSES data set using the metrics included in the MOSES package.¹⁶ Both ReFactor and HyFactor architectures were trained on 80% of the MOSES training set and achieved more than 99% of the reconstruction rate on the remaining 20% of the data used as a validation set. Then, 10 K compounds were randomly selected as seeds for sampling new structures. For each seed compound, 10 virtual structures were generated, so 100K structures were obtained. The generation was based on a log-normal distribution with a mean equal to 0 and a standard deviation ranging from 0.2 to 1.0 with a step of 0.2. Since the original DEFactor source code was not available from the original publication,¹¹ the ReFactor architecture was benchmarked instead.

A preliminary analysis of the results revealed that the MOSES package did not correctly handle typical problems frequently occurring during the structure generation: disconnected molecular graphs were not considered erroneous, and some valence errors were ignored. Therefore, an additional examination of generated structures was performed using the CGRtools package.²¹ The results of the analysis of STD influence on validity, uniqueness, and novelty metrics of the generated structures are given in Table 2.

One can see that the increase in standard deviation leads, on one hand, to the rise of the percentage of new molecules and to the decrease of validity on the other hand. We notice the difference between original and modified workflows for validity checks caused mainly by the trend to return disconnected graphs by ReFactor and HyFactor at high STD, which

Table 3. Results of MOSES Benchmarking for Different Autoencoders; the Similarity Metrics FCD, SNN, and Scaf^a Relate a Set of Generated Structures with the MOSES Test Set^d

model	validity	uniqueness	novelty	FCD	SNN	Scaf	IntDiv
AAE	0.937	0.997	0.793	0.556	0.608	0.902	0.856
VAE	0.977	0.998	0.695	0.099	0.626	0.939	0.856
JTVAE	1.000	1.000	0.914	0.395	0.548	0.896	0.855
ReFactor ^b	0.886	0.265 ^c	0.578	1.743	0.547	0.847	0.868
HyFactor ^b	0.729	0.267 ^c	0.129	0.365	0.614	0.862	0.860

^aThe MOSES benchmarking parameters:¹⁶ Scaf is a cosine similarity based on the occurrence of Bemis–Murcko scaffolds in the compared sets. SNN is an average Tanimoto similarity calculated with Morgan fingerprints between a molecule from the generated set and its nearest neighbor from the test set. FCD is a Wasserstein-2 distance computed on vectors produced by the last layer of the ChemNet neural network between the generated and test sets. IntDiv measures the dissimilarity of structures in the generated set calculated with Morgan fingerprints. See the [Supporting Information](#) for details. ^bSampling for STD = 0.4. All metrics were calculated after the removal of structures with valence errors and disconnected structures using the CGRtools-based workflow. ^cUniqueness was calculated on the entire generated data set after filtration by validity. ^dThe performances of AAE, VAE (SMILES-based), and JTVAE (graph-based) architectures were taken from the article by Polykovskiy et al.¹⁶

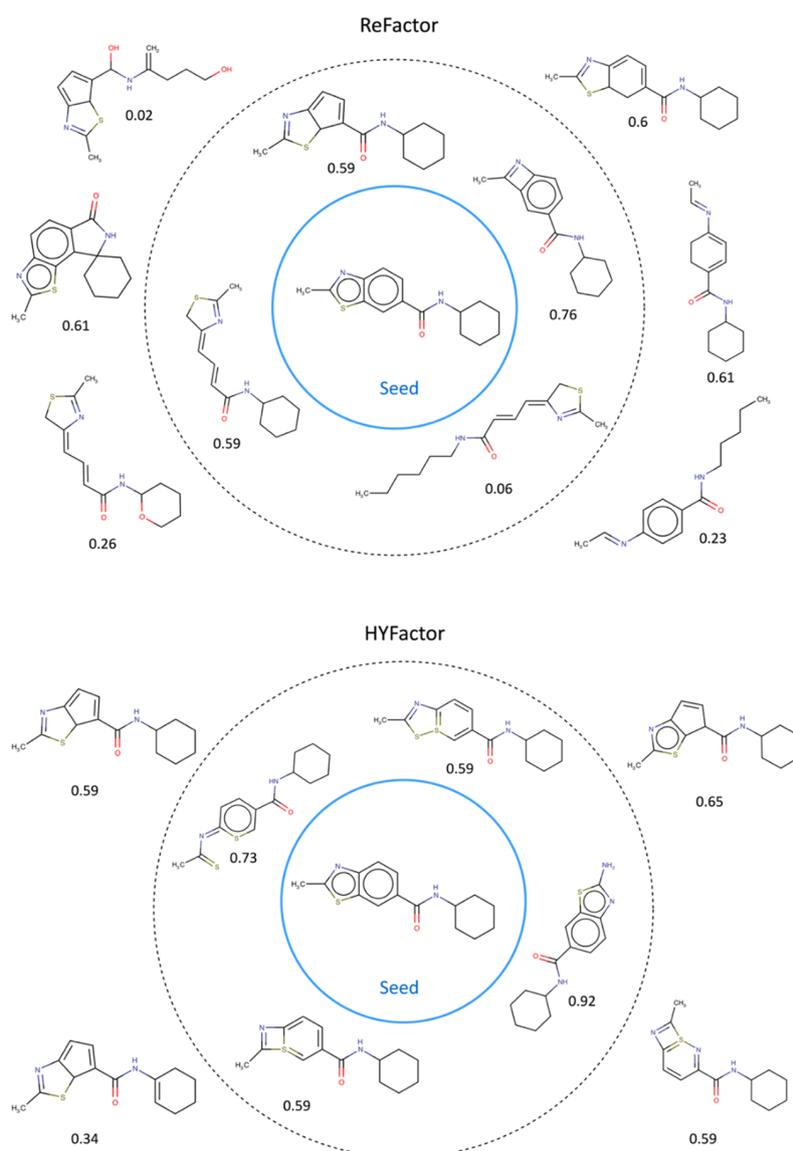


Figure 5. Example of structures generated with ReFactor and HyFactor trained on the MOSES set. Molecules generated with a standard deviation of <0.6 (>0.6) lie within (outside) the dashed circle. All molecules were aromatized with the ChemAxon toolbox. Each number corresponds to a pairwise Tanimoto similarity of a given generated structure with respect to the seed assessed with the atom-centered ISIDA fragments³⁵ involving sequences of atoms and bonds of sizes from 2 to 4 atoms with different labeling of cyclic and acyclic bonds.

Table 4. Training Results on the ChEMBL Data Set

architecture	batch	vector length	number of training parameters (M)		time per epoch ^a (min)	GPU memory ^a (MB)	reconstruction rate (%)	
			encoder	decoder			training set	test set
ReFactor	1024	1024	35.7	14.8	~24.3	~ 22,845	99.8	95.2
HyFactor	1024	1024	25.3	15.1	~16.5	~ 16,755	99.7	95.0

^aMeasured in a “mixed precision” mode, which is available in the TensorFlow package. In this mode, a 16-bit floating-point type is used where it is possible; otherwise, a 32-bit floating-point type is applied.

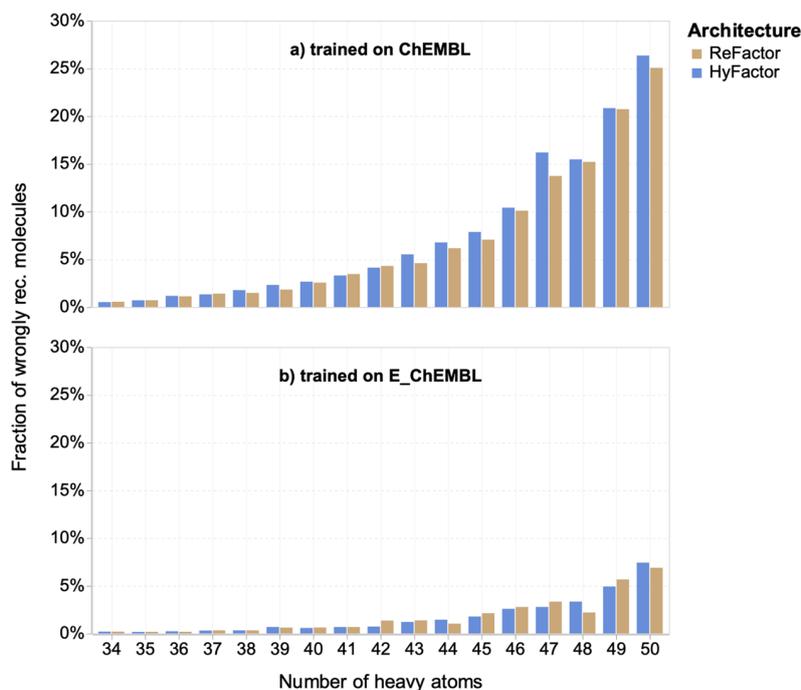


Figure 6. Distributions of errors in the ChEMBL validation set as a function of molecular size for ReFactor (gold) and HyFactor (blue) architectures trained on the (a) ChEMBL data set and (b) E_ChEMBL data set.

nonetheless represent correct structures. Valid structures generated with a high standard deviation parameter ($STD \geq 0.6$) are characterized by high novelty and uniqueness. The validity of structures generated with HyFactor sharply drops for high STD values, which is not the case of ReFactor. Thus, one can suggest that the HyFactor latent space is more discontinuous than that of ReFactor. We notice that the latent space discontinuity is a quite common phenomenon for regular autoencoders without special regularization on latent space, like in variational autoencoders³³ or application of latent vectors for additional task solving.³⁴

However, both architectures achieved high reconstruction rates and reasonable values of the validity, uniqueness, and novelty parameters at $STD = 0.4$ (Table 3). To compare with other autoencoders, we used this standard deviation. The results of MOSES benchmarking are given in Table 3, which includes the most important metrics assessing the generation ability of the proposed architectures. The results for all other metrics available in the MOSES package are given in the Supporting Information (see Table S3).

Since graphs are discrete objects, they occupy particular positions in the continuous autoencoder latent space. The latent vectors corresponding to invalid molecular graphs (e.g., disconnected structures or structures with valence mistakes) in the latent space are located between positions of valid molecules. While the proposed sampling method allows

systematic exploration of the chemical space, the validity of the generated structures can hardly be controlled. Uniqueness and novelty are lower than for other architectures as we generate molecules in the vicinity of seed molecules.

Table 3 shows that the scaffold similarity (Scaf) of structures generated with HyFactor and ReFactor is lower than that for earlier reported autoencoders. Therefore, one can conclude that new architectures are more potent for scaffold hopping, which is crucial in de novo design. Moreover, HyFactor and ReFactor generate molecules with rather high internal diversity (IntDiv), characterizing a broader variety of generated structures. Compared to all benchmarked autoencoders, generated samples from ReFactor have the biggest Fréchet Chemnet distance (FCD) and the smallest similarity to a nearest neighbor (SNN). This demonstrates that the ReFactor's structures are more diverse with respect to the training set than those generated with any other architecture.

Another important issue concerns the neighborhood behavior analysis in the latent space. Thus, it is expected that for small STDs, the distances between generated latent vectors and the seed are rather small, which means that the generated chemical structures are similar to the seed structure. In order to check this hypothesis, for each of the 10K selected seed structures, several molecules were generated with HyFactor and ReFactor for standard deviations varying from 0.3 to 1.0 with a step of 0.02. At each step, 10 molecules were

generated followed by the validity and uniqueness check. These simulations resulted, on average, in 10 and 40 generated structures per seed for HyFactor and ReFactor, respectively.

Generally, the neighborhood behavior is respected, that is, most of the structures generated with a small standard deviation ($STD < 0.6$) are more similar to the seed than those generated with a large STD (Figure 5). However, even with small STDs, ReFactor may occasionally generate very dissimilar structures corresponding to open-chain analogues of the cyclic seed structure. Such dissimilarity explains high FCD and low SNN scores observed for ReFactor compared to HyFactor and other considered architectures.

Data Augmentation: Case Study of the ChEMBL Database. The Achilles' heel of graph-based autoencoders is the reconstruction of molecules with a large number of atoms. Indeed, the probability of error in predicting the atom or bond type increases with molecular size. In this section, we demonstrate how data augmentation may help solve this problem. The experiments with ReFactor and HyFactor were performed on the ChEMBL database containing molecules bearing up to 50 heavy atoms. The results of training are given in Table 4 and specifications of training parameters are reported in the Supporting Information (Table S2).

According to Table 4, HyFactor uses 20% fewer training parameters than ReFactor in order to achieve a similar reconstruction rate, and thus, its training is 33% faster than ReFactor. Although the overall reconstruction rate of both networks is high enough, the reconstruction error sharply increases for molecules containing more than 35 atoms and it reaches almost 30% for ReFactor and HyFactor for molecules containing 50 atoms (Figure 6a). The latter can be explained by the small number of heavy molecules present in the training set (see Figure 1).

In order to confirm these suggestions, 460K virtual structures containing >35 atoms were generated using the Synt-On tool (former SynthI)³⁶ and then added to the ChEMBL set. These structures were generated using a special protocol insisting their similarity to related heavy ChEMBL molecules and synthetic feasibility; see details in the Supporting Information. The enriched ChEMBL set (E_ChEMBL) was then divided into training and test sets in the ratio 4:1 containing 1.6M and 420K structures, respectively. The distribution of molecular size in the obtained data set is given in Figure 1. Both architectures trained on the E_ChEMBL training set achieved reconstruction rates of >95% measured on the E_ChEMBL test set. The enrichment of the initial data set significantly reduced the reconstruction error of heavy molecules: for the molecules containing 50 atoms from ChEMBL, this value drops from some 25% for the models trained on ChEMBL (Figure 6a) to around 7% for the models trained on E_ChEMBL.

CONCLUSIONS

Neural network architecture HyFactor, which uses a hydrogen-count labeled graph as a chemical structure representation, has been developed. In this graph, implicit hydrogen atom counts are used instead of bond types, like in the InChI linear notation. Such a representation allows avoiding most of the problems of molecule standardization (aromatization, functional group standardization, and representation of Kekule structures) that make HyFactor insensible to molecule representation specifics.

For the sake of comparison, we have implemented the ReFactor architecture based on a classical molecular graph. It represents an updated and optimized version of the previously published DEFactor architecture, which uses defactorization of the adjacency matrix for graph generation. Since the latter proceeds in a single-shot manner without autoregressive bond and atom addition, the architecture is time and resource-economic.

Both ReFactor and HyFactor networks demonstrated high (>90%) reconstruction rates both in ZINC250K and ChEMBL datasets, which is similar to (or even better than) earlier reported graph-based or SMILES-based approaches. While the HyFactor architecture achieved the same reconstruction rate as ReFactor, it was also more effective in terms of the network parameters and training time.

Analysis of the dependency of reconstruction rates of HyFactor and ReFactor on molecular size revealed that, for molecules containing more than 35 atoms, the fraction of errors starts to grow, achieving almost 25% for molecules with 50 atoms. We hypothesized that this was related to the underrepresentation of such large molecules in the training dataset. Significant loss on the error rate on large molecules to 7% for the models trained on the dataset enriched by rationally generated virtual molecules supported this hypothesis. We believe that such a problem can be common for other graph- or SMILES-based autoencoders and encourage adding corresponding tests in generative chemistry benchmarking tools.

Both HyFactor and ReFactor can be used for efficient generation of new chemical structures. Since no special regularization of the latent space was used, vectors corresponding to new structures have been sampled around selected training set molecules. Novelty and uniqueness of generated structures increase as a function of the noise level, but the structures' validity drops in the same direction. In the MOSES benchmark, the structures generated by proposed architectures are characterized by greater diversity and scaffold novelty compared to those generated with the help of some other state-of-the-art approaches. This makes the proposed approaches especially promising for constrained molecule generation.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at <https://pubs.acs.org/doi/10.1021/acs.jcim.2c00744>.

Detailed information about data sets used and some complementary modeling results (PDF)

AUTHOR INFORMATION

Corresponding Authors

Timur Madzhidov – Laboratory of Chemoinformatics and Molecular Modeling, Butlerov Institute of Chemistry, Kazan Federal University, 420008 Kazan, Russia; orcid.org/0000-0002-3834-6985; Email: Timur.Madzhidov@kpfu.ru

Alexandre Varnek – Laboratory of Chemoinformatics, UMR 7140 CNRS, University of Strasbourg, 67081 Strasbourg, France; orcid.org/0000-0003-1886-925X; Email: varnek@unistra.fr

Authors

Tagir Akhmetshin – Laboratory of Chemoinformatics, UMR 7140 CNRS, University of Strasbourg, 67081 Strasbourg, France; orcid.org/0000-0002-2549-6431

Arkadii Lin – Laboratory of Chemoinformatics, UMR 7140 CNRS, University of Strasbourg, 67081 Strasbourg, France

Daniyar Mazitov – Laboratory of Chemoinformatics and Molecular Modeling, Butlerov Institute of Chemistry, Kazan Federal University, 420008 Kazan, Russia

Yuliana Zabolotna – Laboratory of Chemoinformatics, UMR 7140 CNRS, University of Strasbourg, 67081 Strasbourg, France

Evgenii Ziaikin – Laboratory of Chemoinformatics and Molecular Modeling, Butlerov Institute of Chemistry, Kazan Federal University, 420008 Kazan, Russia; orcid.org/0000-0001-6316-1301

Complete contact information is available at:
<https://pubs.acs.org/10.1021/acs.jcim.2c00744>

Notes

The authors declare no competing financial interest. The source code of HyFactor and all models obtained in this study are publicly available from our GitHub repository: <https://github.com/Laboratoire-de-Chemoinformatique/HyFactor>.

ACKNOWLEDGMENTS

T.A. thanks the Region Grand Est. for the Ph.D. fellowship.

REFERENCES

- (1) Varnek, A.; Baskin, I. Machine Learning Methods for Property Prediction in Chemoinformatics: Quo Vadis? *J. Chem. Inf. Model.* **2012**, *52*, 1413–1437.
- (2) Button, A.; Merk, D.; Hiss, J. A.; Schneider, G. Automated de Novo Molecular Design by Hybrid Machine Intelligence and Rule-Driven Chemical Synthesis. *Nat. Mach. Intell.* **2019**, *1*, 307–315.
- (3) Segler, M. H. S.; Preuss, M.; Waller, M. P. Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI. *Nature* **2018**, *555*, 604–610.
- (4) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **2018**, *4*, 268–276.
- (5) Baskin, I. I. The Power of Deep Learning to Ligand-Based Novel Drug Discovery. *Expert Opin. Drug Discovery* **2020**, *15*, 755–764.
- (6) Simonovsky, M.; Komodakis, N. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; 2018; Vol. 11139 LNCS, pp. 412–422. DOI: [10.1007/978-3-030-01418-6_41](https://doi.org/10.1007/978-3-030-01418-6_41).
- (7) Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *5th Int. Conf. Learn. Represent., ICLR 2017 - Conf. Track Proc.* 2017, 1–14. DOI: [10.48550/arXiv.1609.02907](https://doi.org/10.48550/arXiv.1609.02907).
- (8) Samanta, B.; De, A.; Jana, G.; Gómez, V.; Chattaraj, P. K.; Ganguly, N.; Gomez-Rodriguez, M. NeVAE: A Deep Generative Model for Molecular Graphs. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 1110–1117.
- (9) Bresson, X.; Laurent, T. A Two-Step Graph Convolutional Decoder for Molecule Generation. *arXiv* **2019**, 1906, No. 03412.
- (10) Zhou, Z.; Kearnes, S.; Li, L.; Zare, R. N.; Riley, P. Optimization of Molecules via Deep Reinforcement Learning. *Sci. Rep.* **2019**, *9*, 10752.
- (11) Assouel, R.; Ahmed, M.; Segler, M. H.; Saffari, A.; Bengio, Y. DEFactor: Differentiable Edge Factorization-Based Probabilistic Graph Generation. *arXiv* **2018**, 1–14.
- (12) Jin, W.; Barzilay, R.; Jaakkola, T. Chapter 11. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *RSC Drug Discovery Series*; 2020; pp. 228–249. DOI: [10.1039/9781788016841-00228](https://doi.org/10.1039/9781788016841-00228).
- (13) Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780.
- (14) Heller, S. R.; McNaught, A.; Pletnev, I.; Stein, S.; Tchekhovskoi, D. InChI, the IUPAC International Chemical Identifier. *J. Cheminf.* **2015**, *7*, 1–34.
- (15) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. *34th Int. Conf. Mach. Learn., ICML 2017* 2017, *4*, 3072–3084.
- (16) Polykovskiy, D.; Zhebrak, A.; Sanchez-Lengeling, B.; Golovanov, S.; Tatanov, O.; Belyaev, S.; Kurbanov, R.; Artamonov, A.; Aladinskiy, V.; Veselov, M.; Kadurin, A.; Johansson, S.; Chen, H.; Nikolenko, S.; Aspuru-Guzik, A.; Zhavoronkov, A. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Front. Pharmacol.* **2020**, *11*, 1–10.
- (17) Gaulton, A.; Bellis, L. J.; Bento, A. P.; Chambers, J.; Davies, M.; Hersey, A.; Light, Y.; McGlinchey, S.; Michalovich, D.; Al-Lazikani, B.; Overington, J. P. ChEMBL: A Large-Scale Bioactivity Database for Drug Discovery. *Nucleic Acids Res.* **2012**, *40*, 1100–1107.
- (18) ChemAxon Ltd. Budapest, Hungary. <https://chemaxon.com/> (accessed 2022-04-19).
- (19) Bourlard, N.; Morgan, H. Generalization and Parameter Estimation in Feedforward Nets: Some Experiments. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*; Touretzky, D., Ed.; Morgan-Kaufmann, 1989; pp. 630–637.
- (20) Pocha, A.; Danel, T.; Podlowska, S.; Tabor, J.; Maziarka, L. Comparison of Atom Representations in Graph Neural Networks for Molecular Property Prediction. In *2021 International Joint Conference on Neural Networks (IJCNN)*; IEEE, 2021; pp. 1–8. DOI: [10.1109/IJCNN52387.2021.9533698](https://doi.org/10.1109/IJCNN52387.2021.9533698).
- (21) Nugmanov, R. I.; Mukhametgaleev, R. N.; Akhmetshin, T.; Gimadiev, T. R.; Afonina, V. A.; Madzhidov, T. I.; Varnek, A. CGRtools: Python Library for Molecule, Reaction, and Condensed Graph of Reaction Processing. *J. Chem. Inf. Model.* **2019**, *59*, 2516–2521.
- (22) Simonovsky, M.; Komodakis, N. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. *Proceedings - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017* 2017, 2017-Janua, 29–38. DOI: [10.1109/CVPR.2017.11](https://doi.org/10.1109/CVPR.2017.11).
- (23) Zitnik, M.; Agrawal, M.; Leskovec, J. Modeling Polypharmacy Side Effects with Graph Convolutional Networks. *Bioinformatics* **2018**, *34*, i457–i466.
- (24) GoogleResearch. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. DOI: [10.5281/zenodo.5949169](https://doi.org/10.5281/zenodo.5949169).
- (25) Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer Normalization. *arXiv* **2016**, DOI: [10.48550/arXiv.1607.06450](https://doi.org/10.48550/arXiv.1607.06450).
- (26) Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP 2014–2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2014; pp. 1724–1734. DOI: [10.3115/v1/d14-1179](https://doi.org/10.3115/v1/d14-1179).
- (27) He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In *Proceedings of the IEEE International Conference on Computer Vision; IEEE, 2015; Vol. 2015 Inter*, pp. 1026–1034. DOI: [10.1109/ICCV.2015.123](https://doi.org/10.1109/ICCV.2015.123).
- (28) Zhuang, J.; Tang, T.; Ding, Y.; Tatikonda, S.; Dvornek, N.; Papademetris, X.; Duncan, J. S. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. *Adv. Neural Inf. Process. Syst.* **2020**, 2020, 1–29.

(29) Sattarov, B.; Baskin, I. I.; Horvath, D.; Marcou, G.; Bjerrum, E. J.; Varnek, A. De Novo Molecular Design by Combining Deep Autoencoder Recurrent Neural Networks with Generative Topographic Mapping. *J. Chem. Inf. Model.* **2019**, *59*, 1182–1196.

(30) Yan, C.; Wang, S.; Yang, J.; Xu, T.; Huang, J. Re-Balancing Variational Autoencoder Loss for Molecule Sequence Generation. *Proc. 11th ACM Int. Conf. Bioinformatics, Comput. Biol. Heal. Informatics, BCB 2020* 2020. DOI: 10.1145/3388440.3412458.

(31) Alperstein, Z.; Cherkasov, A.; Rolfe, J. T. All SMILES Variational Autoencoder. *arXiv* **2019**. DOI: 10.48550/arXiv.1905.13343.

(32) Dai, H.; Tian, Y.; Dai, B.; Skiena, S.; Song, L. Syntax-Directed Variational Autoencoder for Structured Data. *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.* 2018, 1–17.

(33) Kingma, D. P.; Welling, M. Auto-Encoding Variational Bayes. *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.* 2014, 1–14.

(34) Winter, R.; Montanari, F.; Noé, F.; Clevert, D. A. Learning Continuous and Data-Driven Molecular Descriptors by Translating Equivalent Chemical Representations. *Chem. Sci.* **2019**, *10*, 1692–1701.

(35) Varnek, A.; Fourches, D.; Horvath, D.; Klimchuk, O.; Gaudin, C.; Vayer, P.; Solov'ev, V.; Hoonakker, F.; Tetko, I.; Marcou, G. ISIDA - Platform for Virtual Screening Based on Fragment and Pharmacophoric Descriptors. *Curr. Comput.-Aided Drug Des.* **2008**, *4*, 191–198.

(36) Zabolotna, Y.; Volochnyuk, D. M.; Ryabukhin, S. V.; Gavrylenko, K.; Horvath, D.; Klimchuk, O.; Oksiuta, O.; Marcou, G.; Varnek, A. SynthI: A New Open-Source Tool for Synthon-Based Library Design. *J. Chem. Inf. Model.* **2022**, *62*, 2151–2163.

Recommended by ACS

Permutation Invariant Graph-to-Sequence Model for Template-Free Retrosynthesis and Reaction Prediction

Zhengkai Tu and Connor W. Coley

JULY 26, 2022

JOURNAL OF CHEMICAL INFORMATION AND MODELING

READ 

Exploration of Chemical Space Guided by PixelCNN for Fragment-Based De Novo Drug Discovery

Satoshi Noguchi and Junya Inoue

DECEMBER 01, 2022

JOURNAL OF CHEMICAL INFORMATION AND MODELING

READ 

MACAW: An Accessible Tool for Molecular Embedding and Inverse Molecular Design

Vincent Blay, Hector Garcia Martin, *et al.*

JULY 20, 2022

JOURNAL OF CHEMICAL INFORMATION AND MODELING

READ 

Graph-Driven Reaction Discovery: Progress, Challenges, and Future Opportunities

Idil Ismail, Scott Habershon, *et al.*

OCTOBER 03, 2022

THE JOURNAL OF PHYSICAL CHEMISTRY A

READ 

Get More Suggestions >

Supporting Information

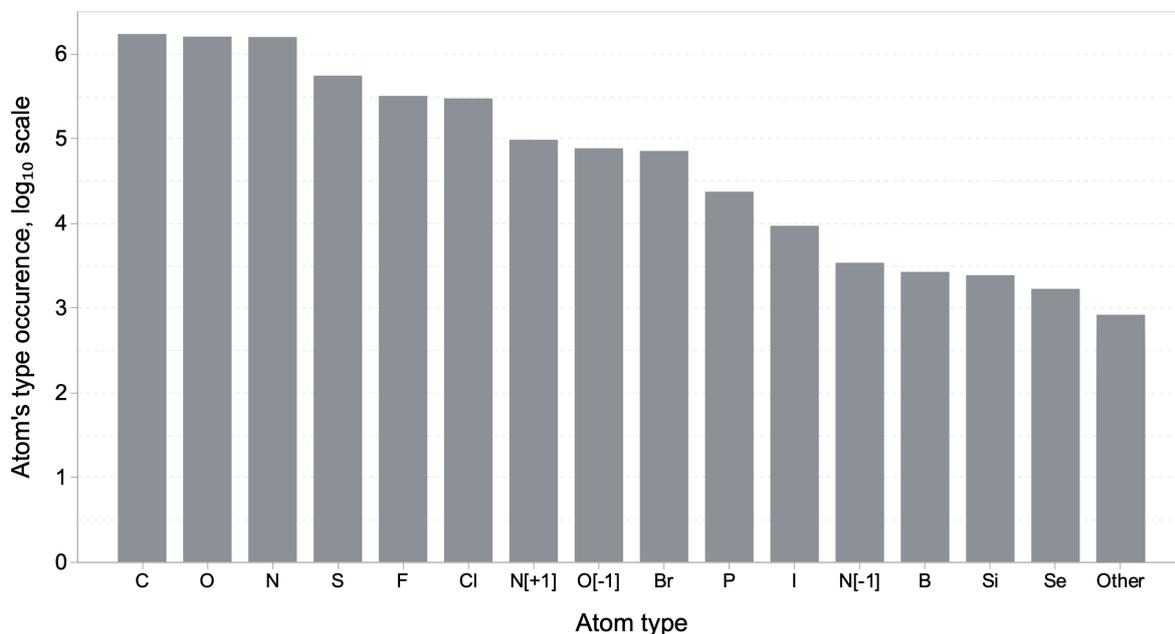


Fig. 3.5 Distribution of molecules from standardised ChEMBL v. 27 as a function of atoms' types presence.

Table 3.1 Results of ZINC 250K data set standardisation.

Filters	Training set	Test set
Number of molecules	244 455	5 000
Duplicates within set	1612	1
Duplicates between training and test sets		73
Remaining molecules	242 770	4 999

Training parameters All calculations were made with NVIDIA QUADRO RTX 6000 GPU with CUDA drivers 11.2. The version of Tensorflow[93] was 2.6. All preprocessing, including transformations of molecular graphs to matrix representation, was done with CGRtools[94] version of 4.1.33. The charts were done with Altair Python package[95, 96]

MOSES benchmarking The MOSES benchmarking[97] is a distribution learning benchmarking. The main goal is to create generative architecture, that will approximate distribution

Table 3.2 Training parameters for each data set. The parameters are equal for both ReFactor and HyFactor architectures.

Dataset	Latent dimension	Batch	Initial learning rate	Max number of atoms	Number of epochs	Number of atom types	ReFactor parameters	HyFactor parameters
ZINC 250K	512	256	0.001	39	100	11	12M	10M
MOSES	512	1024	0.001	28	100	7	12M	10M
ChEMBL & Enriched ChEMBL	1024	1024	0.0008	50	150	15	50M	40M

of real drug-like molecules over known distribution. In the best case, the model should generate valid, unique and novel structures, while the distribution of the generated molecules should be almost the same as real ones.

To measure the quality of learned distribution, several metrics are considered in this tool:

- Validity – a fraction of valid molecules compared to generated ones;
- Unique 10K – a fraction of 10K first unique molecules from the valid ones;
- Novelty - a fraction of novel generated molecules from unique ones;
- Frag (fragment similarity) – a cosine similarity based on the occurrence of BRICS fragments in compared sets. Higher value means that both sets have similar fragments;
- Scaf (scaffold similarity) – a cosine similarity based on the occurrence of Bemis-Murcko scaffolds in compared sets. Higher value means that both sets have similar scaffolds;
- SNN (Similarity to a Nearest Neighbour) – an average Tanimoto similarity calculated on Morgan fingerprints between a molecule from the generated set and its nearest neighbour from the reference set;
- FCD (Fréchet ChemNet Distance)[98] – a Wasserstein-2 distance computed on vectors produced by the last layer of ChemNet neural network between generated and reference sets.

- IntDiv (Internal Diversity) – an average Tanimoto distance between molecules in the generated set based on Morgan fingerprints;
- Filters – a fraction of molecules that pass MCFs (Medicinal Chemistry Filters) and PAINS (Pan-Assay Interfering compounds) medicinal filters[99].

The data used in MOSES benchmarking based on ZINC Clean Leads collection. After standardizing and filtering, 1.9M molecules with non-charged atom types such as C, N, S, O, F, Cl, Br remained. The cleaned set was split into training, test and scaffold test (TestSF) sets in ratio 9:1:1. In the scaffold test set, there were molecules with unique Bemis-Murcko scaffolds that were not present in the training and test sets.

Table 3.3 MOSES benchmarking results

Architecture	Valid	Unique	IntDiv	Filters	Novelty
ReFactor 0.4 std	0.886	0.265	0.869	0.850	0.578
HyFactor 0.4 std	0.729	0.267	0.860	0.959	0.129

Table 3.4 MOSES benchmarking results comparing to test set (Test) and test scaffolds set (TestSF)

Architecture	FCD		SNN		Frag		Scaf	
	Test	TestSF	Test	TestSF	Test	TestSF	Test	TestSF
ReFactor 0.4 std	1.743	2.380	0.547	0.511	0.996	0.993	0.847	0.041
HyFactor 0.4 std	0.365	0.871	0.614	0.566	0.999	0.998	0.862	0.003

Enriched ChEMBL experiments Synthons Interpreter (SynthI)[?] – knowledge-based reaction toolkit for the library analysis and design – was used for creating E_ChEMBL. It combines the RECAP-like[100] fragmentation approach (based on 38 retrosynthetic rules) with a synthons-based way of reagents representation. Synthons are increments of the BB that will be added to the final compound upon a particular chemical reaction. Their distinctive feature is the presence of special markings at the former position of the leaving groups (case of generation

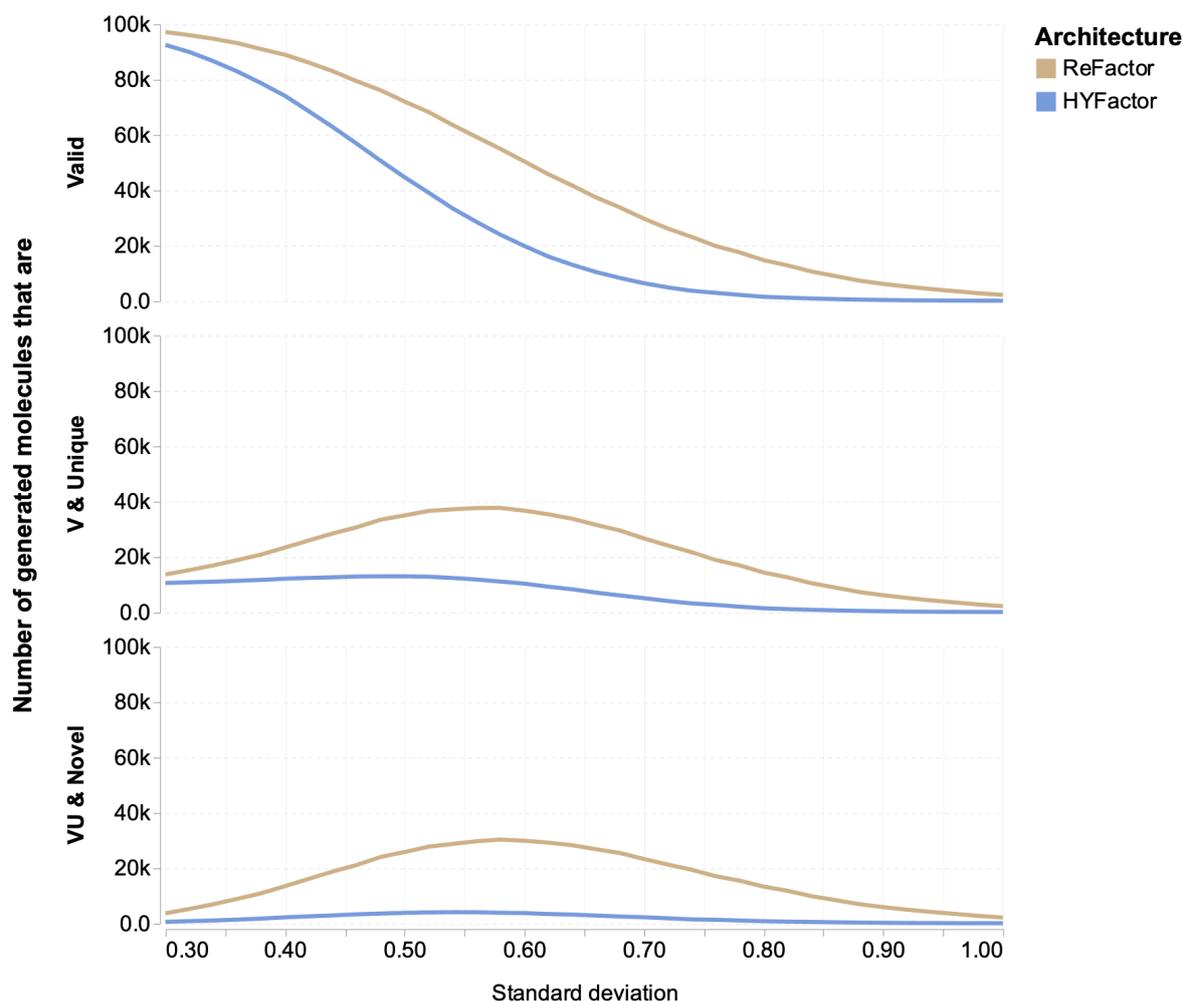


Fig. 3.6 Results of generation with variable standard deviation in range from the 0.3 to 1.0 with the step of 0.02. V corresponds to Valid, U corresponds to Unique.

from BBs, not used in this work) or bond disconnection if derived from compound fragmentation. The type of the mark defines the type of the reaction center – electrophile, nucleophile, radical etc. Those synthons can then be used as a source for the library enumeration.

In this work, as a source of synthons library, needed for the E_ChEMBL generation, the library of 372K ChEMBL synthons obtained in a previous work[101] has been used. As a reference, only molecules from ChEMBL containing 35-50 heavy atoms were used and analogues for them were generated using SynthI-Enumeration module. At the first stage, the reference compounds were fragmented (Figure 3.7 (I)) and the search of the analogues in a source synthons library was performed. Analogous synthons always have the same number of cycles and reaction centers, the types of reaction centers are also preserved, see Figure 3.7 (II and III)). They may differ by a presence or absence of a simple groups like methyl, hydroxyl,

halogen etc., or have a high Tanimoto similarity. On the next stage, all possible compounds are enumerated using the list of analogues synthons by applying the same reaction rules that were used for fragmentation of real ChEMBL molecules. In such a way, the resulting library of analogues still contain the same chemotypes as the reference compounds while introducing some diversity, needed for the training of HyFactor and ReFactor.

Summary for the section 3.2.1

In this work, we introduce a novel autoencoder architecture, referred to as HyFactor, which is based on hydrogen-count labelled graph representations. Through a comparison of HyFactor and ReFactor on the ZINC 250K and ChEMBL datasets, we demonstrate that the autoencoders based on HLG are capable of efficiently creating a vector representation of a molecule without any loss in performance. Additionally, the HLG representation enables a reduction of up to 20% of trainable parameters, which is particularly beneficial for architectures with a large number of parameters, as evidenced by experiments on the ChEMBL dataset.

Both ReFactor and HyFactor were also evaluated in the task of generating analogues, with HyFactor generating structures that were similar to the target structure. Analysis of the generated structures revealed that ReFactor generated a significant number of open-chain analogues of the cyclic target (seed) structure. Additionally, a high number of disconnected structures were generated by both tools, which were not recognised as invalid by the MOSES benchmarking package, highlighting the need for further development of metrics and scoring functions that estimate the structural properties of molecules.

We also recommend adding rationally generated virtual molecules to the training dataset in order to enhance the quality of generative architectures. This approach was proposed as a solution to address the issue of underrepresentation of large molecules in the training dataset, which our findings suggest may be a contributing factor to the increase in error rates for molecules containing more than 35 atoms, reaching nearly 25% for molecules with 50 atoms. Our analysis of the relationship between the reconstruction rates of the HyFactor and ReFactor and molecular size revealed this phenomenon. The implementation of this proposed technique resulted in a significant reduction of the error rate on large molecules to 7% for models trained on such a dataset.

In summary, the new HLG representation is a useful and efficient representation for inverse QSAR and can be beneficial for reducing the number of trainable parameters in architectures.

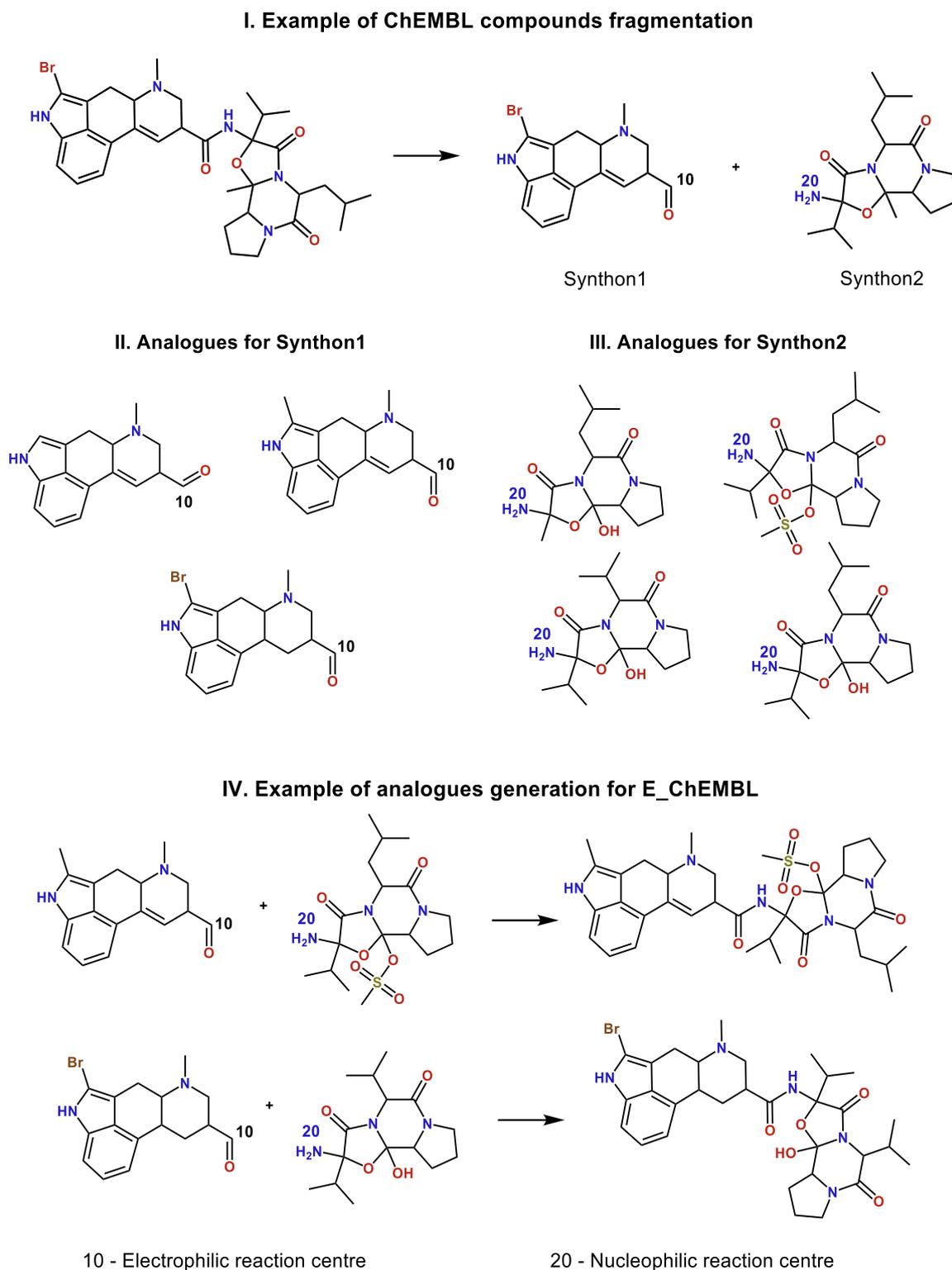


Fig. 3.7 Example of fragmentation of existing ChEMBL compounds (I), search of their analogues in the synthons library prepared from commercial building blocks (II), and generation of virtual compounds for E_ChEMBL (III).

3.2.2 Vector Quantisation Graph AutoEncoder

The previously developed HyFactor graph neural network demonstrated potential as a graph-based architecture for inverse QSAR. However, our analysis revealed that the encoder of HyFactor employed order-dependent operations, resulting in the latent space of HyFactor being corrupted by atom order bias. As this presents a significant limitation, we redesigned the architecture to eliminate this bias in the latent vectors.

In addition, we sought to improve the interpretability of the latent vectors of graph-based autoencoders. Therefore, we developed an autoencoder with a discrete latent vector of fragment counts, where learnable vectors represent fragments. This approach was implemented using a vector quantisation operation adopted from models used for image generation [102]. By learning atomic vectors representing atoms and their environment, this approach allows the application of all the analysis techniques developed for fragment-based descriptors.[103]

This section describes the new Vector Quantisation Graph AutoEncoder (VQGAE) architecture. This autoencoder operates in the discrete latent space and generates molecules in a one-shot manner. VQGAE was trained on the ChEMBL dataset, and its latent vectors were tested in the similarity ranking and QSAR benchmark prepared from the ChEMBL v.27 database for 532 targets. For comparison, fragment descriptors (ECFR and ISIDA) and latent vectors of autoencoders (HyFactor and LatentGAN) were also tested in the benchmarks.

Construction of order-independent molecular fragments space with vector quantised graph autoencoder

Tagir Akhmetshin¹, Arkadii Lin¹, Timur Madzhidov^{2*}, Alexandre Varnek^{1*}

Abstract: Autoencoders represent a promising technique for the inverse quantitative structure-activity relationship (QSAR) task. However, undesirable bias, such as atom ordering, affects the neighbourhood behaviour of autoencoders' latent space and, consequently, usage of the latent vectors as variables in machine-learning models. Here, we report a graph-based autoencoder which implements vector quantisation operation (VQGAE). The latter allows to learn vectorial representation of molecular fragments in an unsupervised manner. The latent vectors or fragment count vectors of VQGAE are permutation invariant and perform well in similarity ranking. In QSAR benchmarks, the VQGAE's latent vectors outperform those derived by some earlier developed SMILES-based and graph-based autoencoders. Finally, VQGAE autoencoder was used in the inverse QSAR task in order to design new A2A adenosine receptor inhibitors.

Keywords: Inverse QSAR, Autoencoders, Deep learning, Generative models, VQVAE, molecular graphs

INTRODUCTION

The current trend in chemoinformatics is the usage of deep learning architectures to learn the structural features of molecules.¹ Deep learning architectures attracted much attention as they can conditionally generate chemical structures based on the learned features. While it is an active field of research, many works showed possible advantages of deep learning in constrained molecular optimisation² and virtual libraries generation³. Nevertheless, few studies have explored the properties of the learned molecular representations.

One group of deep-learning approaches consists of architectures learning molecular vectors in unsupervised manners (e.g., autoencoders, normalising flows, etc.). They apply various operations to transform high-dimensional input into low-dimensional latent vectors. However, some of these operations have undesirable effects. The influence of atom order on the molecular latent vector can be an example of such an impact. It was first shown for SMILES-based architectures.⁴ SMILES-based architectures use sequential operations, which rely on the arrangement of the input symbols. Atoms' order influences the latent space of such architectures. This allows researchers to achieve higher reconstruction rates. At the same time, it damages the topology of the latent space. Bjerrum et al.⁴ studied the order-independence of latent vectors produced by a SMILES-based heteroencoder. The heteroencoder was trained using SMILES data

1. University of Strasbourg, Laboratoire de Chemoinformatique, 4, rue B. Pascal, Strasbourg 67081 (France) *e-mail: varnek@unistra.fr

2. Chemistry Solutions, Elsevier Ltd, Oxford, (United Kingdom) *e-mail: tmadzhidov@gmail.com

augmentation⁵ with different atom orders. It was shown that the latent vectors of the heteroencoder improve performance in the Quantitative Structure-Activity Relationship (QSAR) task compared to an autoencoder trained on canonical SMILES. Higher performance was explained by removing atoms' order dependence. Sadly, the reconstruction ability of such a heteroencoder decreased dramatically compared to an autoencoder based on canonical SMILES.

Another way to overcome order dependence in learned chemical space was proposed by Winter et al.⁶ The authors used a graph-based autoencoder to create a latent space without any influence of atoms' order, where the encoder consisted only of graph convolution networks. The obtained latent vectors were tested on four Quantitative Structure-Property Relationship (QSPR) tasks using MoleculeNet benchmarking⁷. The obtained results have demonstrated slight outperforming of learned molecular representations in comparison to ECFP fragments.

In this work, we develop a new order-independent graph-based autoencoder called Vector-Quantized Graph AutoEncoder (VQGAE). This autoencoder is designed to generate an order-independent molecular representation based on learned fragment vectors. Unlike previous work, it does not require a predefined set of fragments but is able to learn them in an unsupervised manner. Thus, the latent vector of VQGAE is represented by fragment counts, as in commonly used fragment-based descriptors. The atom order independence in the latent space of VQGAE is ensured by using only commutative operations in the encoder.

The performance of VQGAE was tested in representation learning and generative problems. For the first concept, two common virtual screening tasks were used, namely similarity ranking and QSAR benchmarking. The VQGAE latent vectors were compared with latent vectors from graph-based (HyFactor⁸) and SMILES-based (LatentGAN⁹) autoencoders, and with classical chemical descriptors, such as ISIDA¹⁰ fragment descriptors and ECFP fingerprints¹¹. It was shown that the VQGAE latent vectors are both atoms

order independent and capture structural features with the same efficiency as fragment-based descriptors. The generation ability of the autoencoder was tested in the context of Inverse QSAR, where VQGAE was successfully used to generate new antagonists of the adenosine A2A receptor.

METHODS

Data

The ChEMBL database (v. 27)¹² was used for the experiments. It was standardised with ChemAxon JChem's utilities¹³ using the following procedures: 1) dearomatisation, 2) isotopes removal, 3) stereo marks removal, 4) explicit hydrogens removal, 5) small fragments removal, 6) solvents removal, 7) salts strip, 8) neutralisation of charges, 9) functional groups transformation, 10) selection of canonical tautomer form of the molecule, 11) aromatisation, 12) duplicates removal, 13) dearomatisation.

The standardised dataset consisted of 1.6M structures with 15 atomic types (C, O, N, S, F, Cl, N+, O-, Br, P, I, N-, B, Si, and Se). The number of heavy atoms in molecules was no more than 50. The canonical order of atoms in molecules was obtained from a breadth-first search (BFS) algorithm, similar as it was done by Mercado et al.¹⁴

The curated ChEMBL data set was then split into a training set (80% of data or 1.3M molecules) and a test set (20% of data or 327K molecules) as it was done for the training of HyFactor architecture⁸.

Vector quantised graph autoencoder architecture

The vector quantised graph autoencoder (VQGAE) architecture comprises four parts: encoder, decoder, vector quantiser, and several linear layers for predicting structural properties from the feature vector. The general scheme is presented in Figure 1.

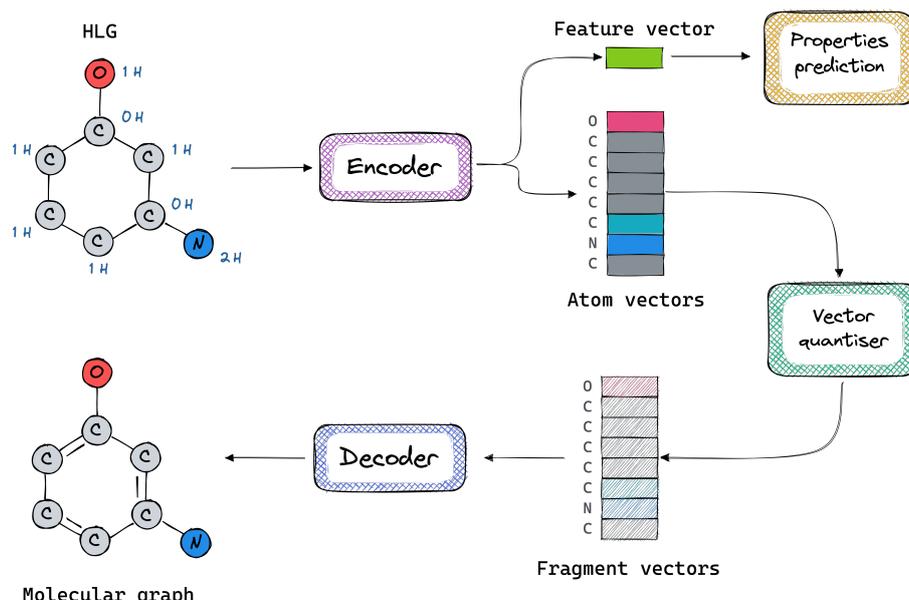


Figure 1. General scheme of VQGAE. HLG refers to the hydrogen-count labelled graph.

Encoding

The input graph is a hydrogen-count labelled graph (HLG)⁸ where each atom is encoded in a vector with the following properties: element number, period, group, number of electrons on the last subshell + atom's charge, number of last shells, the total number of hydrogens, whether or not the atom is in a ring, number of neighbours and counts of single, double, triple and aromatic bonds near current atom. Each property is normalised with a natural logarithm for

numerical stability. As a result, the input for the encoder consists of atom feature vectors with dimension 12 and the graph's adjacency matrix.

The scheme of the encoder is shown in Figure 2. The first operation is an increase of the dimensionality of atoms features by a simple linear operation. Then, these vectors are passed to a Graph Convolution Network (GCN) layer¹⁵ implemented in the Pytorch Geometric package¹⁶. After each GCN layer, the ReLU activation function is applied.

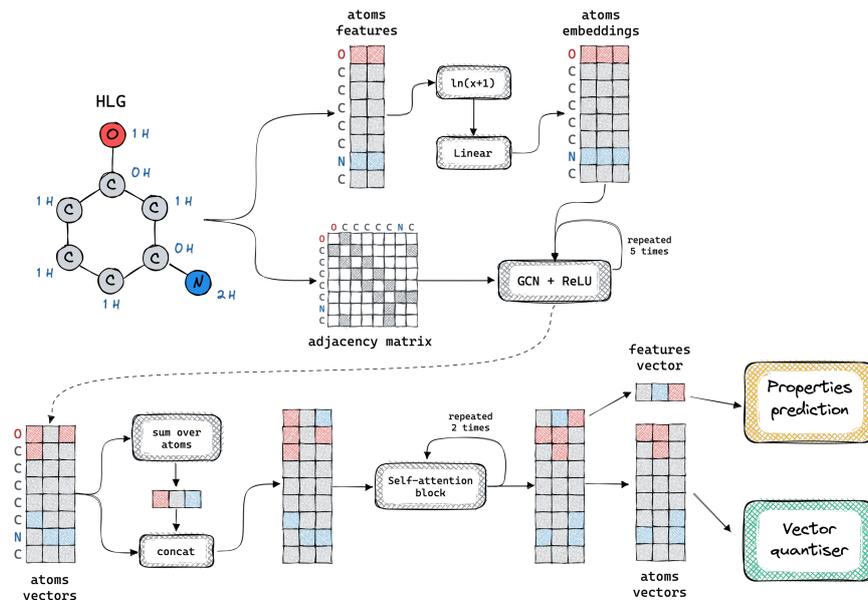


Figure 2. Scheme of VQGAE encoder. N is the maximal number of atoms, and D is the dimension of atoms vectors. GCN refers to the Graph Convolution Network and HLG to the hydrogen-count labelled graph.

The updated atom vectors are passed to the self-attention blocks (Figure 3). These blocks are inspired by class attention layers described in the work of Touvron et al.¹⁷ The authors proposed concatenating the training vector with a matrix instead of pooling it. In contrast, in our implementation, the trainable class (or feature) vector is replaced by a vector of the sum of the atom vectors.

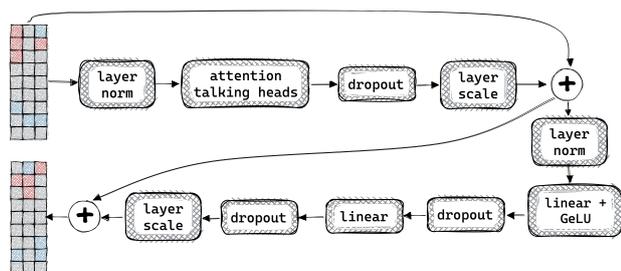


Figure 3. Scheme of self-attention block. The + operation refers to residual summation between input and output vectors.

Following the authors of the article¹⁷, the multi-head attention described in¹⁸ is replaced by the talking-heads attention¹⁹. Furthermore, the scale layer¹⁷ is used, a trainable diagonal matrix initialised by a constant value (0.005). This matrix is multiplied by the output of the dropout layer before residual summation.¹⁸

After self-attention blocks, the molecular feature vector and the final atom vectors are obtained. The molecular feature vector is used to predict eight pre-calculated structural properties, such as hetero atoms count, H-acceptors and H-donors count; chiral centres count; rings, hetero rings and aromatic rings and rotatable bonds count. Each property was calculated with ChemAxon¹⁴ cxcalc package. To predict them, eight independent linear layers are employed.

Vector quantisation

Usually, most architectures use pooling operations to reduce several atom vectors to one molecular latent vector. Thus, the quality of encoding and generation depends on the effectivity of pooling and unpooling operations, respectively. These operations require much computational power, and models become hard to interpret.

Another approach could be the usage of atom vectors directly to generate molecules. However, due to the nature of the graph convolution operation, atom vectors for the same atom with the same environment will differ if they come from different molecules. Therefore, storing all atom vectors and combining them to generate new molecules is technically impossible. But, if it can be assumed that atom vectors with small changes correspond to one fragment, then it is possible to average them to obtain a general (centroid) fragment vector representation. In this way, the number of atom vectors can be reduced without loss of information (Figure 4).

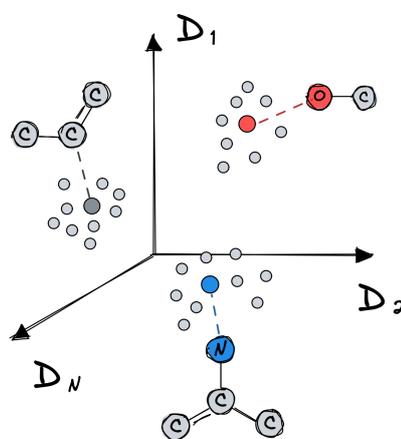


Figure 4. Vectors in N-dimensional fragment space. Here grey circles correspond to atoms vectors after the encoder, and the coloured circles correspond to average vectors (centroids) of the closest atom vectors. It is assumed that centroids represent fragments learned by graph convolution.

A vector quantisation operation is a way to obtain atom vectors' centroids (fragment vectors) during simultaneous training of the encoder and decoder. First proposed for images^{20,21}, this operation allows to create a codebook representing collection of fragment vectors. The vectors in the codebook are initialised randomly and updated during training to achieve higher reconstruction rates. During training and inference, these vectors are used to replace atom vectors from the encoder and then are passed to the decoder (Figure 5).

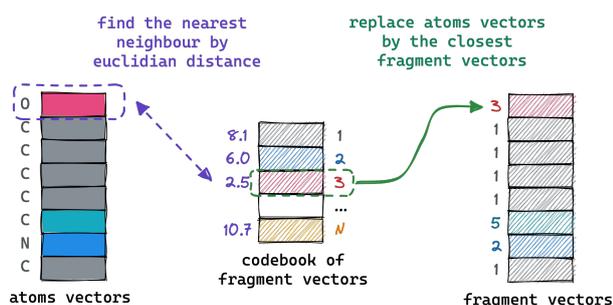


Figure 5. The scheme of vector quantisation step. The nearest neighbours of the atom vectors in the codebook are initially found. Then, the chosen vectors from the codebook are used to replace corresponding atom vectors. Here L represents the length of the codebook.

The update of the chosen vectors from the codebook minimises the distance between them and the corresponding atom vectors (see *loss function* section). Thus, the codebook vectors should move to clusters of atom vectors representing a single fragment to become clusters' centroids.

However, codebook vectors cannot be directly used in virtual screening tasks such as similarity search or QSAR modelling. Therefore, inspired by fragment-based approaches, one can count indices of the chosen codebook vectors and form integer molecular latent vectors (Figure 6).

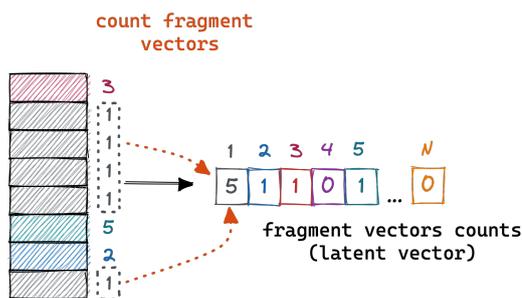


Figure 6. Translation of chosen vectors' indices from the codebook to molecular latent vector. Here L represents the length of the codebook.

Decoding

The obtained codebook atoms vectors are extracted and then passed to the decoder. The gated recurrent unit (GRU)²² layer is applied to break up symmetrical atoms that have the same neighbourhood in the molecule. During GCN, these atoms receive the same updates and it is expected that they have the same properties. Thus, symmetric atoms have identical atomic vectors and any neural network cannot distinguish which neighbour to connect with the atom. Therefore, indexing of symmetric atoms is necessary, which currently requires ordering of atoms in decoder. The hidden states of GRU are then given through layer scale and residually summed with GRU input vectors. Next, codebook atoms vectors are passed to 12 self-attention blocks and, in the end, to defactorization and linear layers, which predict bonds matrix and atoms types, respectively.

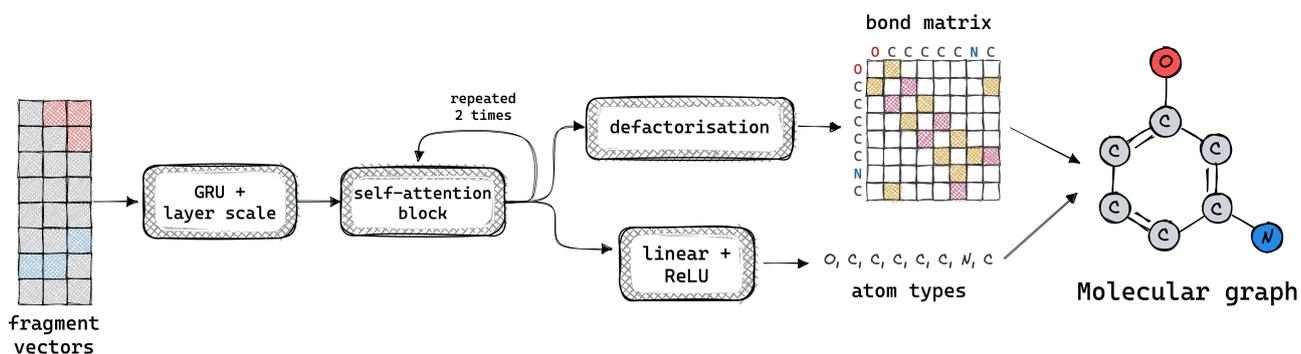


Figure 7. Scheme of the decoder. Here GRU is a gated recurrent unit.

Loss function

The loss function is composed of several components. The main function is

reconstruction loss L_{rec} , a sum of categorical cross-entropy for atoms predictions L_{atoms} (1), and binary cross-entropy for bonds predictions L_{bonds} (2):

$$L_{\text{atoms}} = -\frac{1}{n} \sum A * \log(\tilde{A}) \quad (1)$$

$$L_{\text{bonds}} = -\frac{1}{n^2} \sum_b^4 [E_b * \log(\tilde{E}_b) + (1 - E_b) * \log(\tilde{E}_b)] \quad (2)$$

$$L_{\text{rec}} = L_{\text{atoms}} + L_{\text{bonds}} \quad (3)$$

, where A is a one-hot matrix for atom types, \tilde{A} is a predicted atom types probability matrix, n is the number of atoms in a molecule, and E_b is the true adjacency matrix for bond type b .

However, the vector quantisation operation requires vector extraction by indexing, which is not differentiable. Thus, the gradient from reconstruction loss would not pass through this step, and the encoder would not be trained. To prevent this, as suggested by Van Den Oord et al.^{20,21} for vector quantised variational autoencoder (VQVAE), two losses are used – codebook and commitment losses. The idea of codebook loss is to shift selected codebook variables closer to the atom vectors while the gradient from the commitment loss moves the atom vectors towards the closest codebook vectors. Later, the authors replaced codebook loss with Exponential Moving Averages (EMA). The classical EMA updates each codebook vector as the average of all the nearest vectors from the encoder to a given codebook vector (4):

$$\mathbf{e} = \frac{1}{n} \sum_j^n \mathbf{z}_j \quad (4)$$

where \mathbf{e} is a codebook vector, \mathbf{z}_j – one of the n closest atom vectors to \mathbf{e} .

However, since the training is in batch mode, an online version of EMA is applied. To do so, the number of the nearest vectors in batch \mathbf{n} and their sum \mathbf{z} for the current batch \mathbf{t} are added to the previous count of closest atoms vectors \mathbf{N} (6) and their sum \mathbf{m} (5), respectively. The codebook vector is updated in the following way (7):

$$\mathbf{m}^t = \gamma * \mathbf{m}^{t-1} + (1 - \gamma) * \sum_j^n \mathbf{z}_j^t \quad (5)$$

$$\mathbf{N}^t = \gamma * \mathbf{N}^{t-1} + (1 - \gamma) * n^t \quad (6)$$

$$\mathbf{e}^t = \frac{\mathbf{m}^t}{\mathbf{N}^t} \quad (7)$$

, where \mathbf{e}^t is the codebook vector in the current batch \mathbf{t} , γ was taken as the default value of 0.99.

The EMA function does not require gradient computation, therefore the vector quantisation loss L_{vq} consists only of commitment loss (8):

$$L_{\text{vq}} = \frac{1}{k_t} \sum_i^{k_t} (z(\mathbf{e})_i - \text{sg}(\mathbf{e}_i))^2 \quad (8)$$

, where \mathbf{z}_e is the closest atom to the chosen codebook vector \mathbf{e} , k is the number of atoms vectors in the batch \mathbf{t} , sg – is a “stop gradient” operator, which stops the propagation of the gradient to a given argument.

The property loss is a sum of categorical cross-entropies of each property.

The full loss is calculated as the sum of reconstruction loss L_{rec} , vector quantisation loss L_{vq} and property loss L_{property} (9):

$$L = L_{\text{rec}} + 0.5 * L_{\text{vq}} + 0.01 * L_{\text{property}} \quad (9)$$

Ordering network

Despite VQGAE’s ability to generate molecules from given fragment vector indices, it requires that these indices should be arranged in a “canonical” order learned by GRU for successful generation. To address this issue, a model called an ordering network was trained to predict the canonical order of randomly arranged fragment indices from the VQGAE codebook. The architecture used fragment indices as input, with each index associated with an embedding vector. The embeddings were then processed through six self-attention blocks, identical to those used in the VQGAE. The target output was the “canonical” position of the fragment index in the

vector for the decoder. The loss function was used as a binary cross-entropy. The output of network can be seen as pseudo-permutation matrix. Different from permutation matrix, in the “pseudo” version the sum of values in columns were not equal to one. For extraction of canonical positions of fragment vectors, we also

implemented a beam search-like algorithm. The main goal of this algorithm is to find the combinations of unique positions with the highest sum of predicted probabilities.

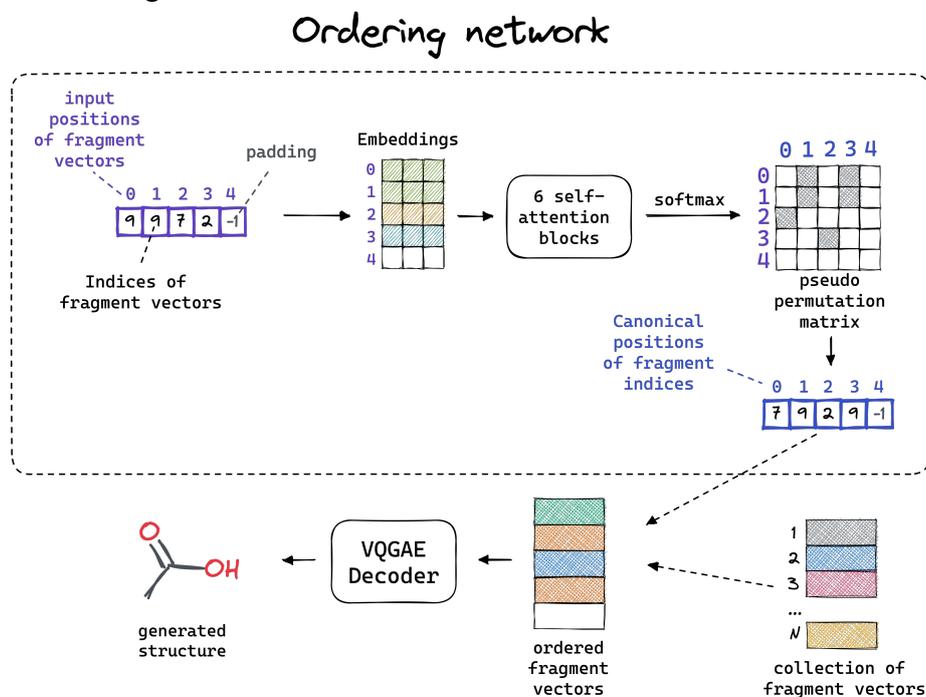


Figure 8. Scheme of the decoder. Here GRU is a gated recurrent unit.

Similarity search analysis

Similarity search is one of the most common methods in cheminformatics that heavily relies on the neighbourhood principle. This approach is one of the fundamental methods for virtual screening, which is still widely used in drug discovery research.

In this work, similarity search was used in a ranking setup, where the goal is to find the most similar structures in a data set for a given molecular query. First, the molecules from the ChEMBL data set were encoded in their vector representation using algorithmic fragment-based and deep learning approaches. It is important to note, that the ordering of atoms was randomised before encoding of molecular structure in a vector representation. Then a group of 33 molecules were taken from the ChEMBL v.30 data set¹³, which were registered in the US Accepted Names (USAN) in 2021, so these

molecules were not present in the ChEMBL v.27 standardised data set. The query molecules were also converted to vector representations with the same parameters as for ChEMBL structures. The 500 most similar structures from the ChEMBL database were identified for each query using the studied vectors. These sets of molecules were then used to compare each approach pairwise and by visual analysis. The choice of similarity metrics was based on the nature of descriptors. Thus, for fragment-based approaches, Tanimoto coefficient was used, and for continuous vectors, Euclidian distance was employed²³.

QSAR benchmarking workflow

One of the tasks where learned molecular features can be used is a Quantitative Structure-Activity Relationship (QSAR) problem. Based on the neighbourhood principle, QSAR can be used to assess the quality of the obtained molecular representation. In this way, it is

expected that the predictions on latent vectors of order-dependent autoencoders will be lower than those of order-independent autoencoders.

The data used for the QSAR benchmarking was collected from the ChEMBL v.27 database. More specifically, the activity information about targets for “Homo sapiens” organism with assay type “Binding assay” and target type “Single protein” was extracted. The obtained data consisted of 387K activity records for 1805 targets. Then, activity records were filtered if 1) they were measured for compounds that did not pass standardisation; 2) if the standard deviation of several IC₅₀ concentrations for the same target and compound was more than 20 nM. It must be noticed that stereo marks were removed during standardisation, so the filtering process also excluded stereo active compounds’ records. The remaining measurements for the “compound-target” pairs were averaged.

IC₅₀ thresholds were calculated for activity class labelling based on the number of active and inactive compounds for each target according to the protocol²⁴. The protocol consists of three steps. First, an active IC₅₀ threshold is selected from the range [10 nM, 50 nM, 100 nM, 300 nM, 500 nM, 700 nM, 1 μM] to determine at least 15 active compounds. Then, an inactive IC₅₀ threshold of 30% or 15 compounds is chosen. Finally, if the threshold for inactive compounds is more than ten times the threshold for active compounds, the threshold for active compounds is updated as the inactive threshold divided by 10 to collect more active molecules.

The final number of the activity records remaining was 189K for 149K compounds and 532 targets. The thresholds for each target, the number of compounds and activity analysis are given in the Supplementary information.

The machine learning algorithm for benchmarking was Random Forest²⁵, implemented in the Scikit-learn Python package²⁶. The following training procedure was used: the activity class labels and corresponding descriptors were extracted for each target, and then data was split into five folds using the StratifiedKFold algorithm²⁶. The standard

scaling was fitted on the training folds for each cross-validation iteration and used to transform the validation fold, except for fingerprints. To explore different parameters, the grid search was used. The metrics were balanced accuracy (BA) and receiver operating characteristic area under the curve (ROCAUC). The parameters of training are given in the Supplementary information.

RESULTS AND DISCUSSION

Analysis of fragments learned by VQGAE

In the methodology section it was assumed that centroids of atoms vectors after the encoder of VQGAE represent vectors of learned fragments. However, it is not possible to retrieve structure of a fragment by passing a single codebook vector to the decoder, as VQGAE will only generate a single atom. Therefore, we decided to extract fragments in an “indirect” manner. In this approach, substructures were extracted from molecules that contained atoms encoded in the same codebook vector, with these atoms serving as the centres of the substructures. The main assumption was that a small size of the environment around the central atom would result in the same extracted substructures that correspond to the learned fragment.

The standardised ChEMBL dataset was used to extract fragments. First, VQGAE was trained on this dataset and achieved a reconstruction rate of 94.5%. Fragment analysis was then performed on the same dataset. The environment around central atoms was used as two (i.e. only the alpha and beta atoms around the central atom were included). Several features of the atoms were kept in the fragment structures: presence of atom in the ring and maximum bond order around the atom.

During analysis of the fragments, we discovered that for one codebook vector there are several corresponding fragments. For example, in Figure 9a, for a single codebook vector, 6 fragments were extracted from 4783 molecules. The most frequent structure corresponds to the C(sp², central atom)-C(sp²)-C(sp³) acyclic fragment. The other fragments look quite similar

to the first one. At the training stage, VQGAE learns enough fragment vectors to maximise the reconstruction rate. At the same time, it is not forced to use as many fragment vectors as possible. In fact, at the end of training, out of a possible 4096 fragment vectors, only 1201 were non-zero. Since the number of non-zero components in the codebook is relatively small, the performance of VQGAE latent vectors in virtual screening tasks might be reduced. One way to increase codebook utilisation is to improve the vector quantization operation by applying noise substitution as suggested in²⁷ or codebook initialisation techniques such as k-means²⁸.

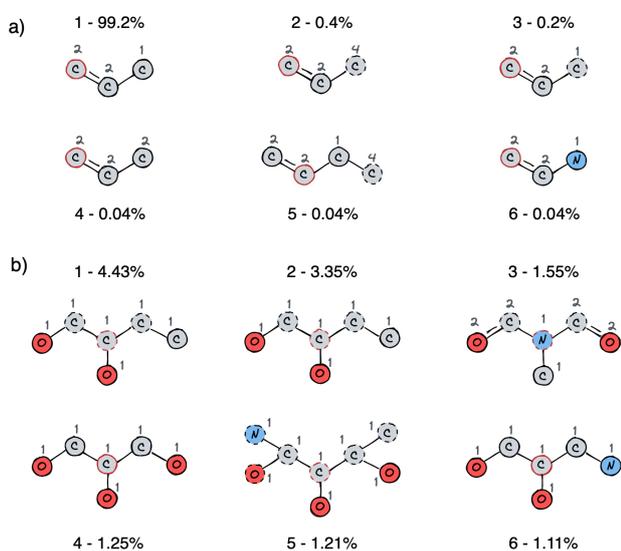


Figure 9. The examples of fragment structures for the two selected codebook vectors. Near the fragment number, the occurrence of the fragment structure in the molecule containing the corresponding fragment vector is given. The central atom is indicated by the red circle. The dashed circles represent atoms included in the ring. The number near the atoms is the maximum bond order around an atom, where 4 corresponds to the aromatic bond.

In the second example (Figure 9b), for one codebook vector more than 5K different structures of fragments from 42K molecules were extracted. Here the top six most frequent fragments are presented. In contrast to the previous example, there is no dominant fragment, as the most popular one was seen less than 5% of the time. The difference between these examples is that in the second case the central atom contains more neighbours.

Similarity search

Once the autoencoder has been trained, it is important to understand the quality of the latent vectors and its generative ability. These characteristics are non-trivial to measure, as there are no existing metrics for them. However, one way to analyse the performance of the autoencoder's latent space is through the similarity ranking task. The similarity search provides insight into the neighbourhood behaviour of the latent space, which is the basis for any modelling using the vector representation of molecules.

The VQGAE latent and feature vectors were compared with the time-proved descriptors for similarity search, which are ISIDA¹⁰ fragment descriptors and ECFP fingerprints¹¹. In addition, these representations were compared with latent vectors from graph-based (HyFactor⁸) and SMILES-based (LatentGAN⁹) architectures. The training details of these architectures are given in the Supplementary information. The similarity metric used for VQGAE feature vectors, HyFactor and LatentGAN latent vectors was the Euclidean distance and for others the Tanimoto coefficient.

A similarity search experiment was performed using the 6 descriptor sets of the 50 most similar structures for each query. Thus, each pair of descriptor sets was compared for overlapping structures for each of the 33 query molecules. The results are given in Figure 10.

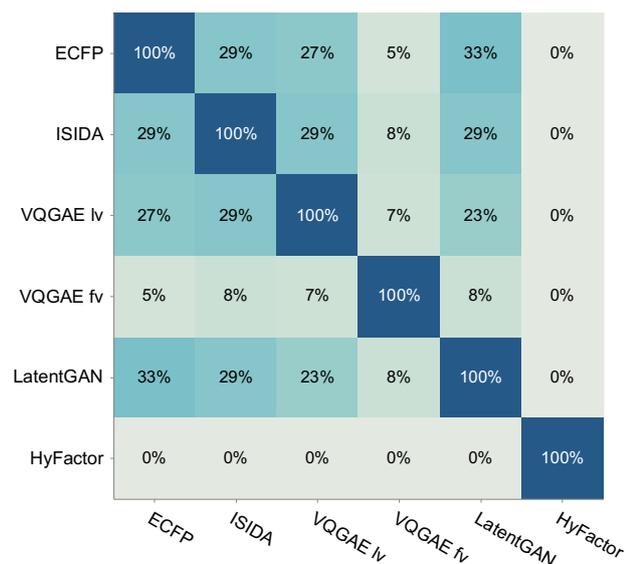


Figure 10. The average percentage of shared structures between the sets of 50 chemical structures most similar to a given query molecule, based on selected vector representations. The pairwise overlaps between these top 50 sets were compared within the query molecule, and the results were averaged across 33 query molecules. Here “VQGAE lv” corresponds to the latent vector and “VQGAE fv” corresponds to the feature vector.

On average, the ECFP and ISIDA methods share 16 common structures as the most similar ones. Regarding the deep learning techniques, the top 50 most similar molecules by VQGAE and LatentGAN latent vectors also have similar number of overlapping structures. Nevertheless, the results of the similarity search based on the

HyFactor latent vectors do not overlap with the results of any other representation.

During the analysis of the HyFactor architecture, it was discovered that the latent vectors of the autoencoder depend on the canonical ordering of atoms. Since its encoder includes a recurrent neural network (RNN) in the graph aggregation step, the neighbourhood behaviour in the latent space also depends on the order of atoms. In contrast, the results demonstrate that autoencoder of LatentGAN does not suffer from this problem, as it uses a heteroencoder with the SMILES augmentation technique.

To better understand the neighbourhood behaviour of all approaches, we conducted a visual analysis of the most similar molecules for two queries. The first query, *Tovinontrine* (Figure 11a), is considered “easy” because it has an analogue with Tanimoto similarity higher than 0.9 based on ISIDA and ECFP descriptors. The second query, *Bersacapavir* (Figure 11b), is considered “hard” as it has no analogues in the ChEMBL 27 data set. Thus, the most similar structure for this query does not exceed a Tanimoto coefficient of 0.65 based on ISIDA, 0.45 in the case of ECFP and 0.55 with VQGAE latent vectors.

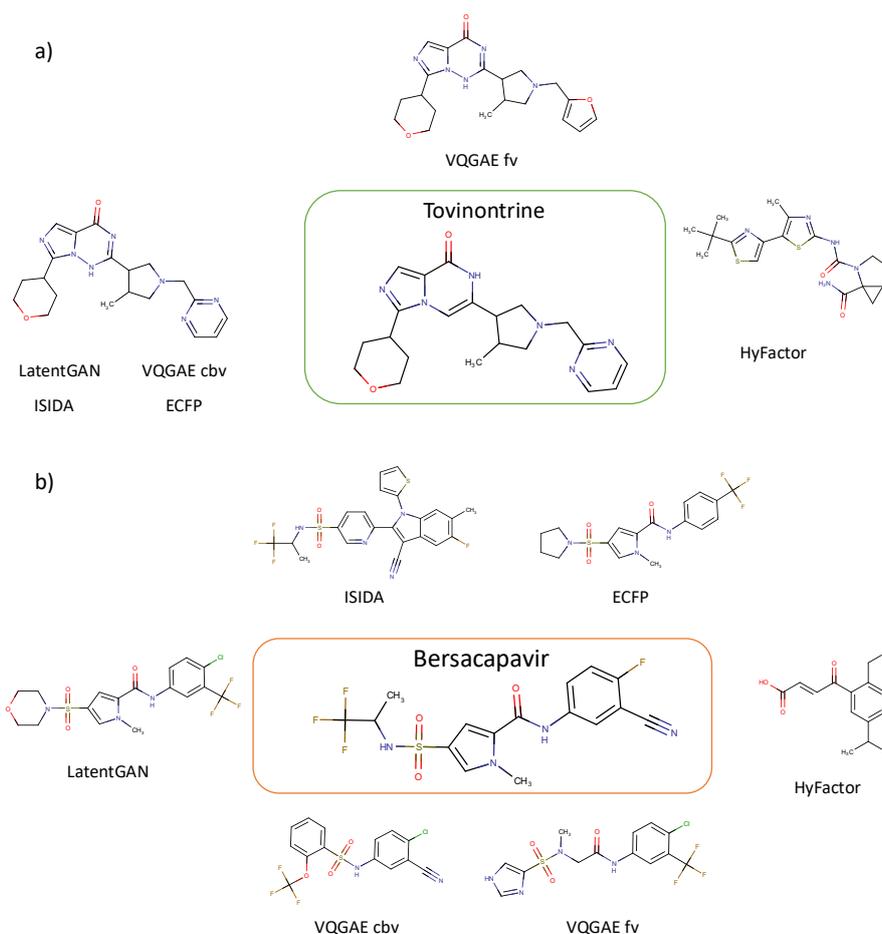


Figure 11. The most similar structures of *Tovinoitrine* (a) and *Bersacapavir* (b) that were chosen based on different vector representations. For the *Tovinoitrine*, every descriptor except VQGAE feature vectors and HyFactor latent vectors found the same structure.

Figure 11a shows that for the “easy” query, the search found close analogues based on all representations except HyFactor. As mentioned earlier, this was due to the order dependence in the HyFactor latent vector and once the order of the atoms was randomised for all molecules, the performance in similarity ranking was annihilated.

In the “hard” case (Figure 11b) for all representations different most similar molecules were found. By visual comparison it can be seen that LatentGAN’s latent vectors and ECFP descriptors provided the most similar structures, keeping the biggest part of scaffold. The search with VQGAE feature vectors was able to

identify relatively close analogue, but, in case of VQGAE latent vectors the structure has different than the given query. A possible explanation is the limited use of codebook vectors mentioned in the fragment analysis section. Since some fragment vectors correspond to different fragments, collision effects similar to those seen with short fingerprints can be expected.

QSAR benchmarking

Another way to assess the quality of latent space concerns their application as variables in machine-learning models linking chemical or biological properties with molecular structure.

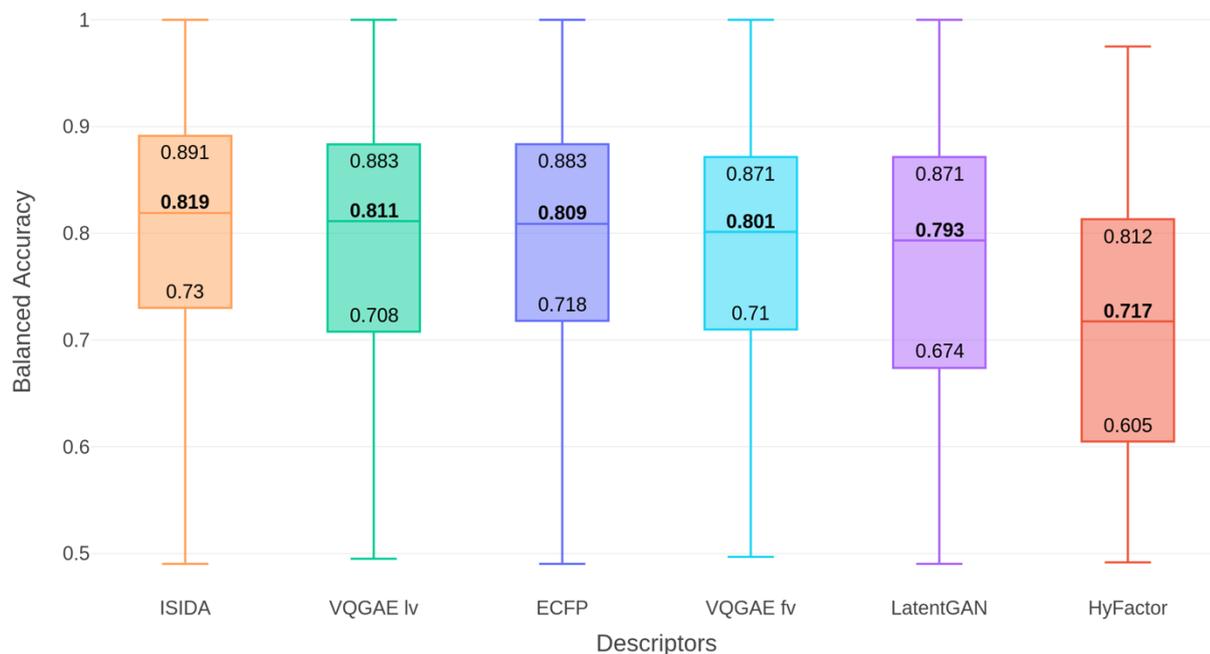


Figure 12. QSAR benchmarking results based on 532 datasets. Here LatentGAN corresponds to the SMILES-based heteroencoder, VQGAE fv corresponds to the feature vector of VQGAE, and VQGAE lv to the vector of codebook indices counts (molecular latent vector).

Here, the VQGAE latent and feature vectors and previously described molecular descriptors were benchmarked in classification models obtained with the random forest method on 532 datasets corresponding to different biological activities extracted from ChEMBL.

Statistical analysis of model performances (balanced accuracy) for different descriptors is given in Figure 12. One can see the model trained on HyFactor latent vectors demonstrated the worst results due to the atoms order dependence in the latent space. Other models showed similar performances, especially model trained on latent vectors of VQGAE showed the same performance as those trained on fragment-based descriptors. The high performance of the model based on VQGAE latent vectors can be explained by good “local” neighbourhood behaviour as shown in the similarity search benchmark. Indeed, biological measurements are often made for analogues of a structure within the same scaffold; therefore, it can be assumed that local effects are more important than long-range effects in the current QSAR benchmark.

Inverse QSAR

In the previous sections the performance of VQGAE to capture structural features was studied. However, the combination of VQGAE with the ordering network can also be used for the discovery of new chemical structures, particularly for the generation of new structures in the context of inverse QSAR. As a proof of concept, the VQGAE was used to design new antagonists of the adenosine A2A receptor.

The adenosine A2A receptor is an example of the G protein-coupled receptor (GPCR) family, which is important in mediating vasodilation, supporting the synthesis of new blood vessels and protecting tissues from collateral inflammatory damage. It is a well-studied receptor with several known molecules for which affinities have been measured. The standardised ChEMBL v.27 dataset contains 3300 molecules with measured inhibition constants (k_i) for the A2A target.

The inverse QSAR setup consisted of genetic algorithm, ordering network and VQGAE architecture. Genetic algorithm (GA) implemented in PyGAD v. 3.0.0 python library²⁹

was used as the method to generate new latent vectors of VQGAE. The fitness or scoring function used was a random forest regressor trained on previously extracted molecules with measured k_i for the A2A target. The ordering network was used to generate structures from the VQGAE latent vectors. It was trained separately from the VQGAE as it requires a fixed set of fragment vectors. The training and generation statistics are given in Supplementary Information.

In the result, 5 new latent vectors were selected as potentially highly active (predicted $k_i < 10$ nM), of which 4 were successfully converted into molecular structures via combination of ordering network and decoder of VQGAE.

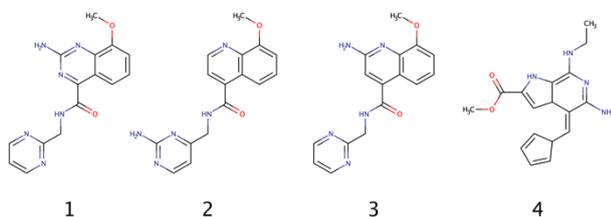


Figure 13. 4 generated structures by VQGAE with ordering network.

The 4 generated molecules are shown in **Figure 13**. Among them, 1-3 share the same Bemis-Murcko framework and highly similar to each other. During analysis of these structures, it was discovered that the structure 1 is present in the SureChEMBL patent database, but not in the ChEMBL. This compound was registered as A2A antagonist with measured $k_i = 43$ nM. Therefore, this result confirms that our approach was able to identify molecules with desired properties.

CONCLUSIONS

Here, we report a new autoencoder (VQGAE) able to learn the vectorial representation of fragments in an unsupervised manner. Its architecture allows to obtain an atom order independent latent space. The latent vectors of VQGAE demonstrated high performance in both similarity search and QSAR benchmarking compared to fragment-based descriptors and latent vectors of previously developed architectures. A combination of

VQGAE with the ordering network can be efficiently applied to the inverse QSAR task which was demonstrated on the example of A2A adenosine receptor inhibitors. We believe that this work opens new perspectives for exploring unsupervised learning of fragments. Follow-up studies are still needed to analyse learned fragment space and to improve generation ability of the proposed autoencoder.

DATA AND SOFTWARE AVAILABILITY

The Python code and pre-trained model weights of VQGAE are available in a GitHub repository: <https://github.com/Laboratoire-de-Chemoinformatique/VQGAE>

ACKNOWLEDGEMENTS

TA thanks the Region Grand Est for the PhD fellowship.

BIBLIOGRAPHY

- (1) Baskin, I. I. The Power of Deep Learning to Ligand-Based Novel Drug Discovery. *Expert Opin. Drug Discov.* **2020**, *15*, 755–764. <https://doi.org/10.1080/17460441.2020.1745183>.
- (2) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **2018**, *4* (2), 268–276. <https://doi.org/10.1021/acscentsci.7b00572>.
- (3) Polykovskiy, D.; Zhebrak, A.; Sanchez-Lengeling, B.; Golovanov, S.; Tatanov, O.; Belyaev, S.; Kurbanov, R.; Artamonov, A.; Aladinskiy, V.; Veselov, M.; Kadurin, A.; Johansson, S.; Chen, H.; Nikolenko, S.; Aspuru-Guzik, A.; Zhavoronkov, A. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Front. Pharmacol.* **2020**, *11*, 1–10. <https://doi.org/10.3389/fphar.2020.565644>.
- (4) Bjerrum, E. J.; Sattarov, B. Improving

- Chemical Autoencoder Latent Space and Molecular de Novo Generation Diversity with Heteroencoders. *Biomolecules* **2018**, *8* (4), 1–17. <https://doi.org/10.3390/biom8040131>.
- (5) Arús-Pous, J.; Johansson, S. V.; Prykhodko, O.; Bjerrum, E. J.; Tyrchan, C.; Reymond, J. L.; Chen, H.; Engkvist, O. Randomized SMILES Strings Improve the Quality of Molecular Generative Models. *J. Cheminformatics* **2019**, *11* (1), 1–13. <https://doi.org/10.1186/s13321-019-0393-0>.
- (6) Winter, R.; Noé, F.; Clevert, D.-A. Permutation-Invariant Variational Autoencoder for Graph-Level Representation Learning. **2021**, No. NeurIPS.
- (7) Wu, Z.; Ramsundar, B.; Feinberg, E. N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; Pande, V. MoleculeNet: A Benchmark for Molecular Machine Learning. *Chem. Sci.* **2018**, *9* (2), 513–530. <https://doi.org/10.1039/c7sc02664a>.
- (8) Akhmetshin, T.; Lin, A.; Mazitov, D.; Zabolotna, Y.; Ziaikin, E.; Madzhidov, T.; Varnek, A. HyFactor: A Novel Open-Source, Graph-Based Architecture for Chemical Structure Generation. *J. Chem. Inf. Model.* **2022**, *62* (15), 3524–3534. <https://doi.org/10.1021/acs.jcim.2c00744>.
- (9) Prykhodko, O.; Johansson, S. V.; Kotsias, P. C.; Arús-Pous, J.; Bjerrum, E. J.; Engkvist, O.; Chen, H. A de Novo Molecular Generation Method Using Latent Vector Based Generative Adversarial Network. *J. Cheminformatics* **2019**, *11* (1), 1–13. <https://doi.org/10.1186/s13321-019-0397-9>.
- (10) Varnek, A.; Fourches, D.; Horvath, D.; Klimchuk, O.; Gaudin, C.; Vayer, P.; Solov'ev, V.; Hoonakker, F.; Tetko, I.; Marcou, G. ISIDA - Platform for Virtual Screening Based on Fragment and Pharmacophoric Descriptors. *Curr. Comput. Aided-Drug Des.* **2008**, *4*, 191–198. <https://doi.org/10.2174/157340908785747465>.
- (11) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50* (5), 742–754. <https://doi.org/10.1021/ci100050t>.
- (12) Gaulton, A.; Bellis, L. J.; Bento, A. P.; Chambers, J.; Davies, M.; Hersey, A.; Light, Y.; McGlinchey, S.; Michalovich, D.; Al-Lazikani, B.; Overington, J. P. ChEMBL: A Large-Scale Bioactivity Database for Drug Discovery. *Nucleic Acids Res.* **2012**, *40*, 1100–1107. <https://doi.org/10.1093/nar/gkr777>.
- (13) ChemAxon Ltd: Budapest, Hungary. <https://chemaxon.com/> (accessed 2022-04-19).
- (14) Mercado, R.; Bjerrum, E. J.; Engkvist, O. Exploring Graph Traversal Algorithms in Graph-Based Molecular Generation. *J. Chem. Inf. Model.* **2021**, No. 2, acs.jcim.1c00777. <https://doi.org/10.1021/acs.jcim.1c00777>.
- (15) Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*; 2017; pp 1–14. <https://doi.org/10.48550/arXiv.1609.02907>.
- (16) Fey, M.; Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric. **2019**, No. 1, 1–9.
- (17) Touvron, H.; Cord, M.; Sablayrolles, A.; Synnaeve, G.; Jégou, H. Going Deeper with Image Transformers. **2021**. <https://doi.org/10.1109/iccv48922.2021.00010>.
- (18) Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *2017-Decem* (Nips), 5999–6009.
- (19) Shazeer, N.; Lan, Z.; Cheng, Y.; Ding, N.; Hou, L. Talking-Heads Attention. 2020. <http://arxiv.org/abs/2003.02436>.
- (20) Van Den Oord, A.; Vinyals, O.; Kavukcuoglu, K. Neural Discrete Representation Learning. *Adv. Neural Inf. Process. Syst.* **2017**, *2017-Decem* (Nips), 6307–6316. <https://doi.org/10.48550/arXiv.1711.00937>.

- (21) Razavi, A.; van den Oord, A.; Vinyals, O. Generating Diverse High-Fidelity Images with VQ-VAE-2. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
- (22) Cho, K.; van Merriënboer, B.; Bahdanau, D.; Bengio, Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. **2014**.
- (23) Willett, P.; Barnard, J. M.; Downs, G. M. Chemical Similarity Searching. *J. Chem. Inf. Comput. Sci.* **1998**, *38* (6), 983–996. <https://doi.org/10.1021/ci9800211>.
- (24) Lin, A.; Baskin, I. I.; Marcou, G.; Horvath, D.; Beck, B.; Varnek, A. Parallel Generative Topographic Mapping: An Efficient Approach for Big Data Handling. *Mol. Inform.* **2020**, *39* (12), 1–12. <https://doi.org/10.1002/minf.202000009>.
- (25) Breiman, L. *Random Forests*; 2001; Vol. 45, pp 5–32.
- (26) Pedregosa, F.; Michel, V.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Vanderplas, J.; Cournapeau, D.; Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Thirion, B.; Grisel, O.; Dubourg, V.; Passos, A.; Brucher, M.; Perrot and Édouardand, M.; Duchesnay. *Scikit-Learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot*; 2011; Vol. 12, pp 2825–2830. <http://scikit-learn.sourceforge.net>. (accessed 2020-09-15).
- (27) Vali, M. H.; Backstrom, T. NSVQ: Noise Substitution in Vector Quantization for Machine Learning. *IEEE Access* **2022**, *10*, 13598–13610. <https://doi.org/10.1109/ACCESS.2022.3147670>.
- (28) Kapre, R.; Ouyang, J. Using Discrete VAEs on T1-Weighted MRI Data to Embed Local Brain Regions; 2021.
- (29) Gad, A. F. PyGAD: An Intuitive Genetic Algorithm Python Library, 2021.

SUPPLEMENTARY INFORMATION

VQGAE training

The VQGAE was trained using ChEMBL v.27. Specifically, the autoencoder network was trained for 100 epochs with a batch size of 500 and a constant learning rate of 0.0002. The dimensions of all atom vectors were set to 512, and the size of the codebook was set to 4000. The number of heads in the multi-head attention (MHA) layers was 16. In the encoder, the number of MHA was 2, and in the decoder, it was 8. The layer scale's initial value was 0.0005, and the dropout probability was set to 0.2. The optimiser was AdaBelief with a gradient clip equal to 1. After training, VQGAE achieved a reconstruction rate of 98.5% on the training set and 94.5% on the validation set.

Other molecular vector representations

ISIDA fragments

The type of ISIDA fragments was as follows: atom-centred fragments based on sequences of atoms and bonds of fixed length from 2 to 4 atoms with a special marker if the atom was in a cycle. The fragments were extracted from 300k random molecules from the ChEMBL database, and 1% of the most popular fragments were retained (4018).

ECFP fingerprints

The experiments used ECFP fingerprints with a radius of 2 and 4096 bits. The RDKit Python package was used to encode the molecules in the fingerprints.

LatentGAN

The heteroencoder part of the LatentGAN architecture was trained on ChEMBL v.27. It was trained for 100 epochs with a batch size of 128 and a constant learning rate of 10^{-3} for the first 50 epochs and an exponential decay thereafter, reaching a value of 10^{-6} in the last epoch. The dimensions of each of the two bidirectional long-short memory units were 256. The latent vector dimension was defined as the dimension of the feed-forward (code) layer, which was 128. The standard deviation of the Gaussian noise smoothing layer was 0.1. The number of stacked unidirectional LSTM layers in the decoder was 3. In addition, standard scaling and principal component analysis (PCA) decomposition were applied to the input vectors. The number of the first PCA components was equal to the dimension of the input vectors (equal to the number of symbols used in the SMILES strings - 37). The resulting loss was 0.065

Ordering network training

The fragment indices and their positions were extracted from the same dataset on which VQGAE was trained. The data was divided into training and validation sets in a 4:1 ratio. The model was trained for 100 epochs with a batch size of 256, using an AdaBelief [140] optimiser with an initial learning rate of 0.0005 and a dropout probability of 0.4. The dimension of all vectors was set to 512. The trained model achieved an F1 score of 0.82.

Inverse QSAR with genetic algorithm.

The crossover type was chosen as scattered crossover, which randomly selects the gene from one of the 2 parents. The scoring (fitness) function used in the GA was a Random Forest (RF) [144] regression model, which was trained on 3300 molecules with known inhibition constants. The RF model was trained using 5-fold cross-validation on VQGAE latent vectors. The model achieved an R2 value of

0.60, and was used as is. The GA optimisation started from the latent vectors of molecules with known inhibition constant and to ensure the applicability domain. The optimisation continued for 20 generations, resulting in 100 vectors that corresponded to new structures.

Summary for the section 3.2.2

In this study, we developed a new graph-based autoencoder for the inverse QSAR task that operates in a discrete latent space, free from the influence of atoms order bias, as demonstrated in similarity ranking and QSAR tasks. Additionally, it achieves a high reconstruction rate comparable to the performance of the order-dependent HyFactor.

Comparison with other approaches revealed that the latent vectors of the proposed VQGAE have the highest performance in QSAR, along with fragment-based approaches. Furthermore, the neighbourhood behaviour of the latent space of VQGAE is similar to ISIDA and ECFP methods, as observed on similarity ranking benchmarks. However, it should be noted that every vector representation showed different neighbourhood behaviour during the similarity search of structures with no analogues in the dataset.

In conclusion, the proposed discrete autoencoder shows promise as a more efficient approach for inverse QSAR compared to the previously proposed HyFactor. With the combination of a generative architecture or combinatorial optimisation method, this approach has the potential to become an essential tool in de novo molecular design. Further experimentation is needed to fully assess its capabilities, such as analysing learned fragment vectors and conducting QSAR based on them.

Chapter 4

Self-learning-based synthesis planning

Exploring the vast and largely uncharted chemical space presents a significant challenge. While current generative approaches have demonstrated high performance in goal-directed tasks for the optimisation of physicochemical properties of interest, the generated structures have to be synthetically feasible. However, designing more diverse and distinct molecules from known compounds will result in less synthesisable structures. Therefore, there is a need for a method that can verify the synthetic feasibility of generated structures and provide possible synthetic pathways to them.

The first generative architectures were combined with Synthetic Accessibility score[5] (SAScore) as a scoring function to avoid generating molecules that are difficult to synthesise. This heuristics-based function considers the presence of fragments in a molecule and its size and number of macrocycles to determine whether it is “easy” or “complex” to synthesise. While SAScore and similar scoring functions align with chemical intuition, they do not ensure the current structure can be synthesised. Moreover, they do not take into account the availability of starting molecules (building blocks) and reactions.

An alternate strategy is to use retrosynthetic planning tools[104], which search for synthetic pathways for the given set of generated structures. This approach is more reliable than SAScore, as it provides examples of different synthetic routes, conditions and starting materials. However, it should be noted that this approach is computationally intensive. To address this limitation, recent advancements in automated retrosynthesis have employed a combination of Monte-Carlo tree search (MCTS) and deep learning models, significantly increasing the speed of the search process[21]. Furthermore, these new MCTS-based retrosynthesis variants have been trained using self-learning techniques, allowing the models to improve continually through experience[105, 2]. This can be used not only for speeding up the retrosynthesis planning but also has the potential to be integrated with robotic synthesis systems, paving the way for the development of fully autonomous systems capable of optimising the synthesis of

various compounds. This idea represents a significant step towards realising the dream of fully autonomous systems that can facilitate routine searches and enhance the development of drugs and materials.

The core of the automated synthesis planning tool is the search algorithm and the scoring functions that guide the search process (Figure 4.1). The scoring functions play a crucial role in determining the search. Furthermore, the search algorithms rely on the use of retrosynthetic transformations, which are employed to decompose a molecule into its potential precursors. These transformations are hand-crafted by human experts or sourced from reaction databases. The latter approach has gained popularity recently, as automated extraction from existing data is more efficient and straightforward than manual creation. Additionally, the recent machine learning scoring functions are also trained on the reaction data. Unfortunately, these databases suffer from non-unique representations of reactions and structures, errors in valence and reaction mechanisms, missing reagents etc. Therefore, given that the quality of retrosynthesis planning is closely tied to the quality of the data, a pre-processing step is necessary. This step involves curating and standardising the reaction data before the extraction and utilisation of retrosynthetic transformations. However, even after the retrosynthetic search has identified potential synthetic pathways, a post-processing step is required to evaluate the reactions in terms of regio- and stereo- selectivity, optimal conditions, and efficiency in terms of thermodynamic and kinetic parameters. All of these components are essential for the successful implementation of a retrosynthetic planning tool capable of providing synthetic pathways for even the most complex compounds.

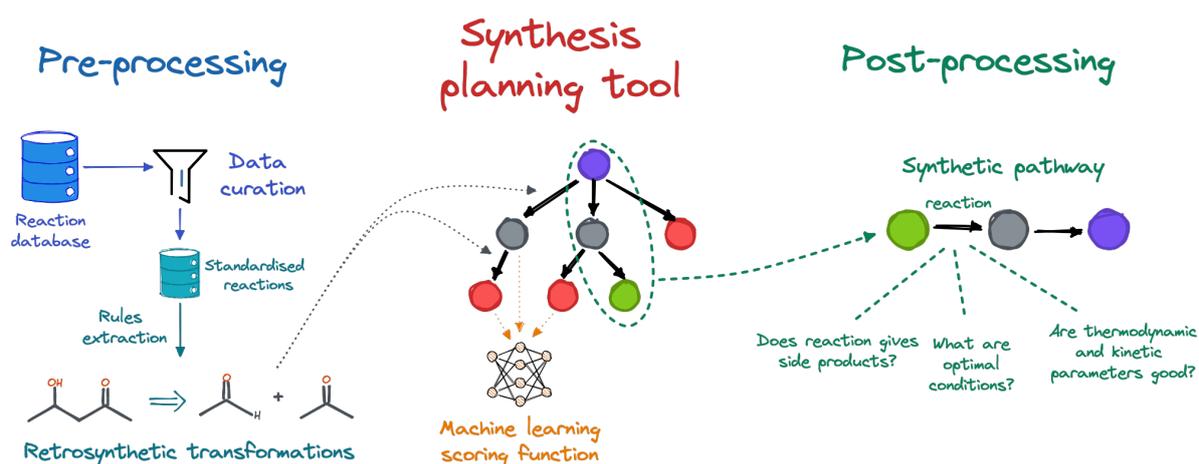


Fig. 4.1 The three main steps for development of high-performant automated retrosynthesis planning tool.

In this chapter, we will focus on the aforementioned challenges associated with automated retrosynthetic approaches. The discussion is outlined as follows:

- Section 4.1 outlines a unified protocol for the curation of reaction data. The discussion summarises all stages of reaction record standardisation. The standardisation protocol was implemented and used to standardise data from three known reaction databases - Reaxys, Pistachio, and USPTO.
- Section 4.2 introduces a novel tool for retrosynthetic planning. This tool is based on the principle of self-learning, which is employed in training the graph-based neural network scoring function. In this manner, the tool is able to adapt and improve retrosynthetic strategies based on previous searches. The effectiveness of this tool was evaluated through comparison with other retrosynthesis approaches and the state-of-the-art tool, AiZynthFinder[3], using both simple and complex compounds for synthesis. The results of these comparisons demonstrate the potential of the proposed tool in advancing the field of retrosynthetic planning;
- Section 4.3 describes two new concepts for validating machine learning models to predict reaction parameters. Predicting reaction pathways under various conditions is a current challenge in advanced retrosynthetic planning. Accurate prediction of kinetic and thermodynamic parameters is a crucial element of this challenge. The proposed validation techniques are employed to assess the unbiased performance of machine learning algorithms for the prediction of reaction physicochemical parameters in new transformations and conditions. The effectiveness of these techniques was evaluated on the task of predicting reaction rate constants for SN₂, E₂, and Diels-Alder reactions.

4.1 Reaction data curation

Previously, retrosynthesis programs used hard-coded retro transformations to generate potential precursors starting from a synthetic target. The creation of these rules required expert input to define the reaction centre and functional groups that are favourable or unfavourable for the reaction. To date, the most comprehensive set of rules was compiled by the authors of Synthia[106], comprising approximately 100K rules. As the authors note, as the database of retro-rules has grown over time, it has become increasingly challenging to add new rules without conflicting with existing ones. In addition, the current database is proprietary, which prevents from testing algorithms using this set of rules. Furthermore, the manual collection of rules is a resource-intensive and time-consuming task, as it requires the participation of an expert for the creation of each rule. Therefore, there is a constant need for alternative approaches to collect retro transformations efficiently.

Another method of obtaining retro-rules is data-driven, which involves the automatic extraction of retro-rules from reaction records. There are currently two possible ways to annotate reaction data: hand-crafted and text mined. Examples of hand-crafted reaction databases are Reaxys[107] with over 50M records and SciFinder[108] compiled by experts who analysed organic chemistry articles and manually added entries into databases. The alternative approach involves the automatic translation of articles and patents into a reaction records format, such as SMIRKS or RDF files. The most popular and only freely available reaction dataset of this type was collected from United States Patent and Trademark Office (USPTO) database by Daniel Lowe[109], comprising approximately 3.75 million reaction records from 1976 to 2016. Another text mined commercial database is Pistachio[110] which was compiled from United States, European and WIPO patents and consists of 9.7M reaction records. Comparing both approaches, hand-crafted databases have an advantage over automatically generated ones as they include more context about reactions, specifically regarding selectivity and conditions, which can aid the search algorithm in producing more accurate results. By contrast, automatic extraction is a cheaper and faster method suitable for big data analysis. However, both approaches can lead to inconsistent representation of reactions and retro-rules obtained from them, which subsequently can reduce the overall performance of the retrosynthesis tool.

It has been well established over several decades that the quality of data is essential for the effective utilisation of chemical knowledge. Much of this research has focused on molecular data and has demonstrated that the lack of proper standardisation in the representation of molecules can degrade QSAR/QSPR models[111]. Over the past 30 years, rules for molecular standardisation have been developed, taking into account various chemical properties such as isomerism and tautomerism. In contrast, the discussion about the standardisation of reactions has emerged more recently with the availability of large reaction databases (consisting of more than one million records). The complexity of standardising reactions is significantly greater than that of standardising molecules due to the additional details that must be considered. As a result, the “structure” or “composition” of reactions is highly dependent on the conditions under which they are conducted. Additionally, many reaction records are imbalanced and multi-step, resulting in missing important information.

In this chapter, we will discuss a unified protocol for the curation of reaction data that consists of four main steps: curation of chemical structures, transformation, reaction conditions, and endpoints (such as yield or thermodynamic and kinetic parameters). We will further implement the first two steps of reaction data standardisation and apply them to the known reaction databases of USPTO, Pistachio, and Reaxys.

doi.org/10.1002/minf.202100119

Reaction Data Curation I: Chemical Structures and Transformations Standardization

Timur R. Gimadiev,^[a] Arkadii Lin,^[b] Valentina A. Afonina,^[c] Dinar Batyrshin,^[c] Ramil I. Nugmanov,^[c] Tagir Akhmetshin,^[b, c] Pavel Sidorov,^[a] Natalia Duybankova,^[d] Jonas Verhoeven,^[d] Joerg Wegner,^[d] Hugo Ceulemans,^[d] Andrey Gedich,^[e] Timur I. Madzhidov,^[c] and Alexandre Varnek^{*,[a, b]}

Abstract: The quality of experimental data for chemical reactions is a critical consideration for any reaction-driven study. However, the curation of reaction data has not been extensively discussed in the literature so far. Here, we suggest a 4 steps protocol that includes the curation of individual structures (reactants and products), chemical

transformations, reaction conditions and endpoints. Its implementation in Python3 using CGRTools toolkit has been used to clean three popular reaction databases Reaxys, USPTO and Pistachio. The curated USPTO database is available in the GitHub repository (Laboratoire-de-Chemoinformatique/Reaction_Data_Cleaning).

Keywords: chemical reactions · data cleaning · big data · Reaxys · USPTO · Pistachio

1 Introduction

Data quality is crucial for the effective storage and exploitation of chemical knowledge. Errors in a chemical structure representation always complicate a modelling task and may cause technical problems in searches for chemical entities. Fourches et al.^[1] pointed out the importance of data curation on Quantitative Structure-Activity Relation (QSAR) models performance and proposed a workflow to clean up compounds datasets in the context of their application in further modelling studies. Despite an expanding importance of reaction data cleaning to enable machine-learning, deep-learning and recently reaction mining applications,^[2–6] this issue has been very little discussed.^[7]

So far, two databases have been intensively used in the modelling studies: Reaxys^{®[8,9]} (<https://www.reaxys.com/>) and USPTO reaction dataset.^[10] Reaxys is a commercial database containing data from 105 patent offices and 16,000+ journals with 55+M manually annotated reactions. Access to this data is limited to partners of the Elsevier company and it has been extensively used for various applications of deep-learning neural networks to retrosynthesis,^[4,11–15] robochemistry,^[16] and prediction of optimal reaction conditions,^[6] as well as for analysis of reaction network.^[17] USPTO is the largest public dataset of chemical reactions extracted from the US patents using text mining techniques.^[18] Nowadays, it contains > 3 M reactions extracted from 9 M patent applications and patents issued in the period of 1976–2016.^[19] This dataset has been extensively used for different applications: analysis of reaction databases,^[20] forward^[21–23] and backward-synthesis (single-step, also called retro-synthesis if multi-step),^[24] reactions classification,^[25,26] atom-to-atom mapping,^[27] yield prediction,^[28–30] and compound role assessment (i.e.

whether a compound is reactant, solvent, or catalyst).^[31] *Pistachio* is an extended commercial derivative of USPTO; it contains reaction data extracted by text mining from recent US and EU patents. A brief description of some other reaction datasets was given in the reference.^[32]

Notwithstanding the extensive use of the abovementioned reaction databases in machine-learning applications, very little effort has been invested to curate reaction data. Most of the suggested structures' standardization workflows,^[20,22,24,25,31,33] recombine selected steps of the molecules cleaning proposed suggested by Fourches et al.^[1,34] We assume that a chemical reaction is a complex

- [a] T. R. Gimadiev, P. Sidorov, A. Varnek
*Institute for Chemical Reaction Design and Discovery (WPI-ICReDD), Hokkaido University
Kita 21 Nishi 10, Kita-ku, 001-0021 Sapporo, Japan*
- [b] A. Lin, T. Akhmetshin, A. Varnek
*Laboratory of Chemoinformatics, UMR 7140 CNRS, University of Strasbourg
4, Blaise Pascal str., 67081 Strasbourg, France
E-mail: varnek@unistra.fr*
- [c] V. A. Afonina, D. Batyrshin, R. I. Nugmanov, T. Akhmetshin, T. I. Madzhidov
*Laboratory of Chemoinformatics and Molecular Modeling, Butlerov Institute of Chemistry, Kazan Federal University
18, Kremlyovskaya str., 420008 Kazan, Russia*
- [d] N. Duybankova, J. Verhoeven, J. Wegner, H. Ceulemans
*Janssen Pharmaceutica
30, Turnhoutseweg str., 2340 Beerse, Belgium*
- [e] A. Gedich
Arcadia Inc., Bol'shoy Sampsoniyevskiy Prospekt, 28 корпус 2, 194044 St Petersburg, Russia

Supporting information for this article is available on the WWW under <https://doi.org/10.1002/minf.202100119>

¹ Copyright © 2020 Elsevier Limited except certain content provided by third parties. Reaxys is a trademark of Elsevier Limited.

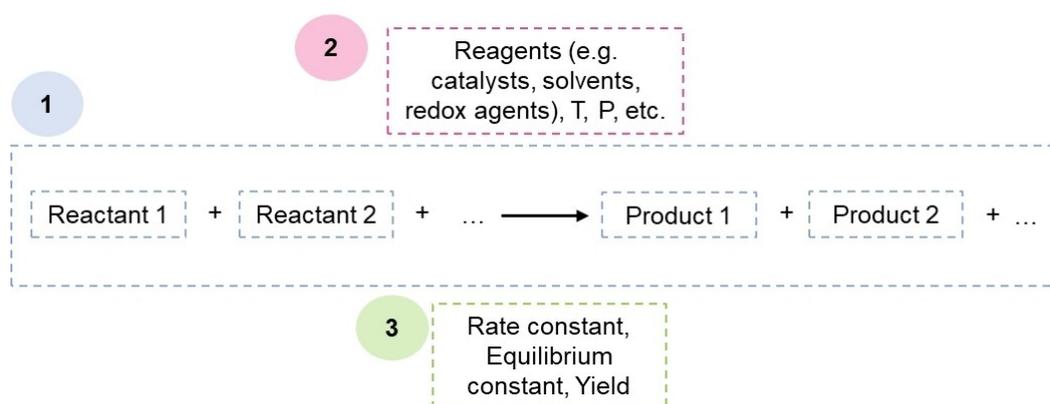


Figure 1. Chemical reaction is described by (1) chemical transformation proceeding under (2) particular reaction conditions and yielding (3) some endpoints. Chemical transformation assembles two types of molecular species: reactants and products.

object that combines a chemical transformation and corresponding reaction conditions (see Figure 1) and, hence, a curation protocol must fully account for this complexity. Here, a *transformation* term corresponds to a reaction equation that links sets of reactants and products. According to IUPAC definition, a *reactant* is “a substance that is consumed in the process of a chemical reaction” whereas a *product* is “a substance that is formed during the chemical reaction”.^[35] Atoms of reactants that change their connectivity and/or formal charge(s) upon the transformation constitute a *reaction centre*. To identify a reaction centre, a one-to-one correspondence between atoms of reactants and products called *atom-to-atom mapping*^[36] (AAM) must be established. Along with reactants and products, some other chemical entities called *reagents* (or additives) can be specified. These are catalysts (or some compounds forming catalytic system), solvents, catalytic poisons, complexation agents, redox agents, detergents, and acids/bases. It should be noted that the IUPAC definition of a reagent as a “substance that is added to the chemical system in order to bring about a reaction”^[35] is not precise enough for automatized reagents recognition. Two alternative definitions have been suggested in the literature for reagents identification. Schneider et al.^[31] attributed a compound to reagents if it wasn't affected in the course of reactions, whereas Gao et al.^[6] considered any compound as a reagent if it did not contribute any carbon atom to the reaction. In turn, *reaction conditions* describe a physico-chemical environment in which a transformation takes place. Generally, a conditions' description includes both numeric properties like temperature, pressure, and reaction duration, as well as reagents. A reaction can be characterised by some *endpoints* representing yield and its kinetic and thermodynamic parameters (Figure 1).

Since errors may occur in each component of a reaction, a curation workflow should consist, at least, of four steps related to the curation of (1) individual chemical structures (reactants, products and reagents), (2) chemical transforma-

tions, (3) reaction conditions, and (4) endpoints (see Figure 2). Only the two first steps are considered in this

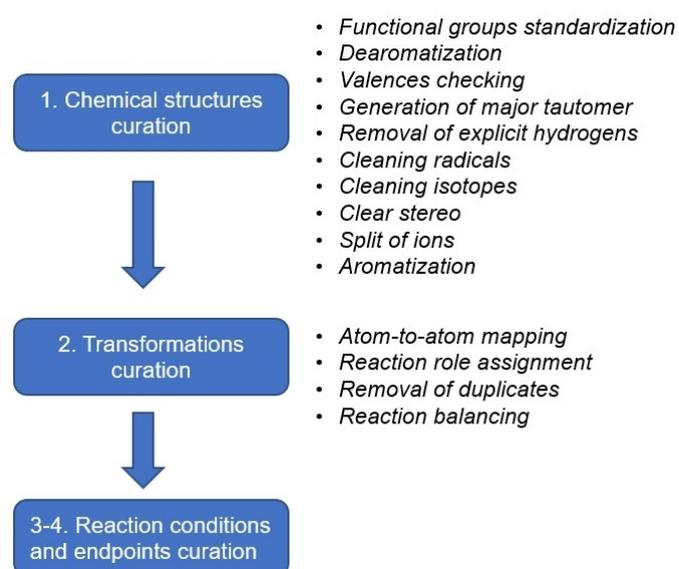


Figure 2. Main steps of reaction curation workflow. Only the two first steps are considered in this work.

work; the others represent a topic for a separate scientific article. Notice that not all the described steps are necessarily needed; the curation protocol may vary as a function of a particular chemoinformatics task (see Table 1).

This paper contains two chapters. The first one describes a protocol of individual structures and transformations curation and relates to the two first steps of the workflow shown in Figure 2. The second chapter reports results of the curation of three popular databases (Reaxys, USPTO and Pistachio) using the developed protocol.

Table 1. Typical reaction modelling tasks and related reaction standardization steps.

Task	Exemplary works	Curation stage(s) required ^a
Template-free forward synthesis	[21,22]	1
Template-free retrosynthesis	[13]	
Template-based reaction generation and reaction rules extraction	[37]	1, 2
Selection of optimal retrosynthetic rules for a given product	[38]	
Rule-based retrosynthesis	[11,12]	
Reaction generation and discovery	[39]	
Prediction of optimal reaction conditions	[6,40]	1, 2, 3
Prediction of kinetic and thermodynamic properties of reaction	[41,42]	1, 2, 3, 4
Prediction of reaction yield	[43,44]	

^a See Figure 2.

2 Reaction Curation Workflow

2.1 Existing Software Tools for Reaction Data Curation

Nowadays, there exist no ready-to-use tools able to perform the ensemble of reaction curation operations mentioned in Figure 2. Thus, the most popular non-commercial libraries – RDKit,^[45] CDK,^[46] Indigo,^[47] and OpenBabel^[48] – can be applied to standardize only individual molecules extracted from reaction equation. Among existing software solutions, only ChemAxon JChem Standardizer^[49] and CGRTools^[50] can perform out-of-the-box standardization of molecules in reactions using common rules described by Fourches et al.^[1]

There are several popular software packages that establish atom-to atom mapping (AAM) between reactants and products. These are non-commercial or free for academia like Indigo,^[47] Reaction Decoder Tool,^[51] RXNMapper^[52] and AutoMapper/ChemAxon JChem Standardize,^[49] commercial tools like Automapper^[53] from BIOVIA and ICMaP^[54] from InfoChem, or web-applications like ReactionMap^[55] and DREAM.^[56] The two latter, however, are only applicable to balanced reactions, which are very rare in a reaction database. A review of mapping software tools was published by Chen et al.^[36]

In some cases, the parsing of chemical records proceeds with errors that occur during the interpretation of chemical data. One of reasons is that reaction data are stored in a particular file format, which may not be readable by a given tool. For instance, in the USPTO database, the records are stored in the XML format that combines an extended reaction SMILES^[57] and related reaction conditions. An extended reaction SMILES string combines a regular SMILES with some supplementary information needed to relate certain substructures separated by a dot and belonging to a complex or salt. However, the extended SMILES format is poorly supported by some cheminformatics tools. For instance, several thousands of USPTO reactions are not readable by RDKit, but are correctly parsed by Indigo and ChemAxon JChem.

The absence of ready-to-use tools for reaction curation motivated us to develop a curation workflow that combines

several selected standardization steps. It was implemented in Python3 language^[58] using CGRTools library^[50] and ChemAxon's facilities.^[49] This tool is available in the GitHub repository `Laboratoire-de-Chemoinformatique/Reaction_Data_Cleaning`.

2.2 Chemical Structures Curation

2.2.1 Specific Features of Structures Curation

Some steps of the structures curation protocol, like functional groups standardization, aromatization and valences checking are similar with those recommended for med-chem databases.^[1] However, several previously recommended options like “removal of inorganics”, “delete fragment”, “strip salts” and “neutralize” lead to deterioration of the reaction equation and to discarding many important organic chemistry reactions (e.g., Grignard^[59]).

Organometallic compounds (usually discarded in med-chem data cleaning) represent a particular problem because they are poorly represented using conventional bond types. Typically, a metal atom is linked to its neighbours via single covalent bonds (Figure 3) or is represented as an isolated cation (Figure 4). Both representations may lead to significant problems upon reactions standardization. For example, in the tetrakis(triphenylphosphino)palladium complex shown in Figure 3, Pd(II) forms the covalent bonds with the P atoms of four triphenylphosphine molecules. This results in assigning wrong valences to the phosphorus atoms (5 instead of 3). Figure 4 (top) shows a reaction of substitution into a cyclopentadiene moiety of ferrocene in which the latter is represented by three separated ionic species. Appearance of the same species (Fe(II) cation and cyclopentadienyl anion) in the reactants and products sides suggests their assignment to reagents and consequent removal from the both sides which leads to a completely wrong reaction equation (see Figure 4, bottom). A special coordination bond type between metal and its environment could be a solution of this problem. For the moment, there exists no tool able to standardize organometallic com-

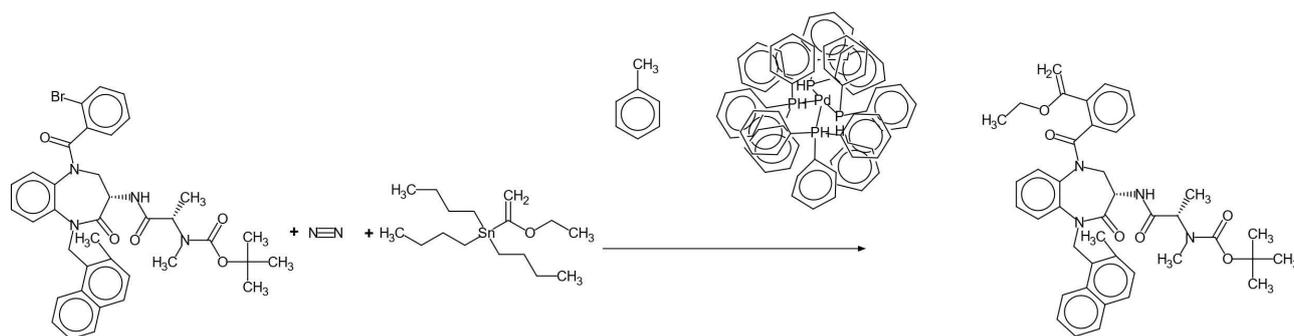
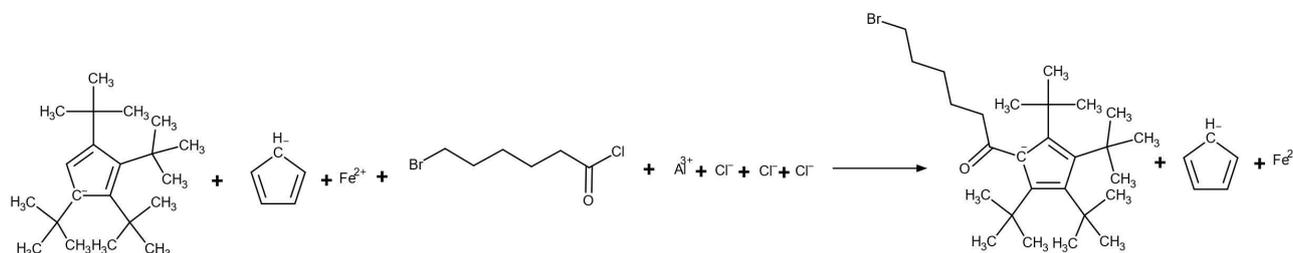


Figure 3. Example of the USPTO record where invalid atom valences were detected with CGRTools. Here, the Pd atom in the tetrakis (triphenylphosphino)palladium complex is connected via covalent bonds to each P atom. As a consequence, Marvin Sketch (ChemAxon) considers phosphorus in triphenylphosphine to be 5-valent whereas its real valence is 3.

Before standardization



After standardization

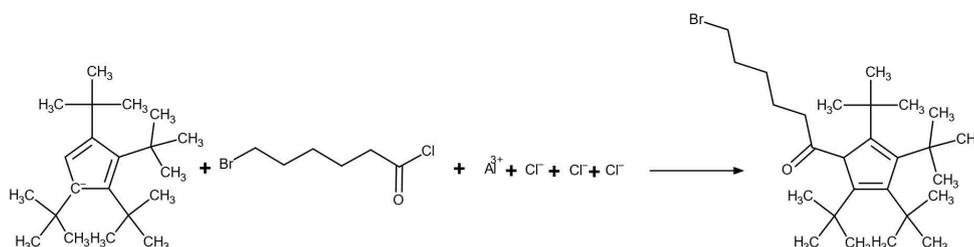


Figure 4. USPTO reaction extracted from the patent US20030137713A1 before (top) and after (bottom) standardization with the protocol developed in this work. Ferrocene is represented as 3 separated ionic species both in reactants and products sides of the reaction equation. One can see that removal of unaffected species destroys the reaction equation.

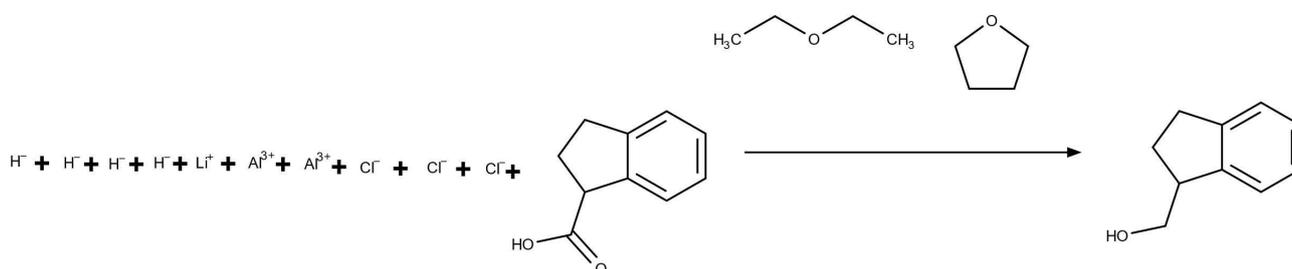


Figure 5. USPTO reaction extracted from the patent number US20010051625A1 in which all components of LiAlH_4 and AlCl_3 are represented by mixture of ions.

pounds, which creates a significant problem to curate reactions that contain organometallics.

A similar problem is observed for some inorganic compounds represented by a mixture of ions. For example, Figure 5 shows a reaction in which all components of LiAlH_4

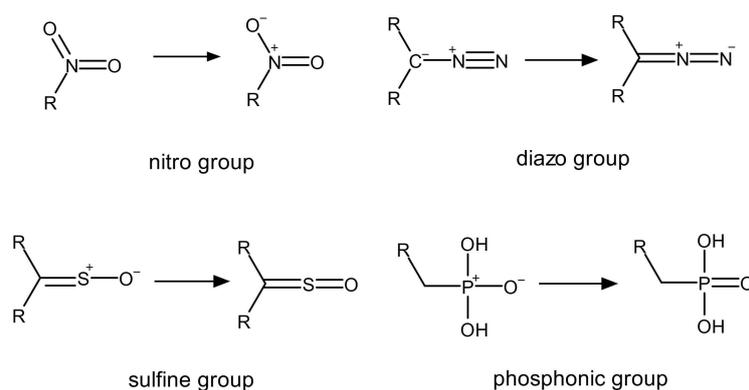


Figure 6. Examples of standardization of functional groups using CGRTools.^[50]

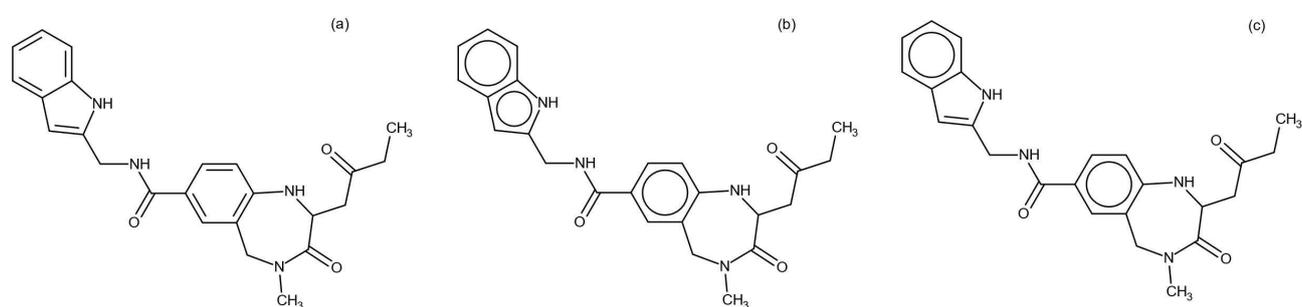


Figure 7. An example of three approaches applied to represent aromatic compounds: (a) Kekule form, (b) Full aromatization (including pyrroles), and (c) Partial aromaticity (i.e. only six-members rings are aromatized).

are represented by mixture of ions. A single covalent bond Al–H is more relevant for the structure encoding because it allows to keep the AlH_4^- structure as an individual species.

2.2.2 Structure Curations Steps

2.2.2.1 Functional Groups Standardization

Big data collections gathered from different sources often suffer from inconsistent representation of certain functional groups. This leads to a reduced number of records retrieved in substructure searches and may introduce noise in the data. Functional groups must be transformed into their standard representations as it is shown in Figure 6. A list of standard representations in CGRTools for 31 popular functional groups is reported by Nugmanov et al.^[50] and is constantly extending in order to tackle some errors found in the Reaxys database (see examples in Figure S1 in SI).

2.2.2.2 Aromatization

Several different algorithms of structure aromatization have been proposed,^[49] see an example of three aromatization styles in Figure 7. Therefore, a database gathering information from different sources may contain structures aromat-

ized according to different aromatization rules. In such a way, the same molecules may not be identified as duplicates, which introduces noise in data mining tasks.

Although aromatization solves the problem of standard representation of arenes and the vast majority of heterocycles, it may cause ambiguity in structure of heterocycles containing both pyrrolic and pyridinic nitrogen atoms, like imidazole. In aromatized structures of such heterocycles, the protonation state of atoms, i.e., their tautomeric form, can hardly be identified. Therefore, SMILES of such structures should contain information about implicit hydrogens, but, on the other hand, the information on implicit hydrogens is not stored in standard MDL file formats. Sometimes, implicit hydrogens in heterocycles are omitted in SMILES notation. This may cause problems with file reading and dearomatization.

In order to achieve the aromatization consistency within a given dataset, we suggest, first, to transform all chemical structures to the Kekulé form and then re-aromatize them again at the end of the “Structures curation” part using selected aromatization algorithm. Among different aromatization algorithms, the “*partial*” option, which keeps heterocycles with pyrrolic nitrogen in the Kekulé form looks rather convenient. Technically, this helps to restore position of hydrogen atoms in heterocycles (see Figure 7). Note that aromatization is considered here only as a part of the

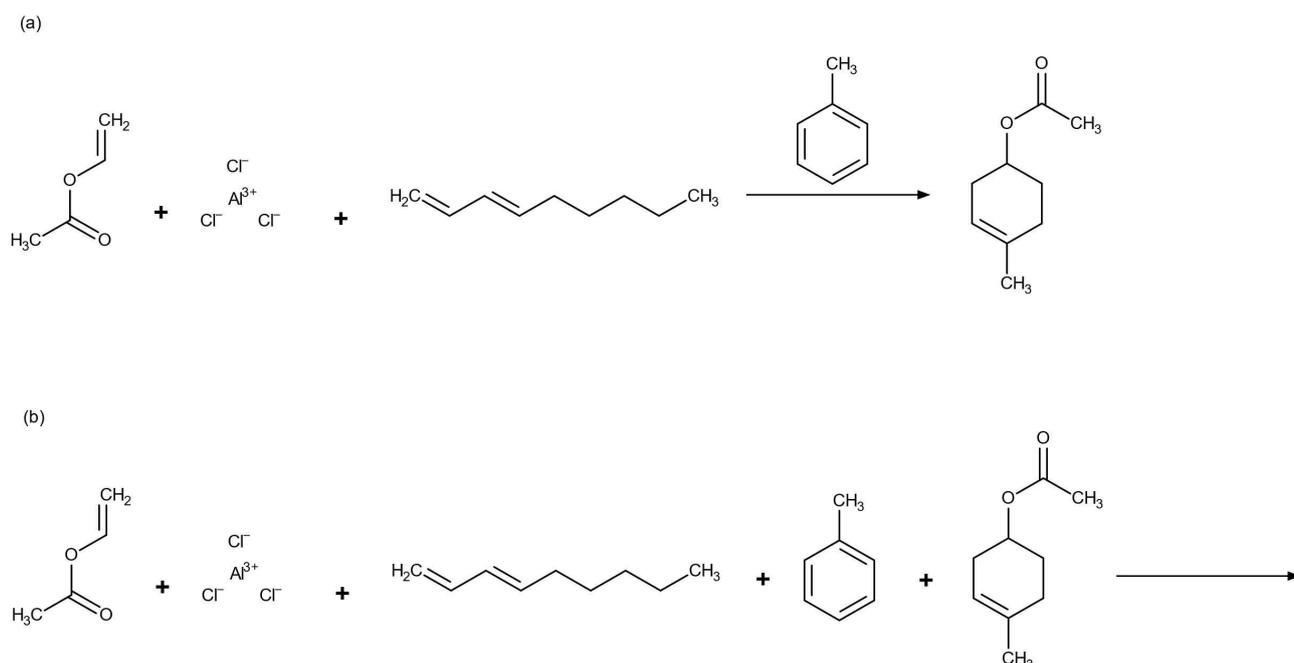


Figure 8. Reaction extracted from the patent US06413448B1 before (a) and after (b) standardization with ChemAxon Standardizer.^[49] One can see that ChemAxon merges all reactants, products and a reagent into one sole MOL block.

structure's standardization protocol, and, hence, we do not pay any attention to its chemical interpretation.

We believe that the dearomatization/re-aromatization procedure described above can also be recommended for data standardization in any database of individual compounds, including medchem databases.

2.2.2.3 Valence Checking

This option checks whether molecules comprising reactions possess any atoms with violated valence. If so, such reactions are to be discarded. Typical example of such reactions is shown in Figure 3.

2.2.2.4 Major Tautomer Generation

The generation of the major tautomer is needed because a given molecule can be recorded in a database in different tautomeric forms. This creates a problem with duplicates detection, substructural or similarity searches and molecular descriptors generation. The RDKit^[45] and ChemAxon^[49] tools used for identification of a major tautomer cannot be directly applied to chemical reactions: RDKit gives an error message, whereas ChemAxon corrupts the data by reassigning each compound to reactants, (see Figure 8). The only solution is to apply a tautomerization tool separately to each individual structure in the reaction equation.

Notice that RDKit, CDK, CGRTools, and ChemAxon Standardizer tools are not able to handle ring-chain tautomerism standardization. Although several rules for

ring-chain tautomers generation have earlier been proposed,^[60] they were not implemented into any standardization workflow. This may introduce an additional noise in data. For instance, cyclic and acyclic forms of glucose were recognized by name-to-structure converter applied to the words "D-glucose" and "glucose" as two different chemical structures upon the text mining of the USPTO.

2.2.2.5 Removal of Explicit Hydrogens

Once the major tautomer is selected, explicit hydrogens should be removed unless they are required in a particular cheminformatics task (e.g., defining stereochemistry or isotopically labelled molecules).

2.2.2.6 Discarding Reactions Containing Radicals

In USPTO, reactions containing radicals frequently result from erroneous recognition of chemical structures by text mining tool or bad reaction balancing (see Figure 9). On the other hand, Reaxys contains some 100 K reactions where radicals, like (2,2,6,6-Tetramethylpiperidin-1-yl)oxyl (a.k.a.-TEMPO), are used as a reactant (see an example in Figure 10). Unfortunately, there are no algorithms able to distinguish erroneously parsed reactions from truly radical reactions. Improvement of the text parsing and reaction balancing techniques may help to solve this problem. So far, we propose to discard any reaction containing radicals to avoid potential mistakes.

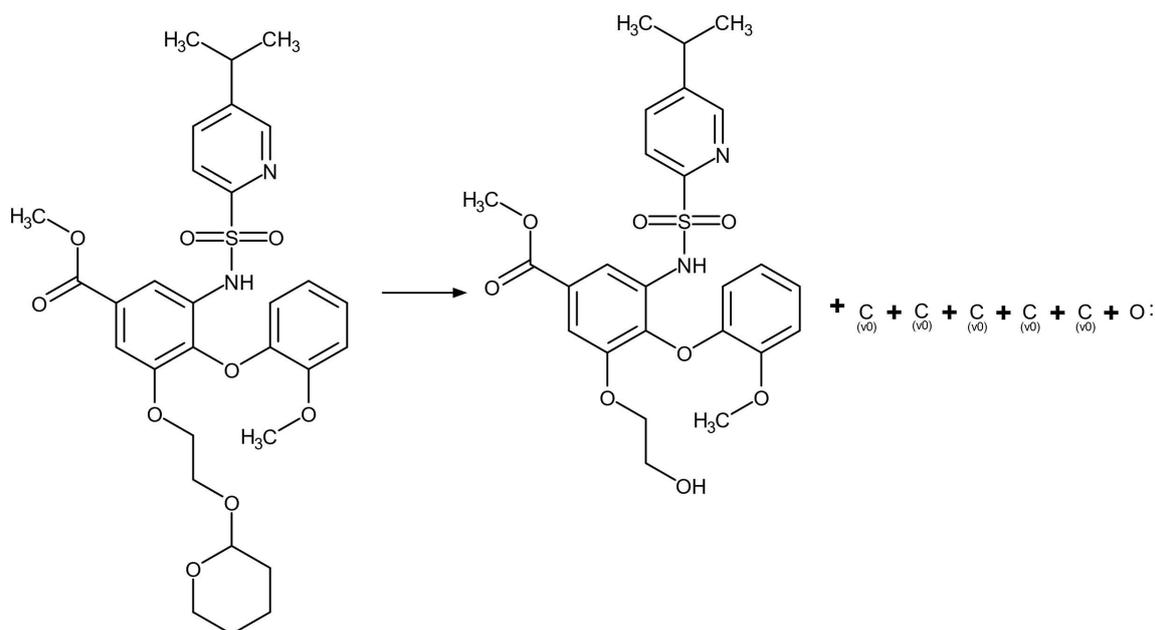


Figure 9. Example of a USPTO reaction balanced in reference^[27] by addition of five neutral carbon atoms and divalent radical of oxygen to the products side.

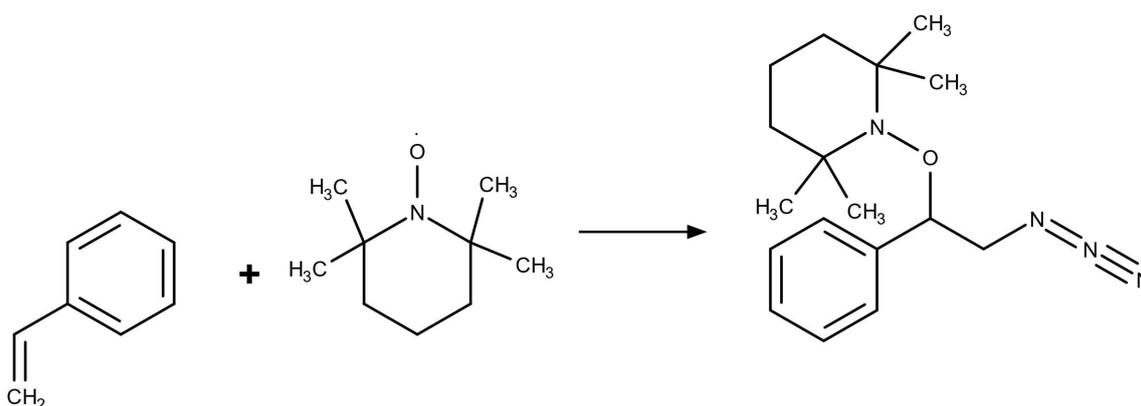


Figure 10. Example of a radical reaction from the Reaxys database with the participation of (2,2,6,6-Tetramethylpiperidin-1-yl)oxyl, a.k.a. TEMPO, reactant (Reaxys ID is 42346215).

2.2.2.7 Cleaning Isotopes

Isotope labels are used in organic chemistry in reaction mechanism elucidation studies.^[61] However, atom-to-atom mappers ignore this information and isotopes do not almost influence reaction characteristics (yield, reaction rate). Therefore, to avoid any confusion, we propose to clean up the isotope labels.

2.2.2.8 Split of Ions

This option helps to detect ions exchange reactions, which proceed without any transformation of covalent bonds. Formally, these reactions have no reaction centre. They are

out of the scope of the cheminformatics tasks listed in Table 1 and, therefore, could be discarded. However, their formal representation as entire salts (as it is done in USPTO and Reaxys) may significantly complicate their identification. For this reason, the ions in salts must be split, as this is shown in Figure 11. After splitting, reactions having the same molecular structures on the left- and right-hand sides of the reaction equation can be easily identified.

2.2.2.9 Clear Stereo

In most of the tasks, stereochemistry can be ignored. In this case, stereochemistry labels of bonds and atoms should be

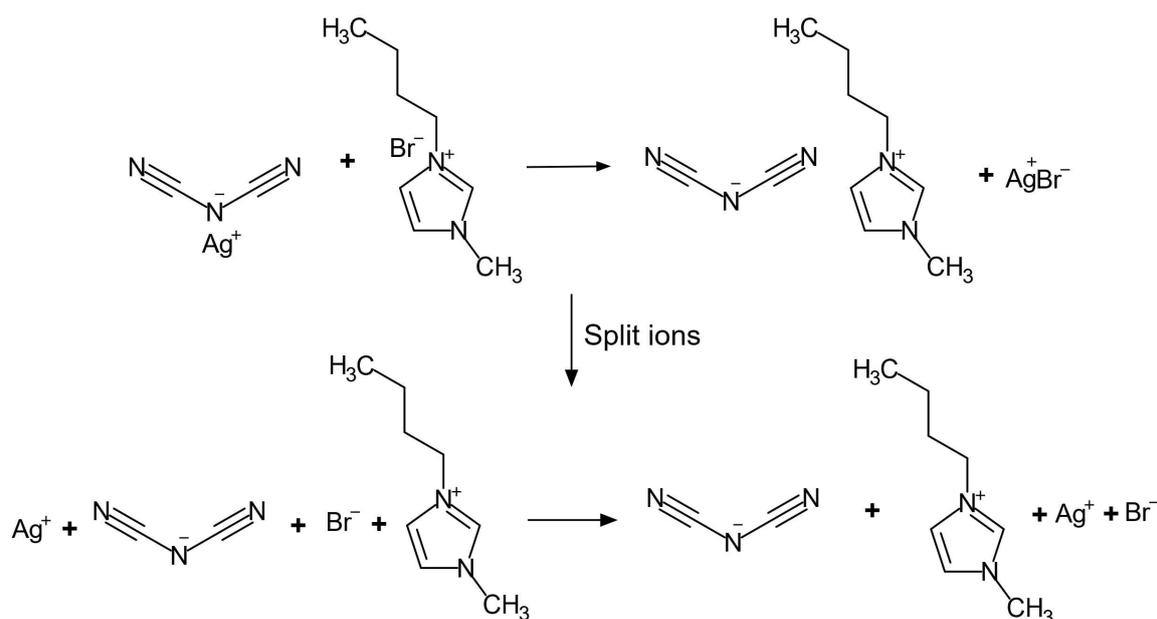


Figure 11. Example of ions exchange reaction (Reaxys 33598213) where the “Split ions” option of CGRTools is applied.

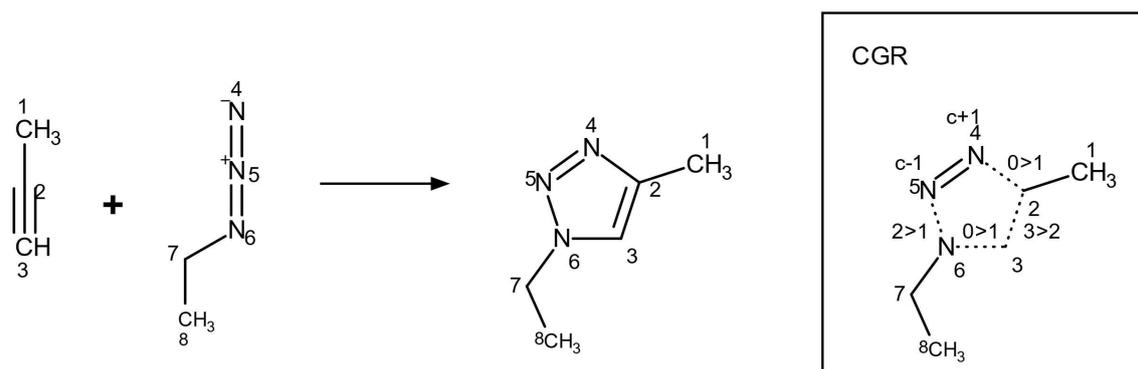


Figure 12. Example of a chemical reaction (left) and related Condensed Graphs of Reaction, CGR (right). The correspondence between the atom numbers of reactants and products in reaction equation is established by an atom-to-atom mapping procedure. Dotted lines in CGR represent dynamic bonds. Indices “c-1” and “c+1” means that the formal atomic charge is lowered and increased by 1, respectively. Indices “0>1”, “2>1”, “3>2” denote formed single bond, transformation of double bond to single and triple to double, correspondingly.

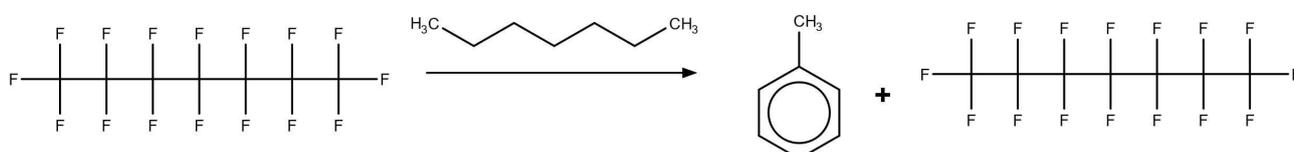
discarded. Thus, reactions involving stereoisomers of some compounds will be later considered as duplicates.

2.3 Transformations Curation

Transformation curation aims to perform the following operations: (i) atom-to-atom mapping, (ii) reaction role assignment, (iii) reaction duplicates removal, and (iv) reaction equation balancing. As a function of reaction equation completeness, operations 2 and 3 can be performed either using canonical reaction SMILES or Condensed Graphs of Reactions (CGR).^[62] A CGR represents an ensemble of reactants and products as a single

molecular graph resulting from superposition of related atoms in reactants and products (see Figure 12). A CGR is described by both conventional and “dynamic” atoms and bonds. Dynamic atoms represent changes in atomic features upon chemical transformations, e.g., the change of atomic charge. Dynamic bonds describe a chemical transformation: breaking or formation of chemical bonds, transformation of a single bond to a double bond, etc. A special CGR/SMILES encoding has recently been suggested using slightly modified SMILES rules.^[39] Unlike reaction SMILES that encode a set of individual reactants and products, CGR and related CGR/SMILES account for both structure of all molecular species (reactants and products) and information about reaction centre issued from atom-to-atom mapping

(a)



(b)

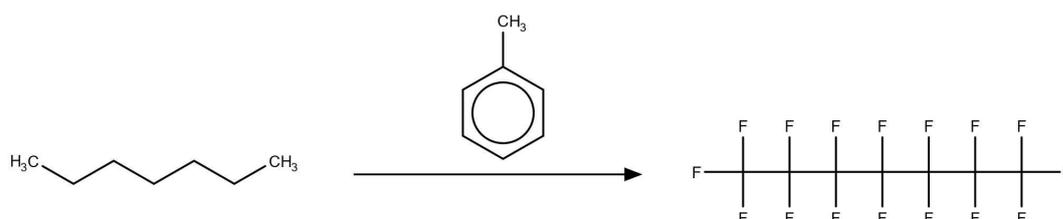


Figure 13. (a) A record from the USPTO database issued from wrongly parsed patent US4029552 A and (b) correct reaction equation. One can see that the text-mining tool erroneously recognized toluene as a product, n-heptane as a reagent and perfluoro-n-heptane as both product and reactant.

(Figure 12). CGR is useful in various reaction modelling tasks, e.g. calculation of descriptors, computing similarity between reactions, data visualization, etc.^[40,42,63–66]

SMILES-based operations don't need any mapping and, therefore, are recommended as a relatively fast procedure to perform reaction role assignment and reaction duplicates removal. Such partial transformation curation allows to decrease the number of records and to simplify some reaction equations because of re-assigning some reactants to reagents. In turn, this helps to reduce both a computation time for atom-to-atom mapping and the number of erroneously mapped reactions. At the next stage, transformation curation can be completed with the help of CGR-based operations.

2.3.1 Atom-to-atom Mapping

Atom-to-atom mapping (AAM) is a one-to-one correspondence between an atom in a reactant and an atom in a product.^[36] AAM is needed to identify the reaction centre and is required as a preliminary step in some data curation procedures. Many AAM algorithms and related software tools are known,^[36] but none of them is perfect. Benchmarking of popular AAM tools (ChemAxon Automapper,^[49] Indigo,^[47] NextMove,^[67] RDTool,^[51] and RXNMapper^[68]) and different approaches for mapping refinement and error identification are reported in our recent article.^[69] Notice that the data standardization process may have to account for the particularities of some tools. Thus, RDTools and RXNMapper have an *in-built* standardization procedure that splits ions in a salt encoded in one MOL block. Both programs also change the order and structural representation of molecules in a reaction equation. Therefore, in this

case, all steps of the standardization workflow should be repeated after completing the AAM. The Indigo does not affect the reaction equation; however, it can leave some atoms unmapped and may establish a many-to-one correspondence that causes a problem with CGR construction. Among the tested tools, only the ChemAxon Automapper produces one-to-one correspondence between reactant and product atoms, does not leave unmapped atoms and does not modify molecular representation.

2.3.2 Reaction Role Assignment

Reaction role assignment is one of the most delicate operation of the reaction data curation. The erroneous assignment of reactants to reagents or vice versa can be introduced by incorrect text parsing, data storage technique, or special rules in particular databases. For instance, in Reaxys, molecules that do not contribute carbon atom to product are considered as reagents. In the USPTO dataset, only compounds recognized by text mining as catalysts and solvents are identified as reagents, and the others are considered reactants. It is confusing to find the same reaction recorded differently in Reaxys and USPTO.

The presence of reagents can also slow down AAM assignments as mapping complexity increases with the number of atoms. Therefore, before the AAM step we suggest to identify reagents as molecules present in both reactants and products and remove them from the reaction equation. Sometimes, this procedure leads to reaction equations without any reactant or product; such reactions should be removed as suspicious. A typical example of an erroneous reaction equation extracted from a text file (patent) is demonstrated in Figure 13 and in Figure S2 in

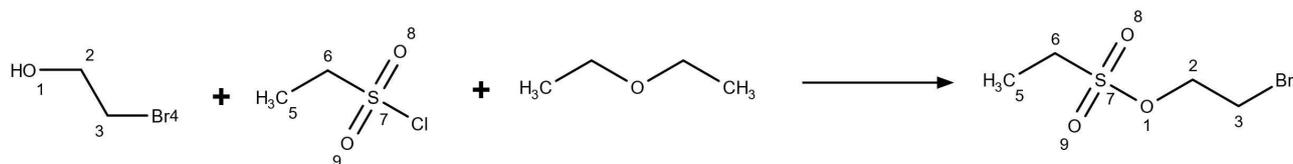


Figure 14. A USPTO reaction from US03930836 patent in which no atom in dimethyl-ether makes part a product and, therefore, this compound should be assigned to reagents.

the Supplementary Information. This patent describes an electrochemical reaction of n-heptane fluorination, where toluene is used as a reagent for the product (perfluoro-n-heptane) separation. However, the text mining tool erroneously recognized toluene as a product, n-heptane as a reagent and perfluoro-n-heptane both as product and reactant. In such a way, perfluoro-n-heptane looks as an unaffected species and, therefore, must be considered as a reagent. This leads to a reaction equation without any reactant, which, therefore, must be discarded. Among the correct reactions that can be removed at this step are ion exchange reactions and racemic resolutions (since information on stereochemistry is erased during the molecule standardization step). Generally, these reactions are out of scope of the cheminformatics tasks listed in Table 1.

In order to assign a reaction role (reactant, reagent, or product) automatically, one can follow the definition from Schneider et al.^[31] which considered compounds unaffected in the course of a reaction as reagents. Such compounds can be automatically detected in AAM as reactants that are not mapped into products. The example presented in Figure 14 shows that any atom of dimethyl-ether in the reactant side does not contribute to 2-bromoethyl ethanesulfonate formation and, therefore, it should be assigned to reagents.

2.3.3 Duplicates Detection

Duplicates detection and removal is a regular data cleaning procedure that aims to keep only unique entities. Two types of duplicates were considered: (i) transformations with the same list of reactants and products and (ii) transformations that contain a different number of reactants and products but corresponding to the same CGR. This is illustrated on the example of addition of furan-2-carbonyl to 2,3-dimethyl-1-(3-phenylpropyl)piperazine, three alternative reaction equations of which are given in Figure 15. Reaction (c) is balanced, whereas reactions (a) and (b) are not because they do not include the minor product HCl. Reactions (a) and (b) can be identified as duplicates using canonical reaction SMILES, whereas for identification of the duplicate (c) a CGR is required.

We recommend duplicate detection using canonical reaction SMILES strings before the AAM step. This helps to decrease the number of reactions considered in time

consuming mapping procedures. For some datasets, like USPTO and Pistachio, SMILES-based duplicates removal leads to cutting dataset almost threefold. As soon as AAM is established, reactions can be converted into CGR (and unique CGR/SMILES) which, in turn, can be used to identify remaining duplicates.

2.3.4 Reaction Balancing

Some cheminformatics tasks, like the preparation of reaction vectors^[70] require fully balanced reactions. The latter can be prepared with the help of a Condensed Graph of Reaction. As one may see from Figure 16, CGR allows to balance a given reaction upon its reversible decomposition. Application of some additional heuristics^[71] can be used to balance redox processes or for compounds that are used many times in a reaction, e.g. reagents for exhaustive alkylation. Alternatively, missing components in reactions can be identified using deep networks.^[72] It should be noted that balancing can only be performed if the list of reactants and/or products are complete enough to produce a realistic CGR. For instance, a hypothetical transformation with an ester as reactant and an alcohol as product cannot be balanced without supplementary information about the second reactant (water or amine).

3 Curation of Popular Reaction Databases

The simplified curation workflow described in Section 2 has been applied to clean three popular reaction databases: Reaxys (single step reactions only), USPTO and Pistachio. This simplified workflow did not include the AAM step and, hence, some other curation procedures (AAM-based reaction role assignment, reaction balancing, CGR-based duplicates removal) that require mapped reaction equations were not considered.

Statistics on reactions discarded at each step of the protocol are given in Table 2. Here, the difference in initial numbers of reactions and transformations in the USPTO and Pistachio databases is explained by the fact that some transformations proceed under different reaction conditions. Data in these databases are stored in the XML format, and, thus, each combination “transformation/reaction condition” represents an independent record. In Reaxys,

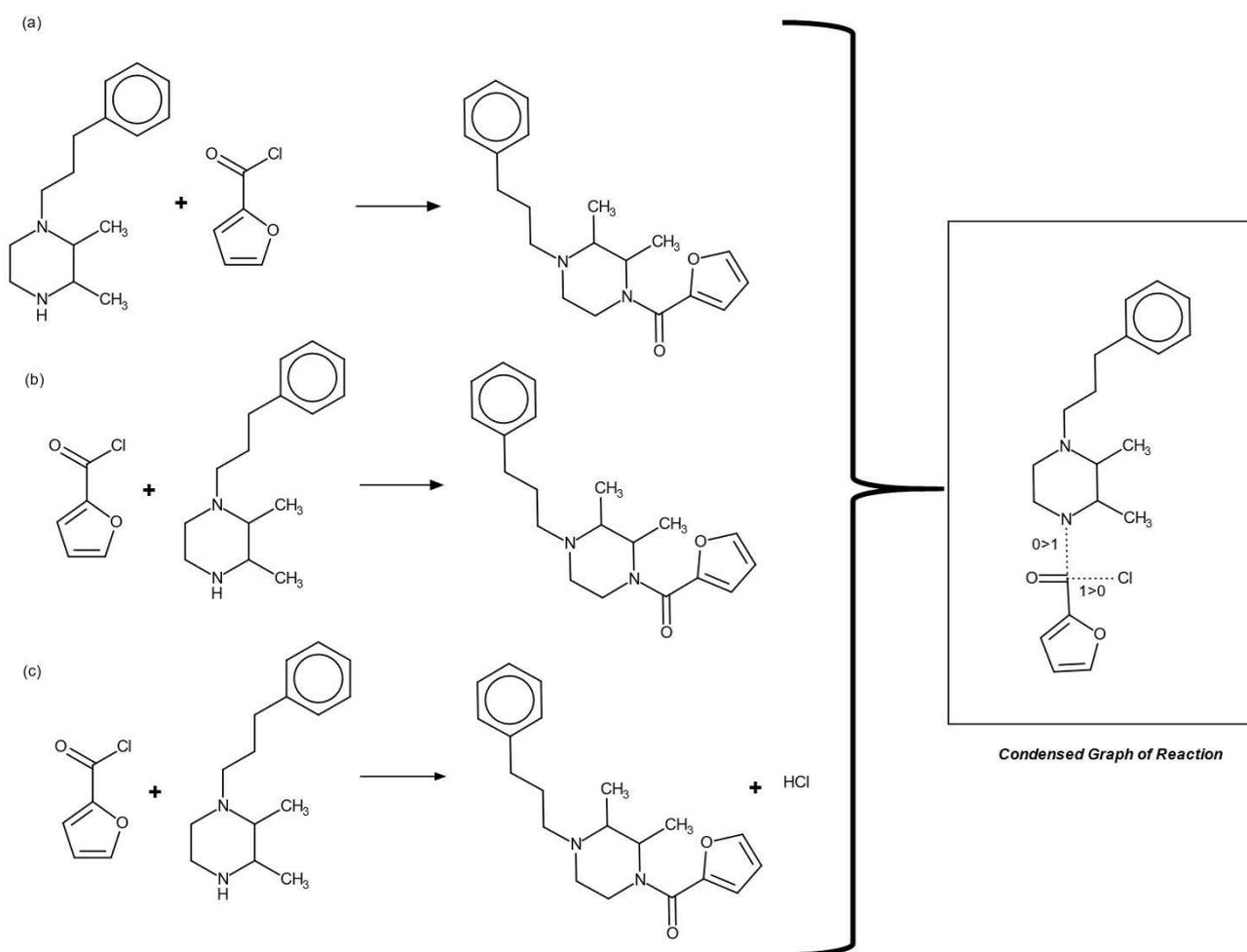


Figure 15. Reactions (a), (b) and (c) are duplicates because they correspond to the same Condensed Graph of Reaction. Reaction duplicates (a) and (b) can be identified using canonical reaction SMILES, whereas for identification of the duplicate (c) a CGR is required.

Table 2. Statistics of data cleaning in three popular databases.

	Reaxys ^a	USPTO	Pistachio
Initial reactions	27.34M ^b	3.75 M	9.7 M
Initial transformations	27.34 M	1.48 M	3.03 M
Unrecognised reaction SMILES	0 ^c	586	2.5 K
Transformations with invalid valences	2.42 M	55.4 K	148 K
Transformations containing isotopes ^d	52.3 K	45	713
Transformations with radicals	102.6 K	1.04 K	0
Transformations without reactants	6.25M	1.8 K	2.8 K
Transformations without products	941.3 K	10.1 K	23.3 K
Transformations without reactants and products	182.9 K	5.8 K	3.8 K
Transformations duplicates	1.65 M	85.3 K	0
Final number of transformations	15.79 M	1.32 M	2.84 M

^a Reaxys database includes the latest update done on 04 March 2021. ^b Reaxys contains 55.56 M reactions. The number here corresponds to single step reactions. ^c Reaxys data was stored in the RDF format, so, no problem with SMILES parsing was detected. ^d These reactions were kept but the isotopic labels were removed.

reactions are stored in the RDF format that already supports the storage of several reaction conditions for the same transformation. Hence, the number of reaction records in

Reaxys fully corresponds to the number of transformations. Surprisingly, 1.65 M duplicated transformations (i.e., some 6% of the initial one-step reactions dataset) were detected

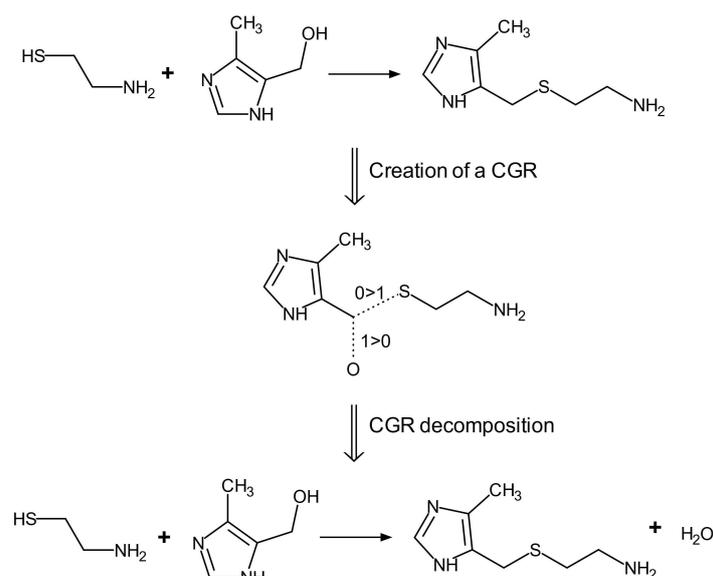


Figure 16. An example of a reaction transformed to CGR followed by the decomposition of the latter to fully balanced reaction. Here, the dashed connections are the dynamic bonds, where “0 > 1” means creation of a single bond, and “1 > 0” means cutting of a single bond.

in Reaxys. This number is significantly smaller for USPTO whereas no duplicates were found in Pistachio. Duplicates in Reaxys mostly result from discarding stereoisomers and isotope labels upon the data cleaning procedure.

A large portion of the transformations with invalid valences in all three databases largely stems from problems with the encoding of organometallic molecule chemical bonds in organometallic compounds (see discussion in Section 2.2 and example in Figure 3). Almost 7.2 M reactions with no reactants or products were found in Reaxys (26% of the initial one-step reactions dataset). The presence of such reactions in Reaxys can be explained by various reasons. For instance, some starting materials like a mixture of polymers or some resin can hardly be represented by a chemical structure. Besides that, reactants can be simply omitted even if they are mentioned in the original paper (see an example of such in Figure 17). Since all Reaxys records are manually prepared, some errors can be explained by a human factor.

Reactions without either reactants or products are much rarer in USPTO and Pistachio (some 0.3%) and are often caused by improper text mining. On the other hand, transformations without reactants or products are very rare given their absence in the initial reaction equation. In some 200 K transformations from Reaxys, 2 K from USPTO and 3.8 K from Pistachio (Table 2), reactants and products coincide, and, therefore, the curation protocol excludes them from reaction equation and moves to reagents. These transformations mostly originate from ionic exchange reactions, stereoisomer resolution reactions, or erroneous text recognition as shown above.

The cleaned Reaxys, USPTO and Pistachio contain 15.79 M, 1.32 M and 2.84 M transformations, respectively.

As the only publicly accessible dataset USPTO was mapped by RXNMapper and published in our GitHub repository [Laboratoire-de-Chemoinformatique/Reaction_Data_Cleaning](https://github.com/Laboratoire-de-Chemoinformatique/Reaction_Data_Cleaning).

4 Conclusions

Chemical reactions are complex processes that involve reactants, products, reagents. Their yield and thermodynamic or kinetic parameters depend on reaction conditions (temperature, pressure, solvent catalyst, etc). The protocol proposed in this work accounts for reaction complexity and consists of four steps related to the curation of (1) individual chemical structures (reactants, products and reagents), (2) chemical transformations, (3) reaction conditions and (4) endpoints. Only the two first steps were treated here.

Most curation steps of individual structures are similar to those suggested by Fourches et al.^[1] However, only few software tools (ChemAxon, CGRtools) can be applied to reactions directly, the others can be applied only to molecular structures extracted from the reaction equation. The curation of organometallic compounds poses a particular problem because it cannot be performed with existing tools. To our opinion, a special coordination bond type should be introduced to tackle this problem.

Transformation curation includes (i) atom-to-atom mapping, (ii) reaction role assignment, (iii) reaction duplicates removal, and (iv) reaction equation balancing. Technically, steps 2 and 3 can be performed using either reaction SMILES or Condensed Graph of Reaction (CGR). Notice that a CGR can be prepared only for atom-to-atom mapped

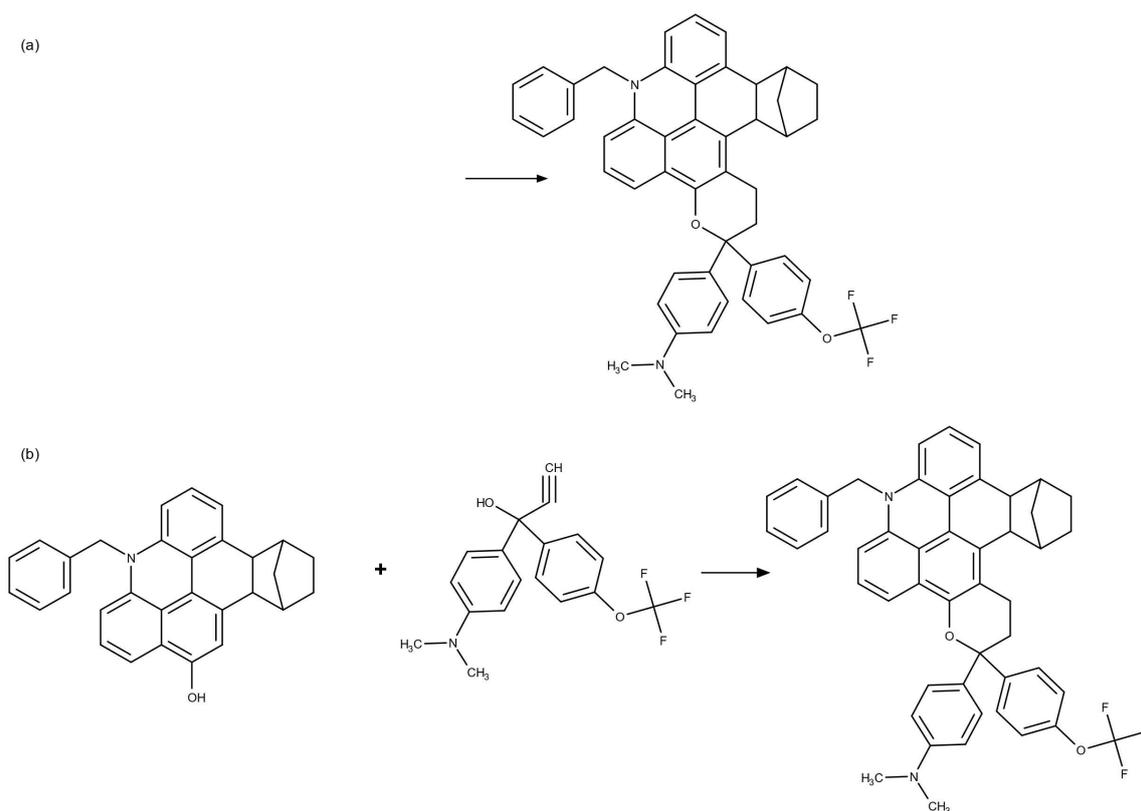


Figure 17. Example of a reaction from the Reaxys database (Reaxys ID is 23023671) with missed reactants. Here, (a) corresponds to the exact Reaxys representation of the reaction, (b) is extracted from EP1445257A1 (2004) patent.

reaction equations. Therefore, less time-consuming SMILES-based operations are recommended as the first step of transformation curation because it may (i) significantly reduce the size of a database due to removal of reaction duplicates, and (ii) simplify reaction equation by re-assigning some reactants to reagents.

The SMILES-based protocol described here has been applied to three popular reaction databases Reaxys, USPTO and Pistachio. The removal of reaction duplicates, transformations without reactants and/or products as well as records containing structures with invalid valences (mostly organometallics) led to the 2–3-fold size reductions.

Author Contribution Statement

The authors equally participated in this work.

Funding

This project was supported by the VLAIO HBC.2018.2287 MaDeSMart (Machine Design of Small Molecules by AI) grant. VA, DB, RN, and TM are grateful to Russian Science Foundation for support of CGRtools development and

upgrade required for the needs of reaction standardization (project No 19-73-10137). Reaxys data were made accessible to our research project via the Elsevier R&D Collaboration Network.

Conflict of Interest

None declared.

Data Availability Statement

A cleaned and atom-to-atom mapped (with RXNMapper) USPTO database containing 1.32 M transformations as well as the standardization protocol implementation in Python3 are freely available in our GitHub repository [Laboratoire-de-Chemoinformatique/Reaction_Data_Cleaning](https://github.com/Laboratoire-de-Chemoinformatique/Reaction_Data_Cleaning).

References

- [1] D. Fourches, E. Muratov, A. Tropsha, *J. Chem. Inf. Model.* **2010**, *50*, 1189–1204.
- [2] I. I. Baskin, T. I. Madzhidov, I. S. Antipin, A. A. Varnek, *Russ. Chem. Rev.* **2017**, *86*, 1127–1156.
- [3] O. Engkvist, P.-O. Norrby, N. Selmi, Y. Lam, Z. Peng, E. C. Sherer, W. Amberg, T. Erhard, L. A. Smyth, *Drug Discovery Today* **2018**, *23*, 1203–1218.
- [4] C. W. Coley, W. H. Green, K. F. Jensen, *Acc. Chem. Res.* **2018**, *51*, DOI 10.1021/acs.accounts.8b00087.
- [5] I. W. Davies, *Nature* **2019**, *570*, 175–181.
- [6] H. Gao, T. J. Struble, C. W. Coley, Y. Wang, W. H. Green, K. F. Jensen, *ACS Cent. Sci.* **2018**, *4*, 1465–1476.
- [7] A. Toniato, P. Schwaller, A. Cardinale, J. Geluykens, T. Laino, *arXiv:2102.01399* **2021**, 1–30.
- [8] J. Goodman, *J. Chem. Inf. Model.* **2009**, *49*, 2897–2898.
- [9] A. J. Lawson, J. Swienty-Busch, T. Géoui, D. Evans, **2014**, pp. 127–148.
- [10] D. M. Lowe, Extraction of Chemical Structures and Reactions from the Literature, University of Cambridge, **2012**.
- [11] M. H. S. S. Segler, M. Preuss, M. P. Waller, *Nature* **2018**, *555*, 604.
- [12] J. S. Schreck, C. W. Coley, K. J. M. Bishop, *ACS Cent. Sci.* **2019**, *5*, 970–981.
- [13] B. Liu, B. Ramsundar, P. Kawthekar, J. Shi, J. Gomes, Q. Luu Nguyen, S. Ho, J. Sloane, P. Wender, V. Pande, *ACS Cent. Sci.* **2017**, *3*, 1103–1113.
- [14] P. Schwaller, R. Petraglia, V. Zullo, V. H. Nair, R. A. Haeuselmann, R. Pisoni, C. Bekas, A. Iuliano, T. Laino, *Chem. Sci.* **2020**, *11*, 3316–3325.
- [15] C. W. Coley, L. Rogers, W. H. Green, K. F. Jensen, *ACS Cent. Sci.* **2017**, *3*, DOI 10.1021/acscentsci.7b00355.
- [16] C. W. Coley, D. A. Thomas, J. A. M. Lummiss, J. N. Jaworski, C. P. Breen, V. Schultz, T. Hart, J. S. Fishman, L. Rogers, H. Gao, *Science* **2019**, *365*, eaax1566.
- [17] J. M. Weber, P. Lió, A. A. Lapkin, *React. Chem. Eng.* **2019**, *4*, 1969–1981.
- [18] D. M. Lowe, Extraction of Chemical Structures and Reactions from the Literature, University of Cambridge, **2012**.
- [19] D. M. Lowe, “The NextMove Patent Reaction Dataset,” can be found under <https://depth-first.com/articles/2019/01/28/the-nextmove-patent-reaction-dataset/>, **2019**.
- [20] N. Schneider, D. M. Lowe, R. A. Sayle, M. A. Tarselli, G. A. Landrum, *J. Med. Chem.* **2016**, *59*, 4385–4402.
- [21] C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, K. F. Jensen, *ACS Cent. Sci.* **2017**, *3*, 434–443.
- [22] P. Schwaller, T. Gaudin, D. Lányi, C. Bekas, T. Laino, *Chem. Sci.* **2018**, *9*, 6091–6098.
- [23] W. W. Qian, N. T. Russell, C. L. W. Simons, Y. Luo, M. D. Burke, J. Peng, *ChemRxiv* **2020**, DOI 10.26434/chemrxiv.11659563.v1.
- [24] I. A. Watson, J. Wang, C. A. Nicolaou, *J. Cheminf.* **2019**, *11*, 1.
- [25] G. M. Ghiandoni, M. J. Bodkin, B. Chen, D. Hristozov, J. E. A. Wallace, J. Webster, V. J. Gillet, *J. Chem. Inf. Model.* **2019**, *59*, 4167–4187.
- [26] N. Schneider, D. M. Lowe, R. A. Sayle, G. A. Landrum, *J. Chem. Inf. Model.* **2015**, *55*, 39–53.
- [27] W. Jaworski, S. Szymkuć, B. Mikulak-Klucznik, K. Piecuch, T. Klucznik, M. Kaźmierowski, J. Rydzewski, A. Gambin, B. A. Grzybowski, *Nat. Commun.* **2019**, *10*, 1434.
- [28] P. Schwaller, A. C. Vaucher, T. Laino, J.-L. Reymond, *Mach. Learn.: Sci. Technol.* **2021**, *2*, 015016.
- [29] P. Schwaller, A. C. Vaucher, T. Laino, J.-L. Reymond, *ChemRxiv* **2020**, DOI 10.26434/chemrxiv.13286741.v1.
- [30] D. Probst, P. Schwaller, J.-L. Reymond, *ChemRxiv* **2021**, DOI 10.33774/chemrxiv-2021-mc870.
- [31] N. Schneider, N. Stiefl, G. A. Landrum, *J. Chem. Inf. Model.* **2016**, *56*, 2336–2346.
- [32] W. A. Warr, *Mol. Inf.* **2014**, *33*, 469–476.
- [33] C. W. Coley, W. H. Green, K. F. Jensen, *J. Chem. Inf. Model.* **2019**, *59*, 2529–2537.
- [34] D. Fourches, E. Muratov, A. Tropsha, *J. Chem. Inf. Model.* **2016**, *56*, 1243–1252.
- [35] A. D. McNaught, A. Wilkinson, *Compendium of Chemical Terminology*, Blackwell Science Oxford, **1997**.
- [36] W. L. Chen, D. Z. Chen, K. T. Taylor, *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2013**, *3*, 560–593.
- [37] P. P. Plehiers, G. B. Marin, C. V. Stevens, K. M. Van Geem, *J. Cheminf.* **2018**, *10*, 11.
- [38] J. L. Baylon, N. A. Cilfone, J. R. Gulcher, T. W. Chittenden, *J. Chem. Inf. Model.* **2019**, *59*, 673–688.
- [39] W. Bort, I. I. Baskin, T. Gimadiev, A. Mukanov, R. Nugmanov, P. Sidorov, G. Marcou, D. Horvath, O. Klimchuk, T. Madzhidov, A. Varnek, *Sci. Rep.* **2021**, *11*, 3178.
- [40] A. I. Lin, T. I. Madzhidov, O. Klimchuk, R. I. Nugmanov, I. S. Antipin, A. Varnek, *J. Chem. Inf. Model.* **2016**, *56*, 2140–2148.
- [41] T. Gimadiev, T. Madzhidov, I. Tetko, R. Nugmanov, I. Casciuc, O. Klimchuk, A. Bodrov, P. Polishchuk, I. Antipin, A. Varnek, *Mol. Inf.* **2018**, *0*, DOI 10.1002/minf.201800104.
- [42] T. R. Gimadiev, T. I. Madzhidov, R. I. Nugmanov, I. I. Baskin, I. S. Antipin, A. Varnek, *J. Comput.-Aided Mol. Des.* **2018**, *32*, 401–414.
- [43] G. Skoraczyński, P. Dittwald, B. Miasojedow, S. Szymkuć, E. P. Gajewska, B. A. Grzybowski, A. Gambin, *Sci. Rep.* **2017**, *7*, 3582.
- [44] D. T. Ahneman, J. G. Estrada, S. Lin, S. D. Dreher, A. G. Doyle, *Science* **2018**, *360*, 186–190.
- [45] “RDKit: Open-source cheminformatics,” **2019**.
- [46] E. L. Willighagen, J. W. Mayfield, J. Alvarsson, A. Berg, L. Carlsson, N. Jeliakova, S. Kuhn, T. Pluskal, M. Rojas-Chertó, O. Spjuth, G. Torrance, C. T. Evelo, R. Guha, C. Steinbeck, *J. Cheminf.* **2017**, *9*, 33.
- [47] A. Savelev, I. Puzanov, V. Samoilov, V. Karnaukhov, Indigo Toolkit, **2019**, <https://lifescience.opensource.epam.com/indigo/index.html>.
- [48] N. M. O’Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, G. R. Hutchison, *J. Cheminf.* **2011**, *3*, 33.
- [49] JChem ChemAxon, <https://chemaxon.com/products/jchem-engines>.
- [50] R. I. Nugmanov, R. N. Mukhametgaleev, T. Akhmetshin, T. R. Gimadiev, V. A. Afonina, T. I. Madzhidov, A. Varnek, *J. Chem. Inf. Model.* **2019**, *59*, 2516–2521.
- [51] S. A. Rahman, G. Torrance, L. Baldacci, S. Martinez Cuesta, F. Fenninger, N. Gopal, S. Choudhary, J. W. May, G. L. Holliday, C. Steinbeck, J. M. Thornton, *Bioinformatics* **2016**, *32*, 2065–2066.
- [52] P. Schwaller, B. Hoover, J.-L. Reymond, H. Strobelt, T. Laino, *Sci. Adv.* **2021**, *7*, eabe4166.
- [53] T. Moock, J. Nourse, D. Grier, W. Hounshell, in: *Chem. Struct.* (Ed.: W. Warr), Springer-Verlag, Berlin, **1988**, pp. 303–313.
- [54] H. Kraut, J. Eiblmaier, G. Grethe, P. Löw, H. Matuszczyk, H. Saller, *J. Chem. Inf. Model.* **2013**, *53*, 2884–2895.
- [55] D. Fooshee, A. Andronico, P. Baldi, *J. Chem. Inf. Model.* **2013**, *53*, 2812–2819.
- [56] E. L. First, C. E. Gounaris, C. A. Floudas, *J. Chem. Inf. Model.* **2012**, *52*, 84–92.
- [57] “Extended SMILES, SMARTS,” can be found under <https://chemaxon.com/marvin-archive/latest/help/formats/cxsmiles-doc.html>, **n.d.**
- [58] T. E. Oliphant, *Comput. Sci. Eng.* **2007**, *9*, 10–20.

- [59] K. Maruyama, T. Katagiri, *J. Phys. Org. Chem.* **1989**, *2*, 205–213.
- [60] L. Guasch, M. Sitzmann, M. C. Nicklaus, *J. Chem. Inf. Model.* **2014**, *54*, 2423–2432.
- [61] D. A. Semenov, J. D. Roberts, *J. Chem. Educ.* **1956**, *33*, 2.
- [62] F. Hoonakker, N. Lachiche, A. Varnek, A. Wagner, *Int. J. Artif. Intell. Tools* **2011**, *20*, 253–270.
- [63] T. I. Madzhidov, T. R. Gimadiev, D. A. Malakhova, R. I. Nugmanov, I. I. Baskin, I. S. Antipin, A. A. Varnek, *J. Struct. Chem.* **2017**, *58*, 650–656.
- [64] W. Bort, I. I. Baskin, T. Gimadiev, A. Mukanov, R. Nugmanov, P. Sidorov, G. Marcou, D. Horvath, O. Klimchuk, T. Madzhidov, A. Varnek, *Sci. Rep.* **2021**, *11*, 3178.
- [65] T. I. Madzhidov, P. G. Polishchuk, R. I. Nugmanov, A. V. Bodrov, A. I. Lin, I. I. Baskin, A. A. Varnek, I. S. Antipin, *Russ. J. Org. Chem.* **2014**, *50*, 459–463.
- [66] R. I. Nugmanov, T. I. Madzhidov, G. R. Khaliullina, I. I. Baskin, I. S. Antipin, A. A. Varnek, *J. Struct. Chem.* **2014**, *55*, 1026–1032.
- [67] NextMove Software, www.nextmovesoftware.com, **2020**.
- [68] P. Schwaller, B. Hoover, J.-L. Reymond, H. Strobelt, T. Laino, *Sci. Adv.* **2021**, *7*, eabe4166.
- [69] A. Lin, N. Dyubankova, T. I. Madzhidov, R. Nugmanov, A. Rakhimbekova, Z. Ibragimova, T. Akhmetshin, T. Gimadiev, R. Suleymanov, J. Verhoeven, J. K. Wegner, H. Ceulemans, A. Varnek, *ChemRxiv* **2020**, DOI 10.26434/chemrxiv.13012679.v1.
- [70] H. Patel, M. J. Bodkin, B. Chen, V. J. Gillet, *J. Chem. Inf. Model.* **2009**, *49*, 1163–1184.
- [71] R. I. Nugmanov, T. I. Madzhidov, I. S. Antipin, A. A. Varnek, *Uch. Zap. Kazan. Univ. Seriya Estestv. Nauk* **2018**, *160*, 32–39.
- [72] A. C. Vaucher, P. Schwaller, T. Laino, *ChemRxiv* **2020**, DOI 10.26434/chemrxiv.13273310.v1.

Received: May 4, 2021

Accepted: August 13, 2021

Published online on August 24, 2021

Supporting information

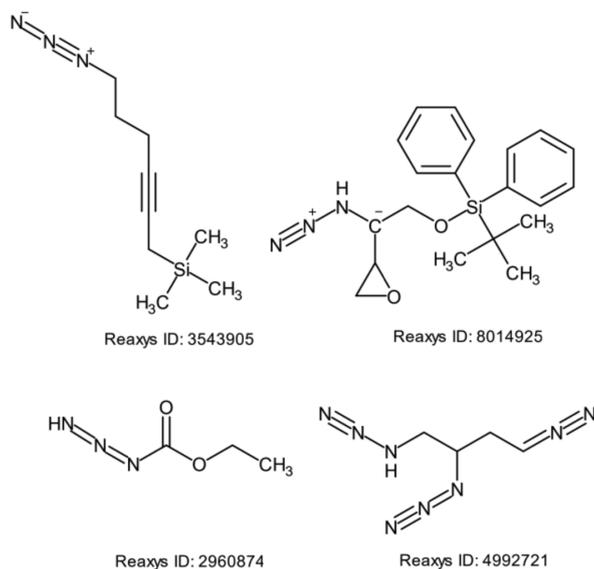


Fig. 4.2 Molecules with a non-standardized azide group extracted from the Reaxys database. In the given molecules, four different forms are used to present the azide functional group, whereas its standard (according to CGRTool) representation is $R - N = [N^+] = [N^-]$.

n-Heptane 1 is subject to electrochemical fluorination 2. The electrochemical cell product effluent 3 preferably is water washed 4 with water 5 to remove water-soluble acidic components. The water washings are discarded 6 or otherwise disposed of. The water-washed product stream 7 is fractionated to produce a first overhead fraction 9 boiling from about 65° C. to about 75° C., a second overhead fraction 11 boiling from about 75° to 100° C., and a bottoms product 13 of material boiling over about 100° C. The bottoms product 13 is discarded. The higher boiling fraction 11 can be recycled 12 to electrochemical fluorination 2. The first overhead fraction 9 is toluene washed 14 employing toluene 15. A resulting stream of toluene and n-heptane 16 is separated 17 to produce recycle toluene 18 and recycle n-heptane 19. The washed product stream 21 is distilled 22 with excess toluene 23, producing an azeotrope overhead 24 of toluene/perfluoro-n-heptane, and a kettle product of excess toluene and impurities. The excess toluene in stream 25 can be separated (not shown), such as by distillation for recycle, and residual kettle product of partially fluorinated products can be returned (not shown) for electrochemical fluorination. The azeotrope 24 is separated to obtain toluene 27, which can be recycled as desired, and substantially pure perfluoro-n-heptane 28.

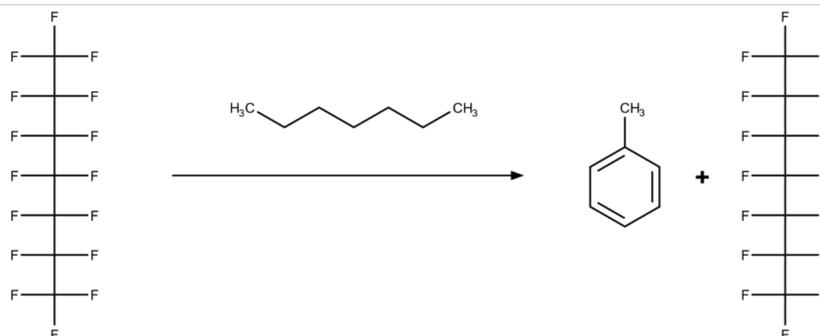


Fig. 4.3 An example of a wrongly parsed reaction from the USPTO database discarded by the standardization protocol as a reaction with no reactants. The reaction was extracted from the patent US4029552A, and the text of the patent is given in the figure.

4.1.1 Summary for the section 4.1

In this section, we discussed techniques for the curation of reaction data, formulated its main steps, and implemented the reaction standardisation protocol. Most of the standardisation rules for reactions follow those previously formulated for molecules. A unique step in the standardisation of reactions is atom-to-atom mapping (AAM), which is crucial in the curation of reactions. It defines the mechanism of the reaction and subsequent steps, such as reaction role assignment and reaction balancing. The performance of AAM algorithms can be sensitive to the representation of chemical structures, and the number of mistakes they make can be reduced through the proper standardisation of molecules and ions.

The developed standardisation protocol was tested on three reaction databases: Reaxys, Pistachio, and USPTO. The analysis showed that regardless of the type of database (hand-crafted or automatically collected), they contained a large number of erroneous reactions or duplicates, resulting in a significant reduction in the size of the databases after curation (42% for the hand-crafted database and 65-70% for the automatically collected ones). The USPTO dataset is the only one that is freely accessible and is provided in our GitHub repository in its standardised form. This dataset will also be used in the following sections as the main reaction database.

4.2 Retrosynthetic planning

Retrosynthetic analysis is a problem-solving technique in chemical synthesis design that was first proposed at the beginning of the 1960s by Elias James Corey, who later received a Nobel Prize for this work. This technique involves a “molecular deconstruction” process converting a synthetic target molecule into simpler precursor structures (retrons) or pseudo-molecular structures (synthons) through a retrosynthetic transformation (retro-rule). Along with Wipke, Corey also developed the first computer-aided organic synthesis (CAOS) software[104], renamed further into “Logic and Heuristics Applied to Synthetic Analysis” (LHASA). In this software, the authors defined key components of retrosynthesis planning algorithms that are still present in modern tools. These contributions can be divided into three main categories: retrosynthetic transformations, a retrosynthetic tree, and a search algorithm for navigating this tree.

The retrosynthetic transformations (Figure 4.4) are characterised by one or more substructures (located to the left of the retrosynthetic arrow) that must be present for the transformation to proceed, as well as descriptions of the structural changes that occur during the transformation. These substructures are comprised of atoms of the reaction centre (atoms around transformed or broken bonds during the reaction) and may also include neighbouring alpha atoms (the first environment), beta atoms (the second environment), and so on. In addition to these structural features, retrosynthetic transformations may also include functional groups that influence the reaction outcome and physicochemical conditions such as solvents and temperature range.

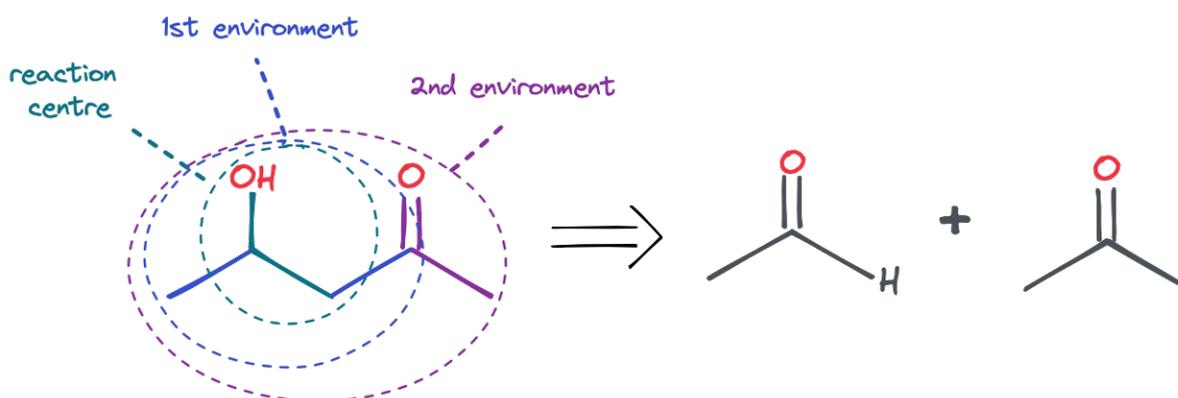


Fig. 4.4 Example of retrosynthetic transformation included in LHASA program that encodes aldol addition.

A retrosynthetic tree is a graphical representation of the various synthetic pathways that can be taken to synthesise a target compound from a set of precursors (Figure 4.5). The tree begins with the synthetic target at the root and is constructed by applying retrosynthetic transformations

to derive precursor compounds successively. The node in the tree represents all the structures obtained through these transformations, with the edges indicating the retrosynthetic reactions that were performed. When a node includes an intermediate structure that must be synthesised, it is chosen as the node's representative structure. In cases where multiple unavailable retrons are present, one is selected randomly to represent the node.

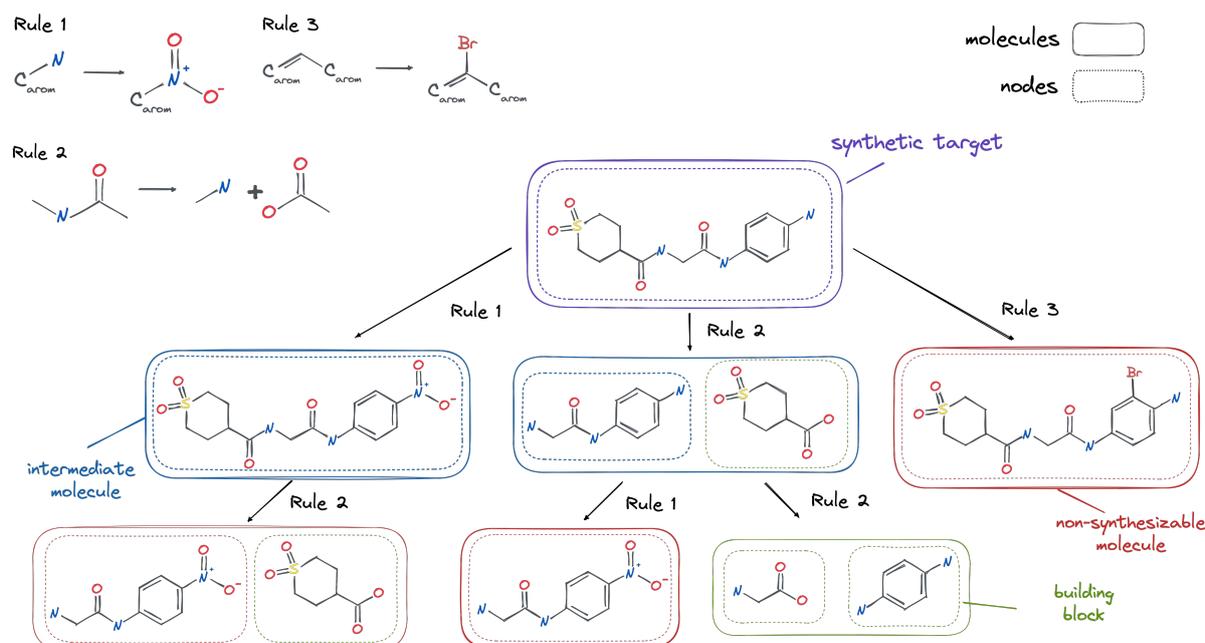


Fig. 4.5 Example of a retrosynthetic tree.

The retrosynthetic search is an iterative process, where after each iteration new nodes of the tree are created. One iteration consists of three key steps: selection of a node, choice and application of retro-rules, and evaluation of the resulting retrons. During selection, a user or algorithm selects a leaf node, which is a node from the last levels of the tree. Once the node is chosen, another module selects retro-rules that meet a predetermined goal, such as simplification of a structure by splitting rings or reducing the molecular size by disconnection of chains. The selected retro-rules are then applied to the representative structure, and the precursors are obtained. The evaluation module assigns them scores based on factors such as validity of chemical structure, availability in the building blocks set, and other user-defined criteria. These scores are used in the next iterations of the tree in the selection stage. The search terminates either when the desired number of iterations has been reached, when a time limit is exceeded, or when the user chooses to stop the search.

The selection of retro-rules and evaluation of retrons during the search may be conducted using heuristics or machine learning techniques. As such, in the following discussion, we

classify retrosynthetic planning tools into those that utilise only heuristics scoring or those that employ machine learning methods.

Heuristic-based approaches

The LHASA program represented a landmark in computer-aided organic synthesis and had a long-lasting impact on future developments in this area. In this tool, the retrosynthetic rules were hand-crafted by human experts. The selection and evaluation steps were carried out by a user, while the transformation step was scored by a “perception” module that analysed the chemical structures for synthetically significant features (for example, rings, ring junctures, functional groups, relations between functional groups and symmetry). If the results were unsatisfactory, meaning that building blocks were not found, the user could return to any existing precursor and continue the search.

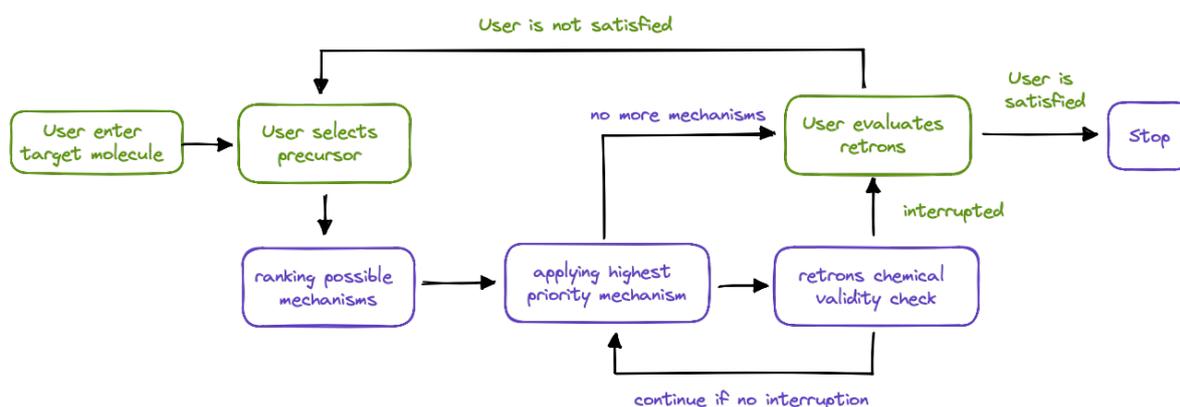


Fig. 4.6 Protocol used in Logic and Heuristics Applied to Synthetic Analysis (LHASA) program.

LHASA can be seen as an interactive retrosynthetic program, as human participates in the whole search process. The next generation (so-called “non-interactive” programs) used heuristics-based scoring functions in the evaluation step to estimate the retrosynthetic feasibility of the structure, which is a possibility to find a pathway to desired building blocks. This approach was first proposed in SYNCHEM[112]. In this program, during the selection stage, the previously obtained scores of chosen transformations and evaluated retrons guided the tree’s growth. Thus, there was no need for the active participation of human experts, which decreased the search time.

The next generation of tools, instead of using retrosynthetic transformations, used reaction mechanisms that are applicable to a large number of reactions. For example, the IGOR[113] program utilised “bond-electron” matrices for the representation of possible reactants. The rules were represented as reaction matrices that encoded changes in bond orders and free valence

electrons connected to atoms during a reaction. Another approach was implemented in the SYNGEN[114] software, where all reactions were described by unit reactions, which encoded the replacement of one kind of bond for another on each involved carbon. These approaches were not limited by known transformations, which allowed them to find new types of reactions during retrosynthetic analysis.

Retrosynthetic transformation-based approaches have demonstrated efficacy in identifying suitable synthetic pathways, yet they did not guarantee the feasibility of individual steps in the synthetic plan, including the identification of potential by-products. This task, known as reaction prediction, is crucial for the successful execution of a synthetic plan. This issue was addressed in the program CAMEO[115], which used a perception module that was able to identify reactive parts in structures based on calculated pK_a . In another tool WODCA[116], the EROS reaction prediction program[117] was employed, which used reaction mechanisms to predict the ease of the forward transformations implied by the retrosynthesis search.

One of the leading retrosynthetic platforms among heuristic-based retrosynthesis tools is Synthia[106] (previously Chematica), which boasts a collection of over 100K hard-coded retro transformations that encode the regioselectivity of reactions. This extensive collection of rules was compiled over the course of two decades, beginning in the early 2000s, and is currently the most comprehensive available. Synthia employs heuristics similar to those used in LHASA and other retrosynthetic approaches and has incorporated scoring functions (e.g., the overall monetary cost, the popularity of substrates involved, complexity estimation of obtained precursors, avoidance of toxic/hazardous intermediates) and a beam-search-inspired priority queue algorithm[106, 118] to improve search performance.

However, the approach of Synthia has been shown to be limited in its scalability.[119] In particular, the addition of new retro-rules requires careful checking to ensure they do not contradict existing ones. These issues were addressed in recent developments that automatically collect retrosynthetic transformations from existing reaction databases. One example is ARChem Route Designer[120], which described a protocol for autonomous retrosynthetic transformation reactions from literature sources. It was tested on the Beilstein reaction database (now known as the Reaxys database[107]) and produced over 93K retrosynthetic rules in 8 days. This is in contrast to the Synthia approach, which required the authors to manually collect a similar number of rules over a period of 20 years. Therefore, the automated extraction of retrosynthetic rules is a promising approach that can accelerate the development of retrosynthesis platforms. However, the main limitation of this approach is the dependence on the quality of reaction data. Furthermore, the reactions in databases are collected based on historical data, and it is possible that some useful transformations for retrosynthesis may be missing.

Machine learning-based approaches

Only a few early retrosynthesis tools used machine learning approaches for reaction prediction. Specifically, regression models were used to estimate the feasibility of reactions within a given reaction class.[121, 122] One of the limiting factors at the time was the availability of reaction data to train these models.

The advancement of reaction databases and deep learning theory has enabled the use of machine learning models to improve the efficiency of retrosynthesis search. A recent study by Schwaller et al.[123] employed a combination of generative deep-learning models and a hypergraph exploration strategy. Instead of utilising automatically extracted templates, the authors proposed training two transformer models based on SMILES representation. One model suggests precursor molecules given a product molecule, while the other scores chemical reactions given precursor-product combinations. These models were trained on the USPTO and Pistachio databases and subsequently coupled with a hyper-graph beam search. The resulting retrosynthesis algorithm was then implemented in the RoboRXN platform[124], enabling automated retrosynthesis using robotic systems.

Recent advancements in reinforcement learning with neural networks have had a significant impact on modern retrosynthesis tools. Researchers at the DeepMind laboratory have successfully integrated the Monte-Carlo tree search (MCTS) algorithm with deep learning models scoring and ranking models, resulting in superhuman performance in a wide range of strategic games, such as chess, go, and shogi.[125] This breakthrough inspired the pioneering work of Segler et al.[21], in which authors presented the first MCTS-based retrosynthesis search tool. Following this idea, numerous variants of MCTS-based retrosynthesis have emerged.

Monte-Carlo tree search is a search method optimised for decision-making problems. This search is an iterative process, where at each iteration, the algorithm creates a new node of a search tree until the maximum number of iterations is exceeded. One iteration consists of 4 steps: selection, expansion, evaluation and update (Figure 4.7).

Selection In the selection stage, the algorithm identifies the most promising nodes by evaluating their scores and the number of visits. The process starts at the root node, where the Upper Confidence Bounds for Trees (UCT) scores of the child nodes are calculated. The child node with the highest score is subsequently selected, and the selection process continues until a leaf node (node without children) is reached. The basic formula of UCT is given in equation 3.1

$$UCT = Q_{prev} + c * \frac{\sqrt{N_{parent}}}{1 + N} \quad (4.1)$$

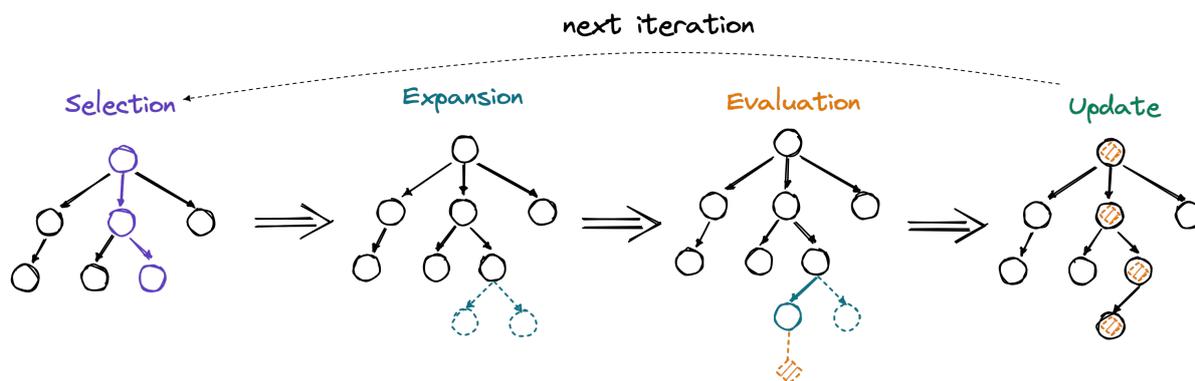


Fig. 4.7 The main stages of Monte-Carlo tree search. Here, on each iteration, one node is created and added to the tree.

where the Q_{prev} – is a score of the node obtained in the previous iterations, c is a non-negative hyperparameter, N – is the number of visits in the current node, and respectively N_{parent} is the number of visits in the parent node. The first term of the equation Q_{prev} reflects the reward of selecting that particular node, and it facilitates the exploitation of the most promising ones that lead to building blocks. Thus, the higher its value, the greater the chance that the node will be chosen in subsequent iterations of the tree search. The second term of the equation is employed to balance exploitation with exploration by introducing a score that depends on the number of visits to the node. This score is highest for nodes that have not yet been visited, thus encouraging the search to explore new solutions. The value of Q is obtained in the following stages (see evaluation and update steps). One of the modifications of this function is used in the Thakkar et al.[3]:

$$UCT = \frac{Q_{prev}}{N} + c * \sqrt{2 * \frac{\ln(N_{parent})}{1 + N}} \quad (4.2)$$

Another variant of UCT is called the probabilistic upper confidence tree (PUCT)[126]. PUCT differs from classic UCT in that it employs predictions from the policy network multiplied with exploration term. In this way, the most promising transformations are explored first, which is especially useful with reliable policy:

$$PUCT = Q_{prev} + p * c * \frac{\sqrt{N_{parent}}}{1 + N} \quad (4.3)$$

where p is a probability of retro-rule that leads from the parent node to the current one. The PUCT used in the Segler et al.[21] with a slight modification:

$$PUCT = \frac{Q_{prev}}{N} + p * c * \frac{\sqrt{N_{parent}}}{1 + N} \quad (4.4)$$

The coefficient c can be specified at the outset (static) or varied during the search process (dynamic). In the case of MCTS-based retrosynthesis, Wang et al.[2] compared different versions of UCT with static and dynamic c , PUCT, and naive selection (which relies solely on the scores of the node). The authors found that UCT with dynamic c performs better when a pre-trained policy network is used.

Expansion In the expansion phase of the MCTS process, the policy function is responsible for identifying the most promising retrosynthetic transformation given a particular molecule represented by a node in the tree. Modern NN-based MCTS algorithms approximate the policy function using a model that assigns probabilities to each retrosynthetic rule or directly transforms the molecule into its possible precursors. These models can be classified based on their input type (such as molecular fingerprints or graphs) and their training objectives.

One common choice for the policy function is a supervised neural network trained on Morgan fingerprints[21, 105, 2, 3, 127] or molecular graphs[128] to rank retro-rules. This approach involves training the network to perform multi-class classification. Thus, for the given reaction we extract retrosynthetic transformation and the product. For this product this retro-rule (which is extracted from the same reaction) is assigned as positive class, where all other retrosynthetic transformations extracted from the reaction database are assigned negative class (as illustrated in Figure 4.8).

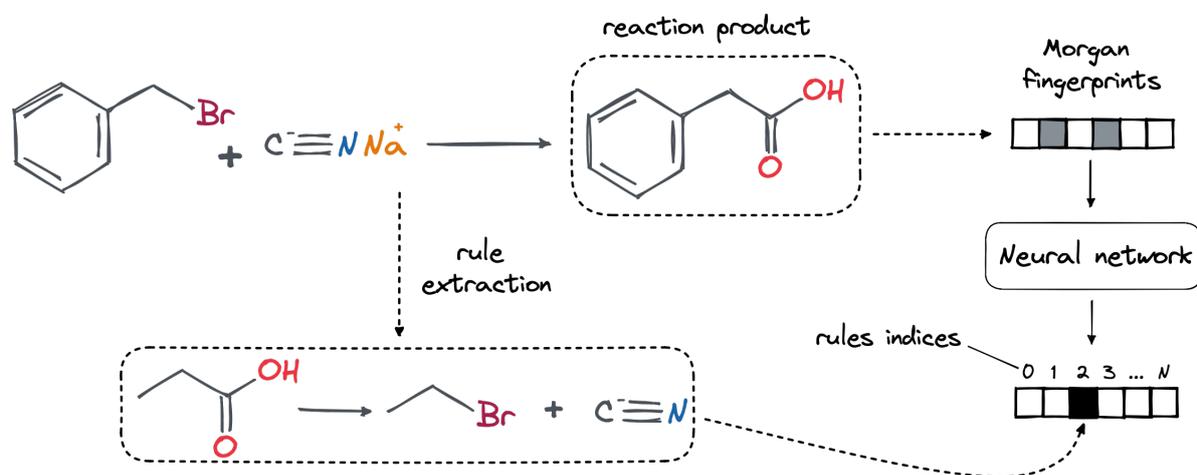


Fig. 4.8 Scheme of data preparation for policy neural network in a manner “1 vs all”. This workflow is used in several retrosynthesis tools, such as AiZynthFinder26 and ASKCOS10.

The main advantage of this technique is that it tends to bias the policy network to prioritise transformations that after application to the input structure will produce reactions similar to the real ones. Thus, even without considering reaction conditions, this approach increases the

chance that the reactions will pass. However, reaction databases consist of historical data and some rules useful in retrosynthesis reactions (coupling or cyclisation) might be underrepresented and will be not used as the first ones to try. Additionally, this policy network does not guarantee that all chosen rules will be applicable.[3]

Additionally, the expansion step can also incorporate the verification of reaction feasibility through the use of a reaction prediction model. For example, Segler et al.[21] employed a classification neural network named "in-scope" that predicts if the reaction is likely to pass based on the way the retrosynthetic transformation was applied to the structure. This approach requires the generation of negative data, which is typically not present in databases. To circumvent this, the authors generated negative data by applying reaction transformations (reverse of retro-rules) to the reactants of reported reactions

An alternative approach of pre-trained policy networks involves the use of generative neural networks that generate potential reactants given products. Lin et al.[129] were among the first to employ a transformer model[130] as a policy network, which directly returned potential precursors for the representative structure of the node. In a way, this approach combined ranking and application of retro-rules in one step. A key advantage of this method is that the transformations learned by model are not restricted to the small area around the reaction centre so, in theory, model can capture "chemical context" or the functional groups that are plausible or implausible for a given reaction. While generative policy networks hold promise, they are less interpretable than retrosynthetic transformations, as the separate identification of reaction types is necessary. Additionally, they tend to be more resource-intensive in comparison to policy ranking-based methods.

Evaluation During the evaluation phase (sometimes referred to as "simulation"), another scoring function (the so-called "evaluation" or "value" function) is used to estimate the retrosynthetic feasibility of a given node. The goal of the evaluation function is to assign a score defining the potential of the node's exploration (term Q in equation 4.1). This step highly influences the tree search strategy, as the tree will grow towards the highly evaluated nodes. The function can be based on either a heuristic or a machine learning approach.

The widely adopted heuristic evaluation function known as "rollout" or "payout" was initially introduced in the early variants of the MCTS for retrosynthesis.[21, 3] During the evaluation, the rollout recursively creates child nodes by selecting the top-1 most probable rules according to the ranking provided by the policy function. The policy function can be used the same as in the expansion step (as was done in Thakkar et al.[3]) or a separate one (as was done in Segler et al.[21]). This process continues until building blocks are found, or the maximum

search depth of the tree is reached (Figure 4.9a). If building blocks are found, the node receives a score of 1, otherwise 0.

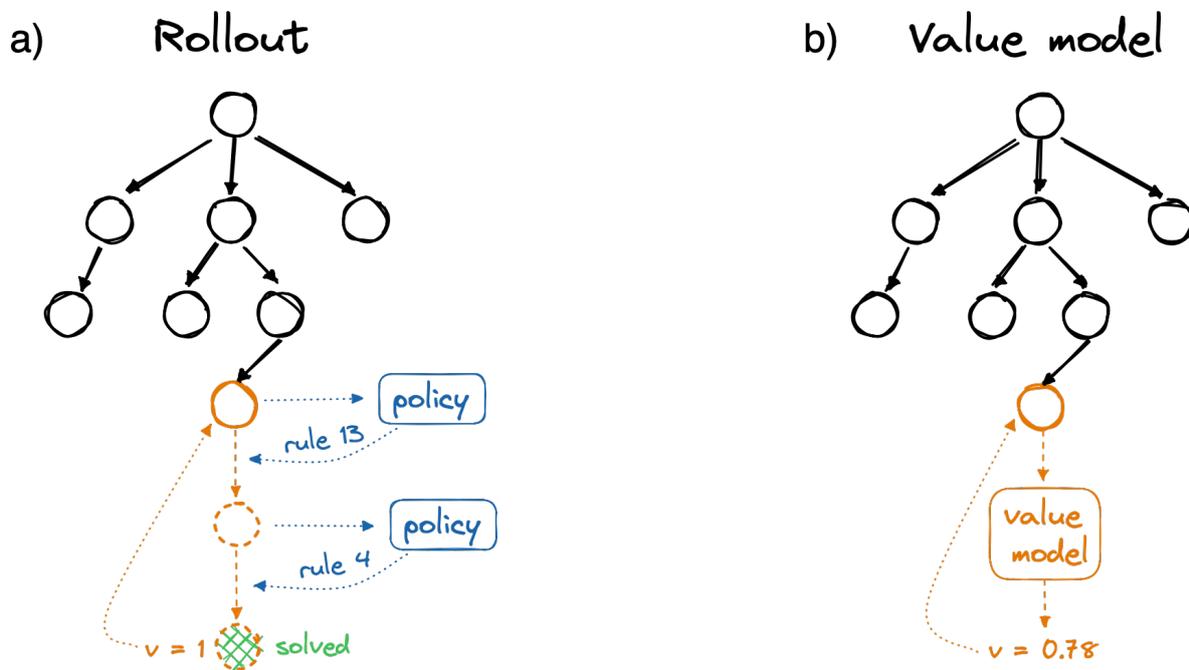


Fig. 4.9 a) A rollout function descends the tree using the top-1 most probable rules according to the policy network. If rollout has found a node consisting only of building blocks, it returns a score of 1, else a score of 0. b) A “value” model that predicts the node’s score based on structures in the node being evaluated.

The performance of the rollout function is critically dependent on the effectiveness of the retro-rules ranking provided by the policy function. The rollout function only employs the highest ranked retro-rule, thus, in the case of a perfect ranking, it will always select the optimal retro-transformation that leads to the building blocks. This, in turn, will result in the identification of the most efficient synthetic pathways during the first iterations of the tree search. However, in practice, policy functions can be noisy and their top-1 ranking accuracy is often low [3]. The high rate of false negatives in ranking may lead to certain promising branches of the tree not being explored during the tree search. In an attempt to improve the rollout function, Ishida et al.[131] introduced a hybrid approach that combines heuristic scores (for example, ring disconnection or selective transformation scores) with it to improve the performance of the retrosynthetic search. While this approach did demonstrate some improvement, the performance gain was not substantial.

Another evaluation function is based on neural networks (4.9b). Given the input molecular structure the neural network predicts the node’s score. Neural networks were not widely used before due to the difficulty of preparing suitable training data for them. Thus, there is no

analytical method for calculating the probability that a molecule would lead to the desired building blocks without performing a retrosynthetic search. This problem was solved when a self-learning approach was proposed[125]. In this method, the neural network learns from the results of tree searches and adjusts itself to more accurately predict the scores of nodes. Thus, the self-learning technique can improve the search strategy over time as the value network is trained. For example, Schreck et al.[105] trained a value network using self-learning to predict the “cost” of a molecule and then choose the most “economic” synthetic pathways. Wang et al.[119] also used self-learning to train a value network to optimise pathways based on “green” or environmentally-friendly criteria, such as lower reaction temperatures and more cost-effective catalysts. Both of these studies demonstrated that self-learning could help to shorten the length of synthetic pathways; however, the training process requires a large amount of time and resources, often requiring millions of CPU hours.

Update In the update stage (also known as “backpropagation”), the score of created node v is used to update the scores of all parents’ nodes (Q_{prev}). A common update function is:

$$Q_{new} = \frac{Q_{prev} * N + v}{N + 1} \quad (4.5)$$

where v – is the score obtained from the value function for the evaluated node, N is the number of visits in the updating node, Q_{prev} and Q_{new} are the previous and new scores of the node, respectively.

Search strategies Retrosynthetic search can be performed using two distinct strategies: the evaluation-first and the expansion-first approaches. In the evaluation-first approach, each child node is assigned a user-defined constant score during the “expansion” step, and then the expanded node is scored during the “evaluation” phase (Figure 4.10). This approach is characterised by a more stochastic selection of nodes for expansion, as the nodes on the last levels of the tree were not yet scored and visited. The evaluation-first strategy is suitable for noisy evaluation functions, such as rollout, and depending on the constant c , this strategy promotes the exploration of neighbouring nodes. Another benefit is the speed of this strategy, as the evaluation function is only called once per iteration. However, with a better evaluation function, this approach will not improve the search quality as selection of nodes the last levels selection will be done randomly. The evaluation-first strategy is used by default in most modern MCTS-based retrosynthesis tools.[21, 3, 127]

Compared to the evaluation-first, the expansion-first strategy is more time-consuming. In this approach, the evaluation function estimates the child nodes instead of the current node (as depicted in Figure 4.11). This stage can be limiting, as the evaluation function will be

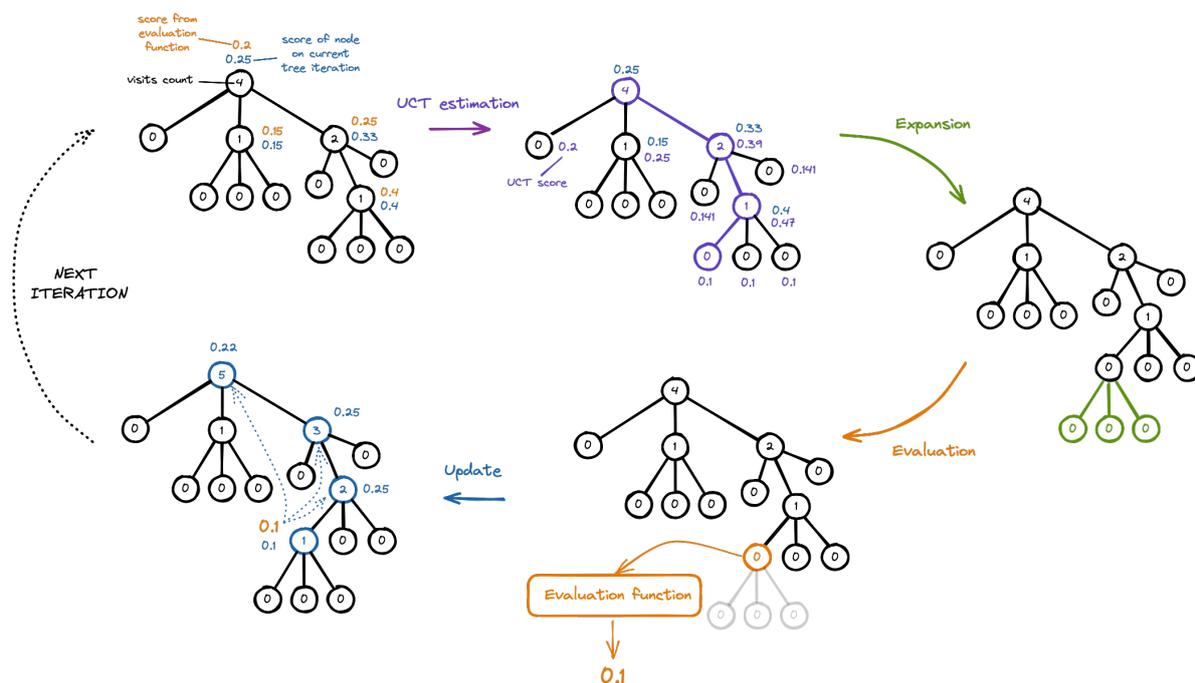


Fig. 4.10 In this example, the classic UCT equation 4.1 was used with $c=0.1$, the update function was taken from equation 4.5, and the tree is on the 5th iteration.

called multiple times per iteration rather than once. As a result, the expanding node in the selection phase will be selected based on its existing score. This strategy is effective when using a reliable value function, for example human expert. In fact, most of early retrosynthetic algorithms employed an evaluation-first strategy.[104, 112, 113]

Self-learning A self-learning method is an approach to training for model-based reinforcement learning architectures (such as value- and policy-based MCTS)[132]. The general idea is to perform several simulations (tree searches in the case of MCTS) and then collect training data about successful and failed decisions for further model fine-tuning. Because of that, this technique became a key to the success of modern MCTS algorithms for games.[125, 133, 132]

Figure 4.12 shows the main scheme of self-learning in case of MCTS-based retrosynthesis. In the initial phase of self-learning, the search algorithm employs a value neural network initialised with random weights to conduct tree searches for a set of synthetic targets. The resulting trees are then used to form a tuning set where the inputs are retrons and labels are based on whether the retron was involved in a path that led to the building blocks or not. The formed tuning set is used to tune the value neural network, which is subsequently utilised by the tree search algorithm in the next iterations (repetitions) of self-learning on the same set of synthetic targets. This process of self-learning continues over several repetitions, also known

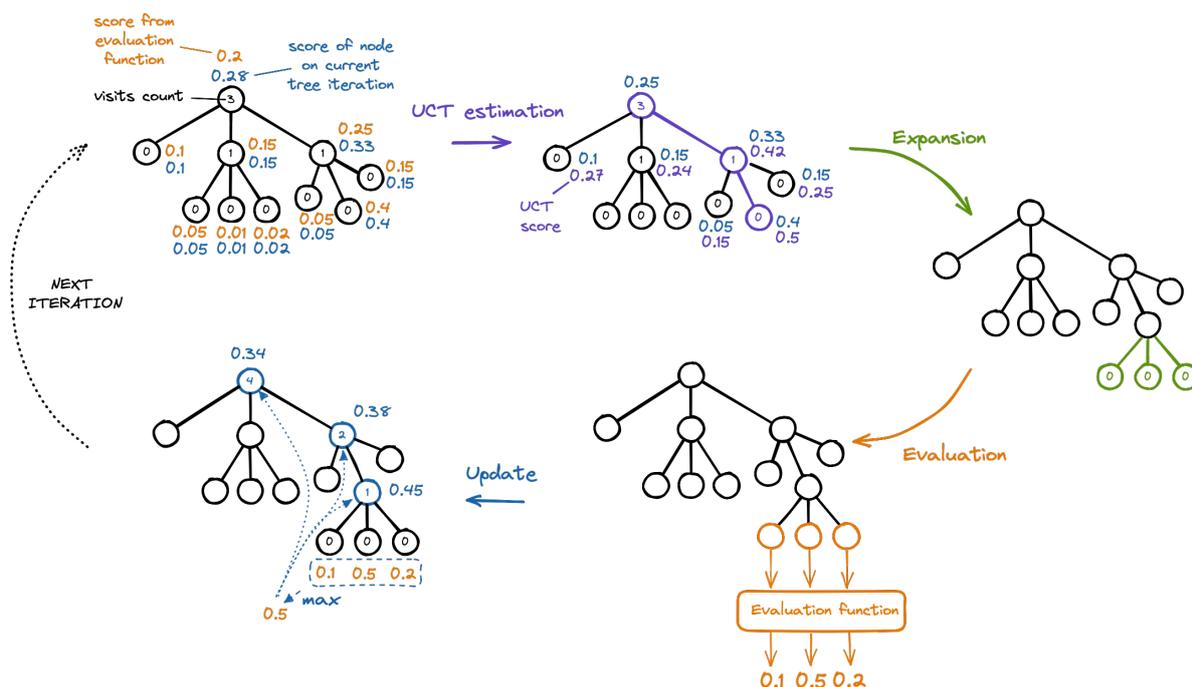


Fig. 4.11 Scheme of MCTS with expansion-first search strategy. In this example, the classic UCT equation 4.1 was used with $c=0.1$, the update function was taken from equation 4.5, and the tree is on the 4th iteration. In this setup, the evaluation function is applied on each created node and during the update stage maximum value of all new nodes is used to update all parent nodes.

as iteration or rounds of self-learning, where the value network's weights for tree searches and tuning are taken from the previous repetition.

Figure 4.13 shows the formation of the tuning set. All retrons that were involved in branches that led to building blocks are initially collected and considered as positive examples. Afterwards, all other retrons in the tree are extracted and are considered as negative examples. It is important to note that if a retron has led to building blocks at least once, it is considered a positive example. The labels or scores for positive and negative examples are assigned depending on the task, which can either be a regression[2] or classification[105] problem. For example, Wang et al.[2] used a special technique known as bootstrapping to assign scores for retrons. In this technique, the building blocks are assigned a score of 1, and the retrons leading to them are assigned a score that is the average of their precursor's scores, multiplied by a discount factor. The retrons on the last levels of the tree that are not available are assigned zero score.

The utilisation of a self-learning approach is a highly attractive technique due to its many benefits, including the lack of necessity for pre-collected training data, the ability to accelerate

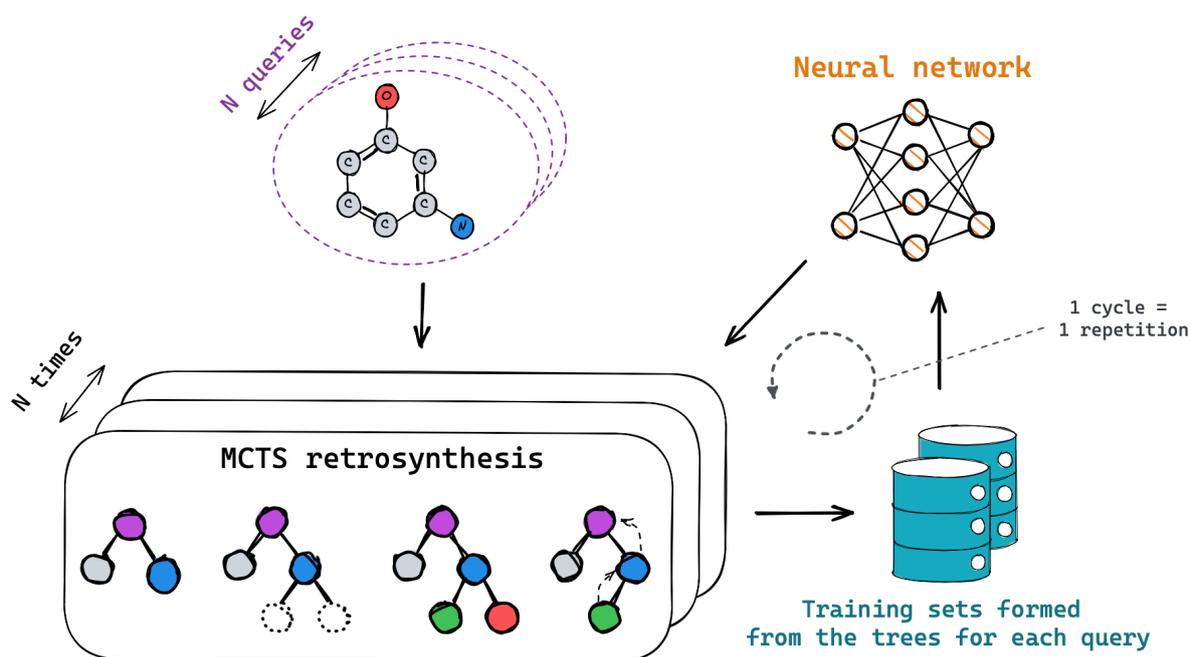


Fig. 4.12 Scheme of self-learning concept

the search process, and the high level of flexibility in optimisation of synthetic pathways. However, the primary limitation of this method is the computational power required. The tree search is already challenging to parallelise and it cannot be used with graphics processing units (GPUs). At the same time during self-learning several repetitions are performed that include thousands of tree searches. Therefore, the computational complexity limits the development of self-learning MCTS-based retrosynthetic algorithms.

MCTS-based approaches In 2017, Segler et al.[21] the first architecture of a data-driven, Monte Carlo Tree Search (MCTS)-based approach for retrosynthesis. The tool, named 3N-MCTS, extracted retrosynthetic transformations from the Reaxys database and utilised three neural networks to select promising retro-rules and predict the feasibility of reactions. The authors demonstrated that the integration of MCTS and deep learning models leads to a significant increase in the speed of retrosynthetic search, through efficient tree exploration. Later, this tool was implemented in the Reaxys platform, which is commercially available.

Since the introduction of MCTS-based retrosynthesis, several further developments have been made in this field. For example, AiZynthFinder[3], is the first fully open-source MCTS implementation that has achieved state-of-the-art results. It was trained on automatically collected reaction database from United States Patent and Trademark Office (USPTO). The authors adopted the similar architecture from work of Segler et al.[21], however they used only

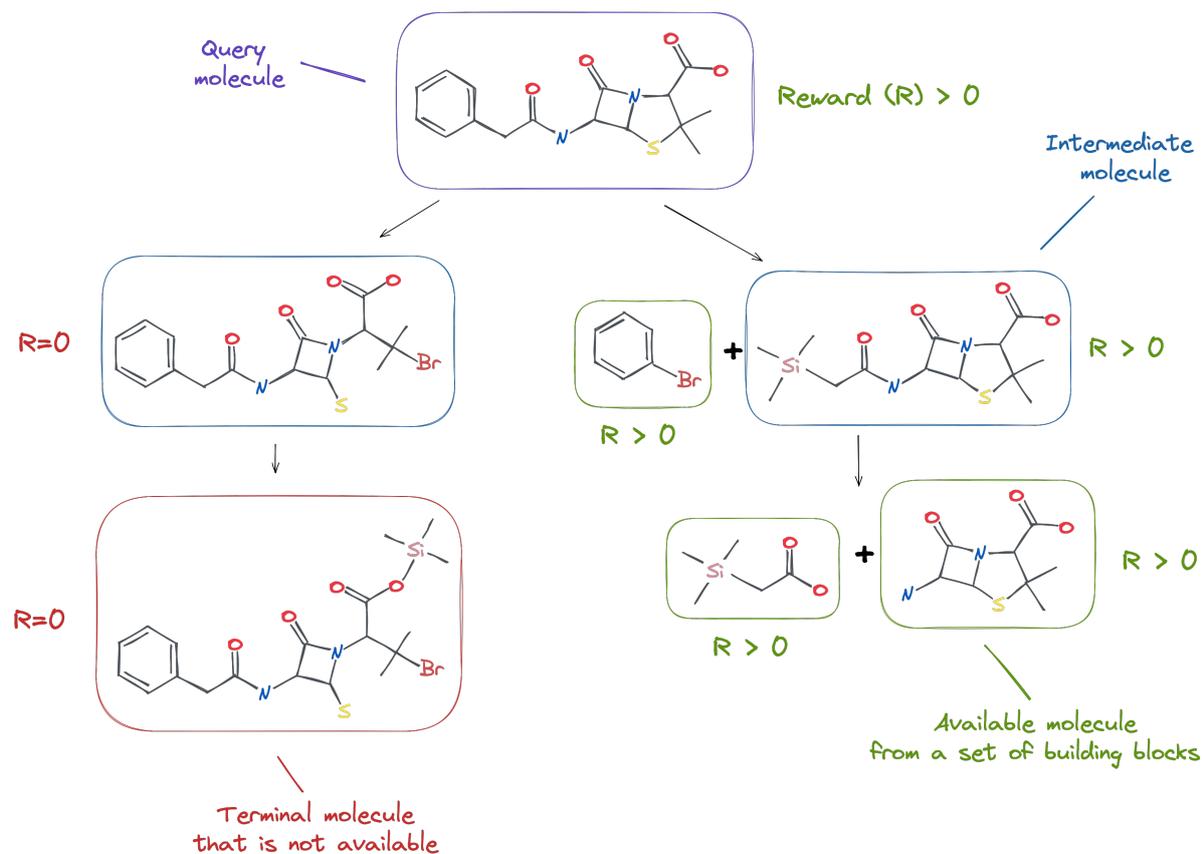


Fig. 4.13 Reward assign during the analysis of a tree search. If the molecule has participated at least once in the root leading to the building blocks, it will receive a positive reward (point); otherwise, it will receive 0.

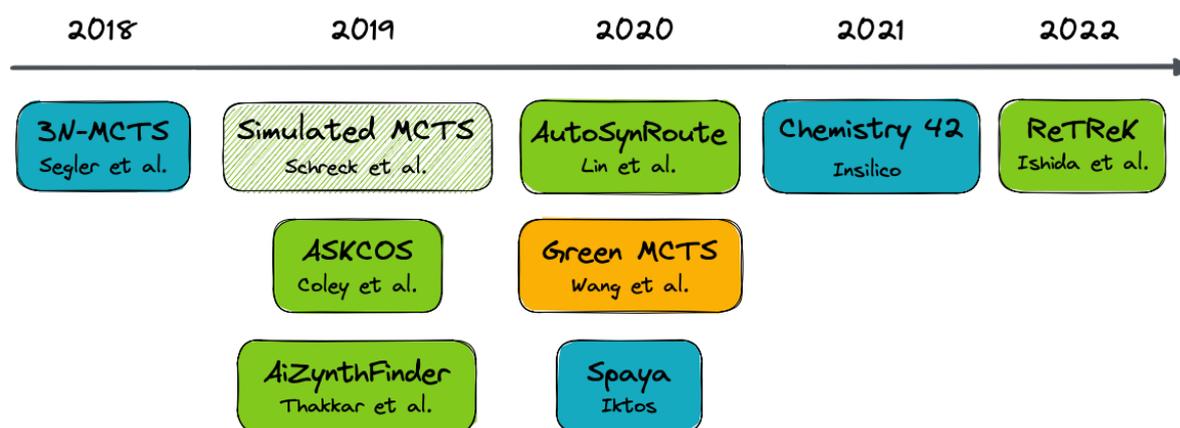


Fig. 4.14 Timeline of the recent developments of MCTS-based retrosynthetic tools. Here tools in green are open-source, green with hachure is partially open-source (the full tool is unavailable), yellow has no open-source code, and blue are commercial platforms.

one neural network for selection of retro-rules. Another open-source retrosynthesis platform ASKCOS[127] augmented the reaction feasibility predictor with forward prediction model that predicts possible side-products and recommends suitable conditions. This retrosynthesis platform was based on both Reaxys and USPTO databases and was combined with a synthetic robot that successfully optimised synthetic routes for 15 known drugs. Some other approaches have also focused on improving retrosynthetic analysis to identify more environmentally friendly synthetic pathways[2], reduce the cost of synthesis[105] or improving retrosynthetic search performance[129, 131].

Except 3N-MCTS, several other commercial tools were presented. In March 2020, IKTOS released the Spaya platform[134], which is capable of calculating the cost of a synthetic pathway based on the source of building blocks and includes filters to ensure the regioselectivity of reactions. However, details about the tool are not widely available. Another MCTS-based variant was patented by Insilico Medicine in 2021, which is similar to the ASKCOS approach with the added feature of a model for ranking the obtained synthetic pathways.[135]

The use of MCTS has accelerated the development of retrosynthesis tools. Among the key benefits of the MCTS-based retrosynthesis is the speed of search. Thus, they are able to propose multiple synthetic pathways within a relatively short time frame of several minutes, even without parallelisation. Furthermore, the implementation of the MCTS algorithm is relatively straightforward, which can be seen by the number of recently developed tools.

On the other hand, current retrosynthesis tools heavily rely on selection functions that prioritise retro-rules, which are trained solely on reaction databases and thus tend to bias selection towards existing reactions. As previously mentioned, these databases consist of historical data and as a result, the selection function prioritises transformations of the most studied reactions, rather than those that may be useful for retrosynthesis. This limitation can be addressed by the introduction of self-learning algorithms, in which scoring functions are learned from the previous experience. While two studies[105, 2] have demonstrated the potential advantages of self-learning (e.g. finding shorter synthetic pathways and increasing search speed), it required a substantial amount of computational time and have yet to be compared with previous developments in the field. Therefore, the future development of self-learning MCTS-based algorithms presents the most interest in the advancement of more powerful retrosynthesis planning tools.

4.2.1 Graph-Based Self-Learning Retrosynthesis

The literature review reveals several challenges associated with automated retrosynthesis. One of the primary challenges is the speed of the search process. In the 1990s, the slow speed of computation hindered the development of retrosynthetic tools, resulting in their limited

ability to solve only relatively simple synthesis molecules.[136] However, with the recent advancements in computing power and the emergence of deep learning theory, new methods combining Monte Carlo tree search with deep learning scoring functions have significantly improved the speed of retrosynthetic search.[21] Additionally, the ease of implementation has made this approach the most popular in current development, as evidenced by the numerous variants that have been implemented.

Still, the problem of synthetic pathways search for natural-product compounds remains unsolved. Despite the existence of tools such as Synthia, which features a comprehensive set of retrosynthetic rules, predefined strategies for functional group protection, and heuristic scoring methods that reflect the logic of human experts, these tools are still only able to solve a limited number of synthetically complex compounds, with many remaining out of reach.[137] The main limitation is that the search process is constrained by the human-based logic included in reaction data and is unable to propose novel synthesis strategies. The similar problem existed in algorithms for strategic games until NN-based Monte Carlo Tree Search (MCTS) algorithms have demonstrated superhuman performance in them.[125, 133, 132] However, most MCTS-based retrosynthesis approaches are still limited by the human bias. The selection of rules and evaluation of precursors heavily depend on policy networks that are trained on historical reaction data, resulting in a bias towards the most popular and well-studied reactions, rather than those that may be more valuable for synthesis planning. Therefore, in order to find more cost-efficient synthetic pathways for complex compounds, the synthetic planning algorithm have to be able to obtain its own experience, rather than rely solely on human knowledge.

One way towards new experience of the search algorithm is self-learning technique. One of the primary benefits is the ability to adapt and learn from previous trials, which is particularly useful in situations where the preparation of training data is challenging. Unlike pre-trained policy and value functions, which are limited by the availability of reaction data, self-learning algorithms have the capability to improve themselves with more repetitions. Additionally, self-learning has been shown to increase the speed of search and decrease the length of synthetic pathways, which are central goals of any search algorithm used in retrosynthetic planning. Furthermore, retrosynthetic tools enhanced with self-learning demonstrate an ability to optimize search in a given direction or under specific constraints. Finally, they are able to improve with time as the number of trials increases. These advantages are crucial for the development of future automated retrosynthetic platforms, where search algorithms are integrated with robotic systems and capable of optimising synthetic pathways without human intervention.

The self-learning approach was previously applied to retrosynthesis search, where evaluation function was replaced from rollout to neural network. However, the authors utilized a pre-trained ranking policy network on reaction data, which restricted the algorithm's potential for

“imagination” of new search strategies. Additionally, these studies did not provide a comparison to previous methods and did not make their source code publicly available. These limitations impede the ability to reproduce and utilise their findings further.

In this study, we address aforementioned issues in a new approach to retrosynthesis called GSLRetro (Graph-based Self-Learning Retrosynthesis) that utilises graph neural networks (GNNs) for both the policy and value functions. This method includes a self-learning training strategy with two evaluation strategies, as well as a rule extraction module. We demonstrate that the self-learning approach, specifically the trained value GNN, significantly increases the number of solved molecules by 20% and enhances the efficiency of retrosynthetic search compared to heuristic evaluation functions such as rollout and random functions. We also compare GSLRetro to the state-of-the-art MCTS algorithm AiZynthFinder[3, 138], and show that GSLRetro outperforms in the number of synthetic pathways found while maintaining high pathway quality.

The previous works were focused only on the performance of MCTS to solve randomly picked molecules or reproduce existing synthetic routes. However, there has been limited investigation into the performance of tree searches for molecules of different complexities. To address this gap, we introduce a new benchmarking dataset composed of evenly distributed “easy-to-synthesise” and natural product-like molecules using the Synthetic Accessibility Score (SAScore) [5]. By evaluating the performance of GSLRetro with different evaluation functions and AiZynthFinder on this dataset, we demonstrate that the trained value network with self-learning significantly outperforms previous approaches for both “easy” and “complex” molecules.

4.2.2 Methods

Proposed algorithms

Policy function As previously discussed in the review section, ranking policy networks are heavily influenced by human bias as they are trained on reaction data. This effect negatively impacts the shaping of the search algorithm’s own experience, therefore in our work, we propose an alternative approach referred to as a “filtering” policy network. The concept behind the filtering network is that instead of learning to rank reaction templates, it learns to keep only those that can be successfully applied to the input structure. The successful application of rule results in structures with no valence errors, which may occur as a result of the presence of leaving groups in reactions from which rules originated. In this way filtering approach eliminates the influence of human bias as the policy network is not trained on reaction data and the search relies solely on the value network, which learns from the results of previous tree

searches. Furthermore, this method allows for the incorporation of new rules without the need for reaction examples, which is beneficial in hybrid approaches that combine automatically extracted rules with those crafted by human experts. However, the number of applicable rules still can be big and it's needed to prioritise those that can be more useful in retrosynthesis. Hence, most perspective rules based on pre-defined heuristics can be chosen in addition to applicable rules. Then, the neural network will simultaneously predict all applicable retro-rules and those that seem more perspective.

In this work, priority retro-rules are selected based on basic intuition in the process of retrosynthetic analysis: preference is given to those retro-rules that split the structure into two or more “large” structures (more than six heavy atoms) or those that break a ring (Figure 4.15). This choice of retro-rules favours coupling and cyclisation reactions, which are the most important in synthesis of complex molecules.

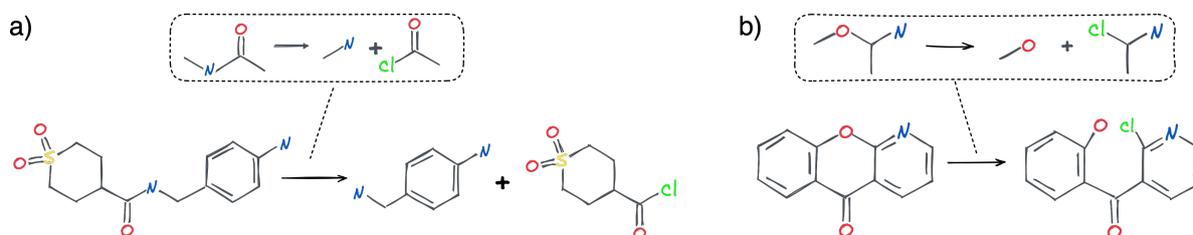


Fig. 4.15 Examples of “priority” rules. a) a “coupling” retro-rule and b) a retro-rule that might correspond to a cyclisation reaction.

Self-learning In training the value network with self-learning, we employ an evaluation-first search strategy combined with an ϵ -greedy policy technique. The evaluation-first strategy is beneficial for increasing the speed of training as the value network has not yet been fully trained. The ϵ -greedy technique promotes exploration by introducing random choices into the selection step with a probability of ϵ .^[105] This method allows the tree search to explore a greater diversity of paths during the self-learning process. In the validation phase, an expansion-first strategy is utilised to evaluate the performance of retrosynthesis with a trained value network. As the trained value network is assumed to be reliable, the expansion-first approach is expected to show better performance.

Retrosynthetic tree The retrosynthetic tree was implemented with special addition for cases, where an application of retro-rule resulted in two unavailable precursors. Such instances are handled by a retrosynthesis queue to which all unavailable molecules are added (Figure 4.16). From this queue, a random structure is selected as the node's representative to which the

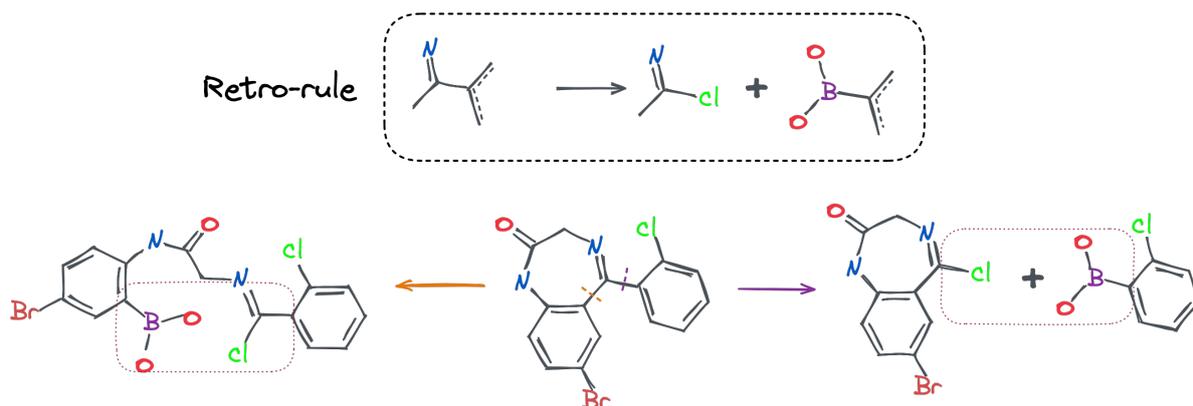


Fig. 4.17 An example of a retrosynthetic rule (a variant of Suzuki coupling) from the USPTO data set that can be applied to phenazepam in different ways and act as a coupling or heterocyclisation reaction rule. This example illustrates a scenario where the confined reaction centre in the retro-rule leads to ambiguity in applying the retro-rule to the molecule.

3. The “reaction distance” filter removes a reaction if the number of changed bonds is more than 6;
4. The “multi-centre reaction” filter removes a reaction record if it includes multiple reaction centres;
5. The “ $C^{sp^3} - C$ breaking” filter removes a reaction if a bond between two sp^3 carbons is broken, as these reactions are energetically unfavourable and may indicate an atom-to-atom mapping error;
6. The “ $C - C$ ring breaking” filter removes a reaction if a bond between two carbons in the same ring of size 5, 6, or 7 is broken. The rings are determined by the “smallest set of smallest rings” (SSSR) algorithm;
7. The “ $C - H$ breaking” filter removes a reaction if a C-H bond is broken with the formation of a C-C bond unless the record is a condensation reaction or reaction with carbenes. This is done by checking presence of heteroatoms neighbours around reacting carbons.

The protocol for the extraction of retro-rules from reactions was developed using the CGRtools python library[94]. This protocol, given a reaction, involves the extraction of substructures containing the atoms of the reaction centre and their immediate environment for each reactant and product. Afterwards, the reactant and product substructures are exchanged. In cases where the reaction includes reagents, they are not incorporated into the retro-rule. All labels that pertain to the atoms of the reaction centre, including hybridisation, the number of neighbours, and the sizes of rings in which the atoms participate, are preserved. In case

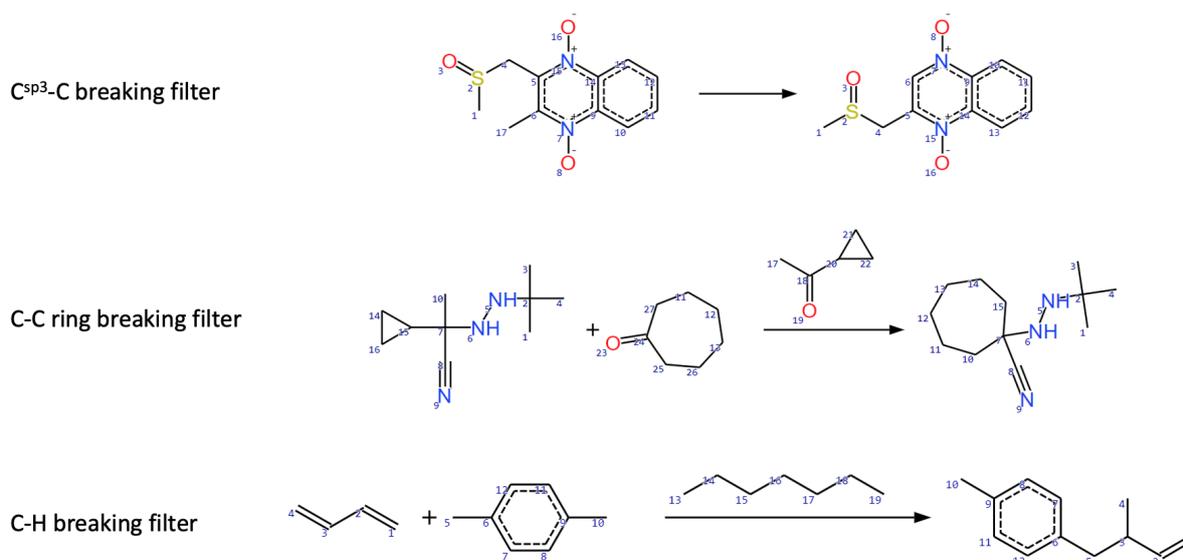


Fig. 4.18 Examples of reactions that did not pass proposed filters. Here are presented filters that check reactions that are probably energetically unfavourable.

of 1st environment atoms only the sizes of rings are preserved. The formed retrosynthetic transformation is subsequently applied to the product of the reaction from which it was extracted and is considered valid if it is able to generate the reactants of the reaction. An example of an obtained retro-rule is shown in Figure 4.19.

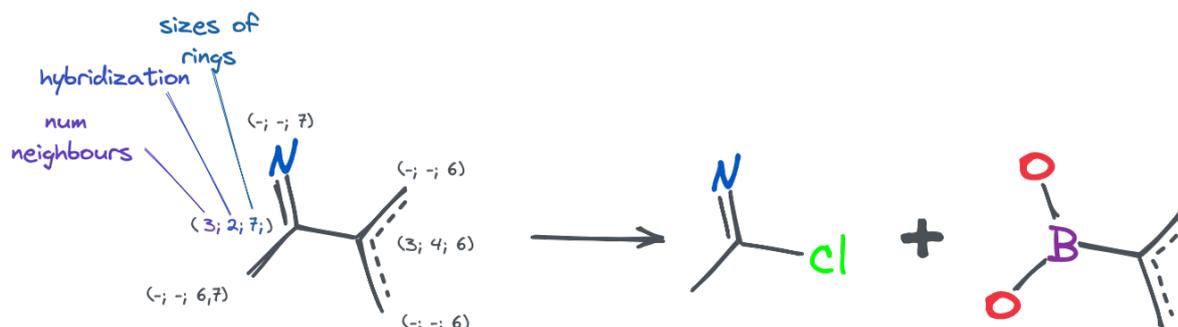


Fig. 4.19 Example of reaction rule used in the GSLRetro. For the rule's "reactant query", atoms of the reaction centre and their neighbours (1st environment) are saved. The atoms of the reaction centre include information about the number of their neighbours, hybridisation and sizes of the rings. The atoms of the first environment only include information about the number of rings in which these atoms were included.

The standardised USPTO database consisted of 1.316 million reactions, of which only 990K reactions passed the filters described above. The reaction rules were then extracted from the filtered USPTO dataset, resulting in a total of 45.318 retro-rules with the first environment.

Among these, 13K retro-rules were deemed “popular,” meaning that they occurred at least in three reactions.

Building blocks The building blocks were taken from previous work[127]. The authors collected building blocks from E-molecules and Sigma-Aldrich stocks. The structures were standardised according to the protocol described in Section 3.2.1. In total, the building block set included 186K of unique molecules.

Benchmarking sets The data used in training and testing was formed from ChEMBL v.27[139] and COCONUT[140] databases. ChEMBL is a database of molecules with drug-like properties and bioactivity data with 1.6M of unique structures, and COCONUT (COLleCtion of Open Natural prodUcTs) is a database that includes 400K known natural-product compounds. Both databases include real compounds with different synthetic complexity. During their analysis, the Synthetic Accessibility Score (SAScore)[5] was used to visualise the distribution of “easy-to-synthesise” and complex molecules (Figure 4.20).

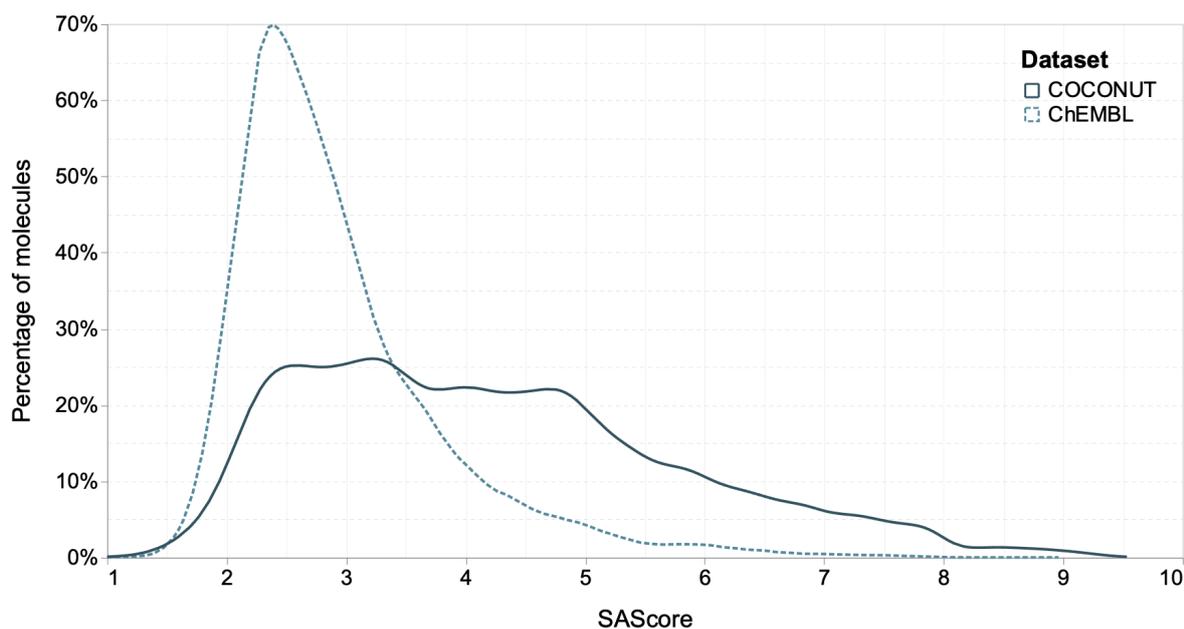


Fig. 4.20 Distribution of synthetic accessibility score (SAScore) among ChEMBL (dashed line) and COCONUT (solid line) databases. The SAScore is a fragment-based metric that ranges from 1 (“easy-to-synthesise” structures) to 10 (“hard-to-synthesise” structures).

From Figure 4.20 it is evident that COCONUT includes more complex compounds to synthesise. However, the total number of compounds is lower than in ChEMBL, so both databases have been merged into one dataset. This dataset was partitioned into seven bins by

SAScore (calculated by the RDKit package[141]) from 1.5 to 8.5 in increments of 1(Figure 4.21). A MaxMin diversity selection was carried out among each split using the Chemfp package v. 4.0[142]. The descriptors for diversity selection were used as Morgan fingerprints with size 16384 and radius 2. For each split, 4100 of the most diverse structures were selected. A random 100 diverse structures from each bin were taken into the test set (700 molecules in total), and the rest formed the training set for self-learning (28K molecules).

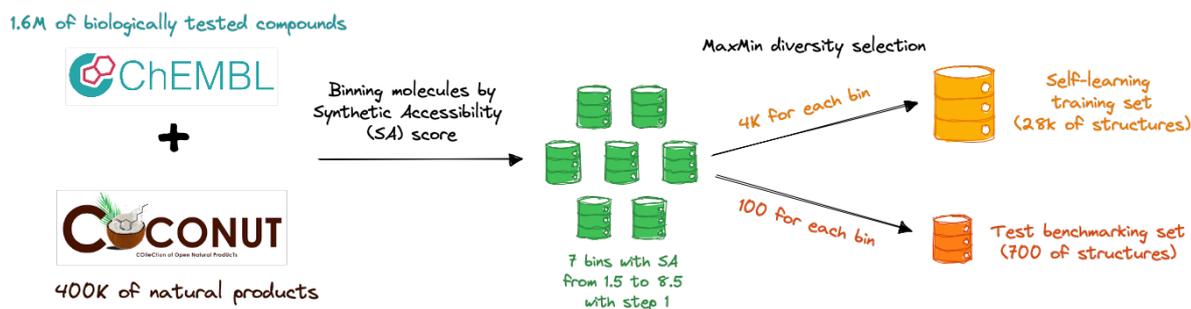


Fig. 4.21 Preparation workflow of training and test molecules from ChEMBL and COCONUT databases.

Policy network training data The policy network dataset was constructed using 600K molecules from the ChEMBL and COCONUT datasets, which were not included in the self-learning training and benchmarking sets. 13K rules extracted from USPTO were applied to each molecule, and the outcome was evaluated. On average, over 304 rules were found to be "applicable" for each molecule, with around 134 of them being recognised as "priority". The formed policy network dataset was randomly divided into training and validation sets in a ratio of 4:1.

Implementation details

MCTS implementation The retrosynthetic tree and search algorithm based on the Monte-Carlo tree search was implemented using Python v. 3.10 and CGRtools library v. 4.1.32[94]. The search was limited by tree parameters: number of iterations, time of the search and size of the tree (total number of nodes). The molecules with a number of heavy atoms of no more than six atoms are assumed to be available. In the selection stage the classical UCT equation (1) was used. The UCT coefficient c was chosen as 0.1. Other hyperparameters were adopted from standard parameters of AiZynthFinder[3] tool. Thus, the maximal depth of the tree search is set to 6, and the number of tree iterations is fixed at 100. For policy network predictions, the top 50 retro-templates are used.

Policy network The policy network was composed of two parts: molecular representation and prediction parts (Figure 4.22). The part responsible for creating a numerical representation of structure (Graph Convolution Network blocks and summation over atoms) adopted from VQGAE architecture. The prediction part was formed from two linear layers with sigmoid activation functions that predict the probabilities for the “applicable” and “priority” rules. These two vectors were summed with a coefficient determined by the hyperparameter α . This approach ensured that the priority rules receive the highest score, followed by other applicable rules, while the remaining rules were discarded.

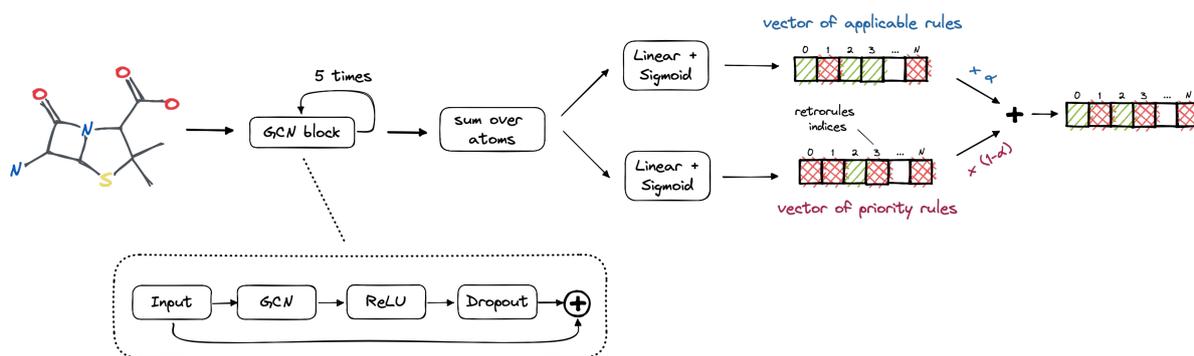


Fig. 4.22 Scheme of policy network used in this work. GCN refers to graph convolution network, ReLU is a rectifier linear unit, plus sign is a residual summation.

The true multilabel “applicability” and “priority” vectors for training the policy were obtained by applying all retro-rules to the policy network training molecules. If the retro-rule created structures without valence errors after application, the retro-rule would be assigned a 1 in the “applicability” vector, otherwise 0. In addition, if the rule split the molecules into two or more structures with number of heavy atoms more than 6 or opens a cycle, the “priority” would be 1; otherwise, 0.

The training was done for 100 epochs with batch size 256, number of GCN blocks equal to 5, an Adabelief optimiser[143] with starting learning rate of 0.0005 and dropout with a probability of 0.4. The dimension of all vectors was set 512. The model achieved 0.947 Balanced Accuracy (BA) for applicable rules and 0.929 for priority rules on the validation set. The coefficient α for a ratio of applicable and “priority” rules was 0.5.

Value network The architecture of the evaluation function is similar to the policy network (Figure 4.23). The molecular representation part of the neural network is the same, and the difference is that only the Linear layer returns one value. For simplicity, the training task of the value network is a classification problem, where positive class corresponds to the structure that leads to building blocks; otherwise, it considered as negative class. The training parameters

employed are consistent with those used for the policy network. The only deviation is the number of training epochs, which is set at 20 for each tuning step.

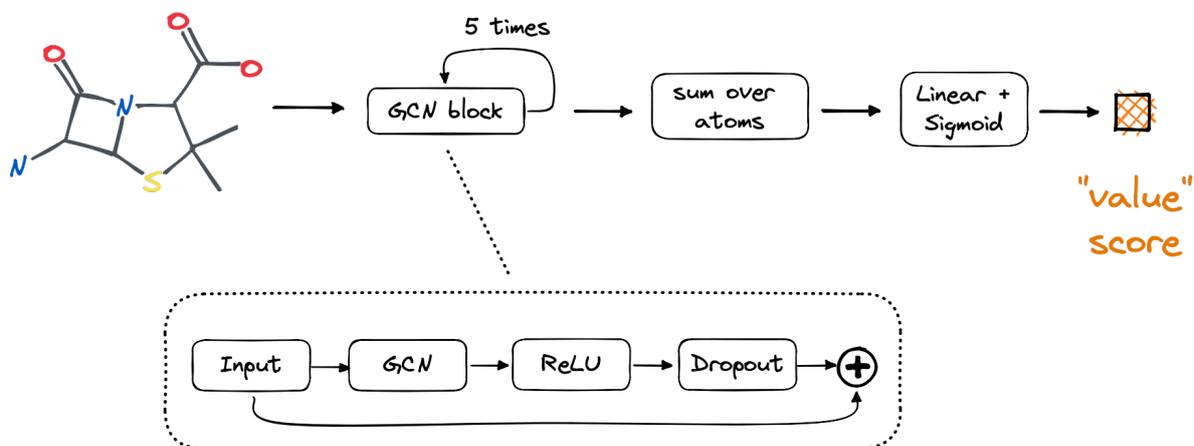


Fig. 4.23 Scheme of value network used in this work. GCN refers to graph convolution network, ReLU is a rectifier linear unit, plus sign is a residual summation.

Self-learning The process of self-learning repetition can be time-consuming, particularly when the size of the synthetic targets set is large. Simultaneously, the process of fine-tuning the value network after a full repetition result in a prolonged optimisation for an optimal value network. To mitigate this issue, we use of mini-repetitions on subsets, or batches, formed from the targets set. In the case of self-learning synthetic targets training set of 28k structures, we employ batches of size 500, thereby performing 56 mini-repetitions (batches) in one “large” repetition.

4.2.3 Results and discussions

AiZynthFinder reproduction

AiZynthFinder is a state-of-the-art, open-source MCTS-based retrosynthesis tool that uses a pre-trained policy neural network on Morgan fingerprints with a rollout evaluation function. In this study, we utilise the version 3.4 of AiZynthFinder as a reference tool for comparison purposes. The reaction data for retro-rules and policy network training used in AiZynthFinder is sourced from the USPTO database. The original database was standardised by the authors and extracted 44K rules with the first environment. However, this version of the USPTO database is not publicly available.

In order to ensure fair evaluation of the search algorithms and models implemented in retrosynthesis tools, it is imperative that the building blocks and reaction data used to extract

retro-rules be consistent across compared tools. Therefore, we employed our own version of standardised and filtered USPTO data as a source of retrosynthetic transformations. We followed the protocol for rule extraction and training of the policy network provided in the AiZynthFinder package and RDChiral library[144] without any modifications. Thus, the version of AiZynthFinder that utilises the trained policy model and prepared retro-rules provided by the authors is referred to as the "original" version, while the version based on our own data is referred to as the "reproduced" version. It is important to mention that the other part (search algorithm and search and training hyperparameters) for both versions were the same.

The retro-rules extraction protocol applied to our version of the USPTO yielded 34K retro-rules with the first environment. The difference between the original rules and the reproduced ones was mainly due to the application of standardisation techniques on the same database. Thus, it was expected that most of the rules would overlap (Figure 4.24a). Nevertheless, the variation in data curation protocols had a substantial effect on the reactions, resulting in only around half of the original retro-rules being present in the reproduced rules.

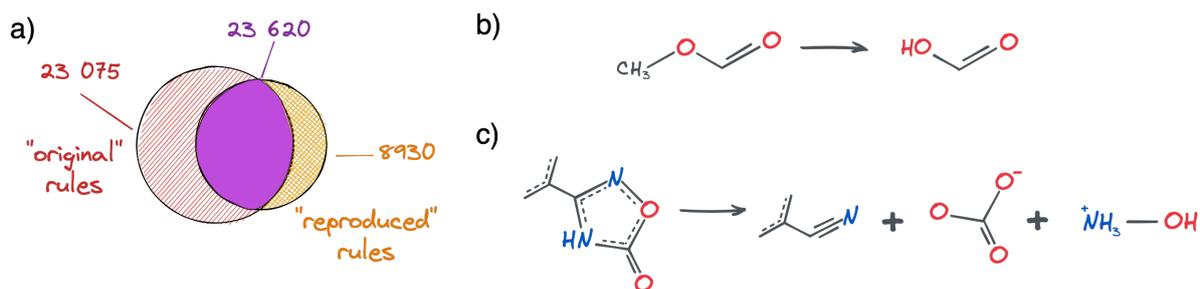


Fig. 4.24 On the left, Venn diagram (a) is based on "original" rules provided in AiZynthFinder and "reproduced" rules extracted from filtered USPTO. On the right are examples of rules that are present only in the "original" (b) and "reproduced" (c) sets of rules.

The extracted rules were then used to train policy network. The architecture of the policy network and its training parameters were adopted from the original paper[3]. The input for the policy network consisted of Morgan fingerprints with a radius of 2 and 2048 bits. The policy network was trained for 100 epochs using the Adam optimiser[?] with a learning rate of 0.001 and a cross-entropy loss function. The primary performance metrics were the top 1, 5, 10, and 50 accuracies, monitored throughout the training process. At the end of the training, the top-1 and top-50 accuracy on the test set were 0.576 and 0.950, respectively. These results can be compared to those reported in the original paper, which showed a top-1 accuracy of 0.595 (the top-50 accuracy was not provided). Based on these results, it can be concluded that the training was successful.

The parameters for the retrosynthetic search were adopted from those provided by the authors of AiZynthFinder[3], with the exception of an increase in the allotted search time to 10 minutes. The standard parameters of AiZynthFinder are following: the maximal depth of the tree search is set to 6, and the number of tree iterations is fixed at 100. The coefficient c in the UCT formula is equal to 1.4. For policy network predictions, the top 50 retro-templates are used.

Figure 4.25 shows the results of a performance test for the original and reproduced versions of AiZynthFinder on a benchmarking dataset. The results demonstrate that the original AiZynthFinder is able to solve more targets than the reproduced version. On average, the synthetic pathways solved by the original AiZynthFinder are 1.2 times shorter, and the number of pathways found is 2.1 times greater. One potential explanation for the lower number of found pathways for the reproduced AiZynthFinder is the strictness of the proposed standardisation protocol. This strictness has resulted in a 1.3 times lower number of retro-rules derived from the standardised USPTO compared to the number of rules provided by the authors. However, the difference in number of retro-rules seemed not only the reason for such a decrease in performance. Therefore, it was decided to visually compare the synthetic paths provided by the original and the reproduced AiZynthFinder.

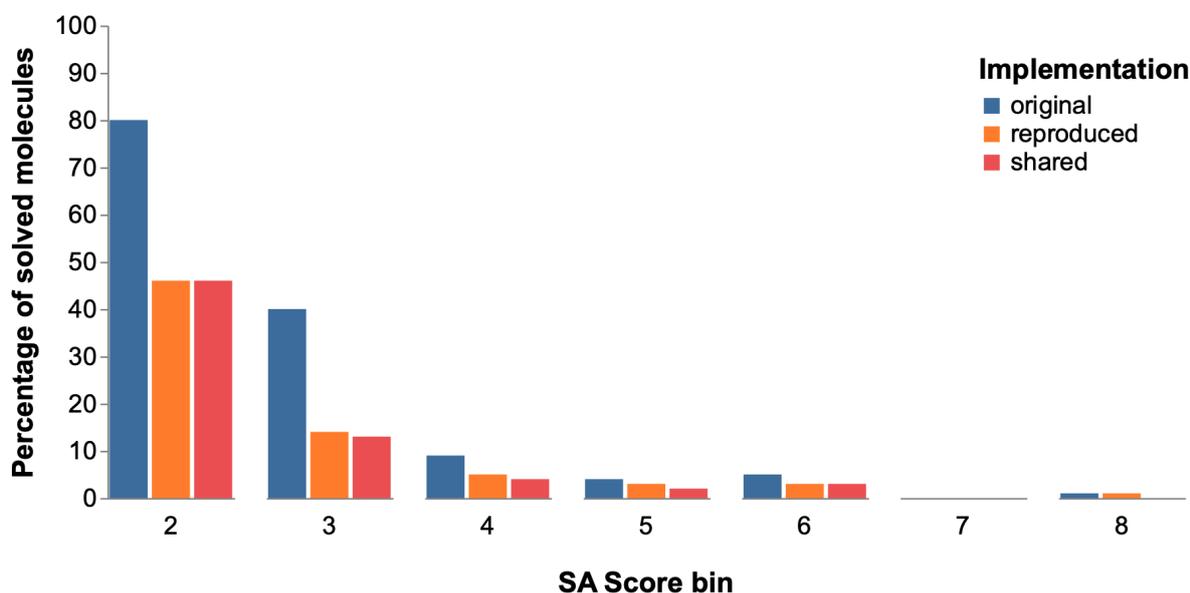


Fig. 4.25 Results of original (blue) and reproduced (orange) AiZynthFinder on the benchmarking test set. The shared (red bar) refers to molecules which were solved by both implementations.

During the visual analysis of the synthetic pathways, it was found that the reproduced AiZynthFinder did not see some building blocks during the search (Figure 4.26). Consequently,

the reproduced algorithm is sometimes forced to “synthesise” building blocks, which lengthens the synthetic pathways and reduce the number of solved query molecules. This observation indicates the presence of a hidden bug that prevents the results from being fully reproduced with the AiZynthFinder. The code for AiZynthFinder was taken from the original repository without any modifications. We were not able to identify the source of the bug (AiZynthFinder itself or in RDChiral library) and we notified authors of its presence.

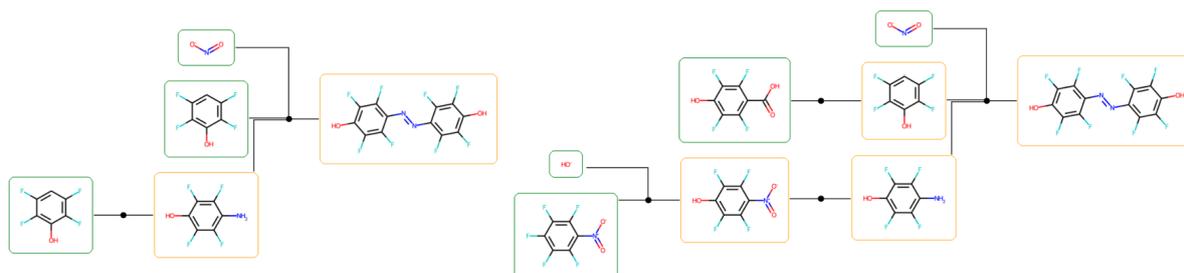


Fig. 4.26 Examples of retrosynthetic pathways for the “original” AiZynthFinder (a) and the reproduced one using our data (b). Here, the original algorithm sees 2,3,5,6-tetrafluorophenol (in red) as a building block, while the reproduced one does not.

GSLRetro implementation

The most popular evaluation function employed in previous NN-based MCTS algorithms was the heuristic-based rollout function. For the fair comparison with value network, we implemented rollout function as a baseline in GSLRetro following 3N-MCTS approach. Additionally, we implemented a random evaluation function, which returned a score derived from a uniform distribution with a range from 0 to 1. For GSLRetro with both functions, the search strategy employed was expansion-first, as the use of an extensive strategy did not result in any performance improvement. Furthermore, the reproduced AiZynthFinder was also utilised in the performance comparison, despite the presence of bugs. The results of comparison are given in Figure 4.27.

In this setup GSLRetro with even a random evaluation function, outperforms reproduced AiZynthFinder for “easy-to-synthesise” molecules. This observation supports the idea that even a naive filtering policy with heuristically chosen “priority” rules can efficiently search for synthetic pathways. Note that the performance of reproduced AiZynthFinder is lowered, but it is unlikely to outperform the GSLRetro rollout results. On the other hand, the rollout evaluation function is slightly better than the random one. As rollout highly depends on the top-1 ranking by policy network, it is evident that the quality of the ranking of the molecules by the filtering + priority policy is low.

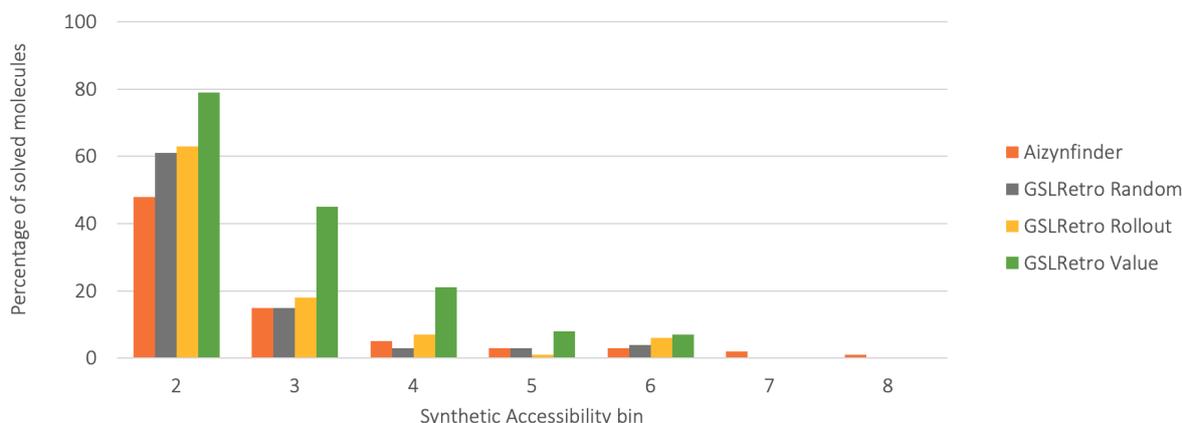


Fig. 4.27 Results of tree searches for the reproduced AiZynthFinder (orange), GSLRetro with filtering policy and random (grey) rollout (yellow), and self-learned value network (green) evaluation functions.

At the moment, the self-learning of GSLRetro value network was performed until the 10th batch of the first “large” repetition. As was previously mentioned, self-learning is time consuming process and in current version of GSLRetro one mini-repetition requires 24 hours.

Figure 4.28 shows the performance of the value network during self-learning on the benchmarking dataset. It can be seen that after training on the first batch, the value network quickly reaches a performance level close to the best and then gradually improves. One potential reason for the slower growth of search performance with the value network may be that it initially solves molecules for which it is relatively easy to find pathways and then gradually learns new synthetic strategies through self-learning. At the same time, it can be seen, that very complex molecules (starting from SAScore bin > 7) are almost never solved. One of possible reasons is the limited amount of tree iterations and the maximum depth of the retrosynthetic tree. Thus, the authors of Synthia[137] showed, that most of natural product like compound require long synthetic pathways (> 10 steps). However, in the current implementation of GSLRetro the increase of tree depth and number of iterations will dramatically increase the computational time. We leave this issue for the future work.

The performance of the GSLRetro with a trained value network was compared to that of heuristic (rollout) and random evaluation functions (Figure 4.27). Compared to other functions, the value network significantly outperforms other approaches. This is mainly seen in the molecules for SAScore bins 3 and 4. At the same time, GSLRetro with value network also outperforms reproduced AiZynthFinder. However, AiZynthFinder solved three molecules from bins 7 and 8, as the compounds with similar cores existed in the building blocks (which means that these retrosynthesis problems were actually easy). One assumption why the GSLRetro

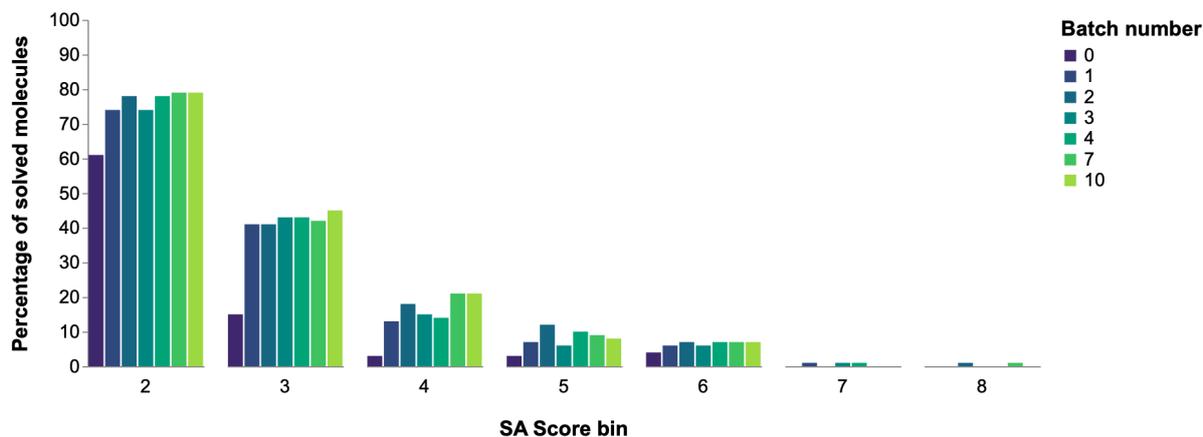


Fig. 4.28 Visualisation of performance of GSLRetro with value network trained after several batches calculated on the benchmarking data set. Here the bar with batch “0” refers to the random model before training and “1” means the value network after tuning on the first batch.

could not find it is due to the filtering policy network, which prioritise rules changing core structure in the first.

Figure 4.29 compares the performance of GSLRetro with the value network and the original AiZynthFinder. The results show that GSLRetro performs similarly to AiZynthFinder on relatively easy-to-synthesise molecules but outperforms AiZynthFinder on molecules of medium synthetic complexity. However, for complex structures, both tools show zero performance. Not surprisingly, both can solve many of the same molecules. As previously noted, the reproduced and original retrosynthetic transformations in AiZynthFinder have 50% of overlap. At the same time, the reproduced rules for AiZynthFinder and GSLRetro rules are extracted from the same version of USTPO data set. Thus, it can be assumed that the same pathways are likely discovered by both AiZynthFinder and GSLRetro.

Analysis of the value function

The trained value network is of interest not only as a method for improving tree search but also as a scoring function for the goal-directed generation of molecules. During self-learning, it should identify the key patterns that lead to the successful discovery of synthetic pathways. Consequently, it can be assumed that molecules with a higher score from the value network are likely to be solved by GSLRetro.

First, the distribution of value network scores (VNScores) was examined. The scores for 800 training molecules with SAScores ranging from 4.2 to 4.5 were obtained and then plotted against the number of heavy atoms and the number of rings (Figure 4.30).

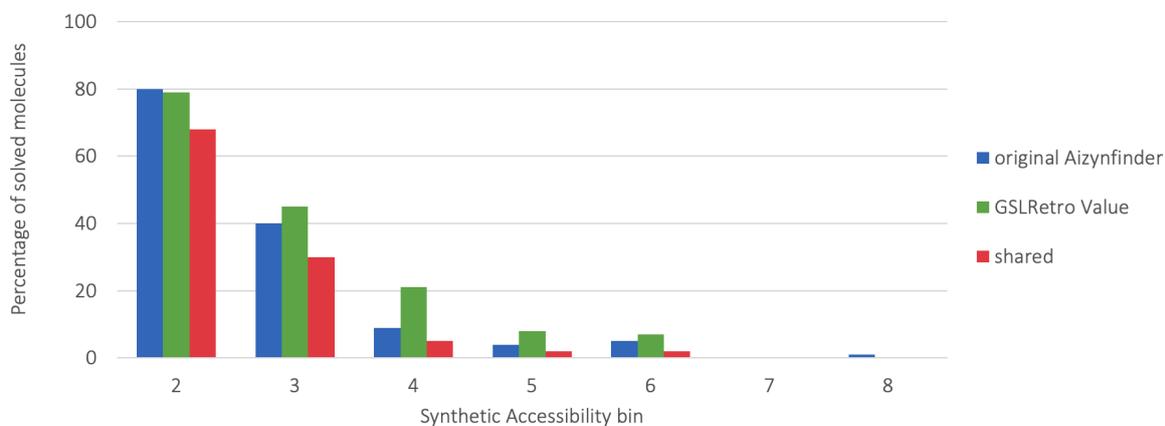


Fig. 4.29 Results of tree searches for the GSLRetro with self-learned value network (green) and original AiZynthFinder (blue). The shared (red bar) refers to molecules which were solved by both tools.

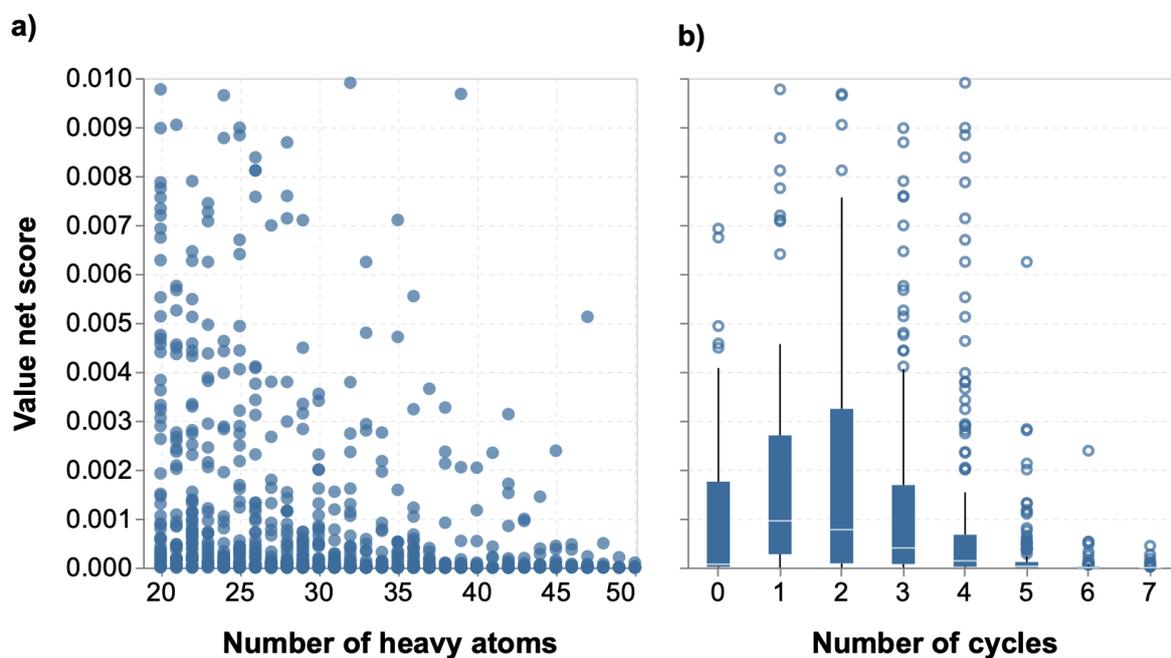


Fig. 4.30 Distribution of a number of heavy atoms (a) and a number of cycles (b) as a function of scores from value network for molecules from self-learning training set with SAScores in the range [4.2; 4.5].

Analysis of the value network scores reveals that, due to an imbalanced training set (with an average of 6% positive examples), the range of scores is narrow and, in the majority of cases (97%), score value does not exceed 0.01. At the same time, there is a weak correlation between the size of the molecule, the number of cycles, and the VNScore. This indicates that the network is not primarily seeking simple dependencies between the retrosynthetic feasibility and structural parameters of the compound but rather giving greater importance to its fragments. We also compared VNScore with SAScores; however, we did not find any correlation between them.

Examples of pathways

Consistent with our observations above, the Monte-Carlo tree search algorithm advanced by the value network are promising in finding the synthetic routes than any other MCTS-variants based on the rollout function. Unfortunately, the number of solved synthetic targets is the only metric to estimate the performance of retrosynthetic tools. Therefore, some examples of found synthetic pathways are given for both GSLRetro with value network and AiZynthFinder.

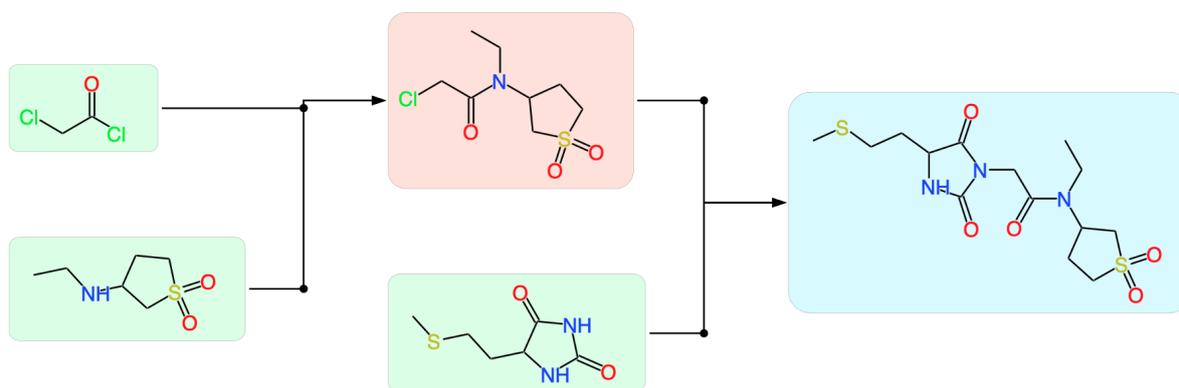


Fig. 4.31 Example of the same synthetic pathway found by original AiZynthFinder and GSLRetro with value network. The SAScore of the target is 3.52.

Figure 4.31 shows a synthetic pathway for a target with SAScore equal to 3.52 that was found by both algorithms. Here a two-step synthesis is proposed, first amide formation from acyl chloride and secondary amine (nucleophilic acyl substitution) followed by imide alkylation. The last step might produce several products, as there is no forward reaction prediction module that validates if product of reaction is major one. However, the presented synthesis provides a general idea that can be easily modified by an expert chemist. Therefore, the question of regioselectivity of reactions falls outside the scope of this work.

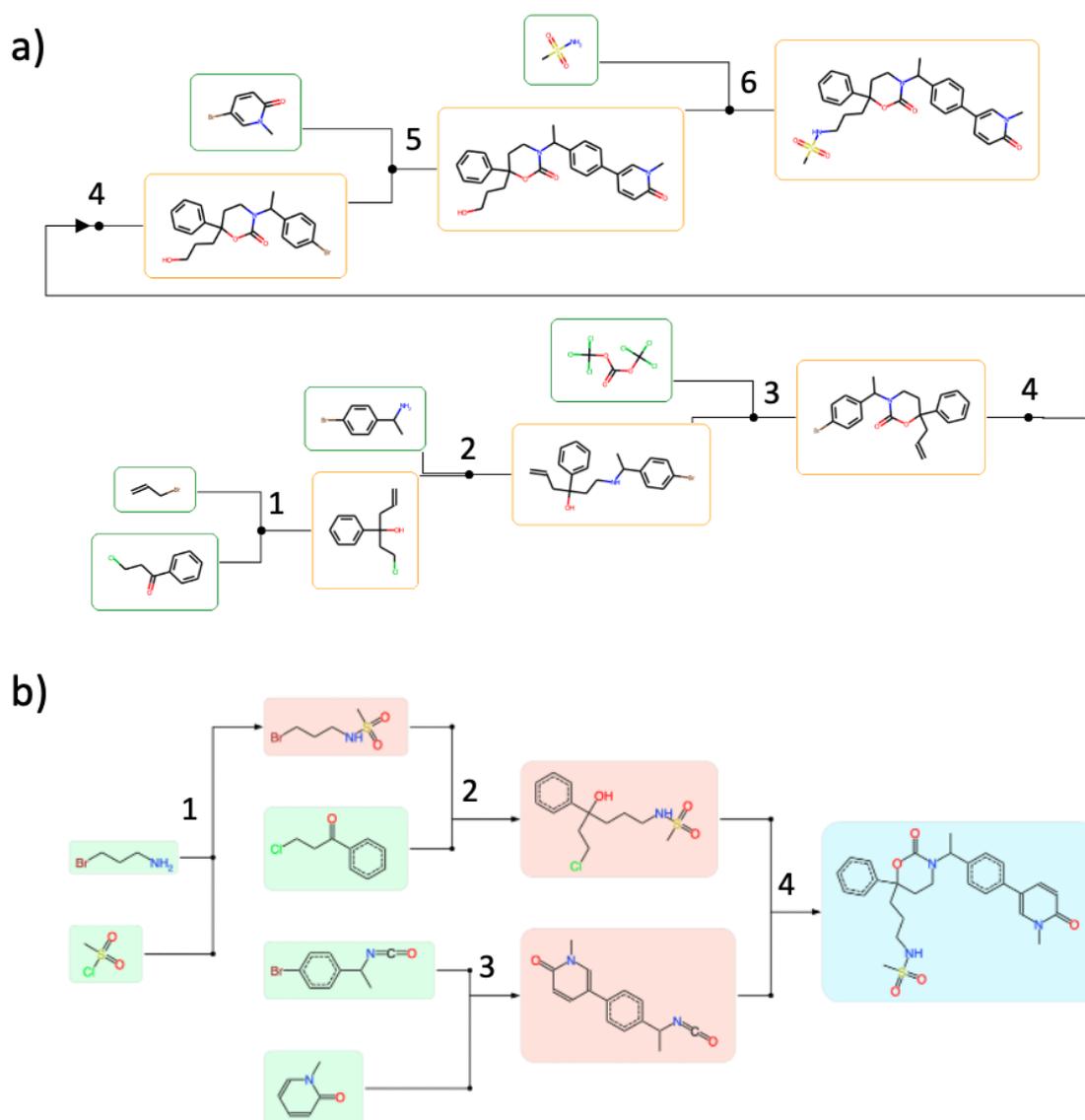


Fig. 4.32 Examples of different synthetic pathways for the same target solved by original AiZynthFinder (a) and GSLRetro with value network (b). Here each number corresponds to a reaction transformation.

However, for some of the shared solved synthetic targets, AiZynthFinder and GSLRetro proposed different synthetic strategies. As illustrated in Figure 4.32, two different pathways were found for the same synthetic target with SAScore 3.55. Specifically, AiZynthFinder proposed a six-step pathway, while GSLRetro proposed a shorter four-step pathway. The use of the rollout function in AiZynthFinder likely contributed to the longer length of the synthetic pathway. Interestingly, both strategies share the same idea of forming the central 1,3-oxazinane cycle. AiZynthFinder's strategy involves reacting the triphosgene with alcohol

and amine groups (Figure 4.32a), reaction 3. However, the Wurtz (or possibly Ulmann) reaction (5) is probably non-selective and require harsh conditions. Reaction 6 will also require harsh conditions not to say that there are doubts in its feasibility. In contrast, GSLRetro's strategy involves reacting the isocyanate group to form the 1,3-oxazinan-2-one cycle (Figure 4.32b, reaction 4), which has a high yield and requires low temperatures[145]. However, the reaction 3 probably require protection of oxygen, since it could possibly react with isocyanate. Also, reaction 2 requiring in situ formation of organometallic reactant could not proceed in presence of sulphonamide, thus it is more reasonable to change order of reaction 1 and 2: first make addition of bromopropylamine to ketone and then generate sulphonamide. Nonetheless, considering synthetic path length and choice of reactions, in this example, GSLRetro more elegant pathway.

In rare cases, AiZynthFinder and GSLRetro solve different synthetic targets. For example, AiZynthFinder found synthetic pathways for natural products with SAScore 4.16 (Figure 4.33a). The main steps of the proposed strategy are the hetero-cyclisation of tetrazole ring with triethyl orthoformate and azide salt (reaction 2) and the addition of 4-(1,2,4-Triazol-1-yl)phenol through Williamson ether synthesis (reaction 4). In another case (Figure 4.33b) only GSLRetro found a pathway for a molecule from the ChEMBL dataset with SAScore 4.02. The main highlights of this strategy are forming a pyrazolidine cycle (reaction 3) and hetero-cyclisation by assembling a sultam group (reaction 6). The feasibility of some reactions due to selectivity issues is questionable, but as we mentioned GSLRetro ignores it for the time being.

As a conclusion, it is worth noting that, despite having fewer reaction templates, GSLRetro's extended search strategy with a value network allows it to propose more solutions of the same or even better quality than previous algorithms.

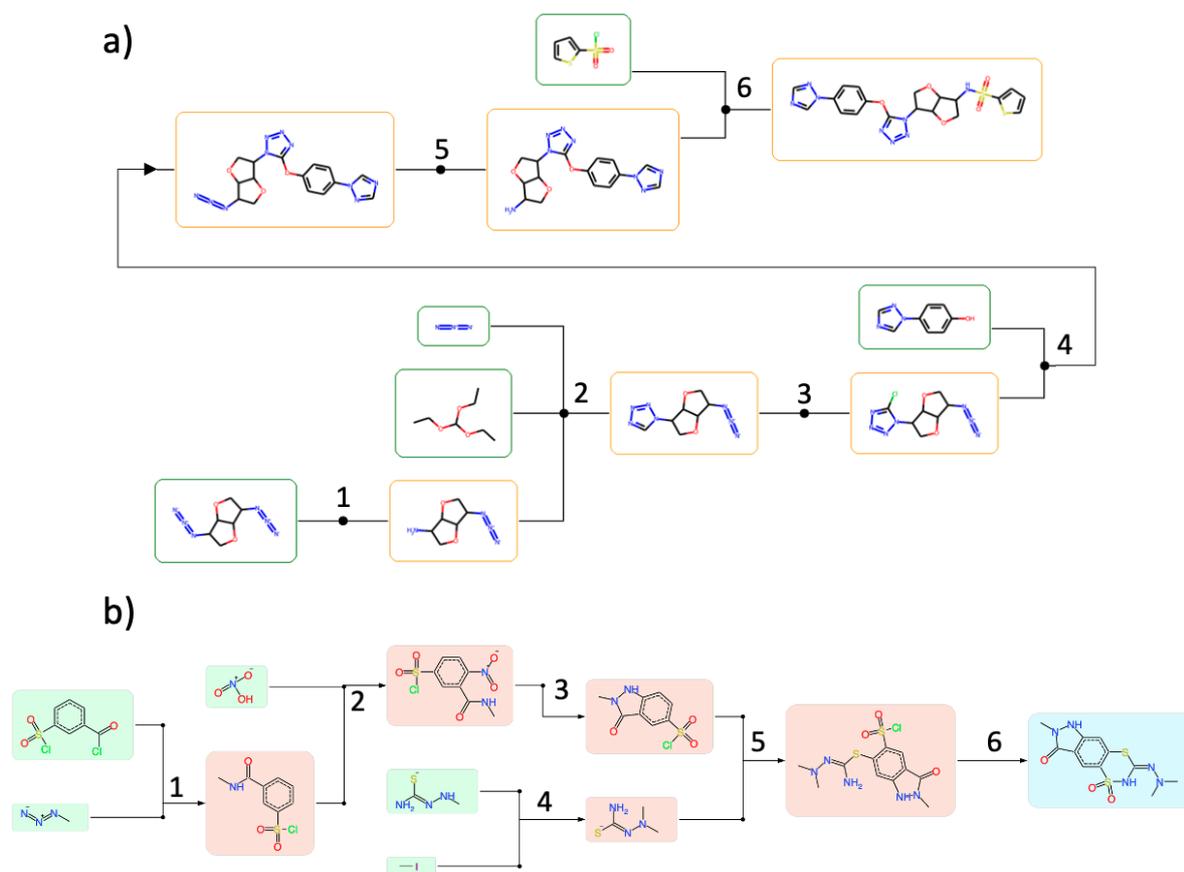


Fig. 4.33 Examples of synthetic pathways for solved targets only by a) AiZynthFinder and b) GSLRetro. Here each number corresponds to a reaction transformation.

Search for synthetic pathways of generated structures

One of the main goals of this work was a combination of generative neural networks for design of new molecular structures with deep learning empowered retrosynthesis planning. Therefore, in this part we want to find synthetic pathways for molecules generated by VQGAE using GSLRetro with value network.

As a result, out of the four structures, three were successfully solved. Examples of the found synthetic pathways are shown in Figure 4.34.

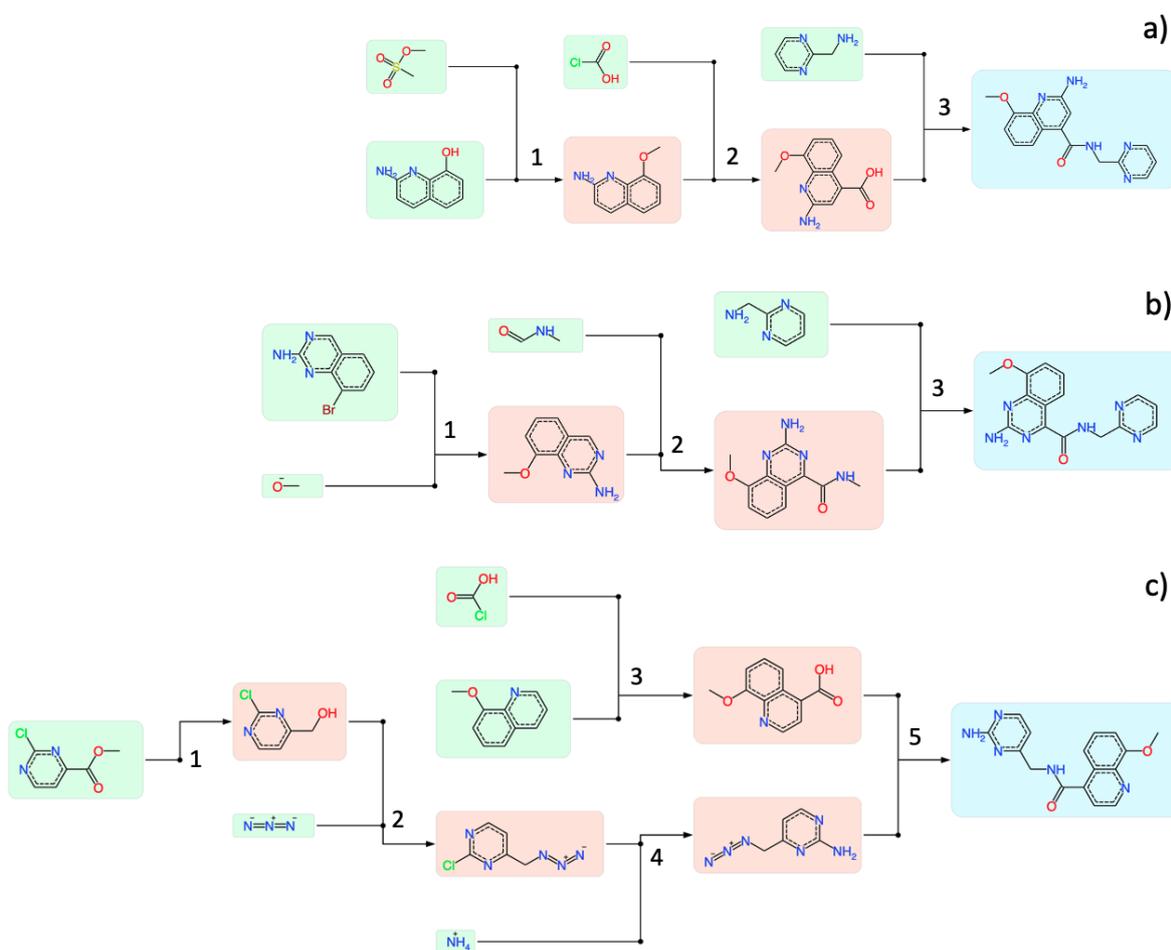


Fig. 4.34 Example of synthetic pathways for the three ligands generated for the adenosine A2A receptor. In blue, a molecule is generated by VQGAE; in red, the intermediate molecules; in green, the building blocks.

The proposed synthetic pathways for all structures look quite reasonable. What is interesting, even minor modifications to the structures could result in significantly different synthesis strategies. At the same time, a common feature among these pathways is the final synthesis step involving forming an amide bond, which is a widely used and well-known reaction in medicinal chemistry due to its high yield and mild conditions. Additionally, these pathways share similar starting building blocks and provide insight into potential starting points for synthesising these structures

4.2.4 Summary for the section 4.2

In this section, we introduce a self-learning variant of the MCTS-based retrosynthesis algorithm called GSLRetro. This algorithm comprises of several key components, including a

rule extraction module, two search strategies, graph-based neural networks for both policy and value functions, a self-learning mechanism, a rollout evaluation function, a synthetic pathway visualization module, and other relevant functionalities. Additionally, we propose the integration of a new approach, referred to as a “filtering” policy network, in conjunction with a value network trained through self-learning. This combination effectively eliminates the influence of human bias in the search algorithms, enabling the exploration of new synthetic strategies and the incorporation of new rules without the need for reaction examples.

The results showed that the trained value neural network significantly outperformed the heuristic rollout function, especially on the quite complex molecules (SAScore between 3 and 6). At the same time, it was compared with the state-of-the-art retrosynthesis algorithm AiZynthFinder with our and provided with the tool retro-rules. It was found that GSLRetro outperforms AiZynthFinder in both setups in the number of solved compounds and it generated more high-quality synthetic pathways for molecules of medium synthetic complexity.

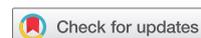
Thus, the GSLRetro was able to improve search performance in a highly constrained setting, characterized by a limited tree depth of 6 and a maximum number of iterations of 100. However, it was unable to “crack” the most complex molecules. This represents a significant challenge that remains to be addressed. There exist two possible explanations for this limitation. Firstly, the source of retrosynthetic transformations, the USPTO database, was found to contain many reactions with errors in atom-to-atom mapping, which resulted in a reduced number of useful retro-rules and hindered the retrosynthesis of natural product compounds. Secondly, it is possible that the depth and iteration limit were too restrictive for the successful retrosynthesis of complex compounds, which may require more than 6 steps for synthesis. However, the current implemented version is not yet efficient, as it requires several months of self-learning even with small number of tree iterations. Thus, we believe that in the future developments the optimisation and parallelisation of code and choice of better source of reaction rules will help in solution of synthetically complex molecules.

In conclusion, our work demonstrates the effectiveness of the self-learning GSLRetro algorithm for retrosynthetic planning, which can be further improved by training the policy and value networks in self-learning mode using more comprehensive reaction rules that take into account the reactivity and regioselectivity of the reactions.

4.3 Validation techniques for prediction of reaction rate constant in different conditions

Despite significant progress in retrosynthesis planning, current tools still need to improve their ability to determine reaction efficiency regarding thermodynamic and kinetic parameters. This is partly because retrosynthetic transformations consider only a limited environment around the reaction centre and do not take into account the reaction conditions. As a result, even when retrosynthetic transformations are successfully applied, the resulting synthetic routes may not be feasible due to unfavourable kinetics or thermodynamics. To overcome this challenge, chemoinformatics researchers are working on methods to optimise reaction conditions and predict relevant physicochemical parameters for each reaction in a synthetic pathway. These efforts are essential to improve the accuracy and efficiency of retrosynthetic design and, ultimately, to enable the synthesis of complex molecules.

One approach to evaluating the feasibility of chemical reactions is the use of quantitative structure-property relationship (QSPR) models based on a graph representation of reactions known as the Condensed Graph of Reaction (CGR). This representation allows for the calculation of fragment descriptors for the reaction and the construction of QSPR models, as well as the incorporation of information about the reaction conditions to predict kinetic, thermodynamic, and yield parameters. However, a key challenge in QSPR modelling is the validation of the model, which can be affected by the fact that physical and chemical properties are often measured under various conditions, and the number of unique reaction transformations is much smaller than the number of records. This can lead to an overly optimistic assessment of predictions in conventional cross-validation techniques, as the test set reactions may be similar to the ones in the training set. Therefore, we introduce novel techniques for validating the predictions of QSPR models for new reaction transformations and in new conditions. These techniques were applied to the training and validation of three models designed to predict the logarithm of the reaction rate constant for SN₂, E₂, and Diels-Alder reactions.



Cross-validation strategies in QSPR modelling of chemical reactions

A. Rakhimbekova ^a, T.N. Akhmetshin ^{a,b}, G.I. Minibaeva ^a, R.I. Nugmanov ^a,
T.R. Gimadiev ^c, T.I. Madzhidov ^a, I.I. Baskin ^{a,d} and A. Varnek ^{b,c}

^aA.M. Butlerov Institute of Chemistry, Kazan Federal University, Kazan, Russia; ^bLaboratory of Chemoinformatics, UMR 7140 CNRS, University of Strasbourg, Strasbourg, France; ^cInstitute for Chemical Reaction Design and Discovery, Hokkaido University, Sapporo, Japan; ^dDepartment of Materials Science and Engineering, Technion – Israel Institute of Technology, Haifa, Israel

ABSTRACT

In this article, we consider cross-validation of the quantitative structure-property relationship models for reactions and show that the conventional k-fold cross-validation (CV) procedure gives an ‘optimistically’ biased assessment of prediction performance. To address this issue, we suggest two strategies of model cross-validation, ‘transformation-out’ CV, and ‘solvent-out’ CV. Unlike the conventional k-fold cross-validation approach that does not consider the nature of objects, the proposed procedures provide an unbiased estimation of the predictive performance of the models for novel types of structural transformations in chemical reactions and reactions going under new conditions. Both the suggested strategies have been applied to predict the rate constants of bimolecular elimination and nucleophilic substitution reactions, and Diels-Alder cycloaddition. All suggested cross-validation methodologies and tutorial are implemented in the open-source software package CIMtools (<https://github.com/cimm-kzn/CIMtools>).

ARTICLE HISTORY

Received 18 September 2020
Accepted 26 January 2021

KEYWORDS

Validation; QSPR; chemical reactions; rate constant prediction; reaction rate; structure-reactivity modelling

Introduction

Nowadays the external validation is considered as an integral component of any Quantitative structure-activity/property relationship (QSAR/QSPR) model, irrespective of the nature of the chemical objects under investigation [1–10]. It was several times pointed out that the correct validation of QSAR/QSPR models is essential [1–3,5,6,11]. It is common to distinguish internal validation, which is used for model selection, and external validation, used for the quality assessment [12]. In most cases, external validation is performed on a dedicated test set [2,11]. Its drawback is possible to bias due to random fluctuations, arbitrary or unfair selection of molecules to test set, which can be overcome by the rational division of the data set into training and test sets [13–18]. Tetko et al. [3] proposed external cross-validation as a more stable alternative to single the external test set. In this procedure, a part of the parent dataset is randomly placed to the external (outer) set used for assessing the predictive performance of the approach, while remaining objects are used for the model

CONTACT T.I. Madzhidov  Timur.Madzhidov@kpfu.ru

 Supplementary data for this article can be accessed at: <https://doi.org/10.1080/1062936X.2021.1883107>

© 2021 Informa UK Limited, trading as Taylor & Francis Group

building including hyperparameters selection based on internal cross-validation. Outer set loops over the whole data set guarantee the involvement of all data points in external prediction. In machine-learning, this technique is also known as nested cross-validation. Since an external (outer) test set is by no means used for model building, it is free from model selection bias [3,19] and, hence, can be used for assessment of model performance.

Data sets of complex chemical objects, such as chemical reactions, polymers, compounds' mixtures often include similar elements (e.g. the same reactant participating in different reactions). Therefore, correct external validation of QSPR models for these objects is much less straightforward compared to 'classical' QSAR/QSPR models for individual molecules [7–9,20,21]. Each of these cases requires designing a specific validation strategy. For the models predicting properties of binary mixtures, 'Mixtures Out', 'Compounds Out', 'Points Out', 'Everything Out' cross-validation strategies have been suggested [7,8]. The 'Donor Out', 'Acceptor Out', and 'Both out' strategies have been successfully applied to validate QSPR models for dissociation free energy of H-bond donor-acceptor complexes [9].

Nowadays, growing attention is attracted to statistical models predicting the kinetic or thermodynamic properties of chemical reactions [22–24]. Chemical reactions represent a complex object because they involve several molecular species of two types (reactants and products) and their properties depend on experimental conditions (solvent, catalyst, temperature, etc). In most of the studies, random split or cross-validation (CV) techniques were used for the validation of models for chemical reactions [25–33]. However, recently we published some studies demonstrating the flaws of conventional validation techniques for reaction characteristics modelling [20,21]. We demonstrated on the bimolecular rate constant data set that the error estimated in conventional CV (RMSE = 0.3 log k units) was lower than both estimated experimental error (about 0.5–1.0 log k units) and that computed for the external test set [20]. This can be explained by the fact that the data set included the same structural transformations of reactants to products studied under slightly different conditions. Since the reaction rate values of these reactions were often very close, they behaved as duplicates. Their presence in both training and test sets in CV loops inevitably leads to very optimistic estimates of predictive ability [20,21]. To overcome this problem, the model's performance was assessed using a subset of reactions studied under one sole condition (called unique data points, UDP) for which such kind of bias is not possible [20]. However, such validation is not a panacea: its results strongly depend on the fraction of UDP in the data set, which can potentially vary from 0 (all reactions were studied at different conditions) to 100% (all reactions were studied at one condition only). Thus, such a type of validation has limited applicability.

Polishchuk et al. [21] suggested the 'product-out' cross-validation strategy in which all reactions with the same main product are placed either in the training or test set for a particular fold. It has been shown that the ranking of models on different descriptors based on 'product-out' validation is significantly different in comparison with conventional cross-validation called 'reaction-out' CV.

Validation for QSAR modelling means the assessment of the predictive power of the model on novel datapoints. For molecular QSAR/QSPR studies, a test set on each cross-validation fold consists of molecules absent in the training set. However, the novelty issue is more complex in the case of reactions modelling and should account for both either

chemical transformation or experimental conditions. Therefore, here we propose two validation strategies that assess predictive performance for a particular novelty type:

- ‘transformation-out’ which estimates the ability to predict characteristics of chemical reactions with a novel reactant-product pair (hereafter we call it transformation),
- ‘solvent-out’ which assesses the quality of prediction for reactions proceeding in a new solvent.

Note that in ‘transformation-out’ validation reactions of the test set proceed under the same conditions as the reactions of the training set. In ‘solvent-out’ validation, the test set includes chemical transformations present in the training set, but solvents are different. The presence of completely new reactions, having both new transformations and solvents is more challenging for the model. In principle, such data points can be used for ‘both-out’ validation. However, they were very scarce or absent in the considered data sets.

As reactions rates are often measured under several conditions and at several temperatures and the number of unique transformations is much lower than the number of reactions (see Table 1), in conventional cross-validation, the test set reactions may have close neighbours in the training set which explains its higher Q^2 and lower RMSE compared to ‘transformation-out’ and ‘solvent-out’ validation.

Proposed validation methodologies have been applied to the modelling of the logarithm of S_N2 , E2, and Diels-Alder reactions rate constants ($\log k$). They have been implemented in the open-source CIMtools software package (<https://github.com/cimm-kzn/CIMtools>) developed for reaction modelling.

Materials and methods

Data set and descriptors

Three data sets for the following types of reactions were used in this study: bimolecular nucleophilic substitution (S_N2), bimolecular elimination (E2), Diels-Alder reactions (DA). These data sets were collected in our previous studies [20,30,31]. An external data set of 90 Menshutkin reactions (S_N2) were also used. The data set was collected and curated in our previous study [20]. Note that this data set comprised novel transformations and the same solvents as in the training set. Thus, it corresponds closely to the ‘transformation-out’ validation strategy. The data set characteristics are presented in Table 1.

Descriptors of the reactions

The descriptor vector for each reaction resulted from a concatenation of structural descriptors and parameters describing experimental conditions (solvent and temperature) as

Table 1. The data set characteristics.

Data set	Number of reactions	Number of unique transformations	Number of unique solvents	Ref.
S_N2	4830	1382	43	[20]
E2	1820	934	21	[30]
DA	1866	812	21	[31]
S_N2 (external)	90	48	3	[20]

proposed in paper [32]. The chemical transformations were encoded by the Condensed Graphs of Reaction (CGR). In the CGR approach, a reaction is represented by a single 2D graph, some sort of pseudomolecule that contains both conventional chemical bonds and so-called dynamic bonds characterizing changed/broken/formed chemical bonds [34,35]. Thus, CGR represents the whole transformation, i.e. reactant-product pair, as a single molecular graph. CGRtools library was used to generate CGRs [36]. ISIDA fragment descriptors were computed for CGR using the ISIDA Fragmentor [37] program. They represent the subgraphs of different topologies and sizes. Each subgraph is considered as a descriptor type whereas its occurrence in a molecule is the descriptor value. In this study, sequences of atoms and bonds containing from 2 to 4 atoms were considered. This fragmentation was successfully used in our previous modelling studies [20,30,31,33].

Descriptors of the reaction conditions

Each solvent was described by 15 descriptors that represent polarity, polarizability, H-acidity, and basicity: Catalan SPP [38], SA [39], and SB constants [38], Camlet-Taft constants α [40], β [41], and π^* [42], four functions depending on the dielectric constant, three functions depending on the refractive index as shown in paper [32]. The latter seven descriptors reflect the polarity and polarizability of the bulk of the solvent. The inverse absolute temperature, $1/T$ (in Kelvin degrees) was also used as a descriptor of temperature influence. Since some of the solvents were a water-organic mixture, the molar ratio of organic solvent was used as a descriptor as well (100% for pure solvent).

Model building and validation

The general procedure of modelling

The models were built using random forest (denoted as RF) approach implemented in the scikit-learn library [43]. The number of trees was equal to 500 in all cases, the optimized hyperparameter was the values of features selected upon tree branching (option `max_features`). The rest parameters were set to their default values.

The predictive performance of the best models was estimated using the nested cross-validation technique. In this approach, an outer (external validation) loop split the initial data set into an external test set (used for assessment of the model performance) and a modelling set (used for the model building including internal cross-validation). Here, three strategies of such split have been tested: conventional random 5-fold cross-validation (denoted as 'reaction-out' CV), 'transformation-out' CV, and 'solvent-out' CV (see below). A hyperparameter (the number of features to consider when looking for the best split) of the Random Forest regression was optimized on modelling set in the conventional 5-fold cross-validation using grid search. Its best value found in internal cross-validation was used to build a model on the whole modelling set, followed by the application of resulting models to the external test set. Predictions for the objects in the external test set folds were merged and then performance metrics (determination coefficient, r^2 , corresponding to Q^2_{F2} formulae in Roy et al. [5], and RMSE) were calculated. Notice, that performance on 'reaction-out' CV, 'transformation-out' CV, and 'solvent-out' CV reflects predictive ability on the external test set.

Strategies for model external validation in QSPR for reaction datasets

Three types of cross-validation strategies were applied. In the cross-validation procedure, the initial data set was divided into a given number of subsets, corresponding to the desired number of folds. Each subset was sequentially considered as an external test set, while the rest was used as the modelling set. The schematic representation of 'reaction-out', 'transformation-out', and 'solvent-out' CV is shown in Figure 1.

The 'Reaction-out' CV approach was simply a regular five times repeated five-fold CV. The sizes of test sets in it are almost equal.

'Transformation-out' CV approach was implemented as k-fold cross-validation (Figure 1). In the 'transformation-out' procedure, all reactions having the same CGR were placed into the same fold (different shapes in Figure 1). The implemented splitting algorithm tried to make the folds approximately equal in size. Therefore, a fold might contain a group of several CGRs (as presented in fold 2, Figure 1). Moreover, the test set contained reactions proceeding in the solvents presented also in the training set, see Figure 1. This allowed us to avoid a bias due to the presence of new solvents in the test set. Thus, unique reactions (corresponding to a unique combination of transformation and solvent) were always placed into the training set, see Figure 1. Since in 'transformation-out' validation all reactions having the same reactants and products were placed into the same subset (Figure 1), this only showed how well reactions with new reactants and products, i.e. CGRs, were predicted.

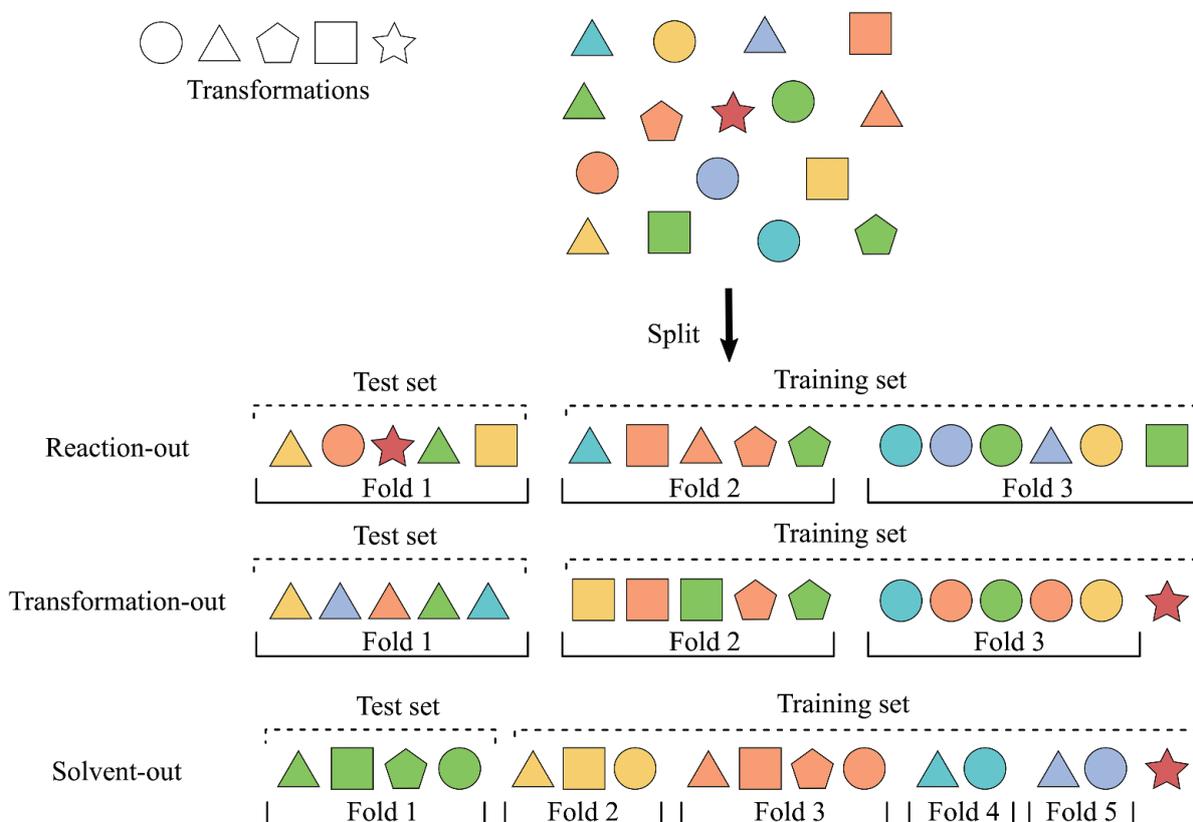


Figure 1. The schematic representation of 'reaction-out', 'transformation out', and 'solvent out' cross-validation. Each shape characterizes a CGR, and different colours indicate the solvent used. The red star is a unique reaction. The unique reactions are excluded from the test set and always complement the training set.

The ‘Solvent-out’ validation approach was implemented as a leave-one-solvent-out (Figure 1), as the number of solvent types is usually low (less than 50). An additional hurdle to the application of k-fold validation was caused by a great imbalance in the number of reactions corresponding to one solvent. In ‘solvent-out’ validation, all reactions carried out in the same solvent were placed into the same subset which is sequentially used as the test set. Each reaction in the test set should have a counterpart in the training set with the same CGR but proceed in a different solvent. Unique reactions (in this case, reactions measured in a single solvent) were always included in the training set and never used in the test set. This method avoids underestimating the model performance because such reactions represent new CGR and new solvent for the trained model. It is worth noting that in this study we have grouped the folds by solvents only, but the developed algorithm can group following several conditions.

Results and discussion

The models were built using RF and fragment descriptors of CGRs and validated using three strategies described above, i.e. ‘reaction-out’ CV, ‘transformation-out’ CV, and ‘solvent-out’ CV. The performance of models for three data sets obtained using ‘reaction-out’ CV, ‘transformation-out’ CV, and ‘solvent-out’ CV strategies are shown in Table 2. In all three cases, the prediction performance metrics for ‘reaction-out’ CV validation were better than for ‘transformation-out’ CV. In ‘reaction-out’ CV strategy reactions having the same reactants and products were simultaneously present in both training and test set. Two reactions with very similar conditions can be present in the training and test set, and thus the prediction performance is overoptimistic. In the ‘transformation-out’ CV strategy, all reactions with the same structural transformation were present in either the training or the test sets, but not both simultaneously. Therefore, RMSE values are bigger than for the ‘reaction-out’ CV strategy (Table 2), but they remain at acceptable levels.

The independent external set consisting of Menshutkin reactions (S_N2) was then used for model validation as well. Prediction on the external set of reactions was used for comparison with ‘reaction-out’ and ‘transformation-out’ validation strategies. The data set contained 48 Menshutkin reactions which had new transformations in comparison with the training set but were carried out in known solvents (corresponding to the training set). As expected, the results on the ‘transformation-out’ validation are close to the one of external validation ($r^2 = 0.51$, $RMSE = 0.99 \log k$ units on the external test set), confirming

Table 2. Coefficient of determination (r^2) and RMSE for different external validation strategies of QSPR for three reaction data sets.

Data set	Strategy of validation					
	‘Reaction-out’		‘Transformation-out’		‘Solvent-out’	
	r^2	RMSE, $\log k$ units	r^2	RMSE, $\log k$ units	r^2	RMSE, $\log k$ units
S_N2	0.83	0.48	0.58	0.76	0.39	0.98
E2	0.73	0.77	0.56	0.98	0.14	1.29
DA	0.85	0.73	0.71	1.04	0.76	0.94

that ‘transformation-out’ validation is a more rigorous and unbiased approach for validating QSPR models of reactions, and, therefore, reliably assesses the accuracy of predictions for novel structural transformations of reactions.

The prediction performance metrics in ‘solvent-out’ validation were quite different in all three cases. Reasonable statistical parameters were observed for the Diels-Alder reactions data set ($r^2 = 0.76$ and $RMSE = 0.94 \log k$ units, respectively). Surprisingly, ‘solvent-out’ validation for bimolecular nucleophilic substitution reaction and bimolecular elimination reaction data sets led to much worse statistical parameters than for DA data set. We suggest that such results in the case of Diels-Alder reactions are due to the fact that most of the reactions in the data set (86%) were carried out in non-polar solvents, moreover, 74% of the reactions were carried out in toluene, benzene, and chlorobenzene (53%, 11%, and 10%, respectively). In the remaining data sets, the reactions were carried out in various solvents of different nature. Moreover, since these reactions have a non-polar transition state, the solvent effect should affect its rate much.

The obtained results in ‘solvent-out’ external validation on the bimolecular nucleophilic substitution reaction data set are discussed in more detail. The data set of S_N2 reactions consists of 43 different solvents where the most popular solvents were ethanol (17.9%), methanol (15.4%), nitrobenzene (13.7%), acetone (12.7%) often used in mixtures with water – more than half the database (59.7%), [Figure 2](#). The total percentage of all reactions carried out in the rest 31 solvents that aren’t shown in the diagram ([Figure 2](#)) was only 9.1%.

The number of reactions carried out in particular solvents, which were used as a test set in ‘solvent-out’ validation, ranged from 1 to 422. Note that in ‘solvent-out’ validation the reactions are not included in the test set if there is no data on rate constant for the same transformations in other solvents. The values of RMSE for all solvents (folds of ‘leave-one-solvent-out’ external CV) are given in [Table 3](#) and discussed below. The same tables for DA and E2 data sets are given in Supplementary Materials (see Tables S1 and S2 in Supplementary materials).

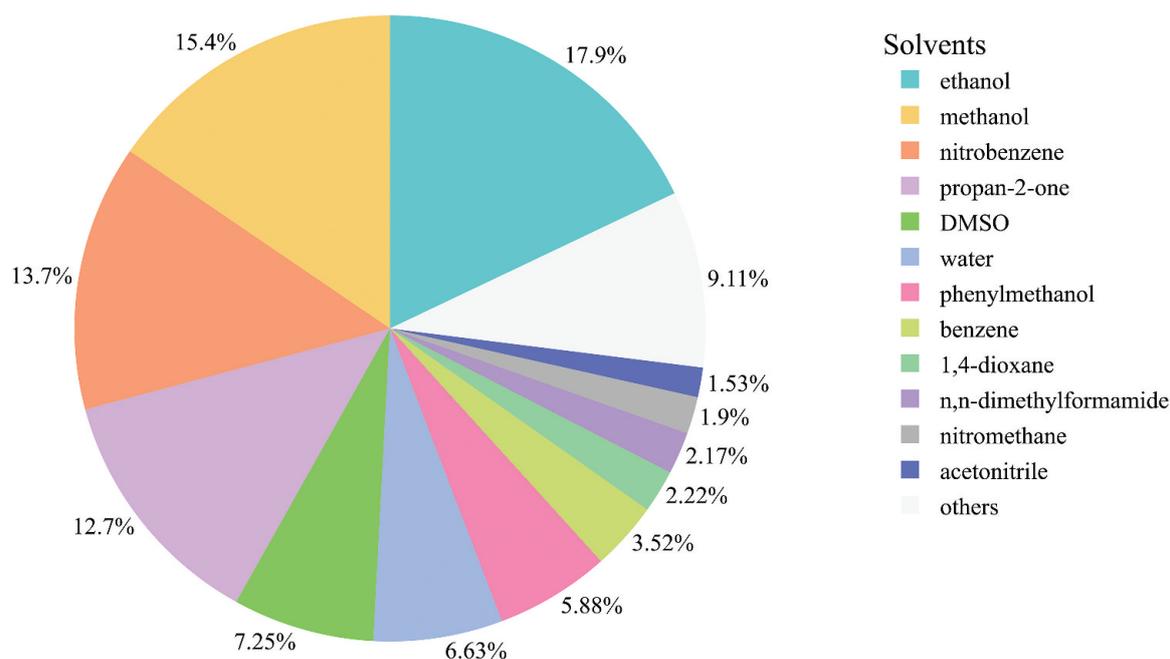


Figure 2. The percentage of reactions carried out in the most popular solvents for the S_N2 data set.

Table 3. Results of external ‘solvent-out’ validation of developed QSPR models. Each fold is characterized by one solvent. Only solvents used in 10+ reactions were shown.

Folds/solvents	Number of reactions in the test set	Number of unique transformations	r^2	RMSE, log k units
Methanol*	422	167	0.406	0.847
Acetone*	327	84	0.540	0.854
Ethanol*	305	89	0.676	0.593
Nitrobenzene	284	87	-0.644	1.481
Dimethyl sulphoxide *	173	25	-0.380	1.371
Benzene	137	53	0.520	0.745
Water	113	36	-0.146	1.039
1,4-Dioxane*	80	27	0.632	0.542
Dimethylformamide	74	36	0.518	1.206
Acetonitrile*	56	41	-0.193	0.973
Nitromethane	40	14	0.140	0.617
Oxolane*	34	14	0.741	0.415
Phenylmethanol	28	12	0.481	0.434
Chlorobenzene	26	15	0.658	0.507
Butan-1-ol	26	11	0.819	0.331
Propan-2-ol	23	16	0.788	0.431
Propan-1-ol	17	9	0.828	0.318
Bromobenzene	16	10	0.645	0.442
Butan-2-one	15	7	0.725	0.194
Toluene	15	9	0.250	0.623
Cyclohexane	14	12	-2.215	1.491
Sulfolane*	14	5	0.175	0.231
Anisole	11	7	0.699	0.186
Heptan-1-ol	11	4	0.851	0.177
3-Methylbutan-1-ol	10	6	0.126	0.538

*mixtures with water

Generally, we found no clear dependence between solvent type, use of water-organic mixtures as a solvent, size of the data set, and the prediction of RMSE for S_N2 reactions. Solvents, for which a lot of rate data exist, are characterized by a larger RMSE (from 0.59 to 1.52). At the same time, a poor prediction is common also for the least popular solvents. Among 8 solvents that are often used in mixtures with water 5 (methanol, ethanol, dimethyl sulphoxide, acetone, acetonitrile) are poorly predicted but 3 (dioxane, sulfolane, oxolane) are predicted well. Prediction error in alcohols has some trend: heavy alcohols (more than 2 carbons) have low values of RMSE (from 0.09 to 0.45), ethanol has a greater error (0.590) and methanol has the greatest (0.82). It is worth noting that water has an even greater error (about 1.02).

We hypothesize that such specificity of ‘solvent-out’ validation can be explained by the insufficient generalization ability of the RF approach, probably due to the application of a large number of weakly informative structural descriptors and a few highly informative solvent descriptors. To support this hypothesis, the descriptors used for trees’ branching were analysed, [Figure 3](#). One can see that RF pays a lot of attention to solvent descriptors: they are selected for branching in about 20–30% of trees, depending on the level of the tree ([Figure 3](#)), while solvent descriptors constitute only 5% of the descriptor set. So, poor prediction of solvents cannot be explained by ignorance of solvent features by the model. It can be seen from [Figure 3](#) that RF is more likely to select solvent descriptors as branching criteria at the first and second level of the tree (i.e. at the root node of the decision tree or right after it) and to use fragment descriptors more frequently in levels 3 and lower ([Figure 3](#)).

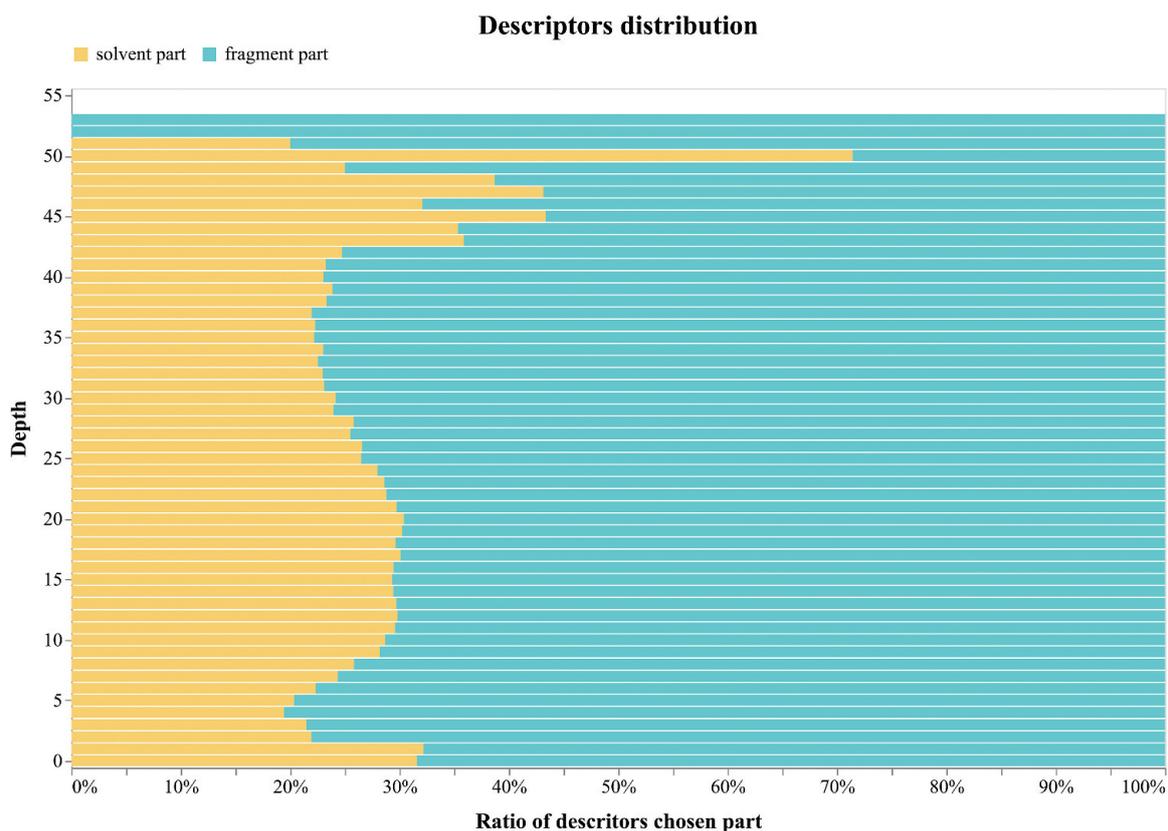


Figure 3. The percentage of solvent and fragment descriptors used for branching at particular depth levels in decision trees inside trained on S_N2 reactions RF model. The root node in the decision tree was considered as having depth 0. Notice that only a few trees had depth greater than 40 which caused spurious fluctuations. Temperature descriptor was ignored.

We believe such a selection of descriptors can be explained as RF tends to internally find correlations within a particular solvent type based on structural descriptors of transformation, creating a kind of family of submodels for each particular solvent type(s). Thus, the model does not try to generalize solvent influence, and RF predicts reactions in new solvents poorer than with new transformations. It also explains the mentioned specificity of alcohols prediction. Heavy alcohols are predicted similarly by the same implicit RF submodel based on the rest of alcohols but since methanol and ethanol are quite different from the rest, their prediction is poorer.

Our test on other machine-learning methods (support vector machines, neural nets) and other fragment descriptors showed similar results: models had a rather poor 'solvent-out' validation performance. It seems that the problem lies mainly in the application of fragment descriptors, but it would be the subject of specific research.

Conclusion

Validation of QSPR models is a very important aspect to understand the reliability of the models for the prediction of new objects not present in the data set. In the case of chemical reactions, test set objects can have new structural transformations or proceed under new conditions (new solvents, new additives, new catalysts, etc.). Thus, different types of novelties can be considered. Model performance on conventional cross-validation that

does not have control over the constitution of the test set has unclear meaning in the case of chemical reaction modelling. To evaluate the ability of the model predicting property of chemical reaction, two strategies of model validation, 'transformation-out' and 'solvent-out', have been suggested. 'Transformation-out' validation provides an estimation of the predictive performance of the models for novel types of structural transformations in chemical reactions and 'solvent-out' CV – for reactions going under new solvents. It is worth noting that 'solvent-out' validation is a special case of 'condition-out' validation, i.e. we can group folds not only by solvents but also by other conditions (new additives, new catalysts, a combination of several conditions). It has been shown that performance on 'transformation-out' validation is similar to the external set one containing new transformations. Thus, the proposed two validation strategies are recommended to be used for an unbiased evaluation of reaction-property models.

On bimolecular nucleophilic substitution reaction and bimolecular elimination reaction data sets, we revealed that the reaction-property models better predict the rate constant for new structural transformations than the rate constants for reactions occurring in novel solvents. It was explained by the inability of applied machine learning methods (RF, support vector machines, neural nets) and different fragment descriptors to correctly generalize dependency of the reaction rate for wide solvent types. But it will require specific research that we plan to be done in the nearest future.

Proposed validation strategies and a tutorial are implemented into open-source CIMtools library for structure-property modelling available on GitHub (<https://github.com/cimm-kzn/CIMtools>).

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by Russian Science Foundation grant No 19-73-10137.

ORCID

A. Rakhimbekova  <http://orcid.org/0000-0002-6820-6385>
T.N. Akhmetshin  <http://orcid.org/0000-0002-2549-6431>
G.I. Minibaeva  <http://orcid.org/0000-0001-7964-4842>
R.I. Nugmanov  <http://orcid.org/0000-0002-8541-9681>
T.R. Gimadiev  <http://orcid.org/0000-0001-5012-0308>
T.I. Madzhidov  <http://orcid.org/0000-0002-3834-6985>
I.I. Baskin  <http://orcid.org/0000-0003-0874-1148>
A. Varnek  <http://orcid.org/0000-0003-1886-925X>

References

- [1] M. Balls, B.J. Blaauboer, J.H. Fentem, L. Bruner, R.D. Combes, B. Ekwall, R.J. Fielder, A. Guillouzo, R.W. Lewis, D.P. Lovell, C.A. Reinhardt, G. Repetto, D. Sladowski, H. Spielmann, and F. Zucco,

- Practical aspects of the validation of toxicity test procedures*, *Altern. Lab. Anim.* 23 (1995), pp. 129–146. doi:10.1177/026119299502300116.
- [2] A. Tropsha, P. Gramatica, and V. Gombar, *The importance of being earnest: Validation is the absolute essential for successful application and interpretation of QSPR models*, *QSAR Comb. Sci.* 22 (2003), pp. 69–77. doi:10.1002/qsar.200390007.
- [3] I.V. Tetko, V.P. Solov'ev, A.V. Antonov, X. Yao, J.P. Doucet, B. Fan, F. Hoonakker, D. Fourches, P. Jost, N. Lachiche, and A. Varnek, *Benchmarking of linear and nonlinear approaches for quantitative structure–property relationship studies of metal complexation with ionophores*, *J. Chem. Inf. Model.* 46 (2006), pp. 808–819. doi:10.1021/ci0504216.
- [4] R. Veerasamy, H. Rajak, A. Jain, S. Sivadasan, C.P. Varghese, and R.K. Agrawal, *Validation of QSAR models - Strategies and importance*, *Int. J. Drug Discov.* 2 (2011), pp. 511–519.
- [5] K. Roy, S. Kar, and R.N. Das, *Validation of QSAR models, in Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment*, K. Roy (ed.), Elsevier, San Diego, 2015, pp. 231–289.
- [6] P. Gramatica and A. Sangion, *A historical excursus on the statistical validation parameters for QSAR models: A clarification concerning metrics and terminology*, *J. Chem. Inf. Model.* 56 (2016), pp. 1127–1131. doi:10.1021/acs.jcim.6b00088.
- [7] I. Oprisiu, E. Varlamova, E. Muratov, A. Artemenko, G. Marcou, P. Polishchuk, V. Kuz'min, and A. Varnek, *QSPR approach to predict nonadditive properties of mixtures. Application to bubble point temperatures of binary mixtures of liquids*, *Mol. Inform.* 31 (2012), pp. 491–502. doi:10.1002/minf.201200006.
- [8] E. Muratov, E.V. Varlamova, V.E. Kuzmin, A.G. Artemenko, N.N. Muratov, S. Mileyko, D. Fourches, and A. Tropsha, *Everything out validation approach for qsar models of chemical mixtures*, *J. Clin. Pharm.* 1 (2014), pp. 1005.
- [9] M. Glavatskikh, T. Madzhidov, V. Solov'ev, G. Marcou, D. Horvath, and A. Varnek, *Predictive models for the free energy of hydrogen bonded complexes with single and cooperative hydrogen bonds*, *Mol. Inform.* 35 (2016), pp. 629–638. doi:10.1002/minf.201600070.
- [10] A.O. Aptula, N.G. Jeliakova, T.W. Schultz, and M.T.D. Cronin, *The better predictive model: High q^2 for the training set or low root mean square error of prediction for the test set?* *QSAR Comb. Sci.* 24 (2005), pp. 385–396. doi:10.1002/qsar.200430909.
- [11] A. Golbraikh and A. Tropsha, *Beware of q^2 !*, *J. Mol. Graph. Model.* 20 (2002), pp. 269–276. doi:10.1016/S1093-3263(01)00123-1.
- [12] P. Gramatica, *Principles of QSAR models validation: Internal and external*, *QSAR Comb. Sci.* 26 (2007), pp. 694–701. doi:10.1002/qsar.200610151.
- [13] J. Gasteiger and J. Zupan, *Neural Networks in Chemistry*, *Angew. Chemie Int. Ed. English* 32 (1993), pp. 503–527. doi:10.1002/anie.199305031.
- [14] J. Huuskonen, *QSAR modeling with the electrotopological state: TIBO derivatives*, *J. Chem. Inf. Comput. Sci.* 41 (2001), pp. 425–429. doi:10.1021/ci0001435.
- [15] I.V. Tetko, V.V. Kovalishyn, and D.J. Livingstone, *Volume learning algorithm artificial neural networks for 3D QSAR studies*, *J. Med. Chem.* 44 (2001), pp. 2411–2420. doi:10.1021/jm010858e.
- [16] M. Snarey, N.K. Terrett, P. Willett, and D.J. Wilton, *Comparison of algorithms for dissimilarity-based compound selection*, *J. Mol. Graph. Model.* 15 (1997), pp. 372–385. doi:10.1016/S1093-3263(98)00008-4.
- [17] A. Golbraikh, *Molecular dataset diversity indices and their applications to comparison of chemical databases and QSAR analysis*, *J. Chem. Inf. Comput. Sci.* 40 (2000), pp. 414–425. doi:10.1021/ci990437u.
- [18] C. Szántai-Kis, I. Kövesdi, G. Kéri, and L. Örfi, *Validation subset selections for extrapolation oriented QSPAR models*, *Mol. Divers.* 7 (2003), pp. 37–43. doi:10.1023/B:MODI.0000006538.99122.00.
- [19] D. Baumann and K. Baumann, *Reliable estimation of prediction errors for QSAR models under model uncertainty using double cross-validation*, *J. Cheminform.* 6 (2014), pp. 47. doi:10.1186/s13321-014-0047-1.

- [20] T. Gimadiev, T. Madzhidov, I. Tetko, R. Nugmanov, I. Casciuc, O. Klimchuk, A. Bodrov, P. Polishchuk, I. Antipinn, and A. Varnek, *Bimolecular nucleophilic substitution reactions: Predictive models for rate constants and molecular reaction pairs analysis*, Mol. Inform. 38 (2019), pp. 1800104. doi:10.1002/minf.201800104.
- [21] P. Polishchuk, T. Madzhidov, T. Gimadiev, A. Bodrov, R. Nugmanov, and A. Varnek, *Structure–reactivity modeling using mixture-based representation of chemical reactions*, J. Comput. Aided. Mol. Des. 31 (2017), pp. 829–839. doi:10.1007/s10822-017-0044-3.
- [22] O. Engkvist, P.-O. Norrby, N. Selmi, Y. Lam, Z. Peng, E.C. Sherer, W. Amberg, T. Erhardn, and L. A. Smyth, *Computational prediction of chemical reactions: Current status and outlook*, Drug Discov. Today 23 (2018), pp. 1203–1218. doi:10.1016/j.drudis.2018.02.014.
- [23] H. Gao, T.J. Struble, C.W. Coley, Y. Wang, W.H. Green, and K.F. Jensen, *Using machine learning to predict suitable conditions for organic reactions*, ACS Cent. Sci. 4 (2018), pp. 1465–1476. doi:10.1021/acscentsci.8b00357.
- [24] I.I. Baskin, T.I. Madzhidov, I.S. Antipin, and A.A. Varnek, *Artificial intelligence in synthetic chemistry: Achievements and prospects*, Russ. Chem. Rev. 86 (2017), pp. 1127–1156. doi:10.1070/RCR4746.
- [25] D.T. Ahneman, J.G. Estrada, S. Lin, S.D. Dreher, and A.G. Doyle, *Predicting reaction performance in C–N cross-coupling using machine learning*, Science 360 (2018), pp. 186–190. doi:10.1126/science.aar5169.
- [26] F. Sandfort, F. Strieth-Kalthoff, M. Kühnemund, C. Beecks, and F. Glorius, *A structure-based platform for predicting chemical reactivity*, Chem 6 (2020), pp. 1379–1390. doi:10.1016/j.chempr.2020.02.017.
- [27] J.A. Kammeraad, J. Goetz, E.A. Walker, A. Tewari, and P.M. Zimmerman, *What does the machine learn? Knowledge representations of chemical reactivity*, J. Chem. Inf. Model. 60 (2020), pp. 1290–1301. doi:10.1021/acs.jcim.9b00721.
- [28] A.A. Kravtsov, P.V. Karpov, I.I. Baskin, V.A. Palyulin, and N.S. Zefirov, *Prediction of rate constants of SN2 reactions by the multicomponent QSPR method*, Dokl. Chem. 440 (2011), pp. 299–301. doi:10.1134/S0012500811100107.
- [29] A.A. Kravtsov, P.V. Karpov, I.I. Baskin, V.A. Palyulin, and N.S. Zefirov, *Prediction of the preferable mechanism of nucleophilic substitution at saturated carbon atom and prognosis of SN1 rate constants by means of QSPR*, Dokl. Chem. 441 (2011), pp. 314–317. doi:10.1134/S0012500811110048.
- [30] T.I. Madzhidov, A.V. Bodrov, T.R. Gimadiev, R.I. Nugmanov, I.S. Antipin, and A.A. Varnek, *Structure–reactivity relationship in bimolecular elimination reactions based on the condensed graph of a reaction*, J. Struct. Chem. 56 (2015), pp. 1227–1234. doi:10.1134/S002247661507001X.
- [31] T.I. Madzhidov, T.R. Gimadiev, D.A. Malakhova, R.I. Nugmanov, I.I. Baskin, I.S. Antipin, and A. A. Varnek, *Structure–reactivity relationship in Diels–Alder reactions obtained using the condensed reaction graph approach*, J. Struct. Chem. 58 (2017), pp. 650–656. doi:10.1134/S0022476617040023.
- [32] T.I. Madzhidov, P.G. Polishchuk, R.I. Nugmanov, A.V. Bodrov, A.I. Lin, I.I. Baskin, A.A. Varnek, and I.S. Antipin, *Structure–reactivity relationships in terms of the condensed graphs of reactions*, Russ. J. Org. Chem. 50 (2014), pp. 459–463. doi:10.1134/S1070428014040010.
- [33] T.R. Gimadiev, T.I. Madzhidov, R.I. Nugmanov, I.I. Baskin, I.S. Antipin, and A. Varnek, *Assessment of tautomer distribution using the condensed reaction graph approach*, J. Comput. Aided. Mol. Des. 32 (2018), pp. 401–414. doi:10.1007/s10822-018-0101-6.
- [34] A. Varnek, D. Fourches, F. Hoonakker, and V.P. Solov'ev, *Substructural fragments: An universal language to encode reactions, molecular and supramolecular structures*, J. Comput. Aided. Mol. Des. 19 (2005), pp. 693–703. doi:10.1007/s10822-005-9008-0.
- [35] F. Hoonakker, N. Lachiche, A. Varnek, and A. Wagner, *Condensed graph of reaction: Considering a chemical reaction as one single pseudo molecule*, Int. J. Artif. Intell. Tools 20 (2011), pp. 253–270. doi:10.1142/S0218213011000140.
- [36] R.I. Nugmanov, R.N. Mukhametgaleev, T. Akhmetshin, T.R. Gimadiev, V.A. Afonina, T. I. Madzhidov, and A. Varnek, *CGRtools: Python library for molecule, reaction, and condensed*

- graph of reaction processing*, J. Chem. Inf. Model. 59 (2019), pp. 2516–2521. doi:10.1021/acs.jcim.9b00102.
- [37] A. Varnek, D. Fourches, D. Horvath, O. Klimchuk, C. Gaudin, P. Vayer, V. Solov'ev, F. Hoonakker, I. Tetko, and G. Marcou, *ISIDA - Platform for virtual screening based on fragment and pharmacophoric descriptors*, Curr. Comput. Aided-Drug Des. 4 (2008), pp. 191–198. doi:10.2174/157340908785747465.
- [38] J. Catalán, V. López, P. Pérez, R. Martin-Villamil, and J.-G. Rodríguez, *Progress towards a generalized solvent polarity scale: The solvatochromism of 2-(dimethylamino)-7-nitrofluorene and its homomorph 2-fluoro-7-nitrofluorene*, Liebigs Ann. 1995 (1995), pp. 241–252. doi:10.1002/jlac.199519950234.
- [39] J. Catalán, C. Díaz, and A. Generalized Solvent, *Acidity Scale: The Solvatochromism of tert-Butylstilbazolium Betaine Dye and Its Homomorph, o'-Di-tert-butylstilbazolium Betaine Dye*, Liebigs Ann. 1997 (1997), pp. 1941–1949. doi:10.1002/jlac.199719970921.
- [40] R.W. Taft and M.J. Kamlet, *The solvatochromic comparison method. 2. The .alpha.-scale of solvent hydrogen-bond donor (HBD) acidities*, J. Am. Chem. Soc. 98 (1976), pp. 2886–2894. doi:10.1021/ja00426a036.
- [41] M.J. Kamlet and R.W. Taft, *The solvatochromic comparison method. I. The .beta.-scale of solvent hydrogen-bond acceptor (HBA) basicities*, J. Am. Chem. Soc. 98 (1976), pp. 377–383. doi:10.1021/ja00418a009.
- [42] M.J. Kamlet, J.L. Abboud, and R.W. Taft, *The solvatochromic comparison method. 6. The .pi.* scale of solvent polarities*, J. Am. Chem. Soc. 99 (1977), pp. 6027–6038. doi:10.1021/ja00460a031.
- [43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, *Scikit-learn: Machine Learning in Python*, J. Mach. Learn. Res. 12 (2011), pp. 2825–2830.

Supporting Information

Table S1. Results of ‘solvent-out’ validation of developed QSPR models for Diels-Alder reaction data set. Each fold is characterized by one solvent

Folds/solvents	Number of reactions in test set	Number of unique transformations	Q ²	RMSE, logK units
toluene	112	38	0.82	0.67
chlorobenzene	91	43	0.93	0.56
benzene	73	28	0.87	0.72
1,4-dioxane	61	34	0.35	2.01
1,2-dichloroethane	37	35	0.83	0.92
trichloromethane	19	16	0.89	0.67
ethyl acetate	17	15	0.61	0.79
acetonitrile	15	11	0.46	0.83
1,4-epoxybutane	12	11	0.65	0.54
propan-2-one	9	7	0.96	0.11
methoxybenzene	9	9	0.90	0.38
dichloromethane	8	8	0.88	0.38
tetrachloromethane	8	8	0.87	0.57
ethanol	7	3	0.84	0.47
nitrobenzene	7	7	0.87	0.58
methanesulfinylmethane	4	2	0.90	0.14
cyclohexane	4	4	0.28	1.73
bromobenzene	2	2	0.83	0.53
nitromethane	2	2	-7.99	0.52
pentan-1-ol	1	1	0.00	0.08
acetic acid	1	1	0.00	0.37

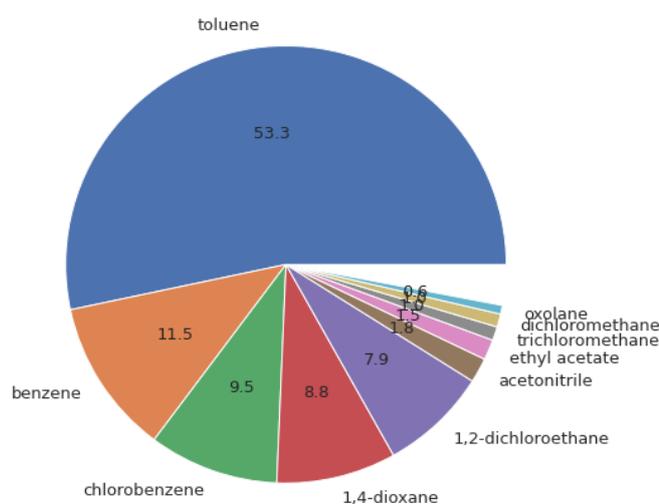


Figure S1. The percentage of Diels-Alder reactions carried out in the most popular solvents

Table S2. Results of ‘solvent-out’ validation of developed QSPR models for bimolecular elimination reaction data set. Each fold is characterized by one solvent

Folds/solvents	Number of reactions in test set	Number of unique transformations	Q ²	RMSE, logK units
ethanol	60	17	0.51	0.75
methanol	58	15	-0.27	1.11
ethane-1,2-diol	46	2	-2.74	1.05
water	44	18	-1.14	1.81
methanesulfinylmethane	32	4	-1.94	2.33
1,4-dioxane	20	14	-0.17	1.58
acetonitrile	18	9	-0.67	1.40
benzene	17	9	-0.33	1.80
propan-2-one	14	10	-0.51	0.89
formamide	12	10	-0.17	0.89
2-methylpropan-2-ol	4	4	0.80	0.44
propan-2-ol	3	1	-7.23	1.09
1,3-dimethylbenzene	3	1	-15.25	1.17
chlorobenzene	3	3	0.65	0.19
trichloromethane	3	3	-1.69	0.87
ethyl acetate	3	3	-2.15	0.54
nitromethane	2	2	-12.07	0.34
hexamethylphosphoramide	2	2	-2.13	0.08

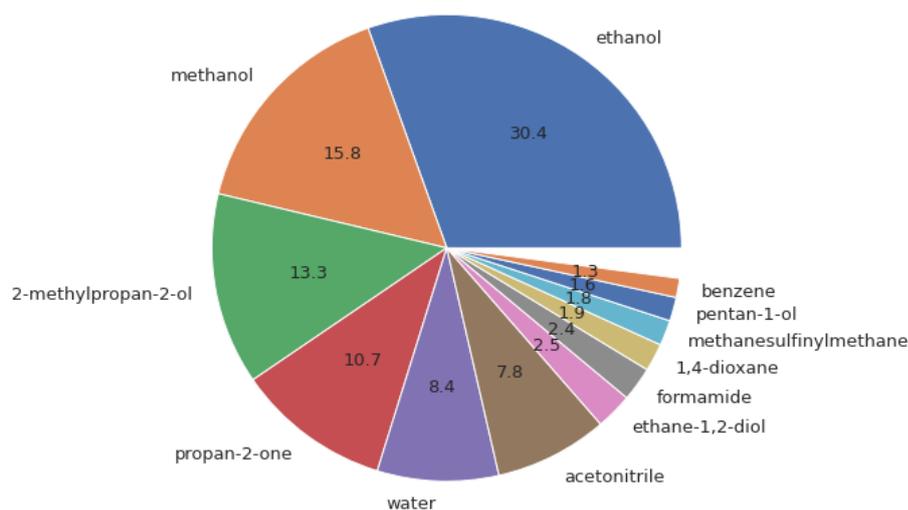


Figure S2. The percentage of E2 reactions carried out in the most popular solvents

4.3.1 Summary for the section 4.3

In this section, we presented novel validation techniques for quantitative structure-property relationship (QSPR) models, which are used to predict the properties of chemical reactions. These techniques, known as “transformation-out” and “solvent-out”, allow for the evaluation of the model’s performance on novel reaction transformations and conditions, respectively. Our analysis of bimolecular nucleophilic substitution and bimolecular elimination reactions showed that the models performed better at predicting the rate constants for new structural transformations than for reactions occurring under novel solvents. This is likely due to the limitations of the machine learning methods and fragment descriptors used, which may not accurately capture the dependency of the reaction rate on a wide range of solvents. Our findings also highlight the need for further research to improve the generalisation of machine learning models to diverse reaction conditions. Overall, the “transformation-out” and “solvent-out” validation strategies presented in this work provide an unbiased evaluation of reaction-property models and should be considered in future studies.

Chapter 5

Conclusions and perspectives

This thesis presents a potential solution to the current challenge in the discovery of synthetically feasible molecular structures with desired properties based on the combination of generative approaches with retrosynthetic search methods, both involving graph-based neural networks.

We encountered two major problems when using deep learning methods to generate new molecular structures: computational efficiency and atom order bias. The computational efficiency problem was tackled by introducing the Hydrogen-count labelled graph (HLG) representation for molecules and related HyFactor autoencoder architecture. Compared to an architecture based on the conventional molecular graph representation (ReFactor), the application of HLG reduces up to 20% of trainable parameters without any performance loss. For a given seed structure, HyFactor generates more similar structures than ReFactor.

Since the encoder part of HyFactor is not permutation invariant, it leads to order dependence in the autoencoder latent space. In order to address this issue, a new Vector Quantised Graph-based AutoEncoder (VQGAE) architecture using permutation invariant operations at the encoding stage was proposed. A good performance in structure reconstruction and generation was achieved due to the application of the vector quantisation technique, allowing to learn the most popular atom-centred fragment vectors. The efficiency of VQGAE latent vectors in similarity search and QSAR modelling was confirmed in a series of benchmarking calculations. Finally, VQGAE was tested in the inverse QSAR task, generating new potent inhibitors against the A2A adenosine receptor target.

Retrosynthetic algorithms typically consist of two main components: retrosynthetic transformation rules and a search strategy which are critical for the performance of the retrosynthetic analysis. Since the quality of the extracted rules strongly depends on the quality of the reaction data, a protocol for reaction data curation was proposed and applied to several widely used reaction databases - USPTO, Reaxys and Pistachio. Since only USPTO was publicly available, we used its standardised and filtered version for the retrosynthetic rules extraction.

Inspired by recent advances in retrosynthesis tools based on Monte-Carlo tree search (MCTS), we developed the Graph Self Learning retrosynthetic tool (GSLRetro). Unlike previous studies implementing a combination of MCTS with heuristic evaluation functions, we focused on the development of a self-learning approach combining MCTS with graph neural networks. Our experiments demonstrated that GSLRetro with a trained value network significantly increased the number of synthetic pathways compared to those found with random and rollout heuristic functions. Moreover, compared to the state-of-the-art tool AiZynthFinder, GSLRetro was able to find 20% more synthetic routes for complex molecules. GSLRetro was successfully used to find synthetic pathways for molecular structures previously generated by VQGAE.

Reaction yield depends on the reaction's thermodynamic and kinetic parameters, particularly the reaction rate constant. To objectively assess the performance of related ML models, we proposed two new validation scenarios for predicting the reaction rate constant for new transformations ("transformation-out" validation) and new reaction conditions ("solvent-out" validation). These scenarios were tested on three manually-collected SN₂, E₂, and Diels-Alder reaction datasets. Our analysis showed that conventional cross-validation overestimates the performance of the ML models, as the same transformations or conditions are often present in the validation set.

Perspectives The developments presented in this thesis mark just the beginning of the journey towards the fully automated design of new molecular structures. We have, indeed, succeeded in developing and implementing some new search strategies in the chemical space, but still, there is a room for their improvement. In the realm of "de novo" molecular design using neural networks, the potential of the developed HLG and VQGAE has yet to be fully explored. The HLG representation was tested in the unsupervised learning task, but its performance in supervised tasks (QSAR/QSPR modelling) needs to be investigated. The ordering network of VQGAE generative part may fail to reconstruct the canonical order of atoms, which needs to be improved. Another point concerns the systematic analysis (size, chemical content) of the learned vector representation of fragments by VQGAE. We also expect that VQGAE trained on large datasets may help to explore more uncharted territories and generate novel and diverse structures.

In the realm of retrosynthesis, the application of the GSLRetro with a trained value network has demonstrated a significant improvement in retrosynthetic search capabilities compared to the tools with heuristic evaluation functions. However, it has been observed that the algorithm needs help to solve more complex compounds. One of the contributing factors mentioned earlier is the limited depth of the tree and the number of iterations. These constraints prevented

GSLRetro from constructing long multi-step pathways, a common case in the synthesis of natural product compounds. This problem can be solved in the future by optimising the GSLRetro code to parallelise the tree search and increase its speed. In addition, the use of retro rules from the patent-biased USPTO database may also be a limiting factor. Therefore, a more reliable tool may be achieved by incorporating data-driven and hand-crafted rules. An alternative approach could be integrating reaction mechanisms instead of automatically extracted rules. Finally, the existing approaches to validate reaction regioselectivity and capability, such as in-scope filters and QSPR models (e.g., developed models for SN2, E2 and Diels-Alder reactions), can be integrated into the GSLRetro to increase the reliability of found synthetic pathways.

References

- [1] E. J. Bjerrum and B. Sattarov, "Improving chemical autoencoder latent space and molecular de novo generation diversity with heteroencoders," *Biomolecules*, vol. 8, no. 4, pp. 1–17, 2018, arXiv: 1806.09300.
- [2] X. Wang, Y. Qian, H. Gao, C. Coley, Y. Mo, R. Barzilay, and K. F. Jensen, "Towards efficient discovery of green synthetic pathways with Monte Carlo tree search and reinforcement learning," *Chemical Science*, vol. 11, no. 40, pp. 10 959–10 972, 2020. [Online]. Available: <http://xlink.rsc.org/?DOI=D0SC04184J>
- [3] A. Thakkar, T. Kogej, J. L. Reymond, O. Engkvist, and E. J. Bjerrum, "Datasets and their influence on the development of computer assisted synthesis planning tools in the pharmaceutical domain," *Chemical Science*, vol. 11, no. 1, pp. 154–168, 2020, publisher: Royal Society of Chemistry.
- [4] T. R. Gimadiev, A. Lin, V. A. Afonina, D. Batyrshin, R. I. Nugmanov, T. Akhmetshin, P. Sidorov, N. Duybankova, J. Verhoeven, J. Wegner, H. Ceulemans, A. Gedich, T. I. Madzhidov, and A. Varnek, "Reaction Data Curation I: Chemical Structures and Transformations Standardization," *Molecular Informatics*, vol. 40, no. 12, pp. 1–16, 2021.
- [5] P. Ertl and A. Schuffenhauer, "Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions," *Journal of Cheminformatics*, vol. 1, no. 1, pp. 1–11, 2009.
- [6] A. Varnek and I. I. Baskin, "Cheminformatics as a Theoretical Chemistry Discipline," *Molecular Informatics*, vol. 30, no. 1, pp. 20–32, Jan. 2011. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/minf.201000100>
- [7] C. W. Coley, "Defining and Exploring Chemical Spaces," *Trends in Chemistry*, vol. 3, no. 2, pp. 133–145, Feb. 2021. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2589597420302884>
- [8] P. G. Polishchuk, T. I. Madzhidov, and A. Varnek, "Estimation of the size of drug-like chemical space based on GDB-17 data," *Journal of Computer-Aided Molecular Design*, vol. 27, no. 8, pp. 675–679, 2013.
- [9] G. Klebe, "Virtual ligand screening: strategies, perspectives and limitations," *Drug Discovery Today*, vol. 11, no. 13-14, pp. 580–594, Jul. 2006. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1359644606001784>

- [10] G. Schneider, “Virtual screening: an endless staircase?” *Nature Reviews Drug Discovery*, vol. 9, no. 4, pp. 273–276, Apr. 2010. [Online]. Available: <http://www.nature.com/articles/nrd3139>
- [11] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney, “Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings IPII of original article: S0169-409X(96)00423-1. The article was originally published in *Advanced Drug Delivery Reviews* 23 (1997) 3–25. 1,” *Advanced Drug Delivery Reviews*, vol. 46, no. 1-3, pp. 3–26, Mar. 2001. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0169409X00001290>
- [12] G. Schneider and U. Fechner, “Computer-based de novo design of drug-like molecules,” *Nature Reviews Drug Discovery*, vol. 4, no. 8, pp. 649–663, 2005.
- [13] M. Hartenfeller and G. Schneider, “Enabling future drug discovery by *de novo* design,” *WIREs Computational Molecular Science*, vol. 1, no. 5, pp. 742–759, Sep. 2011. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/wcms.49>
- [14] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, “Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules,” *ACS Central Science*, vol. 4, no. 2, pp. 268–276, Feb. 2018. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acscentsci.7b00572>
- [15] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, “The rise of deep learning in drug discovery,” *Drug Discovery Today*, vol. 23, no. 6, pp. 1241–1250, Jun. 2018. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1359644617303598>
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: <https://dl.acm.org/doi/10.1145/3065386>
- [17] D. W. Otter, J. R. Medina, and J. K. Kalita, “A Survey of the Usages of Deep Learning for Natural Language Processing,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 604–624, Feb. 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/9075398/>
- [18] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, “Speech Recognition Using Deep Neural Networks: A Systematic Review,” *IEEE Access*, vol. 7, pp. 19 143–19 165, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8632885/>
- [19] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021. [Online]. Available: <https://www.nature.com/articles/s41586-021-03819-2>

- [20] M. H. Segler, T. Kogej, C. Tyrchan, and M. P. Waller, "Generating focused molecule libraries for drug discovery with recurrent neural networks," *ACS Central Science*, vol. 4, no. 1, pp. 120–131, 2018, arXiv: 1701.01329.
- [21] M. H. Segler, M. Preuss, and M. P. Waller, "Planning chemical syntheses with deep neural networks and symbolic AI," *Nature*, vol. 555, pp. 604–610, Mar. 2018, publisher: Nature Publishing Group. [Online]. Available: <http://www.nature.com/articles/nature25978>
- [22] Y. Du, T. Fu, J. Sun, and S. Liu, "MolGenSurvey: A Systematic Survey in Machine Learning Models for Molecule Design," 2022, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/2203.14500>
- [23] J. Jiménez-Luna, F. Grisoni, and G. Schneider, "Drug discovery with explainable artificial intelligence," *Nature Machine Intelligence*, vol. 2, no. 10, pp. 573–584, Oct. 2020. [Online]. Available: <https://www.nature.com/articles/s42256-020-00236-4>
- [24] D. Weininger, "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules," *Journal of Chemical Information and Modeling*, vol. 28, no. 1, pp. 31–36, Feb. 1988. [Online]. Available: <https://pubs.acs.org/doi/abs/10.1021/ci00057a005>
- [25] R. Winter, F. Noé, and D.-A. Clevert, "Permutation-Invariant Variational Autoencoder for Graph-Level Representation Learning," no. NeurIPS, 2021, arXiv: 2104.09856. [Online]. Available: <http://arxiv.org/abs/2104.09856>
- [26] J. Meyers, B. Fabian, and N. Brown, "De novo molecular design and generative models," *Drug Discovery Today*, vol. 26, no. 11, pp. 2707–2715, Nov. 2021. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1359644621002531>
- [27] J. Nam and J. Kim, "Linking the Neural Machine Translation and the Prediction of Organic Chemistry Reactions," 2016, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1612.09529>
- [28] A. Varnek and I. Baskin, "Machine learning methods for property prediction in chemoinformatics: Quo Vadis?" *Journal of Chemical Information and Modeling*, vol. 52, pp. 1413–1437, 2012.
- [29] G. Schneider, "Mind and machine in drug design," *Nature Machine Intelligence*, vol. 1, no. 3, pp. 128–130, Feb. 2019. [Online]. Available: <https://www.nature.com/articles/s42256-019-0030-7>
- [30] R. Todeschini and V. Consonni, *Handbook of Molecular Descriptors*, 1st ed., ser. Methods and Principles in Medicinal Chemistry. Wiley, Sep. 2000. [Online]. Available: <https://onlinelibrary.wiley.com/doi/book/10.1002/9783527613106>
- [31] M. I. Skvortsova, I. I. Baskin, O. L. Slovokhotova, V. A. Palyulin, and N. S. Zefirov, "Inverse problem in QSAR/QSPR studies for the case of topological indexes characterizing molecular shape (Kier indices)," *Journal of Chemical Information and Computer Sciences*, vol. 33, no. 4, pp. 630–634, Jul. 1993. [Online]. Available: <https://pubs.acs.org/doi/abs/10.1021/ci00014a017>

- [32] B. Sanchez-Lengeling and A. Aspuru-Guzik, "Inverse molecular design using machine learning: Generative models for matter engineering," *Science*, vol. 361, no. 6400, pp. 360–365, 2018.
- [33] W. Bort, D. Mazitov, D. Horvath, F. Bonachera, A. Lin, G. Marcou, I. Baskin, T. Madzhidov, and A. Varnek, "Inverse QSAR: Reversing Descriptor-Driven Prediction Pipeline Using Attention-Based Conditional Variational Autoencoder," *Journal of Chemical Information and Modeling*, vol. 62, no. 22, pp. 5471–5484, Nov. 2022. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acs.jcim.2c01086>
- [34] W. W. Wong and F. J. Burkowski, "A constructive approach for discovering new drug leads: Using a kernel methodology for the inverse-QSAR problem," *Journal of Cheminformatics*, vol. 1, no. 1, p. 4, Dec. 2009. [Online]. Available: <https://jcheminf.biomedcentral.com/articles/10.1186/1758-2946-1-4>
- [35] J. Jiménez-Luna, F. Grisoni, N. Weskamp, and G. Schneider, "Artificial intelligence in drug discovery: recent advances and future perspectives," *Expert Opinion on Drug Discovery*, vol. 16, no. 9, pp. 949–959, Sep. 2021. [Online]. Available: <https://www.tandfonline.com/doi/full/10.1080/17460441.2021.1909567>
- [36] G. B. Goh, N. O. Hodas, and A. Vishnu, "Deep learning for computational chemistry," *Journal of Computational Chemistry*, vol. 38, no. 16, pp. 1291–1307, Jun. 2017. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/jcc.24764>
- [37] S. R. Heller, A. McNaught, I. Pletnev, S. Stein, and D. Tchekhovskoi, *InChI, the IUPAC International Chemical Identifier*. *Journal of Cheminformatics*, 2015, vol. 7, publication Title: *Journal of Cheminformatics* Issue: 1 ISSN: 17582946. [Online]. Available: <http://dx.doi.org/10.1186/s13321-015-0068-4>
- [38] J. Handsel, B. Matthews, N. J. Knight, and S. J. Coles, "Translating the InChI: adapting neural machine translation to predict IUPAC names from a chemical identifier," *Journal of Cheminformatics*, vol. 13, no. 1, p. 79, Dec. 2021. [Online]. Available: <https://jcheminf.biomedcentral.com/articles/10.1186/s13321-021-00535-x>
- [39] J. Arús-Pous, S. V. Johansson, O. Prykhodko, E. J. Bjerrum, C. Tyrchan, J. L. Reymond, H. Chen, and O. Engkvist, "Randomized SMILES strings improve the quality of molecular generative models," *Journal of Cheminformatics*, vol. 11, no. 1, pp. 1–13, 2019, publisher: Springer International Publishing. [Online]. Available: <https://doi.org/10.1186/s13321-019-0393-0>
- [40] N. O'Boyle and A. Dalke, "DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures," *Chemistry*, preprint, Sep. 2018. [Online]. Available: <https://chemrxiv.org/engage/chemrxiv/article-details/60c73ed6567dfe7e5fec388d>
- [41] M. Krenn, F. Häse, A. Nigam, P. Friederich, and A. Aspuru-Guzik, "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation," *Machine Learning: Science and Technology*, vol. 1, no. 4, p. 045024, Dec. 2020. [Online]. Available: <https://iopscience.iop.org/article/10.1088/2632-2153/aba947>

- [42] W. Jin, R. Barzilay, and T. Jaakkola, "Chapter 11. Junction Tree Variational Autoencoder for Molecular Graph Generation," in *RSC Drug Discovery Series*, Feb. 2020, pp. 228–249, arXiv: 1802.04364 ISSN: 20413211. [Online]. Available: <http://arxiv.org/abs/1802.04364>
- [43] ———, "Hierarchical Graph-to-Graph Translation for Molecules," *arXiv*, pp. 1–14, Jun. 2019, arXiv: 1907.11223. [Online]. Available: <http://arxiv.org/abs/1907.11223>
- [44] K. Maziarz, H. Jackson-Flux, P. Cameron, F. Sirockin, N. Schneider, N. Stiefl, M. Segler, and M. Brockschmidt, "Learning to Extend Molecular Scaffolds with Structural Motifs," in *ICLR 2022*, Apr. 2022. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/learning-to-extend-molecular-scaffolds-with-structural-motifs/>
- [45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. [Online]. Available: <http://www.nature.com/articles/323533a0>
- [46] A. Radford and K. Narasimhan, "Improving Language Understanding by Generative Pre-Training," 2018.
- [47] U. V. Ucak, T. Kang, J. Ko, and J. Lee, "Substructure-based neural machine translation for retrosynthetic prediction," *Journal of Cheminformatics*, vol. 13, no. 1, p. 4, Dec. 2021. [Online]. Available: <https://jcheminf.biomedcentral.com/articles/10.1186/s13321-020-00482-z>
- [48] D. J. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," 2015, publisher: arXiv Version Number: 6. [Online]. Available: <https://arxiv.org/abs/1505.05770>
- [49] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, Feb. 1991. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/aic.690370209>
- [50] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, Dec. 2014, pp. 1–14, arXiv: 1312.6114. [Online]. Available: <http://arxiv.org/abs/1312.6114>
- [51] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, Mar. 1951. [Online]. Available: <http://projecteuclid.org/euclid.aoms/1177729694>
- [52] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein Auto-Encoders," 2017, publisher: arXiv Version Number: 4. [Online]. Available: <https://arxiv.org/abs/1711.01558>
- [53] Z. Alperstein, A. Cherkasov, and J. T. Rolfe, "All SMILES Variational Autoencoder," *arXiv*, 2019, arXiv: 1905.13343. [Online]. Available: <http://arxiv.org/abs/1905.13343>

- [54] C. Yan, S. Wang, J. Yang, T. Xu, and J. Huang, “Re-balancing Variational Autoencoder Loss for Molecule Sequence Generation,” *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, BCB 2020*, 2020, arXiv: 1910.00698 ISBN: 9781450379649.
- [55] H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song, “Syntax-directed variational autoencoder for structured data,” in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018, pp. 1–17, arXiv: 1802.08786 ISSN: 23318422.
- [56] Q. Liu, M. Allamanis, M. Brockschmidt, and A. L. Gaunt, “Constrained graph variational autoencoders for molecule design,” *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. NeurIPS, pp. 7795–7804, 2018, arXiv: 1805.09076.
- [57] M. Simonovsky and N. Komodakis, “Dynamic edge-conditioned filters in convolutional neural networks on graphs,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 29–38, Apr. 2017, arXiv: 1704.02901 Publisher: Institute of Electrical and Electronics Engineers Inc. ISBN: 9781538604571. [Online]. Available: <http://arxiv.org/abs/1704.02901>
- [58] B. Samanta, A. DE, G. Jana, P. K. Chattaraj, N. Ganguly, and M. G. Rodriguez, “NeVAE: A Deep Generative Model for Molecular Graphs,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1110–1117, Jul. 2019, arXiv: 1802.05283 ISBN: 9781577358091. [Online]. Available: <https://jmlr.org/papers/v21/19-671.html>
- [59] D. Flam-Shepherd, T. C. Wu, and A. Aspuru-Guzik, “MPGVAE: Improved generation of small organic molecules using message passing neural nets,” *Machine Learning: Science and Technology*, vol. 2, no. 4, pp. 0–10, 2021.
- [60] S. Honda, H. Akita, K. Ishiguro, T. Nakanishi, and K. Oono, “Graph Residual Flow for Molecular Graph Generation,” *arXiv*, pp. 1–14, 2019, arXiv: 1909.13521. [Online]. Available: <http://arxiv.org/abs/1909.13521>
- [61] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang, “GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation,” 2020, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/2001.09382>
- [62] C. Zang and F. Wang, “MoFlow: An Invertible Flow Model for Generating Molecular Graphs,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Virtual Event CA USA: ACM, Aug. 2020, pp. 617–626. [Online]. Available: <https://dl.acm.org/doi/10.1145/3394486.3403104>
- [63] M. Kuznetsov and D. Polykovskiy, “MolGrow: A Graph Normalizing Flow for Hierarchical Molecular Generation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, pp. 8226–8234, May 2021. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/17001>
- [64] Y. Luo, K. Yan, and S. Ji, “GraphDF: A Discrete Flow Model for Molecular Graph Generation,” in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, Jul. 2021, pp. 7192–7203. [Online]. Available: <https://proceedings.mlr.press/v139/luo21a.html>

- [65] C. Abate, S. Decherchi, and A. Cavalli, “Graph neural networks for conditional de novo drug design,” *WIREs Computational Molecular Science*, Jan. 2023. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/wcms.1651>
- [66] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges,” 2021, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/2104.13478>
- [67] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural Message Passing for Quantum Chemistry,” 2017, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/1704.01212>
- [68] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković, “Principal neighbourhood aggregation for graph nets,” *Advances in Neural Information Processing Systems*, vol. 2020-Decem, no. NeurIPS, 2020, arXiv: 2004.05718.
- [69] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, Sep. 2017, pp. 1–14, arXiv: 1609.02907. [Online]. Available: <http://arxiv.org/abs/1609.02907>
- [70] R. Assouel, M. Ahmed, M. H. Segler, A. Saffari, and Y. Bengio, “DEFactor: Differentiable Edge Factorization-based Probabilistic Graph Generation,” *arXiv*, pp. 1–14, Nov. 2018, arXiv: 1811.09766. [Online]. Available: <http://arxiv.org/abs/1811.09766>
- [71] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, “Modeling Relational Data with Graph Convolutional Networks,” 2017, publisher: arXiv Version Number: 4. [Online]. Available: <https://arxiv.org/abs/1703.06103>
- [72] L. Gong and Q. Cheng, “Exploiting Edge Features in Graph Neural Networks,” 2018, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/1809.02709>
- [73] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A Comprehensive Survey on Graph Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/9046288/>
- [74] Z. Guo, B. Nan, Y. Tian, O. Wiest, C. Zhang, and N. V. Chawla, “Graph-based Molecular Representation Learning,” 2022, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/2207.04869>
- [75] S. Kearnes, L. Li, and P. Riley, “Decoding Molecular Graph Embeddings with Reinforcement Learning,” *arXiv*, Apr. 2019, arXiv: 1904.08915. [Online]. Available: <http://arxiv.org/abs/1904.08915>
- [76] M. Simonovsky and N. Komodakis, “GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11139 LNCS, pp. 412–422, arXiv: 1802.03480 ISSN: 16113349. [Online]. Available: http://link.springer.com/10.1007/978-3-030-01418-6_41

- [77] X. Bresson and T. Laurent, “A Two-Step Graph Convolutional Decoder for Molecule Generation,” *arXiv*, Jun. 2019, arXiv: 1906.03412. [Online]. Available: <http://arxiv.org/abs/1906.03412>
- [78] K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe, “GraphNVP: An Invertible Flow Model for Generating Molecular Graphs,” 2019, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1905.11600>
- [79] P.-C. Kotsias, J. Arús-Pous, H. Chen, O. Engkvist, C. Tyrchan, and E. J. Bjerrum, “Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks,” *Nature Machine Intelligence*, vol. 2, no. 5, pp. 254–265, May 2020, publisher: Springer US ISBN: 4225602001. [Online]. Available: <http://dx.doi.org/10.1038/s42256-020-0174-5>
- [80] M. Xu, T. Ran, and H. Chen, “De Novo Molecule Design Through the Molecular Generative Model Conditioned by 3D Information of Protein Binding Sites,” *Journal of Chemical Information and Modeling*, vol. 61, no. 7, pp. 3240–3254, Jul. 2021. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acs.jcim.0c01494>
- [81] Y. Li, L. Zhang, Y. Wang, J. Zou, R. Yang, X. Luo, C. Wu, W. Yang, C. Tian, H. Xu, F. Wang, X. Yang, L. Li, and S. Yang, “Generative deep learning enables the discovery of a potent and selective RIPK1 inhibitor,” *Nature Communications*, vol. 13, no. 1, p. 6891, Nov. 2022. [Online]. Available: <https://www.nature.com/articles/s41467-022-34692-w>
- [82] S. Joo, M. S. Kim, J. Yang, and J. Park, “Generative Model for Proposing Drug Candidates Satisfying Anticancer Properties Using a Conditional Variational Autoencoder,” *ACS Omega*, vol. 5, no. 30, pp. 18 642–18 650, 2020.
- [83] A. Zhavoronkov, Y. A. Ivanenkov, A. Aliper, M. S. Veselov, V. A. Aladinskiy, A. V. Aladinskaya, V. A. Terentiev, D. A. Polykovskiy, M. D. Kuznetsov, A. Asadulaev, Y. Volkov, A. Zholus, R. R. Shayakhmetov, A. Zhebrak, L. I. Minaeva, B. A. Zagribelnyy, L. H. Lee, R. Soll, D. Madge, L. Xing, T. Guo, and A. Aspuru-Guzik, “Deep learning enables rapid identification of potent DDR1 kinase inhibitors,” *Nature Biotechnology*, vol. 37, no. 9, pp. 1038–1040, Sep. 2019. [Online]. Available: <http://www.nature.com/articles/s41587-019-0224-x>
- [84] M. Pikusa, O. René, S. Williams, Y.-L. Chen, E. Martin, W. J. Godinez, S. P. S. Rao, W. A. Guiguemde, and F. Nigsch, “De-novo generation of novel phenotypically active molecules for Chagas disease from biological signatures using AI-driven generative chemistry,” *Pharmacology and Toxicology*, preprint, Dec. 2021. [Online]. Available: <http://biorxiv.org/lookup/doi/10.1101/2021.12.10.472084>
- [85] A. Tropsha, “Best Practices for QSAR Model Development, Validation, and Exploitation,” *Molecular Informatics*, vol. 29, no. 6-7, pp. 476–488, Jul. 2010. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/minf.201000061>
- [86] A. Nigam, R. Pollice, G. Tom, K. Jorner, L. A. Thiede, A. Kundaje, and A. Aspuru-Guzik, “Tartarus: A Benchmarking Platform for Realistic And Practical Inverse Molecular Design,” 2022, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/2209.12487>

- [87] I. I. Baskin, “The power of deep learning to ligand-based novel drug discovery,” *Expert Opinion on Drug Discovery*, vol. 15, pp. 755–764, 2020, publisher: Taylor & Francis. [Online]. Available: <https://doi.org/10.1080/17460441.2020.1745183>
- [88] C. W. Coley, N. S. Eyke, and K. F. Jensen, “Autonomous Discovery in the Chemical Sciences Part II: Outlook,” *Angewandte Chemie International Edition*, vol. 59, no. 52, pp. 23 414–23 436, Dec. 2020. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/anie.201909989>
- [89] C. W. Coley, W. H. Green, and K. F. Jensen, “Machine Learning in Computer-Aided Synthesis Planning,” *Accounts of Chemical Research*, vol. 51, no. 5, pp. 1281–1289, May 2018. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acs.accounts.8b00087>
- [90] P. Schwaller, T. Laino, T. Gaudin, P. Bolgar, C. A. Hunter, C. Bekas, and A. A. Lee, “Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction,” *ACS Central Science*, vol. 5, no. 9, pp. 1572–1583, 2019, arXiv: 1811.02633.
- [91] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, “Learning Deep Generative Models of Graphs,” 2018, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1803.03324>
- [92] R. Mercado, T. Rastemo, E. Lindelof, G. Klambauer, O. Engkvist, H. Chen, and E. J. Bjerrum, “Graph networks for molecular design,” *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 025023, Jun. 2021. [Online]. Available: <https://iopscience.iop.org/article/10.1088/2632-2153/abcf91>
- [93] P. B. E. B. Martín Abadi, Ashish Agarwal, G. S. C. A. D. Zhifeng Chen, Craig Citro, S. G. I. G. Jeffrey Dean, Matthieu Devin, M. I. R. J. Y. J. Andrew Harp, Geoffrey Irving, J. L. D. M. M. S. Lukasz Kaiser, Manjunath Kudlur, D. M. C. O. J. S. Rajat Monga, Sherry Moore, K. T. P. T. Benoit Steiner, Ilya Sutskever, F. V. Vincent Vanhoucke, Vijay Vasudevan, M. W. M. W. Oriol Vinyals, Pete Warden, Y. Yu, and X. Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,” 2015.
- [94] R. I. Nugmanov, R. N. Mukhametgaleev, T. Akhmetshin, T. R. Gimadiev, V. A. Afonina, T. I. Madzhidov, and A. Varnek, “CGRtools: Python Library for Molecule, Reaction, and Condensed Graph of Reaction Processing,” *Journal of Chemical Information and Modeling*, vol. 59, pp. 2516–2521, Jun. 2019, publisher: American Chemical Society Genre: brief-report. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acs.jcim.9b00102>
- [95] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, “Vega-Lite: A Grammar of Interactive Graphics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, Jan. 2017. [Online]. Available: <http://ieeexplore.ieee.org/document/7539624/>
- [96] J. VanderPlas, B. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert, “Altair: Interactive Statistical Visualizations for Python,” *Journal of Open Source Software*, vol. 3, no. 32, p. 1057, Dec. 2018. [Online]. Available: <http://joss.theoj.org/papers/10.21105/joss.01057>

- [97] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, A. Kadurin, S. Johansson, H. Chen, S. Nikolenko, A. Aspuru-Guzik, and A. Zhavoronkov, "Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models," *Frontiers in Pharmacology*, vol. 11, pp. 1–10, Dec. 2020, arXiv: 1811.12823. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fphar.2020.565644/full>
- [98] K. Preuer, P. Renz, T. Unterthiner, S. Hochreiter, and G. Klambauer, "Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery," *Journal of Chemical Information and Modeling*, vol. 58, no. 9, pp. 1736–1741, Sep. 2018, arXiv: 1803.09518. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acs.jcim.8b00234>
- [99] J. B. Baell and G. A. Holloway, "New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays," *Journal of Medicinal Chemistry*, vol. 53, no. 7, pp. 2719–2740, Apr. 2010, publisher: American Chemical Society. [Online]. Available: <https://pubs.acs.org/doi/abs/10.1021/jm901137j>
- [100] X. Q. Lewell, D. B. Judd, S. P. Watson, and M. M. Hann, "RECAP - Retrosynthetic Combinatorial Analysis Procedure: A powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry," *Journal of Chemical Information and Computer Sciences*, vol. 38, no. 3, pp. 511–522, May 1998. [Online]. Available: <https://pubs.acs.org/doi/10.1021/ci970429i>
- [101] Y. Zabolotna, D. M. Volochnyuk, V. S. Ryabukhin, D. Horvath, K. S. Gavrilenko, G. Marcou, Y. S. Moroz, O. Oksiuta, and A. Varnek, "A Close-up Look at the Chemical Space of Commercially Available Building Blocks for Medicinal Chemistry," *Journal of Chemical Information and Modeling*, p. acs.jcim.1c00811, Dec. 2021.
- [102] A. Van Den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 6307–6316, Nov. 2017, arXiv: 1711.00937. [Online]. Available: <http://arxiv.org/abs/1711.00937>
- [103] M. Matveieva and P. Polishchuk, "Benchmarks for interpretation of QSAR models," *Journal of Cheminformatics*, vol. 13, no. 1, p. 41, Dec. 2021. [Online]. Available: <https://jcheminf.biomedcentral.com/articles/10.1186/s13321-021-00519-x>
- [104] E. J. Corey and W. T. Wipke, "Computer-Assisted Design of Complex Organic Syntheses: Pathways for molecular synthesis can be devised with a computer and equipment for graphical communication." *Science*, vol. 166, no. 3902, pp. 178–192, Oct. 1969. [Online]. Available: <https://www.science.org/doi/10.1126/science.166.3902.178>
- [105] J. S. Schreck, C. W. Coley, and K. J. M. Bishop, "Learning Retrosynthetic Planning through Simulated Experience," *ACS Central Science*, vol. 5, no. 6, pp. 970–981, Jun. 2019. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acscentsci.9b00055>
- [106] B. A. Grzybowski, T. Badowski, K. Molga, and S. Szymkuć, "Network search algorithms and scoring functions for advanced-level computerized synthesis planning," *WIREs Computational Molecular Science*, Jun. 2022. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/wcms.1630>

- [107] “Reaxys database.” [Online]. Available: <https://www.elsevier.com/solutions/reaxys>
- [108] “SciFinder.” [Online]. Available: <https://www.cas.org/solutions/cas-scifinder-discovery-platform/cas-scifinder>
- [109] D. M. Lowe, “Extraction of chemical structures and reactions from the literature,” Oct. 2012, publisher: Apollo - University of Cambridge Repository. [Online]. Available: <https://www.repository.cam.ac.uk/handle/1810/244727>
- [110] J. Mayfield, D. Lowe, and R. Sayle, “Pistachio,” Patent. [Online]. Available: <https://www.nextmovesoftware.com/pistachio.html>
- [111] D. Fourches, E. Muratov, and A. Tropsha, “Trust, But Verify: On the Importance of Chemical Structure Curation in Cheminformatics and QSAR Modeling Research,” *Journal of Chemical Information and Modeling*, vol. 50, no. 7, pp. 1189–1204, Jul. 2010. [Online]. Available: <https://pubs.acs.org/doi/10.1021/ci100176x>
- [112] H. L. Gelernter, A. F. Sanders, D. L. Larsen, K. K. Agarwal, R. H. Boivie, G. A. Spritzer, and J. E. Searleman, “Empirical Explorations of SYNCHEM: The methods of artificial intelligence are applied to the problem of organic synthesis route discovery.” *Science*, vol. 197, no. 4308, pp. 1041–1049, Sep. 1977. [Online]. Available: <https://www.science.org/doi/10.1126/science.197.4308.1041>
- [113] J. Bauer, R. Herges, E. Fontain, and I. Ugi, “IGOR and computer assisted innovation in chemistry.” *Chimia*, vol. 39, no. 2, pp. 43–53, 1985.
- [114] J. B. Hendrickson, D. L. Grier, and A. G. Toczko, “A logic-based program for synthesis design,” *Journal of the American Chemical Society*, vol. 107, no. 18, pp. 5228–5238, Sep. 1985. [Online]. Available: <https://pubs.acs.org/doi/abs/10.1021/ja00304a033>
- [115] T. D. Salatin and W. L. Jorgensen, “Computer-assisted mechanistic evaluation of organic reactions,” *The Journal of Organic Chemistry*, vol. 45, no. 11, pp. 2043–2051, May 1980. [Online]. Available: <https://pubs.acs.org/doi/abs/10.1021/jo01299a001>
- [116] W.-D. Ihlenfeldt and J. Gasteiger, “Computer-Assisted Planning of Organic Syntheses: The Second Generation of Programs,” *Angewandte Chemie International Edition in English*, vol. 34, no. 2324, pp. 2613–2633, Jan. 1996. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/anie.199526131>
- [117] J. Gasteiger and C. Jochum, “EROS A computer program for generating sequences of reactions,” in *Organic Compounds*. Berlin/Heidelberg: Springer-Verlag, 1978, vol. 74, pp. 93–126, series Title: Topics in Current Chemistry. [Online]. Available: <http://link.springer.com/10.1007/BFb0050147>
- [118] C. Sammut, “Beam Search,” in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, pp. 120–120. [Online]. Available: http://link.springer.com/10.1007/978-1-4899-7687-1_68
- [119] P. Schwaller and T. Laino, “Data-Driven Learning Systems for Chemical Reaction Prediction: An Analysis of Recent Approaches,” in *ACS Symposium Series*, E. O. Pyzer-Knapp and T. Laino, Eds. Washington, DC: American

- Chemical Society, Jan. 2019, vol. 1326, pp. 61–79. [Online]. Available: <https://pubs.acs.org/doi/abs/10.1021/bk-2019-1326.ch004>
- [120] J. Law, Z. Zsoldos, A. Simon, D. Reid, Y. Liu, S. Y. Khew, A. P. Johnson, S. Major, R. A. Wade, and H. Y. Ando, “Route Designer: A Retrosynthetic Analysis Tool Utilizing Automated Retrosynthetic Rule Generation,” *Journal of Chemical Information and Modeling*, vol. 49, no. 3, pp. 593–602, Mar. 2009. [Online]. Available: <https://pubs.acs.org/doi/10.1021/ci800228y>
- [121] D. W. Elrod, G. M. Maggiora, and R. G. Trenary, “Applications of neural networks in chemistry. 1. Prediction of electrophilic aromatic substitution reactions,” *Journal of Chemical Information and Computer Sciences*, vol. 30, no. 4, pp. 477–484, Nov. 1990. [Online]. Available: <https://pubs.acs.org/doi/abs/10.1021/ci00068a020>
- [122] J.-C. Régis, O. Gascuel, and C. Laurenço, “Machine learning of strategic knowledge in organic synthesis from reaction databases,” in *AIP Conference Proceedings*, vol. 330. Nancy (France): AIP, 1995, pp. 618–623, iSSN: 0094243X. [Online]. Available: <http://aip.scitation.org/doi/abs/10.1063/1.47873>
- [123] P. Schwaller, R. Petraglia, V. Zullo, V. H. Nair, R. A. Haeuselmann, R. Pisoni, C. Bekas, A. Iuliano, and T. Laino, “Predicting retrosynthetic pathways using transformer-based models and a hyper-graph exploration strategy,” *Chemical Science*, vol. 11, no. 12, pp. 3316–3325, 2020. [Online]. Available: <http://xlink.rsc.org/?DOI=C9SC05704H>
- [124] “IBM RoboRXN.” [Online]. Available: <https://research.ibm.com/science/ibm-roborxn/>
- [125] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016. [Online]. Available: <http://www.nature.com/articles/nature16961>
- [126] C. D. Rosin, “Multi-armed bandits with episode context,” *Annals of Mathematics and Artificial Intelligence*, vol. 61, no. 3, pp. 203–230, Mar. 2011. [Online]. Available: <http://link.springer.com/10.1007/s10472-011-9258-6>
- [127] C. W. Coley, D. A. Thomas, J. A. M. Lummiss, J. N. Jaworski, C. P. Breen, V. Schultz, T. Hart, J. S. Fishman, L. Rogers, H. Gao, R. W. Hicklin, P. P. Plehiers, J. Byington, J. S. Piotti, W. H. Green, A. J. Hart, T. F. Jamison, and K. F. Jensen, “A robotic platform for flow synthesis of organic compounds informed by AI planning,” *Science*, vol. 365, no. 6453, p. eaax1566, Aug. 2019. [Online]. Available: <https://www.science.org/doi/10.1126/science.aax1566>
- [128] S. Ishida, K. Terayama, R. Kojima, K. Takasu, and Y. Okuno, “AI-Driven Synthetic Route Design with Retrosynthesis Knowledge,” Tech. Rep., 2020, publication Title: ChemRxiv ISSN: 25732293.
- [129] K. Lin, Y. Xu, J. Pei, and L. Lai, “Automatic retrosynthetic route planning using template-free models,” *Chemical Science*, vol. 11, no. 12, pp. 3355–3364, 2020. [Online]. Available: <http://xlink.rsc.org/?DOI=C9SC03666K>

- [130] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017, arXiv: 1706.03762.
- [131] S. Ishida, K. Terayama, R. Kojima, K. Takasu, and Y. Okuno, "AI-Driven Synthetic Route Design Incorporated with Retrosynthesis Knowledge," *Journal of Chemical Information and Modeling*, 2022.
- [132] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, Dec. 2020. [Online]. Available: <http://www.nature.com/articles/s41586-020-03051-4>
- [133] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018. [Online]. Available: <https://www.science.org/doi/10.1126/science.aar6404>
- [134] "Spaya." [Online]. Available: <https://iktos.ai/spaya/>
- [135] A. Konstantinov, E. Putin, B. Zagribelny, Y. Ivanenkov, and A. Zavoronkovs, "Retrosynthesis systems and methods," U.S. Patent US 2022/0 172 802 A1.
- [136] I. I. Baskin, T. I. Madzhidov, I. S. Antipin, and A. A. Varnek, "Artificial intelligence in synthetic chemistry: achievements and prospects," *Russian Chemical Reviews*, vol. 86, no. 11, pp. 1127–1156, Nov. 2017. [Online]. Available: <http://stacks.iop.org/0036-021X/86/i=11/a=1127?key=crossref.2bf52566e7b90f69572ab5b2acd28e67>
- [137] B. Mikulak-Klucznik, P. Gołębiowska, A. A. Bayly, O. Popik, T. Klucznik, S. Szymkuć, E. P. Gajewska, P. Dittwald, O. Staszewska-Krajewska, W. Beker, T. Badowski, K. A. Scheidt, K. Molga, J. Mlynarski, M. Mrksich, and B. A. Grzybowski, "Computational planning of the synthesis of complex natural products," *Nature*, vol. 588, no. 7836, pp. 83–88, Dec. 2020. [Online]. Available: <https://www.nature.com/articles/s41586-020-2855-y>
- [138] S. Genheden, A. Thakkar, V. Chadimová, J. L. Reymond, O. Engkvist, and E. Bjerrum, "AiZynthFinder: a fast, robust and flexible open-source software for retrosynthetic planning," *Journal of Cheminformatics*, vol. 12, no. 1, pp. 1–14, 2020.
- [139] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, and J. P. Overington, "ChEMBL: A large-scale bioactivity database for drug discovery," *Nucleic Acids Research*, vol. 40, pp. 1100–1107, 2012.
- [140] M. Sorokina, P. Merseburger, K. Rajan, M. A. Yirik, and C. Steinbeck, "COCONUT online: Collection of Open Natural Products database," *Journal of Cheminformatics*, vol. 13, no. 1, p. 2, Dec. 2021. [Online]. Available: <https://jcheminf.biomedcentral.com/articles/10.1186/s13321-020-00478-9>

- [141] G. Landrum, “rdkit/rdkit: 2022_09_2 (Q3 2022) Release,” Nov. 2022. [Online]. Available: <https://zenodo.org/record/7357998>
- [142] A. Dalke, “The chemfp project,” *Journal of Cheminformatics*, vol. 11, no. 1, p. 76, Dec. 2019. [Online]. Available: <https://jcheminf.biomedcentral.com/articles/10.1186/s13321-019-0398-8>
- [143] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. S. Duncan, “AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients,” *Advances in Neural Information Processing Systems*, vol. 2020-Decem, pp. 1–29, 2020, arXiv: 2010.07468. [Online]. Available: <http://arxiv.org/abs/2010.07468>
- [144] C. W. Coley, W. H. Green, and K. F. Jensen, “RDChiral: An RDKit Wrapper for Handling Stereochemistry in Retrosynthetic Template Extraction and Application,” *Journal of Chemical Information and Modeling*, vol. 59, no. 6, pp. 2529–2537, Jun. 2019. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acs.jcim.9b00286>
- [145] F. Aricò, S. Bravo, M. Crisma, and P. Tundo, “1,3-Oxazinan-2-ones via carbonate chemistry: a facile, high yielding synthetic approach,” *Pure and Applied Chemistry*, vol. 88, no. 3, pp. 227–237, Mar. 2016. [Online]. Available: <https://www.degruyter.com/document/doi/10.1515/pac-2015-1004/html>

List of figures

1.1	HyFactor scheme	2
1.2	HyFactor scheme	3
1.3	VQGAE scheme	5
1.4	Results on QSAR benchmarking	6
1.5	Resume MCTS scheme	8
1.6	Resume evaluation functions	9
1.7	Resume self-learning protocol	10
1.8	Resume benchmarking sets preparation	11
1.9	GSLRetro results	11
1.10	Resume synthetic pathways of AiZynthFinder and GSLRetro on the same target	12
1.11	Resume synthetic pathways of AiZynthFinder and GSLRetro on the same target	13
1.12	Resume shared synthetic pathway of AiZynthFinder and GSLRetro	13
1.13	VQGAE resume generated molecules	14
1.14	GSLRetro resume paths	15
3.1	Types of iterative generators	26
3.2	Types of autoencoders	28
3.3	General scheme of message-passing neural networks	30
3.4	GNN approaches for multigraphs	32
3.5	Atom types in ChEMBL	48
3.6	HyFactor generation results	51
3.7	E_ChEMBL	53
4.1	Retrosynthesis intro	74
4.2	Reaxys non-standardized molecules	92
4.3	USPTO wrongly parsed reaction	92
4.4	Classical Retrorules	94
4.5	Retrosynthetic tree	95
4.6	LHASA algorithm	96
4.7	General algorithm of MCTS	99
4.8	Scheme of ranking policy network	100
4.9	Types of evaluation functions	102
4.10	Evaluation-first search strategy	104
4.11	Expansion-first search strategy	105
4.12	Scheme of self-learning concept	106
4.13	Scheme of self-learning data collection	107
4.14	Timeline of MCTS-based retrosynthesis planning tools	107
4.15	Examples of “priority” rules	111

4.16	Branched synthetic pathways	112
4.17	Non regio-selective retro-rule	113
4.18	Reaction filters examples	114
4.19	Example of rule used in GSLRetro	114
4.20	Distribution of SAScore in ChEMBL and COCONUT	115
4.21	GSLRetro benchmarking sets preparation	116
4.22	GSLRetro policy network	117
4.23	GSLRetro value network	118
4.24	RDchiral rules analysis	119
4.25	AiZynthFinder reproduction results	120
4.26	Example of AiZynthFinder's bug	121
4.27	General retrosynthesis results	122
4.28	Self-learning results	123
4.29	GSLRetro and AiZynthFinder comparison	124
4.30	Analysis of value network scores	124
4.31	Example of the same synthetic pathway found by GSLRetro and AiZynthFinder	125
4.32	Examples of synthetic pathways for the same targets found by GSLRetro and AiZynthFinder	126
4.33	Examples of synthetic pathways for different targets found by GSLRetro and AiZynthFinder	128
4.34	Synthetic pathways for generated antagonists of A2A	129

List of tables

1.1	Résultats du benchmarking des autoencodeurs HyFactor et ReFactor sur le jeu de données ZINC 250K.	4
1.2	Résultats du benchmarking des autoencodeurs HyFactor et ReFactor sur le jeu de données ZINC 250K.	4
3.1	Results of ZINC 250K data set standardisation.	48
3.2	Training parameters for each data set. The parameters are equal for both ReFactor and HyFactor architectures.	49
3.3	MOSES benchmarking results	50
3.4	MOSES benchmarking results comparing to test set (Test) and test scaffolds set (TestSF)	50

Réseaux neuronaux à base de graphes pour la génération de structures moléculaires synthétiquement accessibles

Résumé

Cette thèse est dédiée au développement de générateurs par réseaux de neurones artificiels de graphes et d'un outil de planification rétrosynthétique amélioré par des méthodes d'apprentissage automatique profond. Les modèles génératifs s'appuient sur le concept d'autoencodeur, très populaire dans les tâches de conception moléculaire *de novo* et d'analyse QSAR inverse. L'architecture proposée, HyFactor, basée sur le graphe étiqueté par le nombre d'hydrogènes par atome, se révèle performante et utile pour générer des analogues moléculaires. La seconde architecture proposée, VQGAE, donne des résultats comparables à ceux d'HyFactor dans la tâche de reconstruction, mais la représentation latente se montre plus performante pour la recherche par similarité et pour le QSAR. Ces performances sont illustrées par la génération de ligands très actifs pour les récepteurs A2A. La faisabilité synthétique des structures générées a été vérifiée à l'aide d'un nouvel outil de rétrosynthèse, GSLRetro. Cet outil est conçu sur le concept d'auto-apprentissage qui améliore la stabilité des solutions grâce aux précédentes solutions proposées. Le protocole de curation des données de réaction ainsi que de nouvelles techniques de validation croisée pour les modèles QSPR basés sur les réactions sont également présentés. En somme, la combinaison de ces outils constitue une étape importante vers la conception automatisée de molécules.

Résumé en anglais

This thesis is dedicated to the development of graph-based generative neural networks and a retrosynthetic planning tool enhanced by deep learning architectures. The generative networks are based on the autoencoder concept, which has gained popularity in *de novo* molecular design and inverse QSAR tasks. The proposed architecture, HyFactor, based on the hydrogen number labelled graph, was computationally efficient and helpful in generating molecular analogues. The following architecture, VQGAE, performed as well as HyFactor in the reconstruction task, while its latent vectors showed the best performance in the similarity search and QSAR tasks. As a proof of concept, VQGAE was used to generate highly potent ligands of A2A receptors. The synthetic feasibility of the generated structures was verified using a new retrosynthesis tool, GSLRetro. This tool is based on a self-learning concept by which it can improve the search quality by training on previous searches' results. In addition to the retrosynthetic planning tool, the protocol of reaction data curation and new cross-validation techniques for reaction-based QSPR models were proposed. Finally, the combination of these tools represents a significant step towards automated molecular design.