Université Paris Cité
ED Frontières de l'Innovation en Recherche et Education - 474

Unité de Bioinformatique Structurale, CNRS UMR3528 - Institut Pasteur
Centre de Biologie Computationelle - Mines ParisTech

Geometric Deep Learning for Structural Bioinformatics

Presentée par Vincent MALLET

Une thèse de doctorat en
Génétique, omiques, bioinformatique et biologie des systèmes

Dirigée par Michael NILGES et Jean-Philippe VERT

Le 16 Novembre 2022

Membres du Jury :

Alessandra CARBONE	Professeur	Université Paris Sorbonne	Rapportrice
Frédéric CAZALS	Directeur de recherche	INRIA	Rapporteur
Véronique STOVEN	Professeur	Mines ParisTech	Examinatrice
Karsten BORGWARDT	Professeur	ETH Zurich	Examinateur
Emanuele RODOLA	Professeur	Université de Rome, Sapienza	Membre invité
Michael NILGES	Directeur de recherche	Institut Pasteur	Directeur
Jean-Philippe VERT	Directeur de recherche	Google Brain	Directeur

Résumé

Mots clés : Apprentissage profond géométrique, Bioinformatique structurale, *Drug design*

L'apprentissage automatique a permis plusieurs percées dans la gestion des données tabulaires, d'images ou de texte. Il a également commencé à aider la science, par exemple avec Alphafold, mais son application aux questions scientifiques n'est pas immédiate. Le premier défi consiste à modéliser des objets naturels avec des objets mathématiques représentés dans un ordinateur - comme des images - tout en respectant leurs propriétés physiques. Le deuxième défi est d'étendre les méthodes d'apprentissage à de nouveaux objets mathématiques et numériques avec plus de structure, un domaine de recherche connu sous le nom d'apprentissage profond géométrique. Avoir un éventail plus large d'objets mathématiques nous donne plus de liberté pour modéliser efficacement nos objets naturels pour l'apprentissage automatique.

La biologie structurale est un domaine scientifique visant à comprendre le vivant en utilisant les structures tridimensionnelles de molécules importantes, disponibles grâce à des outils expérimentaux et informatiques. Ce domaine s'appuie donc sur des données structurées qui pourraient se prêter à l'apprentissage automatique si les deux défis ci-dessus étaient relevés. Parmi les principales applications de la biologie structurale figure la découverte de médicaments, qui vise à trouver de potentiels médicaments dans un vaste espace de composés chimiques. Dans l'approche centrée sur les cibles thérapeutiques, les structures tridimensionnelles de celles-ci sont utilisées pour sélectionner ces potentiels médicaments. Cette approche pourrait être révolutionnée par l'utilisation de l'apprentissage profond géométrique.

Nous commençons par un apport méthodologique qui permet de respecter la structure des molécules d'ADN représentées sous forme de chaînes de caractères. En effet, une telle représentation néglige la symétrie du brin complémentaire qui découle de l'appariement des deux brins de l'ADN. En utilisant la théorie de l'équivariance, nous caractérisons la classe de modèles d'apprentissage automatique qui respectent cette structure supplémentaire. Nous montrons empiriquement qu'utiliser cette classe de modèles améliore la précision de la prédiction de la liaison des facteurs de transcription.

Nous préconisons ensuite l'utilisation d'un type spécifique de graphe pour représenter l'ARN en conjonction avec des méthodes d'apprentissage profond pour les graphes. Ce type spécifique de graphes est une représentation gros-grain et discrète introduite par les biochimistes il y a vingt ans. Nous montrons que l'utilisation de cette représentation est supérieure à l'utilisation de graphes de base et suffisante pour extraire un signal pertinent pour la découverte de médicaments ciblant l'ARN. De plus, nous pouvons tirer parti de ce cadre d'apprentissage pour détecter efficacement des motifs structuraux dans l'ARN, en relâchant les contraintes imposées à ces motifs par les outils préexistants. Nous avons publié un package pour utiliser cette représentation dans les applications d'apprentissage automatique.

Enfin, nous présentons trois outils pour aider à la découverte de médicaments centrés sur les cibles thérapeutiques qui reposent sur l'apprentissage automatique. Nous proposons un outil dédié à la recherche de sites de liaison aux sites d'interaction protéine-protéine en prédisant simultanément la liaison aux petites molécules et aux protéines. Nous proposons également un outil pour regrouper efficacement les conformations d'une trajectoire de dynamique moléculaire, permettant la sélection de conformations représentatives pertinentes. Enfin, nous proposons une méthode qui génère des populations de composés avec une affinité accrue pour une cible donnée.

Summary

Keywords : Geometric deep learning, Structural bioinformatics, Drug design

Machine learning has enabled several breakthroughs in tabular, image and text data management. It has also started to help science, for instance Alphafold, but its application to scientific questions is not necessarily straightforward. The first challenge is to model natural objects as mathematical objects that can exist in a computer - like pictures - while respecting their physical properties. The second challenge is to expand machine learning methods to work on mathematical and numerical objects with more structure, in a field known as geometric deep learning. Having a wider range of mathematical objects gives us more freedom to efficiently represent our natural objects for machine learning.

Structural biology is a scientific field aiming at understanding life by using the 3D structures of important molecules, available thanks to experimental and computational tools. This field thus relies on structured data that could be amenable to machine learning if the two challenges above were addressed. Among the major applications of this structural biology is drug discovery that aims at finding potential drugs in a vast space of chemical compounds. In the target-centric approach, the 3D structures of target molecules are used to select these promising compounds. This approach could be revolutionized by the use of geometric deep learning.

We start by a methodological contribution that helps respect the structure of DNA molecules represented as strings. Indeed, such representation traditionally overlooks the reverse-complement symmetry that stems from the double-stranded nature of DNA. By using the theory of equivariance, we characterize the class of machine learning models that respect this additional structure. We empirically show that using this class of models enhances the accuracy of the prediction of transcription factor binding.

We then advocate for the use of a specific type of graph to represent RNA in conjunction with deep learning methods for graphs. This specific type of graph is a coarsened, discrete representation that was introduced by biochemists twenty years ago. We show that using this representation is superior to using basic graphs and sufficient to extract a signal relevant to RNA drug discovery. Moreover, we can leverage this learning framework to efficiently detect structural motifs in RNA, relaxing constraints imposed on these motifs by pre-existing tools. We release a package to use this representation in machine learning applications.

Finally, we introduce three tools to help target-centric drug discovery that rely on machine learning. We offer a tool that is dedicated to finding the binding sites at protein-protein interaction sites by simultaneously predicting small-molecules and protein binding. We also propose a tool to efficiently cluster together conformations from a molecular dynamics trajectory, enabling for selection of relevant representative conformations. Lastly, we propose a method that iteratively generates populations of compounds with higher and higher predicted affinity to a given target.

Résumé substantiel

Dans le contexte d'une thèse rédigée en anglais en France à l'université Paris-Cité, nous incluons un résumé plus long (12 000 caractères) qui reprend le déroulé de la thèse en français

Apprentissage profond géométrique pour la bioinformatique structurale

L'apprentissage automatique consiste en un ensemble de méthodes algorithmiques qui utilisent de l'expérience pour améliorer leur performance. Le domaine a commencé à apparaître avec un enthousiasme modéré dès les années 50. En 2012, un réseau de neurones remporte une compétition de classification d'images, marquant le début d'une explosion d'intérêt autour de ce domaine. Depuis cette époque, l'approche dominante d'apprentissage automatique est d'ajuster les valeurs des paramètres d'un modèle pour en minimiser l'erreur sur des données connues. Les progrès de ces méthodes d'apprentissage ont révolutionné de nombreux domaines du numérique et le traitement automatisé de leurs données. Ces méthodes ont été particulièrement optimisées pour gérer des objets mathématiques bien précis qui sont les représentations numériques du texte, de l'image ainsi que des données tabulaires.

La recherche scientifique est basée sur l'accord entre des modèles explicatifs avec des mesures expérimentales. Il est donc naturel de vouloir appliquer des méthodes d'apprentissage à des questions scientifiques, avec des applications à succès comme AlphaFold. Celà étant, représenter un objet naturel par un objet numérique tout en respectant ses propriétés physiques représente un défi de modélisation. Le deuxième défi est d'étendre les méthodes d'apprentissage à de nouveaux objets mathématiques et numériques avec plus de structure, un domaine de recherche connu sous le nom d'apprentissage profond géométrique. En effet, avoir un éventail plus large d'objets mathématiques pour l'apprentissage automatique nous donne plus de liberté pour modéliser efficacement nos objets naturels.

La biologie structurale est un domaine scientifique visant à comprendre le vivant en utilisant les structures tridimensionnelles de molécules importantes. Les structures de ces objets sont une représentation qui se prête bien à des simulations physiques puisque celles-ci impliquent souvent des forces dépendantes de la distance géométrique. Malheureusement, les simulations physiques peinent à simuler les interactions entre grosses biomolécules. Par ailleurs, des données structurales sont disponibles grâce à des outils expérimentaux et informatiques. Il est donc raisonnable de vouloir s'appuyer sur ces données et l'apprentissage automatique pour répondre aux questions insolubles par des méthodes traditionnelles. Ces données structurées ne sont ni du texte ni des images, si bien que les exploiter avec de l'apprentissage automatique nécessite de relever les deux défis mentionnés ci-dessus.

Parmi les principales applications de la biologie structurale figure la découverte de médicaments, qui vise à trouver de potentiels médicaments dans un espace de composés chimiques immensément vaste. Ce processus est encore très long et coûteux (15 ans et 2 milliards de dollars en moyenne). Il dispose lui aussi de données et des méthodes d'apprentissages ont déjà été déve-

loppées pour explorer plus efficacement l'espace chimique. Ces petites molécules agissent en se fixant à des biomolécules, appelées cibles thérapeutiques, dont elles modifient le comportement. C'est là qu'intervient la biologie structurale. Dans l'approche centrée sur les cibles thérapeutiques, les structures tridimensionnelles des cibles sont utilisées pour sélectionner les composés les plus prometteurs. En raison du caractère structuré de ces données, peu de méthodes d'apprentissage ont été utilisées dans cette approche. Celle-ci pourrait donc être révolutionnée par l'utilisation de l'apprentissage profond géométrique.

Dans ce travail nous essayons d'aborder les deux défis susmentionnés et de répondre à la question de comment faire des algorithmes d'apprentissages efficaces sur les structures des biomolécules. Outre des applications fondamentales, nous utilisons le domaine d'application de la découverte de médicaments pour jauger de l'efficacité de nos méthodes. Par ailleurs, nous essayons aussi de contribuer dans ce domaine en identifiant des tâches qui peuvent se prêter à l'application de nos méthodes et ainsi être réalisées plus efficacement.

Nous commençons par une contribution méthodologique qui permet de respecter la structure mathématique des molécules d'ADN représentées sous forme de chaînes de caractères. En effet, la manière la plus classique de récolter des données sur l'ADN est d'utiliser un ensemble de technologies appelées des séquenceurs. Ces séquenceurs lisent indifféremment les deux brins de l'ADN ce qui aboutit à la présence de deux résultats possibles et équivalents pour le même objet. La représentation par chaîne de caractères néglige cette symétrie du brin complémentaire. En utilisant la théorie de l'équivariance, nous caractérisons les fonctions linéaires et les fonctions de non linéarités ponctuelles qui respectent cette symétrie. Nous montrons empiriquement que prendre en compte cette propriété améliore la précision de modèles qui prédisent la liaison des facteurs de transcription. Nous rendons disponible notre implémentation à la communauté.

Nous nous sommes aussi intéressés à la modélisation de l'ARN pour l'apprentissage automatique. La flexibilité de l'ARN le rend difficile à modéliser. De plus, elle complique sa détermination structurale, ce qui diminue la quantité de données disponibles pour les algorithmes d'apprentissages. Il faut donc utiliser une représentation numérique qui encode déjà beaucoup de propriétés importantes de ces molécules. Un type spécifique de graphes mathématiques, qu'on appelle graphe 2.5D, est une représentation gros-grain et discrète introduite par les biochimistes il y a vingt ans. Les nœuds représentent des nucléotides d'ARN et les arêtes sont divisées en 13 catégories qui représentent des types d'interactions chimiques. Dans les cinq dernières années, des méthodes permettant de conduire de l'apprentissage sur des graphes ont été développées. Nous préconisons l'utilisation de graphes 2.5D pour représenter l'ARN en conjonction avec des méthodes d'apprentissage profond pour les graphes pour apprendre sur des structures d'ARN.

Dans une première contribution, nous associons à des sites de liaison ARN, un pharmacophore qui est un ligand idéal permettant de trier les potentiels ligands de cette poche. Pour ce faire, nous utilisons un algorithme d'apprentissage sur les sites de liaisons représentés par des graphes 2.5D. Nous montrons que cette méthode permet de trouver un signal supérieur aux méthodes existantes, et pertinent pour la découverte de médicaments ciblant l'ARN. Par ailleurs, en entraînant également l'algorithme sur des graphes classiques, nous montrons que l'utilisation de graphe 2.5D est supérieure à l'utilisation de graphes standards.

Motivés par ce résultat, nous tirons parti de l'utilisation jointe de cette représentation et de méthodes d'apprentissage sur graphe pour détecter efficacement des motifs structuraux dans l'ARN. Ces motifs sont des sous-structures observées fréquemment dans l'ARN. Une représentation latente, continue et vectorielle de sous structures dans ces graphes est apprise par une

méthode sur graphes. Cette représentation permet d'aligner les structures expérimentales de l'ARN dans l'espace latent et d'y trouver des sous structures abondantes et qui coexistent fréquemment. Notre méthode permet de trouver des graphes similaires à un motif de graphe donné en entrée. Elle permet également de chercher automatiquement les motifs présents dans une base de données. Cette formulation permet de relâcher les contraintes sur les motifs structuraux imposées par les outils préexistants. Nous montrons que notre méthode permet de retrouver et étendre des motifs connus. Elle permet également de découvrir de nouveaux motifs dans l'ARN.

Nous avons de plus publié un paquet Python pour faciliter l'utilisation de graphes 2.5D pour représenter l'ARN pour l'apprentissage automatique. Ce paquet inclut la représentation de toutes les structures d'ARN connues par des graphes 2.5D. Il comprend également de nombreuses fonctionnalités pour télécharger, manipuler, comparer et dessiner ces graphes. Enfin plusieurs modèles d'apprentissages sont disponibles ainsi que des procédures d'entraînement. Nous espérons que les résultats positifs obtenus et la disponibilité de ce paquet favoriseront l'adoption des graphes 2.5D pour l'apprentissage sur la structure de l'ARN.

Enfin, nous présentons trois outils qui reposent sur l'apprentissage automatique pour aider à la découverte de médicaments centrés sur les cibles thérapeutiques. Dans un premier temps, nous proposons un outil dédié à la recherche de sites de liaison aux sites d'interaction protéine-protéine. Ces sites sont des cibles thérapeutiques intéressantes car elles correspondent souvent à une partie fonctionnelle d'une protéine : par exemple le site de liaison d'un pathogène à son hôte. Cependant ces cibles sont plus délicates car les interfaces d'interactions protéine-protéine sont souvent moins favorables à la liaison de petites molécules et les exemples sont moins nombreux dans les bases de données. Nous appuyons notre méthode sur une base de données développée en interne et récemment publiée qui contient ces exemples filtrés. Nous entraînons ensuite un algorithme à prédire simultanément pour une protéine donnée, ses sites de liaison aux petites molécules et aux autres protéines. Cet algorithme utilise la structure de la protéine donnée. Il a été ajusté automatiquement en collaboration avec IBM. Nous montrons qu'il est capable de détecter efficacement les sites de liaisons de petites molécules en général et tout particulièrement ceux qui sont à un site potentiel d'interaction protéine-protéine. Nous permettons aussi le suivi de trajectoires de dynamiques moléculaires pour avoir une estimation plus fine qui prend en compte l'aspect dynamique des structures de protéines. Enfin, nous avons aussi implémenté une interface utilisateur simple d'accès pour favoriser l'adoption par la communauté.

Nous proposons également un outil pour regrouper efficacement les conformations d'une trajectoire de dynamique moléculaire, permettant la sélection de conformations représentatives pertinentes. Cet outil repose sur l'algorithme des cartes auto-organisatrices et avait déjà été proposé il y a quelques années. Nous proposons une nouvelle implémentation qui utilise une carte graphique permettant d'aller plus de cent fois plus vite. Cette implémentation permet également un apprentissage par lots qui permet l'utilisation de la méthode sur des données de taille arbitraire.

Enfin, nous proposons une méthode pour générer des petites molécules optimisées pour une cible thérapeutique. Elle part du constat que de nombreux modèles génératifs ont été développés pour les petites molécules dans les cinq dernières années. Ces modèles, entraînés sur des millions de molécules existantes permettent notamment d'explorer de nouvelles molécules. Dans notre projet, nous proposons notre propre modèle génératif qui mélange des représentations de petites molécules par graphes et par chaînes de caractères. Nous montrons qu'il a des performances comparables à l'état de l'art tout en étant un ordre de grandeur plus rapide. De plus,

ces modèles permettent de faire de l'optimisation moléculaire pour trouver des composés avec certaines propriétés, une approche dite centrée sur les ligands. L'inclusion d'information sur la cible est difficile car pour chaque composé, elle nécessite des calculs plus coûteux que de simples propriétés moléculaires. Nous proposons d'utiliser une méthode d'optimisation compatible avec un calcul coûteux en conjonction avec notre modèle génératif. Nous montrons que notre outil est capable de générer des populations de composés avec une affinité accrue pour une cible.

Dans l'ensemble, ce travail étudie comment l'apprentissage automatique peut aider la biologie structurale et son application à la conception de médicaments. Nous montrons que des méthodes mathématiques dédiées ainsi que des représentations appropriées pour modéliser les biomolécules améliorent les performances des approches par apprentissage automatique. Nous montrons par ailleurs que l'utilisation de modèles d'apprentissage automatique peut être utile pour assister et potentiellement révolutionner la découverte de médicaments.

Contents

List of Figures

List of Tables

Glossary

ABPN : Augmented Base Pairing Network
AI : Artificial Intelligence
AuROC : Area under the Receiver Operator Characteristic
BO : Bayesian Optimization
CNN : Convolutional Neural Networks
CPU : Core Processing Unit
Cryo-EM : Cryogenic Electron Microscopy
DFT : Density Functional Theory
DNA : DeoxyriboNucleic Acid
EF : Enrichment Factor
FEP : Free Energy Pertubation
GAN : Generative Adversarial Network
GCN : Graph Convolutional Networks
GED : Graph Edit Distance
GPU : Graphical Processing Unit
HD : Hetero Dimer
HPO : Hyper Parameter Optimization
iPPI : inhibitors of Protein-Protein Interaction
LBO : Laplace Beltrami Operator
MAA : Motif Aggregation Algorithm
MACCS : Molecular Access Keys
MCMC : Markov Chain Monte Carlo
MD : Molecular Dynamics
MOA : Mechanism Of Action
MPNN : Message Passing Neural Networks
MSE : Mean Squared Error
NRM : Nuclear Magnetic Resonance
PDB : Protein Data Bank
PL : Protein Ligand

PPI : Protein-Protein Interaction
QED : Quantitative Estimate of Druglikeness
QSAR : Quantitative Structure-Activity Relationship
RCPS : Reverse Complement Parameter Sharing
RC : Reverse Complement
RGB : Red Green Blue
RGCN : Relational Graph Convolutional Network
RL : Reinforcement Learning
RNA : RiboNucleic Acid
SOM : Self-Organizing Maps
TF : Transcription Factor
VAE : Variational AutoEncoder

Chapter 1

Introduction

Contents

Abstract

Machine learning methods have revolutionized our approach to problems that previously seemed out of reach. In applications with abundant data, these methods surpass any other. On the other hand, classical approaches fail to answer some questions in the field of structural biology and drug discovery, while an increasing amount of structural data is available. This raises two related questions : How to conduct machine learning on the structure of biomolecules ? Can machine learning help drug discovery ? These two questions are related because a better toolbox means increased opportunities to use it, but to judge the relevance of the tools, we need an end goal performance to enhance.

To answer the first question, we decompose the problem into a mathematical problem and a modeling problem. Classical machine learning methods apply to vector spaces, and dedicated tools must be developed to learn over objects with more mathematical structure. This development widens the set of objects we can use to model our biomolecules. We must then decide which representation to use for which molecule, based on prior domain knowledge.

Equipped with machine learning for biomolecules, we can leverage the available structural data to help drug discovery. This represents an additional challenge as one needs to find new learning formulations for problems that have been addressed with classical methods. Our results show that using the right tools and the right representation yields better results for learning algorithms on the structure of biomolecules. We also show that they enable solving difficult real-world problems in an efficient way.

Résumé

Les méthodes d'apprentissage automatique ont révolutionné notre approche à l'égard de problèmes auparavant inatteignables. Pour les problèmes où beaucoup de données sont disponibles, ces méthodes surpassent toutes les autres. D'autre part, les approches classiques peinent à répondre à certaines questions dans le domaine de la biologie structurale et de la découverte de médicaments, alors qu'une quantité croissante de données structurales est disponible. Cela soulève deux questions liées : Comment mener un apprentissage automatique sur la structure des biomolécules ? L'apprentissage automatique peut-il aider à la découverte de médicaments ? Ces deux questions sont liées car de meilleurs outils signifient plus d'opportunités d'utilisation, mais pour juger de la pertinence des outils, nous avons besoin d'un objectif final à améliorer.

Pour répondre à la première question, nous décomposons le problème en un problème mathématique et un problème de modélisation. Les méthodes classiques d'apprentissage automatique s'appliquent aux espaces vectoriels, et des outils spécifiques doivent être développés pour apprendre sur des objets avec plus de structure mathématique. Ce développement élargit l'ensemble des objets que nous pouvons utiliser pour modéliser nos biomolécules. Nous devons ensuite décider quelle représentation utiliser pour quelle molécule, sur la base de notre connaissance préalable du domaine.

Équipés d'un apprentissage automatique pour les biomolécules, nous pouvons tirer parti des données structurales disponibles pour aider à la découverte de médicaments. Cela représente un défi supplémentaire car il faut trouver de nouvelles formulations par apprentissage pour des problèmes jusqu'alors abordés par des méthodes classiques. Nos résultats montrent que l'utilisation des outils et de la représentation pertinents donne de meilleurs résultats pour des algorithmes apprenants sur la structure des biomolécules. Nous montrons également qu'ils permettent de résoudre efficacement des problèmes réels difficiles.

1.1 Preface

My PhD lies at the intersection of several fields. I have thus chosen to write a short primer on machine learning in general along with a primer on structural biology and drug discovery, to set the stage for the actual dissertation. One can see this as a useful reference point for machine learning experts interested in the content but who need some introduction to the biology side, or vice versa. Newcomers can read it entirely and practitioners skip it altogether.

Statistical machine learning (see section A.1) is the process of approximating a theoretical relationship between theoretical objects, based on empirical observations. The observations we have are considered as samples from a theoretical underlying distribution and are then further represented into a format that a computer can deal with. Then the theoretical relationship is modeled with a flexible parametric function between our represented samples. An optimization is conducted to find the parameters of the function that give the lowest error for our model - this process is called *learning*.

The field of biology is concerned with the study of living organisms, who display a hierarchical organization. The bottom up approach to understanding life, named molecular biology, studies bio-molecules and their interactions to explain biological processes. Structural biology explains these molecular interactions using the spatial structure of these molecules. This motivates the development of experimental and computational tools to determine this structure and to predict function from structure. See section A.2 for more details.

Exactly solving quantum chemistry equations is not tractable for the structure of the large molecules of structural biology. Besides, more and more structural data is available thanks to experimental progress and to decades of joint community effort. Therefore, a promising avenue to answer structural biology questions is to leverage these empirical measurements with machine learning. This raises the question of how to conduct machine learning on the structures of biomolecules.

The first challenge is to respect the mathematical properties of structural data, because it cannot trivially be represented as the vectors that are manipulated by classical machine learning algorithms. We investigate dedicated machine learning methods in section 1.2 and also expand this toolbox in chapter 2. This broadens the set of mathematical objects that we know how to perform machine learning on. The second challenge is to respect the known properties of biomolecules. We need to understand what are the important properties and suitable representations of each family of molecules (section 1.3). We then empirically investigate which of these representations are best for learning on structures throughout the thesis.

Understanding life also opens the door to human intervention in those natural processes, for instance to fix the molecular malfunctions that cause a disease. The most classical way to do so is small-molecule drug discovery (see section A.3). Drug discovery has already started to benefit from machine learning methods. However, the structural information that is used by traditional *target-centric* drug discovery is still relatively under-exploited by machine learning methods (see section 1.4). This is partly due to the question of how to learn on these objects being unanswered. We chose structure-based drug discovery as our main application for geometric deep learning methods and validated that these methods enhanced existing drug discovery tools in chapters 3, 6 and 8.

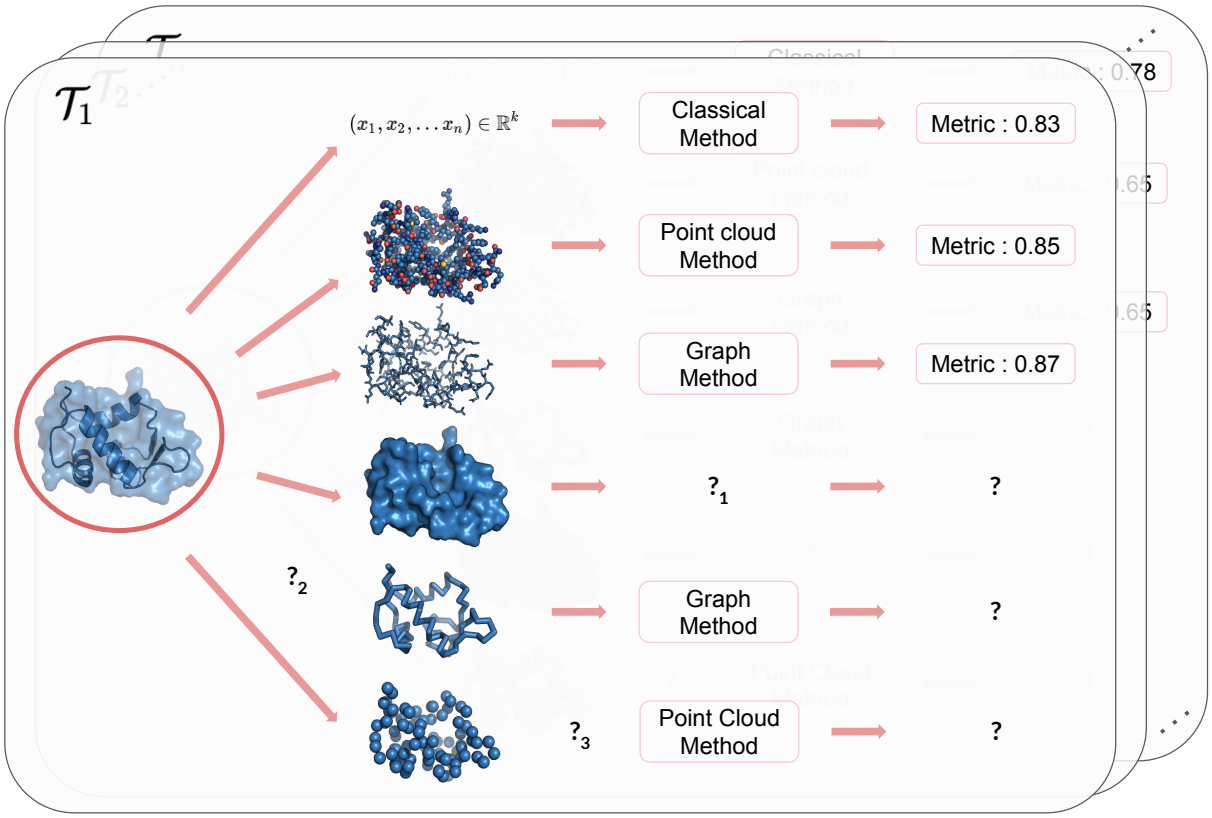


Figure 1.1 – Visual abstract of the context of the thesis : We try to solve a task $\mathcal{T}_1$ using machine learning, yielding a result in terms of a metric ranging from zero to one (we show arbitrary values). The classical vector representation of the protein as a k-dimensional vector yields a metric value of 0.83. However we could learn on different protein representations, using graphs or surfaces for instance. This raises the question of which numerical objects can be used in machine learning. For instance the methods to learn on surfaces are not fully established yet ($?_1$). It also raises a question of modeling, for instance there are several ways to represent a protein as a coarse-grained graph ($?_2$). Even equipped with a known mathematical model and methods that are able to learn on this mathematical structure, most combinations still need to be explored. For instance, to our knowledge, there are no applications of point cloud learning method to point clouds of residues ($?_3$). Finally, finding the tasks $\{\mathcal{T}_1, \mathcal{T}_2, \dots \mathcal{T}_n\}$ of relevance for drug discovery is also a challenge. We believe that all these questions must be addressed simultaneously, because they have related answers.

1.2 Geometric Deep Learning

Traditional machine learning manipulates vectors without mathematical structure. However, we sometimes know that our data has mathematical properties beyond those of a simple vector space. This is especially true of structured objects, which are of paramount importance for our structural biology applications. Geometric deep learning is concerned with methods that we obtain from these mathematical properties of our data [Bronstein et al., 2017, 2021].

A first source of mathematical structure is the existence of a distance between points that is not the euclidean distance - motivating the term *geometric* learning. The first main illustration of such spaces is surfaces - more formally manifolds - , where the distance between two points is given by the geodesic distance. The other main application is mathematical graph data where points are related by a neighborhood function. We will see that the main challenge with these structures is to take the induced neighborhood relationship into account in learning algorithms.

Another source of mathematical structure comes from group symmetries of the data. This symmetry often arises because we need to represent our data in a computer using an arbitrary decision. For instance, the unordered set $\{1, 2\}$ will be represented in our computer as one of the two arrays $[1, 2]$ or $[2, 1]$ - we arbitrarily pick an order. The major source of arbitrary choices in our application is the choice of a basis to represent 3D coordinates. The right framework to deal with such symmetries is the group equivariance framework. We will now detail how to deal with these two mathematical structures in machine learning methods.

1.2.1 Equivariant networks

These networks are useful when a group acts on our input or output. Groups are a very basic algebraic notion, where a set is endowed with an operation that combines its element to give another element of the set. The set of real numbers endowed with additions, or invertible matrices with matrix multiplication are groups. On a finite vector space, representing a group simply consists in associating an invertible matrix to each element of the group.

Formally speaking, a group G is a set endowed with an operation $\bullet : G \times G \to G$ that is associative, has an identity element and an inverse element for each element of the set. Now let $\mathcal{X}$ be a set, a group action ρ is a map $\rho : G \to \text{Bij}(\mathcal{X})$ such that $\forall g_1, g_2 \in G, \rho(g_1 \bullet g_2) = \rho(g_1) \circ \rho(g_2)$. If $\mathcal{X}$ is a vector space, and the image set of ρ is restricted to bijections that are linear, ρ is said to be a linear representation of the group. Linear representations were extensively studied and we refer the reader to [Serre, 1977] for an excellent introduction. The easiest example of a representation is the **trivial representation** that associates the identity on $\mathcal{X}$ to all elements in the group : $\rho_{triv}(g) = I_{\mathcal{X}}$. For a finite group, let us consider the group elements as the basis elements of a vector space $\mathcal{X}$. The **regular representation** is defined as $\rho_{reg}(h)(\lambda_i g^i) = \lambda_i (h \bullet g^i)$.

Let us now introduce a key property of group representations. Let $\mathcal{Y}$ be a vector subspace of $\mathcal{X}$, we say it is stable under ρ if $\forall g \in G, \rho(g)(\mathcal{Y}) \subset \mathcal{Y}$. The restriction of the representation to $\mathcal{Y}$ is a representation of G denoted as a subrepresentation. A representation without sub-representation except for the restriction to zero is called an irreducible representation. They depend only on G and are called the **irreps** of G. For any finite dimensional vector space $\mathcal{X}$, any representation ρ decomposes into a direct sum of irreducible representations.

Given two group representations ρ_A and ρ_B on two spaces A and B, a function $\psi : A \to B$ is **equivariant** if $\psi \circ \rho_A = \rho_B \circ \psi$. This means that the function on a group-perturbed input yields the same result as applying the group perturbation on the output of the function on the

native input. A notable instance of equivariance is invariance when the representation on B is the trivial representation. The function output is then constant with regards to input group actions.

As an illustration, we can think of the set of real numbers between 0 and 2π endowed with addition modulo 2π. This constitutes a group, also known as SO(2), the group of 2D rotations, where the number is the 'angle' of the rotation. A natural group action on $\mathbb{R}^2$ is ρ_A, that rotates vectors canonically. This group can also act on higher dimensional vectors by a representation ρ_B rotating their first two coordinates and leaving the others put. These two representations for a rotation of angle α can be mathematically written as :

$$\rho_A(\alpha)(x_1, x_2) = cos(\alpha).\ x_1 - sin(\alpha).\ x_2, \quad sin(\alpha).\ x_1 + cos(\alpha).\ x_2$$
$$\rho_B(\alpha)(x_1, x_2, \cdots, x_k) = \rho_A(\alpha) \otimes \rho_{triv}(\alpha) = \rho_A(\alpha)(x_1, x_2), x_3, \cdots, x_k\ .$$

Now let us imagine that we want to color 2D grayscale images with a parametric function $f_\theta : \mathbb{R}^2 \to \mathbb{R}^2 \times [0, 255]^3$. For each pixel of the image, this function returns three values between 0 and 255. We want this function to be equivariant with regards to the group actions mentioned above, because rotating the image should not affect its colors : $\rho_B \circ f_\theta = f_\theta \circ \rho_A$. This is illustrated in Figure 1.2

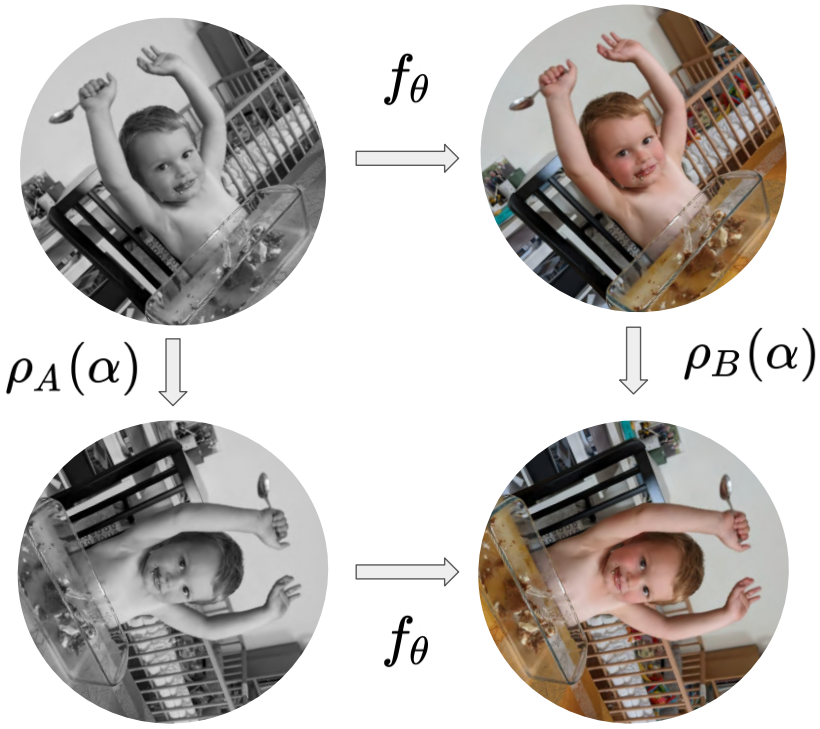


Figure 1.2 – Illustration of the concept of equivariance. If an equivariant colorizing function f_θ is used, rotation of the input yields a consistent output

Equivariant deep neural networks were introduced in a seminal paper [Cohen and Welling, 2016] and have sparked a great enthusiasm in the community. This paper introduced the formal-

ism of group representation for the group p4 acting on input images through rotations. Then, the authors choose a specific representation for the intermediate feature spaces, namely the one induced by the regular representation. They designed equivariant layers with regards to this group and group representation and empirically showed increased performances. This work has sparked great enthusiasm in the community. The same authors expanded their theoretical framework by enabling the use of any representations instead of only the regular ones, using the irreps decomposition [Cohen and Welling, 2017]. Then they expanded again their theoretical framework to include any group acting on a homogeneous space - for which the group action enables going from a point to any other - of this group [Cohen et al., 2019b]. The solutions exhibited in these papers are actually all equivariant linear maps. This was proven for regular representations first [Kondor and Trivedi, 2018], and for any representation then, by construction [Cohen et al., 2019b]. Another interesting approach for the same result was provided by Esteves [2020] while it was shown that all continuous equivariant functions could be approximated by a stack of equivariant layers [Dym and Maron, 2020], a classic universality result on classical neural networks.

Motivated by these theoretical results, several papers propose equivariant layers for specific groups. Dealing with rotation was chief among the motivation for equivariant networks. The first line of paper was mostly dealing with 2D rotations [Dieleman et al., 2016; Weiler et al., 2018b; Worrall et al., 2017] as equivariance can be obtained from engineering. Several papers were proposed also for 3D rotations [Anderson et al., 2019; Kondor, 2018; Kondor et al., 2018; Thomas et al., 2018; Weiler et al., 2018a]. However these methods involve using a basis of equivariant kernels that rely on the spherical harmonics. These harmonics can be precomputed on a grid kernel and then used within the Volumetric framework [Weiler et al., 2018a]. Spherical harmonics require a small grid step to capture high frequencies, resulting in high memory usage. They can also be computed at discrete locations on the fly [Thomas et al., 2018], resulting in a computational overhead. The general implementations [Geiger et al., 2022; Weiler and Cesa, 2019] should be saluted. Some papers also introduce equivariant attention frameworks, going beyond the equivariant linear layers [Fuchs et al., 2020]. To address the computational burden of equivariant methods, an avenue is to use less general but more efficient methods [Satorras et al., 2021]. Equivariant networks were shown to help learning in several applications : medical imaging [Winkels and Cohen, 2019], reinforcement learning [van der Pol et al., 2020] or histopathology [Graham et al., 2020; Lafarge et al., 2021].

However, these networks were mostly developed in the context of euclidean spaces, where one could leverage that euclidean spaces are homogeneous spaces for the group of translations. Actually, this results carries over easily to the homogeneous setting of the 3D sphere and SO(3) group [Cohen et al., 2018]. However, some spaces of interest do not have a euclidean geometry and need specific tools.

1.2.2 Non euclidean spaces

In non euclidean spaces, the objects cannot be added or translated trivially. There exists however a meaningful notion of distance, that induces neighboring relationships that retain a lot of information. We will focus mainly on graphs and manifolds as they are both the main focus of the methods development and the most useful tools for our structural biology applications.

Graphs are defined as a pair $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is an unordered collection of objects, called

nodes, and $\mathbb{E}$ is a set of pairwise connections between these objects, called edges. We can define a path on this graph as a sequence of contiguous edges going from one object to another and the length of a path as the length of the sequence. The **shortest-path distance** is a pairwise distance defined as the shortest path between the nodes of the pair. Famous graphs include social networks with users and connections between them, or genealogical trees.

Manifolds formalize the notion of surfaces as objects that look like euclidean space locally. The most classical manifold example is the round surface of the earth. Locally it looks flat and can be represented accurately with flat maps, which is not true globally. One can go from one point of Riemannian manifold to another following the surface. The shortest such path again induces a distance that can differ widely from the euclidean distance. For instance, Paris and Auckland are eighteen thousands kilometers away while the diameter (longest straight line in a sphere) of the earth is only about thirteen thousand kilometers. Another example is how we can touch our hips even though our hands are *far* from our hips if we follow the surface of our body : our skin. Using this surface distance conveys useful information, for instance height can be robustly estimated using the distance from head to toe, even in a sitting position.

More formally, let M be a topological space and U $\subset$ M be an open subset of M. A chart (U, ϕ) is an homeomorphism from U to an open subset of the euclidean space of dimension n. An atlas on M is defined as collection of charts $\{(U_i, \phi_i), i \in \mathbb{N}\}$ such that $\cup_i U_i = M$. It is said to be differentiable if $\forall i, j, \quad \phi_i \circ \phi_j^{-1}$ is differentiable on $U_i \cup U_j$. Smooth manifolds are topological spaces equipped with a differentiable atlas. Given a point $p \in U$, one can define the set of derivable functions $\gamma : U \to \mathbb{R}$ such that $\gamma(0) = p$ and an equivalence relation if their derivative at 0 coincide : $\gamma_1 \sim \gamma_2 \iff \gamma_1'(0) = \gamma_2'(0)$. The equivalence class of this relation induce a vector space called the tangent space at p and denoted T_pM. A Riemannian metric is a set of inner product in T_pM defined for each p, and there always exist one for smooth manifolds. A Riemannian manifold is a smooth manifold equipped with a Riemannian metric. Given a differential curve $\gamma[a, b] \to M$, one can compute its length integrating the Riemannian norm of the curve derivative. This enable us to define the **geodesic distance** between p and p', as the minimum of the length of curves going from p to p'. One can discretize manifolds as graphs with vertices sampled on the manifold, and connections between points with a small geodesic distance. These two mathematical objects are illustrated in the Figure 1.3.

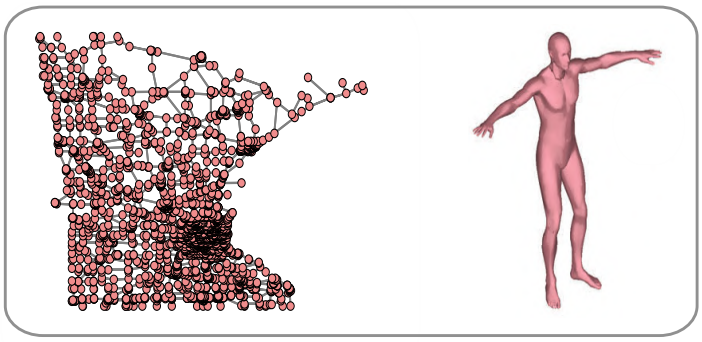


Figure 1.3 – Graph and Riemannian manifold examples. *Left :* The Minnesota road network where grey lines are the edges and represent roads while red circles are the nodes that represent road intersections. *Right :* Human represented as a coarse surface or manifold, sampled from the SCAPE database [Anguelov et al., 2005].

Now that we have introduced our mathematical objects, we want to derive parametric functions that take into account their properties. A prior information that was heavily used in computer vision is the fact that objects are made of components in a hierarchical organization, and that these components can appear in different parts of the input object. Road signs can appear in different parts of pictures and similarly, chemical functional groups can appear at several places in a molecule. However, if we want a model to learn to drive or to predict molecular toxicity, we would like to be able to learn these sub-components without regard to their localization. This has motivated the use of **convolution** with learned filters - Convolutional Neural Networks (CNNs) - in the euclidean domains [Fukushima and Miyake, 1982; LeCun et al., 1989, 1998]. This architecture was the foundation of AlexNet [Krizhevsky et al., 2012], a CNN that reached superhuman accuracy on image classification, allegedly the spark of the deep learning frenzy.

CNNs are often seen as a template matching algorithm, where the template is localized - small compact support - and learnt. This template is then *translated* on top of the image to produce an output signal. One can actually define convolutions as the linear operators that are equivariant to translation. On a regular grid topology, there is a very efficient implementation and the procedure has been extensively used. Convolutions have a tight connection to **Fourier** space through the convolution theorem. This theorem states that the Fourier representation of the convolution of two signals is the pointwise product of their Fourier decomposition. On non-euclidean spaces though, there is no easy way to even define convolution, because of the lack of a meaningful translation operator. To retrieve convolutions, there are two main avenues : the spectral and spatial ones.

Building convolution on graphs was pioneered using the spectral approach [Bruna et al., 2013; Henaff et al., 2015] and relies on the convolution theorem. Defining convolution as a product in the Fourier space alleviates the need for a translation operator, but poses the problem of generalizing Fourier to non-euclidean domains. This is done by noticing that the Fourier basis is the eigen basis of the **Laplacian** operator in 1D. There are generalized Laplacians such as the Laplace Beltrami Operator (LBO) or the Graph Laplacian, on manifolds and graphs respectively. By diagonalization of these operators, we obtain an eigen basis for our signals, analogous to a Fourier basis. This basis is exemplified in Figure 1.4. In this basis, we can then make pointwise products that correspond to spatial convolutions. In practice this approach has the major limitation of making the learning rely on the diagonalization step, both computationally costly and unstable across different graphs or manifolds. These limitations were addressed by ChebNet [Defferrard et al., 2016] that noticed that polynomials of the eigenvalues could be expressed as polynomials of the Laplacian - resulting in state-of-the-art performance. This approach served as a foundation for more modern methods like DiffusionNets [Sharp et al., 2022] and as a theoretical justification for spatial approaches.

The spatial approaches adapt the template matching view by defining ways to make local neighborhoods look euclidean, which is almost the definition of a manifold. Then a learnt template is applied in each of those neighborhoods. For instance, one can use a convolution with small spatial support in radial coordinates, where the radius is the geodesic distance and the angle is with regards to an intrinsic orientation [Masci et al., 2015]. The notion of local radial coordinates was then extended with variations yielding better performances [Boscaini et al., 2016; Monti et al., 2017]. For graphs, since the notion of angles does not make sense, one can compute filters that only depend on the shortest path distance or immediate neighborhoods

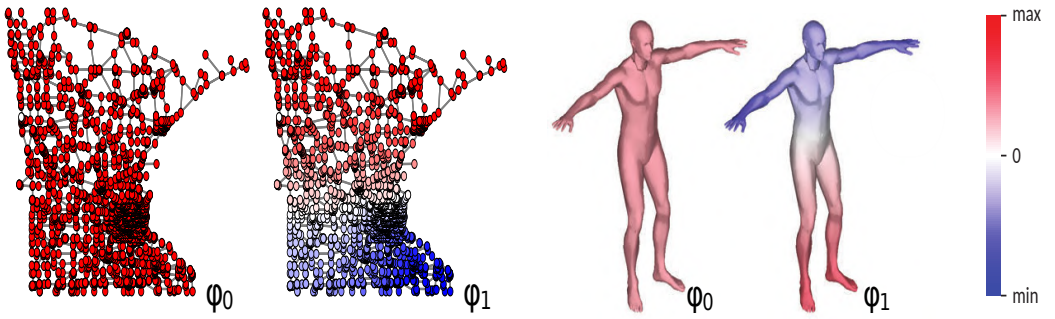


Figure 1.4 – The first two eigenvectors of the Laplacian on a graph and a manifold. The first one is a constant vector associated with a zero eigenvalue while the other one captures the longest dimension of the object (slowest diffusion). Figure adapted from [Bronstein et al., 2017].

[Kipf and Welling, 2016]. This framework was then extended to take edge types into account, simply by learning different filters by edge type [Schlichtkrull et al., 2018]. These methods were also discovered independently by the computational chemistry community to represent molecules [Duvenaud et al., 2015; Kearnes et al., 2016]. The general key components are convolution filters applied to each node and a symmetric function that aggregates the messages of all neighbors - a framework known as the Message Passing Neural Networks (MPNN) [Gilmer et al., 2017].

Classical machine learning is conducted over vector spaces. Geometric deep learning methods **extend the set of mathematical objects** we can perform machine learning on, to graphs, manifolds and sets acted on by a group. This field is still actively researched to keep expanding this toolbox, understand better these methods theoretically and make their implementation more efficient.

The development of these tools raise questions about their **applicability and usefulness**. Machine learning most successful methods operate on text and image data, which are generated in copious amounts by modern technologies, justifying these approaches. Geometric deep learning enables dealing with more sophisticated objects that arise in many more domains. The application of these methods often relies on a step of modeling natural objects, making the design of meaningful benchmarks more challenging. This disrupts the classical way machine learning research is conducted and calls for interdisciplinary collaborative design of models.

1.3 Representation of biomolecules

Some man-made objects are mathematical, as well as their canonical representation. When learning on such objects, aforementioned methods make complete sense. For instance, social networks *are* graphs, so to perform machine learning on social networks, one should use graph neural networks. On the other hand, bio-molecules are natural objects that are only **modeled** with one of these mathematical representations. The classical biomolecules representations include 3D images of voxels, atoms or residues represented as point clouds, possibly with edges

representing the chemical connectivity, coarse-grained representations, molecular surfaces or simply descriptors (see Figure 1.5). Which representation of biomolecules is the best for machine learning ?

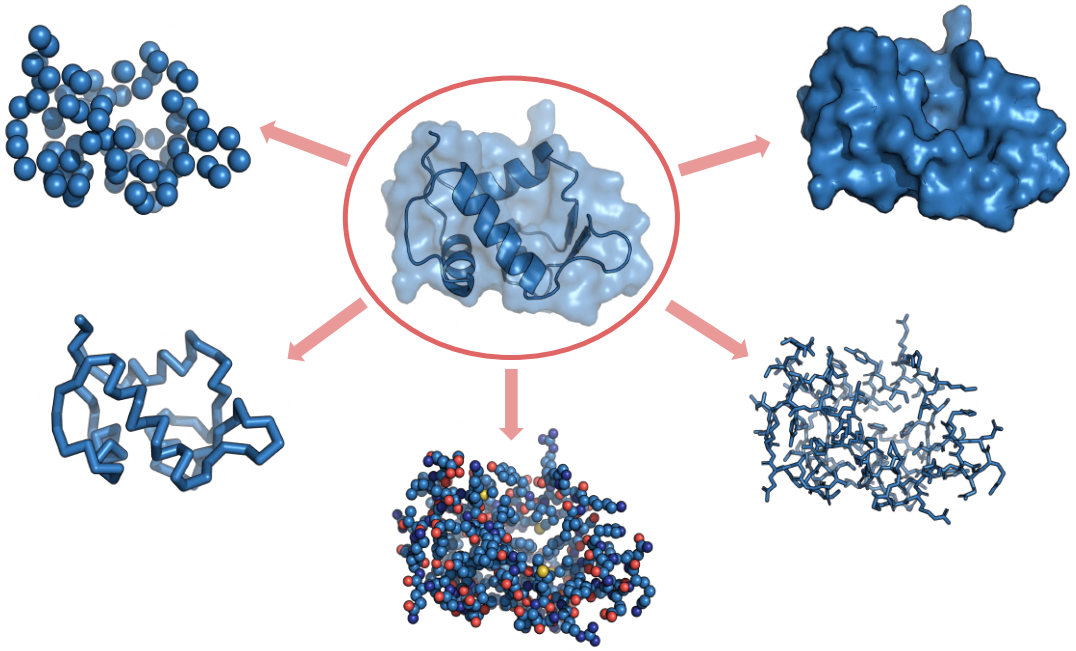


Figure 1.5 – The problem of choosing a representation exemplified on the modeling of a protein. Chain A of pdb 1ycr can be represented as its residues or its atoms, connected or not, as well as its surface

We will introduce and compare ways to represent each family of bio-molecules in a suitable way for machine learning and provide example applications. Because these representations are only a model, using one representation or the other is a matter of choice. This choice depends on the volume of data at hand, the task at hand and the prior we want to inject in our model. In their recent and timely work, ATOM 3D [Townshend et al., 2021] released a systematic benchmark of tasks on molecules. They also started comparing graph-based, voxel-based and equivariant methods. Several methods are still missing in this benchmark. To go further, models blending several concurrent representations are still rare [Gligorijević et al., 2021] but a promising direction to investigate.

1.3.1 DNA

Because of its stable double stranded helix structure, DNA is mainly represented as a 1D sequence. Therefore, the main machine learning models developed to functionally annotate usually rely on sequence method techniques [Alipanahi et al., 2015]. There is however a symmetry in the output of DNA sequencing : the sequencer will read both strands of the helix, yielding redundant output. To leverage this symmetry, several methods were developed. We generalize them and frame this redundancy in the equivariance framework in chapter 2. The 3D structure of DNA is an expanding field of study with novel experimental methods like Hi-C [Lieberman-Aiden et al.,

2009], and is a promising research direction.

1.3.2 RNA

RNA molecules have long been solely considered an intermediary messenger between DNA genomic information and protein structure. However, because only one strand is expressed, RNA has a non-trivial folding process and is able to hold a variety of functions in cells. Actually, a fraction (∼2% [Collins et al., 2004; Ezkurdia et al., 2014]) of it is translated into proteins, while the rest only holds a functional role. For this reason, it holds huge promise for drug discovery as it represents numerous novel targets [Warner et al., 2018]. RNA is however considered a more challenging target than proteins because of its **more flexible** structure. This greater flexibility disrupts the traditional pocket pipeline and also makes the structure resolution more challenging, resulting in a lot less data available (∼1.9k structures as opposed to ∼160k protein structure in the PDB [Berman et al., 2000]).

Traditional machine learning was used on RNA with feature vectors for a long time [Brouard et al., 2016; Kim et al., 2006]. The data sparsity makes the use of deep learning methods less amenable. One possible solution is to use transfer learning and to pretrain an RNA model on protein data, a solution very recently investigated in [Möller et al., 2022]. Another one is to use a representation with a strong inductive prior. In base-pairing networks, nodes represent nucleotides and the edges represent the interaction between two nucleotides, either backbone or the traditional base pair. These networks define the secondary structures of RNA and folding algorithms infer such connectivity from sequence [Gruber et al., 2008]. Base-pairing networks were used as features for machine learning [Maticzka et al., 2014; Uhl et al., 2019]. Actually, to represent the full 3D structure of an RNA molecule in a coarsened way, biochemists have extended these graphs with 12 categories of non canonical base-pairs [Leontis et al., 2002; Stombaugh et al., 2009]. This **2.5D graph** representation also includes a domain-knowledge prior, that can be paired with the use of graph neural networks. In RNAmigos (chapter 3), we showed that this representation increases performance compared to canonical graphs. We then leveraged this representation to mine structural motifs in Vernal (chapter 4) and released a library called RNAGlib (Chapter 5) that promotes the use of this representation for Machine Learning tasks using graph neural networks.

1.3.3 Proteins

Proteins are the biomolecules whose structures were most studied. This is especially true of the two protein families of enzymes and receptors. These families are the main drug targets of existing drugs [Hughes et al., 2011]. Proteins have a relatively stable structure - despite having some more deformable parts that are often the functional sites [Grünberg et al., 2006]. For this reason, structural data is most abundant for this class of molecules, along with machine learning approaches.

However, the most efficient way to represent a protein is still an open question. The framework that we found to be the most successful for our applications was the Volumetric CNN one, introduced in Jiménez et al. [2017]. This framework represents data on a 3D grid by Gaussian interpolation, and then leverages standard 3D convolution. It is a popular framework and was used in several works [Skalic et al., 2019c; Stepniewska-Dziubinska et al., 2020]. This is the

one we used for instance in InDeep (Chapter 6). This framework is not rotation equivariant by design, but approximate equivariance can be obtained from rotational augmentation.

Rotation equivariant networks are appealing for learning on structures represented in an arbitrary basis. However, as mentioned above, their computational limitations have limited their use on whole structures. There is an interesting usage of equivariant networks to represent binding pockets efficiently [Simonovsky and Meyers, 2020]. New equivariant methods are developed at a very rapid pace and some recent methods were successful on specific tasks on the structure of proteins [Stärk et al., 2022]. We can envision that equivariant networks practical limitations will be alleviated in the future with improvements in the methods as well as in the computing power available.

Graphs can also be used to represent a protein's connectivity. The nodes represent either the residues [Fout et al., 2017; Torng and Altman, 2019] or the atoms directly [Townshend et al., 2019], and the edges represent chemical bonds and distances. A limitation of these methods is that general graphs do not incorporate the notion of angles directly - even though a complete graph obviously encodes the geometry implicitly. This limitation disappears with the formalism of graphs embedded in 3D with an equivariant message passing scheme [Fuchs et al., 2020; Satorras et al., 2021], with no known applications to learning on proteins.

Finally surface methods are promising as they naturally encode the rotational invariance prior. They also encode the screening effect : atomic potentials decay at least as r^{-6}, making the interaction of buried residues with potential partners negligible compared with the surface residues. This representation was very successfully applied to protein data [Gainza et al., 2020; Sverrisson et al., 2021]. We believe the main obstacle to their adoption is the involved formalism of the manifolds and the lack of easy to use methods.

All of these representations are concurrent valid representations. In our experiments, despite their theoretical limitations, Volumetric CNNs have performed very well. We have repeatedly tried to leverage rotation equivariant networks to encode the rotational invariance prior, with little success. These findings were confirmed recently by ATOM 3D [Townshend et al., 2021]. They empirically find that for machine learning tasks on proteins, the best representation are either graph methods - for tasks with few data - or the volumetric approach - for tasks with more abundant data. They also report disappointing performance of equivariant methods both in terms of accuracy and computing time and argue that these methods are more suited to deal with small objects than full size proteins, also for memory issues. We believe that this benchmark lacks surface methods and are working on including them.

1.3.4 Small molecules

In the context of drug discovery, small molecules refer to compounds made of organic atoms that follow certain rules that make them potential drugs. The data available for small molecules is a bit different in nature as the one previously mentioned. First of all, it can be built from enumeration of those rules, generating a whopping 10^{23}∼10^{60} compounds estimated druggable chemical space [Bohacek et al., 1996; Ertl, 2003; Polishchuk et al., 2013]. Moreover, 3D conformations of small compounds can be efficiently generated in-silico, making available a **tremendous number of drug-like structures**.

For this reason, machine learning has been used to predict small molecules properties for a long time. A plethora of traditional feature vectors were developed to represent molecules for

machine learning [Durant et al., 2002; Glen et al., 2006; Rogers and Hahn, 2010]. These feature vectors are often deemed as molecular fingerprints and encode the presence or absence of given chemical groups.

This community has pioneered the use of geometric deep learning, introducing graph convolutional networks in a research stream independent from the geometric deep learning stream [Duvenaud et al., 2015]. Small molecules applications have also motivated a lot of the equivariant literature [Anderson et al., 2019; Thomas et al., 2018] and are pinned by Atom 3D as the main successful field of application of equivariant networks for molecular data. Several representations of the raw data can be used in machine learning with an increasing level of complexity, using a SMILES-like 1D string representation [Krenn et al., 2019; Weininger, 1988], the very classical molecular graph [Jiang et al., 2021; Kearnes et al., 2016], graphs embedded in 3D [Schütt et al., 2017] or the 3D atom positions of a given conformation [Thomas et al., 2018]. A review of these representations for learning was published very recently [Atz et al., 2021].

Finally, the abundance of data was leveraged successfully by **unsupervised models**, following the seminal Gómez-Bombarelli et al. [2018]. The goal of these papers is to learn a vector representation from the large quantity of data represented in a certain way. This learnt representation can then be used for other tasks, such as optimization. These papers rely on generative models, usually auto-encoders : an encoder network maps an input onto a vector space and a decoder network maps the vector back to the input space. The networks are trained to minimize the reconstruction error. These methods were originally applied for textual applications and so the first methods to apply it on molecules chose string molecular representations. However not all decoded strings are valid molecules, motivating the introduction of decoder constrained by syntactic rules [Dai et al., 2018; Kusner et al., 2017] and of SELFIES, a syntax where all generated strings are valid molecules [Krenn et al., 2019]. Some more recent methods are based on graphs [Jin et al., 2018a, 2020b] which benefit from a more efficient encoding, but graph decoding is not yet fully established despite great improvements in recent years. In OptiMol (chapter 8), we advocate for graph encoding and SELFIES decoding, taking the best of both worlds. Recent methods are based on chemical synthetic pathways [Gao et al., 2021], to avoid generating compounds that are not synthesizable, a known caveat of existing methods [Gao and Coley, 2020].

1.3.5 Molecules should not be represented as static objects.

The aforementioned methods mention molecules as objects that have a structure beyond vectors. They all consider these objects as one static object, such as the output of a crystallographic experiment. However, biomolecules are actually **dynamic**, meaning that each molecule constantly interacts with its neighborhood and changes shape. The most classical depiction of the dynamic nature is based on sampling conformations - a set of representative states of the molecule. An average of the property of each conformation robustly estimates the molecule's property. The most common method to get these sample conformations is Molecular Dynamics (MD). This method is based on integration of Newton's law of motion with force fields and has been the focus of thousands of papers [Abraham et al., 2015; Brooks et al., 2009; Case et al., 2005; Humphrey et al., 1996]. It is computationally expensive for large systems, and other methods based on Markov chain Monte Carlo can be used to get the samples for big systems [Alber et al., 2008]. Finding a set of conformations to obtain an accurate and unbiased estimate of a given molecule's

property is still an open problem.

The static representation being more simple and tractable, it has been overwhelmingly dominant in the machine learning papers. However, several papers have used machine learning models on each snapshot of a dynamic [Kozlovskii and Popov, 2020]. Moreover one can learn over this set of conformations using frameworks like multi-instance learning [Ilse et al., 2018]. This was pioneered in two recent papers, for small molecules [Zankov et al., 2021] and protein structures [Wu et al., 2022] but a more extensive investigation is promising.

Depending on the application, one might want to be sensitive or robust to a change in conformation. Intrinsic surface methods are isometry invariant, meaning that some deformations of the surface do not affect their outputs. Thus, surface representations could be more suitable for applications where we seek robustness with regards to conformation, whereas other representations using 3D spatial convolutions might be more sensitive.

We see that for each family of molecules, there are several possible representations that encode different priors and provide different properties of stability and performance to our models. Using **several different representations** with an ensemble of models or even a hybrid model that would leverage all of them is a promising direction. We note that the best representation can only be defined as the most useful for each specific task. We hope that general properties make the use of certain models more relevant than others overall, but our choices and conclusions ultimately depend on the puzzles we try to solve.

1.4 Application to drug discovery

We have chosen drug discovery as the main application of our methods. AI-augmented drug discovery is a major research direction, with existing success stories [Stokes et al., 2020; Zhavoronkov et al., 2019]. Major efforts have been put into ligand-centric approaches leveraging generative models. We however believe that with the development of geometric methods, learning methods are now ready to take into account the structure of targets and to revolutionize structure-based approaches.

Central to the structure-based drug discovery approach is the concept of binding pockets. A binding pocket is a region of the surface of a biomolecule that can interact with a ligand. Such a region is said to be ligandable. Moreover, to be a site of interest, this binding should result in a biological effect [Pérot et al., 2010]. A given site that checks both of these conditions is said to be druggable. The structure-based pipeline can be decomposed in two main steps of finding a pocket promising to be druggable and then finding promising compounds for this pocket.

1.4.1 Finding binding pockets

The task of finding binding pockets given a protein structure has a long history with computational chemistry with geometric methods like fpocket [Guilloux et al., 2009] being a classic in the field. Then other tools also try to assess the druggability of these pockets by giving them a score, with programs such as VolSite [Desaphy et al., 2012]. More recently, this problem was

approached with structured machine learning methods, like DeepSite [Jiménez et al., 2017] and many follow-ups [Kozlovskii and Popov, 2020; Mylonas et al., 2021; Stepniewska-Dziubinska et al., 2020].

However these methods are restricted to finding conventional pockets - e.g. enzymes and receptors binding sites. In the field of drug discovery, there is a need to investigate the druggability of other pockets to have **new targets** and thus search new regions of the chemical space for potential drugs. Two big avenues for this process of expanding targets are Protein-Protein Interaction (PPI) sites and RNA. As mentioned above, only a fraction of RNA is translated into proteins, making potential RNA targets numerous. Moreover, a different chemistry - articulated around RNA base pairing, riboses, and negatively charged phosphate groups - governs RNA pockets' microenvironments. Future RNA drugs are thus anticipated to be different from the patented chemical space. RNA targeted drug discovery is still in its infancy, with only one FDA approved drug targeting RNA. Thus, still only a few methods try to find RNA binding pockets [Möller et al., 2022; Su et al., 2021; Wang et al., 2018].

The other avenue for finding new targets is to look at PPI sites. Modulators of such sites could either prevent an important binding - such as the one between ACE2 and the Spike Protein - or stabilize one, disrupting a potential pathological pathway. To investigate the druggability of such sites, one needs dedicated databases, such as the iPPI-DB [Torchet et al., 2021] and tools, such as InDeep (chapter 6).

1.4.2 Estimating binding affinity

Once equipped with the structure of the target pocket, one wants to predict the affinity between this pocket and a molecular compound. The overwhelmingly dominant approach for this task is **molecular docking** [Amaro et al., 2018; Lang et al., 2009; Luo et al., 2019; Shoichet et al., 2002; Trott and Olson, 2010], with millions of molecules routinely docked by pharmaceutical companies and a long history of fine tuning these methods. They are based on a two-step process of sampling poses of the ligand in the binding pocket and scoring these poses. The scoring step uses various approaches such as force-field based, semi empirical or knowledge-based methods. The main shortcomings of docking are its limited precision and high computational cost ($\sim$1 minute per compound). Machine learning methods to estimate binding affinity from only the structures of protein target and molecular compounds exist [Hoffmann et al., 2010; Stärk et al., 2022], bypassing the pose sampling step. However they are not accurate enough yet to replace the traditional two-step pipeline.

The main avenue to enhance the docking protocol is to better score the generated poses. To take into account entropic, solvation and dissociation effects, Free Energy Perturbation (FEP) methods use MD sampling and energy computations to get an molecular affinity estimate [Wang et al., 2015]. Density Functional Theory (DFT) computations yield an even more accurate estimation of molecular affinity, at the cost of an even harder computational burden [Hohenberg and Kohn, 1964]. Another option is to learn a scoring function, training on bound structures along with an affinity measure [Ballester and Mitchell, 2010; Jiménez et al., 2018]. The learnt scoring function is then used to re-score the most promising poses, based on the native score. Geometric deep learning was recently used for these scoring functions with superior results and remains a very promising research direction [McNutt et al., 2021]. One of the inherent problems is again the fact that the affinity is not a function of one pose but rather of the ensemble of

poses. The current score aggregation functions often rely on the max function; we anticipate that a great benefit could be obtained by using another aggregation function, following the multi-instance learning framework [Ilse et al., 2018].

1.4.3 Finding the right compounds

To obtain an actual drug, one needs to find small molecules that exhibit a large set of specifications [Hughes et al., 2011]. The most obvious one is the binding affinity to our binding pocket, because we want our compound to disrupt the functioning of our target. However there are a lot of other factors to take into account. First of all, we need the molecule to be synthesizable and affordable. Some chemical space regions are under patents, that need to be avoided to find reasonable candidates for a new drug. Then we also need the drug to make its way to the protein we are interested in, adding some membrane permeability and solubility constraints. Finally we also need some specificity, to avoid its binding to other proteins and disrupting their normal functioning, especially for functionally important ones like the hERG family [O'Brien, 2014; Sanguinetti and Tristani-Firouzi, 2006].

All of these objectives need to be **jointly optimized**, because failing on any of these levels will cause the drug to malfunction, even with a high affinity. However there is no easy deterministic way to estimate most of these properties. Quantitative Structure-Activity Relationship (QSAR) models are the most common approach to solving this problem [Cherkasov et al., 2014]. They are machine learning models that extrapolate the values of molecular properties from empirical measurements on a limited set of molecules. Even affinity prediction can be formulated as a QSAR task by creating a model trained on experimental affinity for each target, only implicitly encoding the target information. Despite originally relying on molecular descriptors coupled with linear regression, more complex machine learning models were developed in the last 30 years. These models are actually one of the motivations for the introduction of graph neural networks [Duvenaud et al., 2015] and equivariant networks [Thomas et al., 2018]. Once we know how to predict these objectives, they can be blended into a composite score that we want to optimize.

Because of the noise of the experiments, the amount of factors to blend in and the limited accuracy of the predictive models, the score that we compute is only a noisy estimate of the drug-likelihood (see Figure 1.6). Therefore, we cannot suggest just one chemical compound and the preferred approach is to suggest a list of compounds to try out. Wet-lab assays are well suited to trying out a few hundreds of compounds (sometimes tens, sometimes thousands), which is approximately the number of compounds we will try to offer. Moreover, because we believe that similar compounds have a higher chance of having similar properties and thus to fail for a similar, unexpected reason, we want our suggestion to be as **diverse** as possible : *"The discovery of a single, yet structurally novel, hit ligand can be much more beneficial and informative in the early stages of a drug design project than the identification of many close analogues of a known ligand"* [Broccatelli and Brown, 2014]. We are then facing a conflicting objective : finding compounds that are optimized with regards to the score and finding compounds that are diverse.

The classical way to fulfill these objectives is to use the scoring function to screen existing databases and to retrieve the most promising with a constraint on diversity. A concurrent and rising approach relies on the aforementioned unsupervised generative models, that are trained to produce compounds similar to an existing database, the prior distribution. Such models

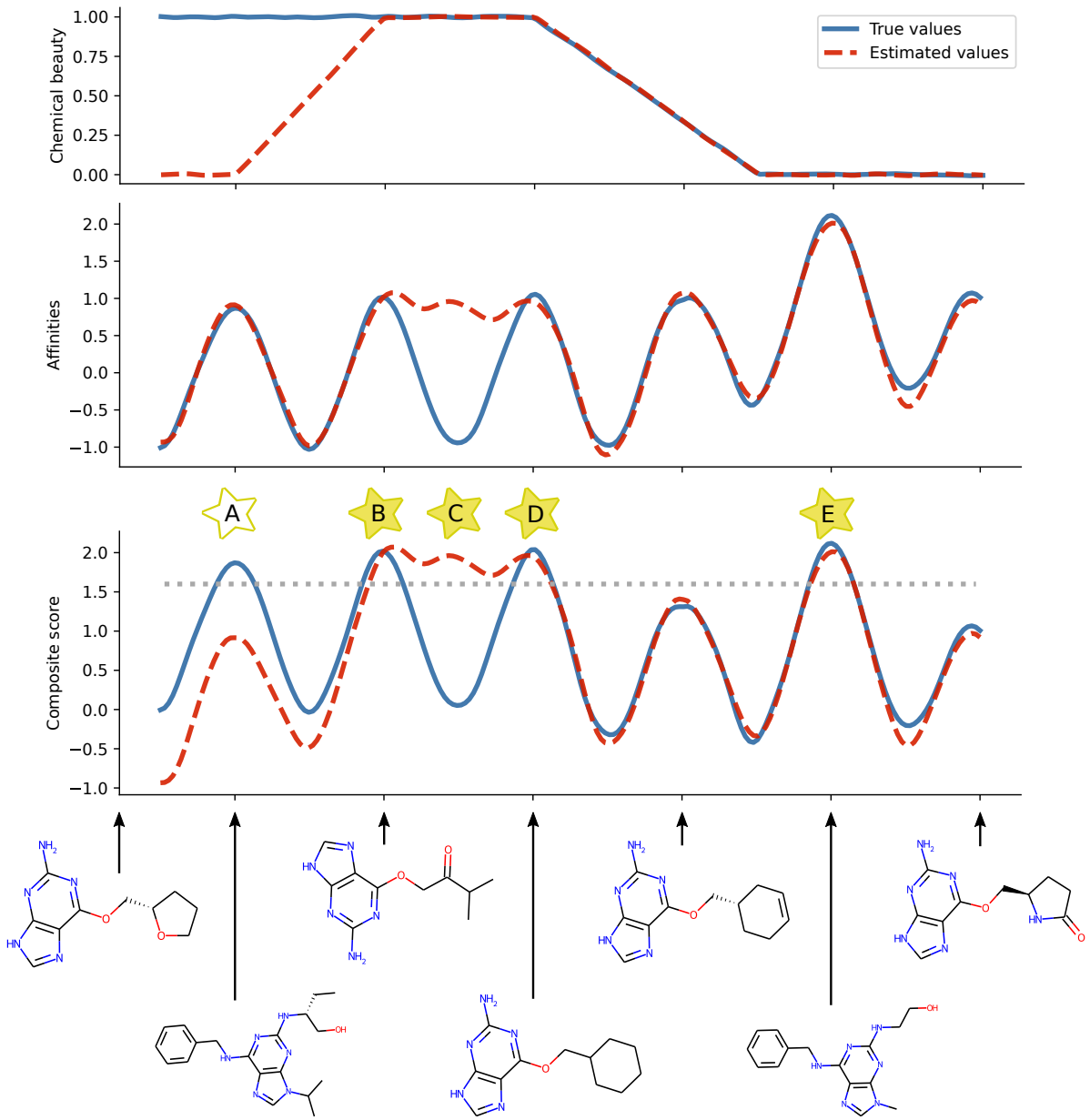


Figure 1.6 – Schematic illustration of chemical optimization. The x-axis represents the chemical space. Exact and computed Chemical Beauty (QED) and affinity values are plotted on the top, with optimistic and pessimistic model errors. These two scores are then aggregated (summed) into an example composite score.
This illustrates the difficulty of finding the right compounds. Let us imagine that our algorithm returns all local maxima above the dotted gray line threshold of the estimated score. We denote the results with full stars. Compounds B and D are satisfactory results. Compound A is wrongly discarded because of our pessimistic QED estimation. Compound C is retained but is an artifact resulting from our optimistic affinity evaluation. Compound E is selected because of its high affinity, but has a forbidding chemical beauty of zero, showing the shortcomings of the sum as a composite score. Some additional effects could exist that are not included in the score and depicted in this illustration, for instance compound B and its vicinity could be lethal for rats.

represent compounds in a vector space, opening the door to continuous optimization [Gómez-Bombarelli et al., 2018]. Several approaches were developed using direct optimization [Winter et al., 2019a,b], adversarial methods [Prykhodko et al., 2019], reinforcement learning [Olivecrona et al., 2017; You et al., 2018] or flow networks [Bengio et al., 2021]. Because the target based oracle - docking or DFT - is computationally expensive, target structure inclusion is still rare and needs special design (see [Skalic et al., 2019b] and chapter 8).

1.5 Contributions

We now introduce our work that follows the questions raised in this introduction. We start by investigating methods and representations for learning on biomolecules. We apply the equivariance framework to DNA sequences, to cope with the Reverse Complement symmetry in chapter 2. We show that representing RNA with the aforementioned 2.5D graphs yields superior machine learning performances in chapter 3. We then leverage this graph representation to mine structural motifs in RNA in a flexible way in chapter 4 and package several tools and utilities for this representation into a Pypi package in chapter 5.

We then apply structured machine learning methods to help drug discovery. This was already started in chapter 3, where the task we aim to solve is to associate a pharmacophore to an RNA binding pocket. In chapter 6, we address the druggability of PPI sites by detecting binding sites and following these sites along MD trajectories. We also propose a GPU-based tool, `quicksom`, to efficiently cluster MD trajectories and distribute it as another PyPi package in chapter 7. Finally we offer a tool to generate optimized populations of molecular compounds with regards to a user-defined, potentially costly score, compatible with docking simulations in chapter 8. We summarize these contributions and detail the role played in each of them below.

1.5.1 RC-Equivariant Networks for DNA Sequences

In this paper we are interested in the problem of learning models over DNA, represented with sequences. A translational symmetry prior in these sequences have motivated the extensive use of CNNs. There is however another additional structure in DNA, resulting from the reverse-complement symmetry. The two strands of DNA are paired deterministically so one can get the content of one from the other. The sequencing results do not take that redundancy into account.

We frame this additional structure as a group action on the input and design equivariant networks for DNA. We include a generality result, exposing all possible equivariant linear layers and all equivariant point-wise linearities : the two main components of classical neural networks. We then show how other existing methods can be framed as a subset of our framework. Finally we implement our networks and show that enabling optimization over this larger functional space yields better results.

This chapter was made in collaboration with Jean-Philippe Vert and was published in NeurIPS 2021 [Mallet and Vert, 2021]. I was the first author in this paper. Jean Philippe and I contributed equally to the theory and the writing and I implemented the networks and conducted the experiments.

1.5.2 Augmented base pairing networks encode RNA-small molecule binding preferences

In `RNAmigos`, we want to learn a model that associates a binding pocket to a pharmacophore of its corresponding ligand. Our prediction can be used as a prefiltering tool for finding binders in chemical libraries. We represent RNA binding sites as 2.5D graphs and use a graph neural network model.

We show that our prediction places the true ligand in the 71st percentile in two decoy libraries, showing a significant improvement over several baselines, and a state of the art method. This shows that RNA binding data contains structural patterns with potential for drug discovery. Furthermore, we observe that RNA 2.5D graphs are the only representation able to uncover a significant signal, suggesting that they are the right representation for conducting machine learning on RNA.

This chapter was made in collaboration with Carlos Oliver, Roman Sarrazin Gendron, Vladimir Reinharz, William L Hamilton, Nicolas Moitessier, Jérôme Waldispühl and was published in Nucleic Acid Research in 2020 [Oliver et al., 2020]. I was the second author in this paper. I contributed to the idea of the project, implemented part of the experiments and contributed to the writing.

1.5.3 Vernal: a tool for mining fuzzy network motifs in RNA

Motivated by the success of `RNAmigos`, we used graph neural networks to approximate the feature map of a structural kernel, defined on RNA 2.5D subgraphs centered around nodes. Then the dot product in this learnt representational space approximates the kernel, that in turn approximates an RNA graph edit distance. This procedure thus maps structurally similar nodes in the same regions of a high dimensional space. Embedding the graphs in this space, the frequent substructures are now organized in clusters and larger frequent subgraphs appear in the form of highly connected clusters. However, allowing the cluster to have a non zero spatial extension, we allow for similar but non-isomorphic structures to be grouped together. We then group together co-occurring clusters in an iterative process to discover larger motifs. We show that our method allows us to rediscover and expand known structural motifs, as well as to mine new fuzzy motifs.

This chapter was made in collaboration with Carlos Oliver, Pericles Philippopoulos, William L Hamilton and Jérôme Waldispühl and was published in Bioinformatics in 2022 [Oliver et al., 2022]. I was co-first author in this paper. We got the idea of the project together. Carlos conducted most of the data pipeline while I did the meta-graph formalization and implementation. We conducted the experiments and wrote together.

1.5.4 `RNAglib`: a python package for RNA 2.5 D graphs

Because our results advocated that RNA 2.5D graphs are a suitable representation for conducting machine learning on RNA structures, we decided to release a database of RNA in this representation with pre-annotation for machine learning tasks. We also developed, standardized and documented our data processing tools and machine learning pipelines and released them in the form of a package. This chapter was made in collaboration with Carlos Oliver, Jonathan Broadbent, William L Hamilton and Jérôme Waldispühl and was published in Bioinformatics

Application Notes in 2022 [Mallet et al., 2022b]. I was co-first author in this paper. Carlos and Johnathan conducted most of the data pipeline and I did the learning and documentation packaging.

1.5.5 `InDeep`: 3D fully convolutional neural networks to assist in silico drug design on protein–protein interactions

`InDeep` serves a more applied purpose : the automatic detection of binding pockets at PPI sites. We used a joint model that predicts both the interaction and the ligandable sites, from a protein structure. This model was trained on PPI modulators on a dedicated database - iPPI-DB - and its output was postprocessed with a dedicated procedure. The architecture and post-processing were optimized using automatic model optimization in collaboration with IBM research.

We show that the resulting model is competitive even on classic ligands and far more accurate on these PPI modulators. Moreover, we can track the ligandability along an MD trajectory and show the model identifies favorable conformations. Finally the tool ships with a visualization plugin in PyMol to ease its adoption by biochemists.

This chapter was made in collaboration with Luis Checa Ruano, Alexandra Moine Franel, Michael Nilges, Karen Druart, Guillaume Bouvier and Olivier Sperandio and was published in Bioinformatics, 2022 [Mallet et al., 2022a]. I was the first author in this paper. Guillaume and Olivier launched the project. Karen conducted most of the data pipeline. I implemented most of the project helped by Guillaume and improved the architecture and training procedure. We conducted the validation experiments also with the help of Luis. We wrote together.

1.5.6 `quicksom`: Self-Organizing Maps on GPUs for clustering of molecular dynamics trajectories

A few years ago, a tool based on SOMs to cluster molecular dynamics was released. In `quicksom`, we expanded this tool a lot, offering GPU support as well as enhanced data loading. This enables faster computation time and better scaling properties. We benchmark this version of the code against other implementations and show it is state of the art. After publication, we kept pushing this tool by offering a Jax [Bradbury et al., 2018] support. This chapter was made in collaboration with Michael Nilges and Guillaume Bouvier and was published in Bioinformatics Application Notes in 2021 [Mallet et al., 2021]. I was the first author in this paper. I implemented the method and packaged it. Guillaume helped with the experiments. We wrote together.

1.5.7 OptiMol: Optimization of Binding Affinities in Chemical Space for Drug Discovery

`OptiMol` aims at generating optimized populations of small-molecules compounds, as described in the drug discovery section. We introduced a new graph to SELFIES generative model for small-molecules. We formulated the problem of finding diverse and optimized compounds as a statistical conditioning of a prior distribution on higher and higher values of the property. This enabled us to leverage a statistical framework based on iterated updates, that can take advantage of a computational cluster. Therefore `OptiMol` can be used even if the score to optimize is computationally expensive to compute, as is the case for docking scores.

We showed our generative model to be more computationally efficient than existing methods. We also show that `OptiMol` is successful at biasing the prior distribution towards high predicted affinity compounds while remaining tractable. We moreover show that the tool can optimize a multi objective score beyond affinity.

This chapter was made in collaboration with Jacques Boitreaud, Carlos Oliver and Jerome Waldispuhl and was published in the Journal of Chemical Information and Modeling in 2020 [Boitreaud et al., 2020]. I was co-first author in this paper. We got the idea and the formalization of the project together. Jacques implemented most parts about the generative model while I did the optimization parts. We conducted the experiments and wrote together.

Chapter 2

Reverse-Complement Equivariant Networks for DNA Sequences

This chapter was made in collaboration with Jean-Philippe Vert and was published in NeurIPS 2021 [Mallet and Vert, 2021].

Contents

Abstract

As DNA sequencing technologies keep improving in scale and cost, there is a growing need to develop machine learning models to analyze DNA sequences, e.g., to decipher regulatory signals from DNA fragments bound by a particular protein of interest. As a double helix made of two complementary strands, a DNA fragment can be sequenced as two equivalent, so-called *Reverse Complement* (RC) sequences of nucleotides. To take into account this inherent symmetry of the data in machine learning models can facilitate learning. In this sense, several authors have recently proposed particular RC-equivariant convolutional neural networks (CNNs). However, it remains unknown whether other RC-equivariant architectures exist, which could potentially increase the set of basic models adapted to DNA sequences for practitioners. Here, we close this gap by characterizing the set of all linear RC-equivariant layers, and show in particular that new architectures exist beyond the ones already explored. We further discuss RC-equivariant pointwise nonlinearities adapted to different architectures, as well as RC-equivariant embeddings of k-mers as an alternative to one-hot encoding of nucleotides. We show experimentally that the new architectures can outperform existing ones.

Résumé

Avec les avancées en termes de coût et d'efficacité des méthodes de séquençage ADN, il y a un besoin croissant de méthodes dédiées à l'analyse de ces données, par exemple pour déchiffrer les signaux contenus dans une séquence ADN qui se lie à une protéine d'intérêt. Du fait de sa structure en double hélice, faite de deux brins complémentaires, un fragment d'ADN donne lieu à deux résultats de séquençage aux contenus équivalents, appelés des *Compléments Inverses* (CI). La prise en compte de cette symétrie par les modèles d'apprentissage automatique favorise leur entraînement. Plusieurs papiers ont récemment proposés certains réseaux convolutionnels CI-équivariants. Il n'existe pas pour l'instant de résultats sur l'existence d'autres réseaux respectant cette symétrie. Nous dérivons un tel résultat en caractérisant toutes les couches linéaires CI-équivariantes, qui englobent mais dépassent les couches existantes. Nous caractérisons aussi les non-linéarités ponctuelles CI-équivariantes et des représentations par k-mers CI-équivariantes pour les séquences de nucléotides. Enfin, nous montrons expérimentalement que ces architectures peuvent donner de meilleurs résultats que les architectures existantes.

2.1 Introduction

Incorporating prior knowledge about the structure of data in the architecture of neural networks is a promising approach to design expressive models with good generalization properties. In particular, exploiting natural symmetries in the data can lead to models with fewer parameters to estimate than agnostic approaches. This is especially beneficial when the amount of available data is limited. A famous example of such an architecture is the convolutional neural network (CNN) for 1D sequences or 2D images, which is well adapted to problems which are invariant to translations in the data, while exploiting multiscale and local information in the signals. Motivated by the success of CNNs, there has been a fast-growing body of research in recent years to build the theoretical underpinnings and design architectures and efficient algorithms to systematically exploit symmetries and structures in the data [Bronstein et al., 2021].

A central idea that has emerged is to formalize the symmetries in data by a particular *group action* (e.g., the group of translations or rotations on images), and to create multilayer neural networks which, by design, "behave well" under the action of the group. This is captured formally by the concept of *equivariance*, which states that each equivariant layer should be designed to be subject to the group action (e.g., we should be able to "translate" or "rotate" the signal in each layer), and that when an input data is transformed by a particular group element, then its representation in an equivariant layer should also be transformed according to the same group element. While it is easy to see that convolutional layers in CNNs are equivariant to translations, Cohen and Welling [2016] formalized the concept of group equivariance CNN (G-CNN) for more general groups and showed in particular how to design convolutional layers equivariant not only to translations but also to reflections and to a discrete set of rotations. Following this seminal work, the theoretical foundations of group equivariant neural networks were then expanded, going beyond regular representations [Cohen and Welling, 2017], for more groups [Anderson et al., 2019; Hoogeboom et al., 2018; Thomas et al., 2018; Weiler et al., 2018a], in less regular spaces [Cohen et al., 2019a, 2018] or with more general results on their generality and universality [Cohen et al., 2019b; Dym and Maron, 2020; Esteves, 2020; Kondor and Trivedi, 2018]. The main applications were developed with the groups of rotations in 2D and 3D, mostly to computer vision problems, but also in biology with histopathology [Graham et al., 2020; Lafarge et al., 2021], medicine [Winkels and Cohen, 2019] and quantum chemistry [Schütt et al., 2021].

In this paper, we explore and study the potential benefits of equivariant architectures for an important class of data, namely deoxyribonucleic acid (DNA) sequences. DNA is the major form of genetic material in most organisms, from bacteria to mammals, which encodes in particular all proteins that a cell can produce and which is transmitted from generation to generation. The study of DNA in humans and various organisms has led to tremendous progress in biology and medicine since the 1970s, when the first DNA sequencing technologies were invented, and the collapsing cost of sequencing in the last twenty years has accelerated the production of DNA sequences: there are for example about 2.8 billion sequences for a total length of $\sim 10^{13}$ nucleotides publicly available at the European Nucleotide Archive (ENA[1]). Unsurprisingly in such a data-rich field, machine learning-based approaches are increasingly used to analyze DNA sequences, e.g., in metagenomics to automatically predict the species present in an environment from randomly sequenced DNA fragments [Liang et al., 2020; Menegaux and Vert, 2019; Tampuu et al., 2019; Vervier et al., 2016] and to detect the presence of viral DNA in human samples

[1] As of May, 2021: https://www.ebi.ac.uk/ena

[Tampuu et al., 2019], in functional genomics to predict the presence of protein binding sites or other regulatory elements in DNA sequences of interest [Ghandi et al., 2014; Lee et al., 2015a; Oubounyt et al., 2019; Stormo, 2000; Zeng et al., 2016; Zhang et al., 2019], to predict epigenetic modifications [Levy et al., 2020], or to predict the effect of variations in the DNA sequence on a phenotype of interest [Alipanahi et al., 2015; Zhou and Troyanskaya, 2015].

Due to the sequential nature of DNA and the translation-equivariant nature of the questions addressed, many of these works are based on 1D CNN architectures, although recently transformer-based language models have also shown promising results on various tasks [Clauwaert and Waegeman, 2021; Ji et al., 2021; Zaheer et al., 2020]. However, besides translation, DNA has an additional fundamental symmetry that has been largely ignored so far: the so-called reverse complement (RC) symmetry, due to the fact that DNA is made of two strands oriented in opposite directions and encoding complementary nucleotides. In other words, a given DNA segment can be sequenced as two RC DNA sequences, depending on which strand is sequenced; any predictive model for, e.g., DNA sequence classification should therefore be RC-invariant, which calls for RC-equivariant architectures. While strategies based on data augmentation and prediction averaging has been commonly used to handle the need for RC invariance [Alipanahi et al., 2015; Quang and Xie, 2019], one translation- and RC-equivariant CNN architecture has been proposed and led to promising results [Brown and Lunter, 2019; Onimaru et al., 2020; Shrikumar et al., 2017]. However, it remains unclear whether that architecture is the only one that can encode translation- and RC-equivariance, or if alternative models exist to complement the toolbox of users wishing to develop deep learning models for DNA sequences.

Using the general theory of equivariant representations, in particular steerable CNNs [Cohen and Welling, 2017], we answer that question by characterizing the set of all linear translation- and RC-equivariant layers. We show in particular that new architectures exist beyond the ones already explored by [Brown and Lunter, 2019; Onimaru et al., 2020; Shrikumar et al., 2017], which in the language of equivariant CNNs only make use of the regular representation [Cohen and Welling, 2016] while more general representations lead to different layers. We further discuss RC-equivariant pointwise nonlinearities adapted to different representations, as well as RC-equivariant embeddings of k-mers as an alternative to one-hot encoding of nucleotides. We test the new architecture on several protein binding prediction problems, and show experimentally that the new models can outperform existing ones, confirming the potential benefit of exploring the full set of RC-equivariant layers when manipulating DNA sequences with deep neural networks.

2.2 Methods

2.2.1 Group action of translation and reverse complementarity on DNA sequence

DNA is a long polymer made of two intertwined strands, forming the well-known double-helical structure. Each strand is a non-symmetric polymer that can be described as an oriented chain of four possible monomers called nucleotides and denoted respectively $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. The two strands are oriented in opposite directions, and their nucleotides face each other to form hydrogen bonds. They interact at each position in a deterministic way because only two nucleotides pairings can happen: `(A,T)` and `(G,C)`. Thus, given a nucleotide sequence on one strand, we can deduce the so-called RC sequence of its corresponding strand by complementing each nucleotide and

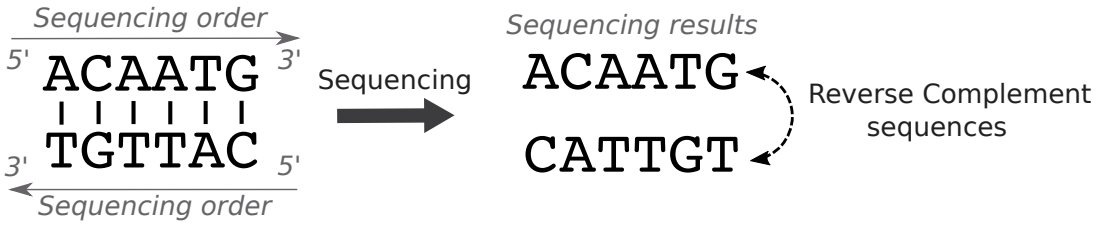


Figure 2.1 – Illustration of the reverse-complement symmetry. Both DNA strands get sequenced in opposite directions resulting in redundant information.

reversing the order (**Figure 2.1**). When a double-stranded DNA fragment is sequenced, the two strands are first separated and, typically, only one of them is randomly selected and is decrypted by the machine. This implies that any given DNA fragment can be equivalently described by two RC sequences of nucleotides. Moreover, several genomic learning tasks amount to a sequence annotation that does not depend on the strand. For example, a protein can bind a double-stranded DNA fragment, and both strands of the bound part can get sequenced. This motivates the search for equivariance to this RC-action for the prediction functions. Moreover, the sequencing often results in long sequences where the relevant parts of the sequence do not correlate with their position. The task of prediction over genomic sequences is thus largely translation equivariant, which explains why the community settled on the use of CNNs to train and predict on arbitrary length segments.

To formalize mathematically the translation and RC operations on DNA sequences, we first encode the raw genetic sequence as a signal function in $F_0 = \left\{f : \mathbb{Z} \to \{0,1\}^4\right\}$, as the one hot encoding of the nucleotide content for each integer position. Because of the finite length of this polymer, we assume that beyond a compact support this function takes a constant value of zero. The group $(\mathbb{Z}, +)$ of translations acts naturally on this encoding by $T_u(f)(x) = f(x-u)$, for a translation $u \in \mathbb{Z}$, and the RC operations amounts to the following : $RC(f)(x) = \sigma(-1)[f(-x)]$, where $\sigma(-1)$ is the 4×4 permutation matrix that exchanges complementary bases `A`/`T` and `C`/`G` (while we denote by $\sigma(1)$ the 4×4 identity matrix). We notice that RC is a linear operation on F_0 that satisfies $RC^2 = I$, and thus that the RC operation is a group representation on F_0 for the group $\mathbb{Z}_2 = \{1, -1\}$ endowed with multiplication.

To jointly consider translations and RC actions, we naturally consider the semi-direct product group $G = \mathbb{Z} \rtimes \mathbb{Z}_2$. Elements $g \in G$ can be written as $g = ts$ with $t \in \mathbb{Z}, s \in \mathbb{Z}_2$ and the group G acts on F_0 by the action π_0 defined by:

$$\forall ts \in G\,, \quad \forall (f,x) \in F_0 \times \mathbb{Z}\,, \quad \big(\pi_0(ts)f\big)(x) = \sigma(s)[f(s(x-t))]\,.$$

In other words, π_0 is the representation of G on F_0 induced by the representation σ of RC on $\mathbb{R}^4$ [Cohen and Welling, 2017].

2.2.2 Features spaces of equivariant layers

Let us now describe the structure of intermediate layers of a neural network equivariant to translations and RC. Following the theory of steerable CNNs [Cohen and Welling, 2017], we consider successive representations of the input DNA sequence in the following way:

Definition 1. *Given ρ a representation of $\mathbb{Z}_2$ on $\mathbb{R}^D$ for some $D \in \mathbb{N}^*$, a* ρ-feature space *is the set of signals $F = \{f : \mathbb{Z} \to \mathbb{R}^D\}$ endowed with the G group action π, known as the representation induced by ρ :*

$$\forall ts \in G\,, \quad \forall (f, x) \in F \times \mathbb{Z}\,, \quad \big(\pi(ts)f\big)(x) = \rho(s)[f(s(x-t))]\,. \tag{2.1}$$

With this definition, we see in particular that the one-hot encoding input layer maps the input DNA sequence to a σ-feature space, and that the dimension (i.e., number of channels in the language of deep learning) and group action of ρ-feature space are fully characterized by the representation ρ. Interestingly, the theory of linear group representations allows us to characterize more precisely *all* such representations:

Theorem 1. *For any representation ρ of $\mathbb{Z}_2$ on $\mathbb{R}^D$, there exist $a, b \in \mathbb{N}$ such that $a + b = D$ and an invertible matrix $P \in GL(\mathbb{R}^D)$ such that*

$$\forall s \in \mathbb{Z}_2\,, \quad \rho(s) = P\ Diag(I_a, sI_b)P^{-1}\,.$$

In other words, combining Definition 1 and Theorem 1, we see that any ρ-feature space that we will use to build translation- and RC-equivariant layers is fully characterized by a triplet (P, a, b), which we call its *type*, and which characterizes both its dimension $D = a + b$ and the action of the group G by (2.1). By slight abuse of language, we also refer to (P, a, b) as the type of ρ.

Theorem 1 is a standard result of group theory, which explicits the decomposition of any representation ρ in terms of so-called irreducible representation, or *irreps*. In the case of $\mathbb{Z}_2$, there are exactly two irreps which act on $\mathbb{R}$, namely, $\rho_1(s) = 1$ and $\rho_{-1}(s) = s$. If ρ has type (P, a, b), then it means that it can be decomposed as a times $\rho_1(s)$ and b times $\rho_{-1}(s)$. In the particular case where P is the identity matrix, i.e., when we consider a type (I, a, b), then $\rho(s)$ is a diagonal matrix for any $s \in \mathbb{Z}_2$, and each channel of F is acted upon by a single irrep. In that case, we will call the channels of type "1" (resp. "-1") if they are acted upon by ρ_1 (resp. ρ_{-1}), and we will say that F is an "irrep feature space".

Now, let us introduce another special case. Since $\mathbb{Z}_2$ is finite of cardinality 2, let us consider the *regular representation* ρ_{reg} of $\mathbb{Z}_2$ on $\mathbb{R}^2$ defined by:

$$\rho_{reg}(1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\,, \quad \rho_{reg}(-1) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\,.$$

One can easily check that ρ_{reg} is of type $(P_{reg}, 1, 1)$, where $P_{reg} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. It corresponds to a ρ-feature space with two channels, where the RC operations flips the two channels (and of course the sequence coordinates).

Let us now consider feature spaces of interest. In the input layer, nucleotides are one-hot encoded in a certain order, let us say $(\mathtt{A}, \mathtt{T}, \mathtt{G}, \mathtt{C})$. As stated above, this input space is acted upon by σ, a 2−cycle that swaps bases $\mathtt{A}/\mathtt{T}$ and $\mathtt{C}/\mathtt{G}$. We see that we can rewrite $\sigma = (\rho_{reg} \oplus \rho_{reg}) := (\rho_{reg}^{\oplus 2})$, where $\oplus$ is the bloc-diagonal operation. Because ρ_{reg} is of type $(P_{reg}, 1, 1)$, we can diagonalize σ with $(P_{reg}^{\oplus 2})$ and the diagonal would be alternated +1 and -1 values. Thus, there exists a permutation Π such that σ is of type $(P, 2, 2)$, with $P = \Pi(P_{reg}^{\oplus 2})\Pi^{-1}$. These concepts are illustrated in Supplementary Section B.1

Interestingly, all RC-equivariant layers proposed so far in [Brown and Lunter, 2019; Onimaru et al., 2020; Shrikumar et al., 2017] follow a similar pattern: the channels go by pair, and the RC action amounts to flipping the channel values within a pair and reversing the sequence coordinates. In our formalism, this corresponds to channels of type (P, a, a), where $a \in \mathbb{N}^*$ is the number of pairs of channels, and where up to a permutation of channels the matrix P satisfies $P = \Pi(P_{reg}^{\oplus a})\Pi^{-1}$. Following [Shrikumar et al., 2017], we will refer to these layers as *Reverse Complement Parameter Sharing* (RCPS) layers below.

This highlights the fact that translation- and RC-equivariant layers explored so far are equivariant according to Definition 1, but that there exists potentially many other equivariant layers, obtained in particular by allowing ρ-feature spaces of types (P, a, b) where $a \neq b$, on the one hand, and where P is not a direct sum of P_{reg}, on the other hand. We investigate such variants below.

2.2.3 Equivariant linear layers

While Definition 1 characterizes ρ-feature space in terms of structure and group action, an equivariant multilayer neural network is built by stacking ρ-feature spaces on top of each other and connecting them with equivariant layers. Cohen et al. [2019b, Theorem 2] gives us a general result about such equivariant mappings. Here, we apply this result to our specific data and group, and characterize the class of equivariant linear layers, i.e., the linear functions $\phi : F_n \to F_{n+1}$ that satisfy $\pi_{n+1}\phi = \phi\pi_n$, where π_n and π_{n+1} are respectively the group action on F_n and F_{n+1}.

Theorem 2. *Given two representations ρ_n and ρ_{n+1} of $\mathbb{Z}_2$, of respective types (P_n, a_n, b_n) and $(P_{n+1}, a_{n+1}, b_{n+1})$ with $a_n + b_n = D_n$ and $a_{n+1} + b_{n+1} = D_{n+1}$, and respective ρ_n- and ρ_{n+1}-feature spaces F_n and F_{n+1}, a linear map $\phi : F_n \to F_{n+1}$ is equivariant if and only if it can be written as a convolution:*

$$\forall (f, x) \in F_n \times \mathbb{Z}\,, \quad \phi(f)(x) = \sum_{y \in \mathbb{Z}} \kappa(y - x) f(y)\,, \tag{2.2}$$

where the kernel $\kappa : \mathbb{Z} \to \mathbb{R}^{D_{n+1} \times D_n}$ satisfies:

$$\forall x \in \mathbb{Z}\,, \quad \kappa(-x) = \rho_{n+1}(-1)\kappa(x)\rho_n(-1)\,, \tag{2.3}$$

or equivalently:

$$\forall x \in \mathbb{Z}\,, \quad \kappa(x) = P_{n+1} \begin{pmatrix} \alpha(x) & \beta(x) \\ \gamma(x) & \delta(x) \end{pmatrix} P_n^{-1}\,, \tag{2.4}$$

where $\alpha : \mathbb{Z} \to \mathbb{R}^{a_{n+1} \times a_n}$ and $\delta : \mathbb{Z} \to \mathbb{R}^{b_{n+1} \times b_n}$ are even, while $\beta : \mathbb{Z} \to \mathbb{R}^{a_{n+1} \times b_n}$ and $\gamma : \mathbb{Z} \to \mathbb{R}^{b_{n+1} \times a_n}$ are odd functions.

As stated in Cohen et al. [2019b], "Convolution is all you need" to define linear layers which are equivariant to our group. In addition, Theorem 2 characterizes all the convolution kernels that ensure equivariance through the two equivalent constraints (2.3) and (2.4).

To illustrate this result, let us consider two RCPS feature spaces F_n and F_{n+1} of respective types $(\Pi_n(P_{reg}^{\oplus a_n})\Pi_n^{-1}, a_n, a_n)$ and $(\Pi_{n+1}(P_{reg}^{\oplus a_{n+1}})\Pi_{n+1}^{-1}, a_{n+1}, a_{n+1})$. Then, the channels in F_n and F_{n+1} go by pair, and if we consider a slice $\tilde{\kappa} : \mathbb{Z} \to \mathbb{R}^{2\times 2}$ of the convolution kernel κ

describing how a pair of channels in F_n maps to a pair of channels in F_{n+1}, (2.3) gives the constraint:

$$\tilde{\kappa}(-x) := \begin{pmatrix} \tilde{\kappa}_{11}(-x) & \tilde{\kappa}_{12}(-x) \\ \tilde{\kappa}_{21}(-x) & \tilde{\kappa}_{22}(-x) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tilde{\kappa}(x) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \tilde{\kappa}_{22}(x) & \tilde{\kappa}_{21}(x) \\ \tilde{\kappa}_{12}(x) & \tilde{\kappa}_{11}(x) \end{pmatrix}.$$

We recover exactly the constraints of the RCPS filters first proposed by [Shrikumar et al., 2017], proving as a consequence of Theorem 2 that RCPS convolution filters describe exactly *all* equivariant linear mappings between RCPS feature spaces.

Moreover, if we now consider any two feature spaces F_n and F_{n+1} of respective types (P_n, a_n, b_n) and $(P_{n+1}, a_{n+1}, b_{n+1})$, then Equation (2.4) tells us that up to multiplications by matrices P_{n+1} and P_n^{-1}, the kernel is expressed in terms of even and odd functions, which can be trivially implemented with parameter sharing. For example, to represent the even function α, one just needs to parameterize the values of $\alpha(x)$ for $x \geq 0$, and complete the negative values by parameter sharing $\alpha(-x) = \alpha(x)$. Hence, the parameter sharing idea used in RCPS [Shrikumar et al., 2017] extends to any equivariant linear map.

Instead of using (2.4) to parameterize equivariant convolution kernels, one may also directly write the constraints (2.3) for specific representations, and potentially save the need of multiplication by P_{n+1} and P_n^{-1} in (2.4). This is for example the case in RCPS layers [Shrikumar et al., 2017], and more generally for channels acted upon by the regular representation; for the sake of completeness, we derive in Appendix B.4 the constraints to go from and to the regular representation or the irreps, and use them in our implementation.

2.2.4 Equivariant nonlinear layers

Besides equivariant linear layers, a crucial component needed for multilayer neural networks is the possibility to have equivariant nonlinear layers, such as nonlinear pointwise activation functions or batch normalization [Ioffe and Szegedy, 2015]. In this section, we discuss particular nonlinearities that are adapted to various equivariant layers.

Pointwise activations. Let us begin with pointwise transformations, that encompass most activation functions used in deep learning. Pointwise transformations are formally defined as follows: given a function $\theta : \mathbb{R} \to \mathbb{R}$ and a vector space $V = \mathbb{R}^A$ for some index set A, the pointwise extension of θ to V is the mapping $\bar{\theta}_V : V \to V$ defined by $\bar{\theta}_V(f)(a) = \theta\left(f(a)\right)$, for any $(f, a) \in V \times A$. For a D-dimensional representation ρ of $\mathbb{Z}_2$ and a ρ-feature space F with G-group action π, we say that a pointwise extension $\bar{\theta}_F : F \to F$ is equivariant if it commutes with π, i.e., $\pi\bar{\theta}_F = \bar{\theta}_F\pi$. By definition of the group action (2.1), this is equivalent to saying that the pointwise extension $\bar{\theta}_{\mathbb{R}^D}$ of θ to $\mathbb{R}^D$ commutes with ρ. The following theorem gives an exhaustive characterization of a large class of equivariant pointwise extensions for any ρ-feature space:

Theorem 3. *Let ρ be a representation of $\mathbb{Z}_2$ and $\theta : \mathbb{R} \to \mathbb{R}$ be a continuous function with left and right derivatives at 0. Let F be a ρ-layer and $\bar{\theta}_F : F \to F$ be the point-wise extension of θ on this layer. Then $\bar{\theta}_F$ is equivariant if and only if at least one of the following cases holds:*

1. *θ is a linear function.*
2. *θ is an affine function, and $\rho(-1)\mathbf{1} = \mathbf{1}$.*

3. *θ is not an affine function, and there exists a permutation matrix Π, integers $a, b, c, d \in \mathbb{N}$, and scalars $(\lambda_1, \dots, \lambda_a) \in (\mathbb{R}_+^*)^a$, such that ρ decomposes as*

$$\Pi^{-1}\rho(-1)\Pi = \bigoplus_{i=1}^{a} \begin{pmatrix} 0 & \lambda_i \\ \lambda_i^{-1} & 0 \end{pmatrix} \oplus \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}^{\oplus b} \oplus (1)^{\oplus c} \oplus (-1)^{\oplus d}. \tag{2.5}$$

In that case,

- *Either $b = d = 0$ and $\forall i, \lambda_i = 1$ and θ is any function,*
- *Or $b = d = 0$ and $\exists i, \lambda_i \neq 1$ and θ is a leaky ReLu function.*[2]
- *Or $b + d > 0$ and $\forall i, \lambda_i = 1$ and θ is an odd function,*

The first case in Theorem 3 is of little interest, since pointwise linear functions are always equivariant to linear group actions. The second case essentially says that adding a constant to a pointwise linear function is only equivariant for representations ρ such that the sum of all rows of $\rho(-1)$ is equal to 1. This holds for example for the regular representation and the RCPS layers, but not for an irrep feature space of type (I, a, b) with $b > 0$, since in that case, some rows have a single "-1" entry. The most interesting case is the third one, since it describes what pointwise nonlinearities one can apply. The condition (2.5) on the decomposition of ρ essentially excludes all representations that have more than one nonzero value in at least one row of $\rho(-1)$. Among valid ρ's that decompose as (2.5), we see that the regular representation (corresponding to the first block in (2.5) with $\lambda_i = 1$)), used in RCPS, stands out as the only that allows *any* nonlinearity, besides of course invariant channels of type "+1" (third block in (2.5)). Replacing a "1" in the regular representation by a scalar $\lambda_i \neq 1$ (in the first block of (2.5), with $b = d = 0$) creates a valid representation ρ, however only leaky ReLu pointwise nonlinearities can be applied in that case. Another case of practical interest is the irrep feature space of type (I, c, d) for some $c > 0$ and $d > 0$. By Theorem 3, only odd nonlinearities are allowed in that case, such as the hyperbolic tangent function. Finally, one should keep in mind that other representations, which do not satisfy the conditions listed in Theorem 3, do not allow any equivariant nonlinear pointwise transform; this is for example the case of $\rho(-1) = \begin{pmatrix} 0 & -1/2 \\ -2 & 0 \end{pmatrix}$, which is a valid representation of $\mathbb{Z}_2$ but does neither meet the condition to accept affine activations (case 2), nor to accept nonlinear activations (case 3) because $\rho(-1)$ does not decompose according to (2.5).

Other activation functions Besides pointwise transformations from a ρ-feature space to itself characterized in Theorem 3, the set of nonlinear equivariant layers is tremendous and the design choices are endless. A first extension is to keep pointwise activation, but to allow different nonlinearities on different channels, e.g., by using any function on the "+1" channels and an odd function on the "-1" channels of an irrep feature space. Another relaxation is to use different input and output representations. While odd functions will not affect the field type, even functions will turn a field of type "-1" into a "+1" type. It is well known that any function decomposes into a sum of an odd and even function. Therefore, given ρ, a representation decomposed as in

[2] A leaky ReLu function is $\theta(x) = \alpha_{sign(x)}x$ for some $(\alpha_+, \alpha_-) \in \mathbb{R}^2$.

(2.5), any pointwise non-linearity can be used in a ρ-feature space by first decomposing it into its odd and even components and applying each component separately for the second and fourth blocks.

Other possibilities exist and include creating new representations by tensorization, which amounts to taking pointwise products between different channels [Dym and Maron, 2020; Kondor, 2018; Thomas et al., 2018]. or using non point-wise activation layers that act on several coupled dimensions, such as the ones used in [Thomas et al., 2018]. For instance, we could apply the max function to paired channels. These possibilities are discussed in [Weiler and Cesa, 2019]

Batch normalization An equivariant batch normalization was introduced by [Shrikumar et al., 2017]. It considers a feature map and its reverse complement as two instances, which is easy to do because the reverse complement feature map is already computed when using regular representation. We propose another batch normalization for irrep feature spaces that also gives the result we would have had if the batch contained all the reverse complement of its sequences. For the "+1" dimensions, it amounts to scaling as we would have the same values twice. For the "-1" dimensions, we enforce a zero mean and compute a variance estimate based on this constraint.

K-mers. Instead of the standard one-hot encoding of individual nucleotides as input layer, we propose to one-hot encode k-mers for $k \geq 1$, i.e., overlapping blocks of k consecutive nucleotides. This technique is known to improve performance in several tasks [Liang, 2012; Menegaux and Vert, 2019]. In order to implement it into an equivariant network, we need to know how the group acts on the k-mers space, made of 4^k elements. The simplest idea is to pair the index of the channels of two RC k-mers. Because some k-mers are their own reverse complement, the canonical way to do so is to have a representation that is a blend of "+1" irrep and regular representation. An alternative is to make the regular representation act on the k-mers instead by redundantly encoding these k-mers into paired dimensions. This is the strategy we follow in our implementation, to be more coherent with the usual input group action.

2.3 Experiments

We assess the performance of various equivariant architectures on a set of three binary prediction and four sequence prediction problems used by Zhou et al. [2020] to assess the performance of RCPS networks. The binary classification problems aim to predict if a DNA sequence binds to three transcription factors (TFs), based on genome-wide binarized TF-ChIP-seq data for Max, Ctcf and Spi1 in the GM12878 lymphoblastoid cell-line [Shrikumar et al., 2017]. The sequence prediction problems aim to predict TF binding at the base-pair resolution, using genome-wide ChIP-nexus profiles of four TFs- Oct4, Sox2, Nanog and Klf4 in mouse embryonic stem cells. For a more detailed explanation of the experimental setup, please refer to Zhou et al. [2020]. We report "significant" differences in performance below when the P-value of a Wilcoxon signed rank test is smaller than 0.05.

Models. We build over the work of Zhou et al. [2020] for both the binary and the sequence prediction problems. They benchmarked an equivariant RCPS architecture and a corresponding non-equivariant model, with the same number of filters and trained with data augmentation,

which we respectively refer to as "RCPS" and "Standard" models below. The data augmentation scheme for the "Standard" model consists in adding to the training set the reverse complement sequences of all training sequences, which is a natural procedure to let the model "learn" the equivariance without encoding it explicitly in the architecture of the network. We checked empirically that data augmentation significantly improves the performance of non-equivariant models (Appendix B.7). In addition, we extend the RCPS architecture with one-hot encoding of k-mers as input layers, which we refer to as "Regular" below. Finally, we add to the comparison a new equivariant network where each RCPS layer is replaced by an (I, a, b) layer with the same number of filters, which we call "Irrep" below. We also use k-mers and vary the ratio $a/(a+b)$ in this model. We combine the regular and "+1" dimensions with *ReLu* activations and the "-1" dimensions with a *tanh* activation.

Influence of hyperparameters in equivariant models To assess the impact of different hyperparameters in the family of equivariant models we propose (k-mer length for Irrep and Regular, $a/(a+b)$ ratio for Irrep), we train equivariant models with different combinations of hyperparameters on the training set and assess their performance on the validation set, repeating the process ten times with different random seeds. We assess the performance of each run in terms of Area under the Receiver Operator Characteristic (AuROC), and show in Figure 2.2 the average performance reached by all runs with a given ratio $a/(a+b) \in \{0, 1/4, 1/2, 3/4, 1\}$ (left) and with a given $k \in \{1, 2, 3, 4\}$ (right). We see a clear asymmetry in the performance as a function of $a/(a+b)$, with poor performance when $a = 0$ and optimal performance for $a = 0.75$, significantly better than all other ratios tested. This confirms that exploring different irreps may be valuable. As for the k-mer length, setting $k = 3$ gives the best performance and significantly outperforms all other values of k tested. This confirms that going beyond one-hot encoding of nucleotides in equivariant architectures can be beneficial.

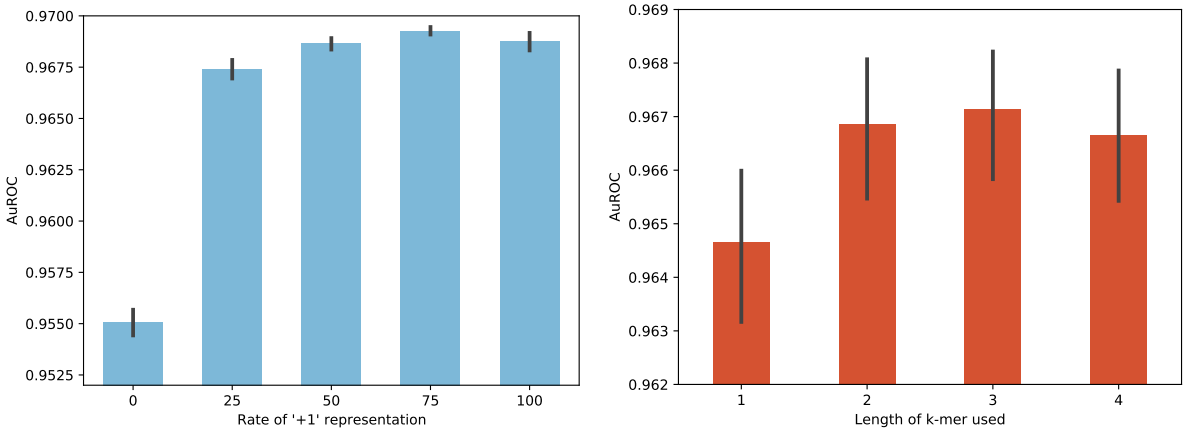


Figure 2.2 – Average AuROC performance across four TFs and 10 random seeds for the Irrep model as a function of $a/(a+b)$ (*left*, also averaged over k values) and for the Irrep and Regular models as a function of k (*right*, also averaged over $a/(a+b)$ values for Irrep).

Binary task. We then compare the test set performance of three different models for the binary classification task: 1) Standard, 2) RCPS, and 3) the best Irrep or Regular equivariant model,

where hyperparameters are selected based on the AuROC on the validation set, which we denote as "Best Equivariant". Figure 2.3 (left) shows the performance of each model on each TF task and overall. As already observed by [Shrikumar et al., 2017], the equivariant RCPS architecture has a strong lead over the Standard, non-equivariant model in spite of data augmentation. Interestingly, we see that Best Equivariant is significantly better than RCPS on all tasks, and that the performance gain from RCPS to Best equivariant is of the same order as the performance gain from Standard to RCPS. This demonstrates that the family of equivariant architectures we introduce in this paper can lead to significant improvement over existing architectures.

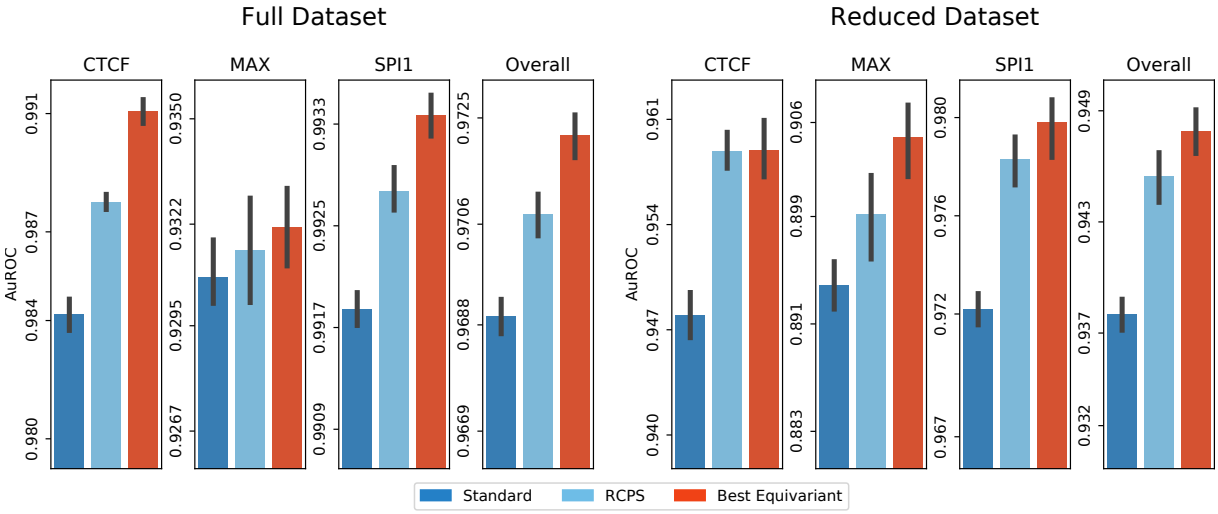


Figure 2.3 – AuROC performance of the three different models (Standard, RCPS and Best equivariant after hyperparameter selection on the validation set) on the three binary classification problems CTCF, MAX and SPI1, as well as their average. Error bars correspond to an estimate of the standard error on 10 repeats with different random seeds. The left plot is the performance on the full datasets, while the right plot shows the performance where models are trained on a subset of 1,000 sequences only (notice the differences of AuROC values on the vertical axis in both plots).

Reduced models. Since equivariant architectures are meant to be particularly beneficial in the low-data regime [Fuchs et al., 2020], we further assess the performance of the three models on the same binary classification problems but with only 1,000 sequences used to train the models, and show the results on Figure 2.3 (right). Overall, the performances are worse than in the full-data regime (Figure 2.3, left), which confirms that this is a regime where more data helps. We also see that the relative order of the three different methods remains overall the same, with Best Equivariant outperforming RCPS, which itself outperforms Standard. Interestingly, the gaps between the best and worse models widens in the low-data regime, showing that the prior is more useful in this setting. More precisely, there is a large gap of about 1% between Best Equivariant and Standard in the low data regime, compared to a gap of about 0.3% on the full dataset. We also investigated whether equivariant models converge faster to their solutions, but found no noticeable difference (Appendix B.8).

On post-hoc models. Zhou et al. [2020] introduced the so-called *post-hoc* model, another equivariant method obtained by averaging the predictions of a Standard model over a sequence and its reverse-complement, and showed that it is competitive with and often outperforms RCPS. The post-hoc model only requires training and storing one network, but aggregates two predictions for each sequence at inference time. Because of that, the good performance of post-hoc may be due in part to the aggregation step common to all ensemble models [Dietterich, 2000]. To decipher the respective contributions of the network architecture, on the one hand, and of the aggregation of predictions, on the other hand, we add to the comparison an ensemble of two Standard models trained with different random seeds (*Ensemble Standard*) and an ensemble of two equivariant Irrep models (*Ensemble Irrep*) and present the results in Figure 2.4. We see that Ensemble Irrep strongly outperforms Best Equivariant, and both post-hoc and Ensemble Standard widely outperform the Standard architecture. This confirms that ensembling equivariant or non-equivariant models through post-hoc of ensemble aggregation is always useful (at the cost of increased computational time). We see that Ensemble Standard is not significantly different from post-hoc Standard on CTCF and SPI1, but that post-hoc Standard is better on MAX, suggesting that most of the benefits of post-hoc Standard indeed comes from the ensembling effect. Regarding the impact of the architecture for a given budget of predictions, we saw earlier than Best equivariant significantly outperforms Standard when a single prediction per test sequence is allowed, and see now that Ensemble Irrep strongly outperforms both post-hoc and Ensemble Standard when two predictions are allowed, thus confirming the benefit of equivariant architectures in all settings. We also see that a single Best equivariant model outperforms post-hoc and Ensemble Standard, indicating that enforcing equivariance throughout the network is not only faster but also more accurate than averaging a non-equivariant model over group transformed inputs.

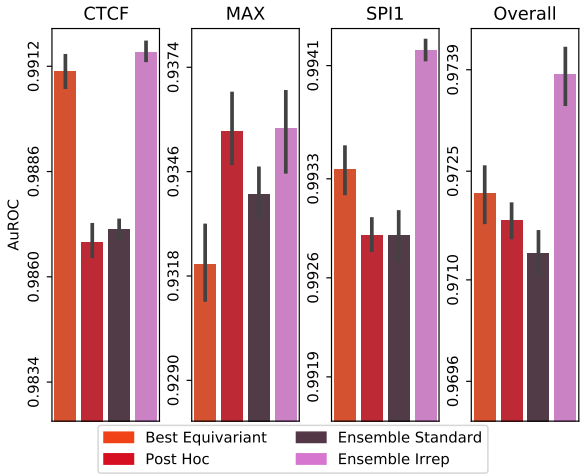


Figure 2.4 – AuROC performance on the three binary classification problems, for the Best Equivariant model, the post-hoc Standard model, and an ensemble of two Standard or Irrep models. Error bars correspond to an estimate of the standard error on 10 repeats with different random seeds.

Profile task. We now compare the performance of different models on the profile prediction tasks. To limit the carbon footprint of this study, and based on the influence of hyperparameters on the binary task (Figure 2), we only test two equivariant models in addition to Standard and RCPS: a Regular model with $k = 3$, and an Irrep model with $k = 3$ and $a/(a + b) = 75\%$. We also assess the performance of post-hoc Standard (the best model in [Zhou et al., 2020]), and an ensemble of two models of the best performing equivariant model. Figure 2.5 shows the performance of all models in terms of Spearman correlation between the target profile and the predicted ones, on the full dataset (left) or a reduced experiment with only 1,000 training sequences (right).

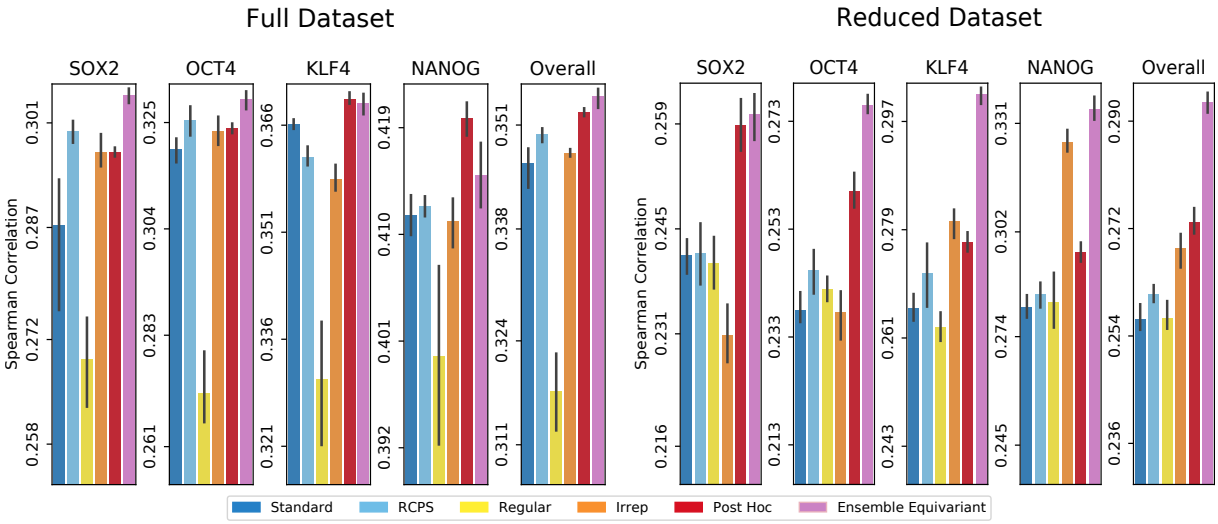


Figure 2.5 – Spearman Correlation between true and predicted profiles by different methods for four data sets.

First of all, we observe as before that in the low-data regime, the gap between standard and equivariant networks grows in favor of equivariant ones. We also observe, surprisingly, that Irrep, which outperformed RCPS on the binary task, now underperforms it. A possible explanation could be that since this task aims to annotate an individual nucleotide, encoding the nucleotide level information using k-mers makes the signal blurry and decreases performance. However, in the reduced setting, Irrep performs better again. These results indicate that for now the best model should be chosen empirically on a validation set. Finally, despite good performance of post-hoc Standard, the ensemble equivariant model once again performs better for the same computational cost at inference.

Experiment settings and computational cost. All experiments were run on a single GPU (either a GTX1080 or a RTX6000), with 20 CPU cores. The binary classification experiments were shorter to train. To limit our carbon footprint, we chose to run more experiments on this task, e.g., for hyperparameter tuning and to reduce the number of replicates for the profile task. The total runtimes of each of those tasks were approximately a week.

2.4 Conclusion

In this paper, we addressed the problem of including the RC symmetry prior in neural networks. Leveraging the framework of equivariant networks, in particular steerable CNNs, we deepened existing methods by unraveling the whole space of linear layers and pointwise nonlinearities that are translation and RC-equivariant. We also investigated the links between the linear representations and the non-linear layers of neural networks, exposing the special role of the regular representation in equivariant networks. Finally, we implemented new linear and nonlinear equivariant layers and made all these equivariant layers available in Keras [Chollet, 2015] and Pytorch [Paszke et al., 2019]. [3] We then explored empirically how this larger equivariant functional space behaves in terms of learning. Our best results improve the state of the art performance of equivariant networks, showing that new equivariant architectures can have practical benefits. In the future we plan to test more deeply the newly proposed architectures on prediction tasks involving double-stranded DNA, such as DNA-protein binding prediction, epigenetics or metagenomics. On the theoretical side, we characterized equivariant pointwise nonlinearities that preserve the layer type, but more general nonlinear transforms (e.g., not pointwise, or changing the layer type) remain to be fully characterized.

[3] code available at github.com/Vincentx15/Equi-RC

Chapter 3

Augmented base pairing networks encode RNA-small molecule binding preferences

This chapter was made in collaboration with Carlos Oliver, Roman Sarrazin Gendron, Vladimir Reinharz, William L Hamilton, Nicolas Moitessier, Jérôme Waldispühl and was published in Nucleic Acid Research in 2020 [Oliver et al., 2020].

Contents

Abstract

RNA-small molecule binding is a key regulatory mechanism which can stabilize 3D structures and activate molecular functions. The discovery of RNA-targeting compounds is thus a current topic of interest for novel therapies. Our work is a first attempt at bringing the scalability and generalization abilities of machine learning methods to the problem of RNA drug discovery, as well as a step towards understanding the interactions which drive binding specificity. Our tool, `RNAmigos`, builds and encodes a network representation of RNA structures to predict likely ligands for novel binding sites. We subject ligand predictions to virtual screening and show that we are able to place the true ligand in the 71st-73rd percentile in two decoy libraries, showing a significant improvement over several baselines, and a state of the art method. Furthermore, we observe that augmenting structural networks with non-canonical base pairing data is the only representation able to uncover a significant signal, suggesting that such interactions are a necessary source of binding specificity. We also find that pre-training with an auxiliary graph representation learning task significantly boosts performance of ligand prediction. This finding can serve as a general principle for RNA structure-function prediction when data is scarce. `RNAmigos` shows that RNA binding data contains structural patterns with potential for drug discovery, and provides methodological insights for possible applications to other structure-function learning tasks.
The source code and data is freely available at csb.cs.mcgill.ca/RNAmigos.

Résumé

La liaison d'une petite molécule à l'ARN est un mécanisme important qui peut stabiliser la structure tridimensionnelle d'un ARN et activer sa fonction. La découverte de composés ciblant l'ARN est donc un sujet central pour de nouvelles thérapies. Notre travail tente d'apporter la scalabilité et la généralisation des méthodes d'apprentissage automatique au problème de la découverte de médicaments à d'ARN, et représente une étape vers la compréhension des interactions qui déterminent cette liaison. Notre outil, `RNAmigos`, représente les structures d'ARN par un réseau pour prédire les ligands probables étant donné un nouveau site de liaison. Nous soumettons nos prédictions à un criblage virtuel et montrons que nous sommes capables de placer le vrai ligand dans le 72eme centile parmi des decoys, une amélioration significative relativement à l'état de l'art. De plus, nous observons que la représentation par un réseau augmenté d'interactions non canoniques entre paires de bases est la seule capable de découvrir un signal significatif, ce qui suggère que ces interactions sont une source nécessaire à la spécificité de liaison. Nous constatons également que le pré-entraînement avec une tâche d'apprentissage auxiliaire améliore considérablement les performances de prédiction des ligands. Cette découverte peut servir de principe général pour prédire la fonction d'un ARN à partir de sa structure, lorsque les données sont rares. `RNAmigos` montre que les données de liaison d'ARN contiennent des motifs structurels avec un potentiel pour la découverte de médicaments et fournit des indications méthodologiques pour d'autres tâches d'apprentissage de liens entre structure et fonction.
Le code source et les données sont disponibles à l'adresse :
csb.cs.mcgill.ca/RNAmigos.

3.1 Introduction

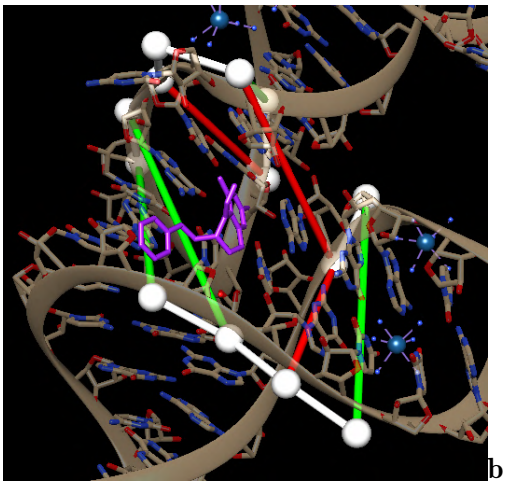
Recent studies have identified small organic molecules as important non-covalent regulators of RNA function [Donlic and Hargrove, 2018]. These discoveries contribute to a better understanding of pathways present in all organisms, but also pose RNA molecules as a large class of promising novel drug targets. For example, Ribocil, which has recently been uncovered through a phenotypic assay to target the FMN riboswitch, is currently undergoing clinical trials as a novel antibiotic [Howe et al., 2015]. Various other small molecule-activated RNA systems are also being proposed[Porter et al., 2017; Rauch et al., 2020; Wagner et al., 2018]. Notable among these is the application to CRISPR activation regulation [Kundert et al., 2019]. The list of possible therapies is likely to expand given the observations of KD Warner and co-workers that only a small fraction of the genome is translated into protein (1.5%) while the vast majority is transcribed into potentially druggable non-coding RNA (70%) [Warner et al., 2018].

3.1.1 RNA Structural Organization

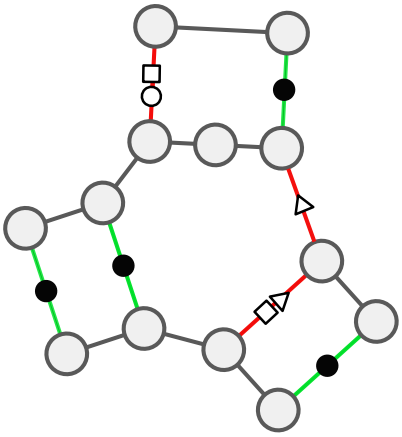
RNAs possess multiple levels of structural organization which together determine function, and by extension, ligand binding ability. At the simplest level, RNA is a string of monomers {`A,U,C,G`} linked by a chain of covalent bonds known as the backbone. This is commonly known as the primary structure of RNA. Non-covalent pairwise interactions between nucleotides (bases) in the chain give rise to the secondary and tertiary structure. Canonical pairs (i.e. `A-U`, `C-G`) give rise to the secondary structure. Notably, these pairs form loops and stacks (helices), assembling a stable scaffold for the full structure [Tinoco Jr and Bustamante, 1999]. The experimental determination of binding energies for these pairs [Freier et al., 1986] prompted a boom of algorithms for sequence to secondary structure prediction such as `RNAfold`, [Lorenz et al., 2011]. In seminal work, Leontis and Westhof identified 11 additional types of base pairing occurring in 3D structures [Leontis and Westhof, 1998, 2001], known as non-canonical base pairs. These interactions can occur between any pair of nucleotides and are defined by the relative orientations of three faces of the interacting bases in 3D. By considering all combinations of faces and a *cis* and *trans* orientation, we arrive at 12 possible base pairing geometries. Whereas canonical pairs form stable helices, non-canonical pairs are typically found in loops (i.e. regions without canonical pairs) and create more complex structural patterns [Leontis and Westhof, 2003; Petrov et al., 2013b]. These pairings fine-tune RNA function by defining structure at the 3D level [Leontis et al., 2006]. Interestingly, non-canonical pairs were also found to be enriched in ligand binding sites [David-Eden et al., 2010; Kligun and Mandel-Gutfreund, 2013], which corroborates with the observation that increased structural complexity is associated with binding specificity [Warner et al., 2018].

These observations, together with the well-studied role of secondary structure in RNA ligand binding [Thomas and Hergenrother, 2008], led us to hypothesize that studying RNA structures at the augmented base-pairing level (i.e. including non-canonical pairs) holds useful spatial and chemical information about ligand binding. However, studying RNA at this level of structure comes with major algorithmic challenges, such as the lack of binding energies and more complex interaction patterns. For these reasons, non-canonical interactions are typically modeled with statistical methods, and represented using more general data structures such as graphs [Cruz and Westhof, 2011]. In practice, this means that a graph using vertices to represent nucleotides and multi-relational edges to encode base-pairing interactions could offer a signature for RNA

ligand binding sites (See **Figure 3.1** for an example of a binding site and its associated base pairing network). We call this graphical representation of RNA sites annotated with canonical and non-canonical interactions an Augmented Base Pairing Network (ABPN) since we consider base pairs beyond the canonicals. Indeed, similar representations of RNA base pairing networks have been exploited in various tools [Cruz and Westhof, 2011; Petrov et al., 2013b; Reinharz et al., 2018a; Sarrazin-Gendron et al., 2019] for their ability to capture RNA-specific interactions in an interpretable manner. This paradigm distinguishes RNA from protein-ligand interactions where surface-cavity topologies tend to drive binding preferences [Luo et al., 2019], hence direct use of atomic coordinates can be more appropriate.



a Binding site atomic coordinates



b Graph encoding of binding site as an augmented base pairing network (ABPN)

Figure 3.1 – RNA structure representation of the THF riboswitch binding site (PDB: `4LVV`) as atomic coordinates using UCSF Chimera [Pettersen et al., 2004](left) and resulting augmented base pairing network (ABPN) (right). We superpose the ABPN in the 3D visualization. Nodes are drawn as white spheres, backbone connections are in white, and canonical and non-canonical base pairs are green and red tubes respectively. We color the edges simply to guide the eye to the corresponding base pairs but note that edge color has no special meaning to our graphs. We annotate the graph representation with the standard Leontis-Westhof nomenclature for pairing type symbols. In this case, the binding site has three canonical interactions denoted (●), and three non canonicals of types (□ ○, ▷, □ ▷).

3.1.2 Structure-based Drug Discovery and RNA Base Pairing Networks

The central aim of structure-based drug discovery is to identify compounds with high affinity to a given site or set of binding sites. A natural problem to address in this context is the prediction of binding affinity from a binding site-ligand pair. Machine learning models which solve this task can be used as alternatives to computationally expensive docking simulations to screen ligand databases for strong binders [Kitchen et al., 2004]. And in some cases have shown superior performance to methods built on explicit chemico-physical knowledge [Gómez-Bombarelli et al., 2018]. This setting is quite feasible in the protein domain as affinities and drug screens are abundant, hence various methods have been proposed [Jiménez et al., 2018]. Recently, some repositories of RNA small molecule data have been made public [Wang et al., 2018] however,

only a handful of binding affinities are known. Given a pose for a ligand inside a binding site, various scoring approaches have been proposed; DrugScoreRNA, and LigandRNA [Pfeffer and Gohlke, 2007; Philips et al., 2013], SPA-LN [Yan and Wang, 2017] which are built on *a priori* chemical knowledge and rely on accurate docking. While RNA docking methods which search for the optimal pose in a binding site are still showing limited success [Luo et al., 2019; Sun et al., 2017b].

The fundamental commonality in these tools is that they all require a binding site and ligand as input. Therefore, identifying a binder consists of docking and scoring all combinations of RNA and small molecules from a desired library in all putative poses, which can be prohibitive. In this work, we ask whether base pairing patterns, which creates a scaffold for the 3D structures and is easier to obtain, can be used to accelerate these searches. Or in other words, if a coarse-grained representation of RNA structures provides sufficient information about potential ligands.

To our knowledge, the closest contribution to this work is a template-based approach named Inforna [Disney et al., 2016]. Inforna searches through an input sequence and secondary structure for motifs that are similar to those found in a library of small-molecule structural binding motifs, and returns candidate ligands. Here, we propose two major innovations to such approaches. First, we take the first step towards *learning* a *generalizable* RNA binding landscape that can be used to infer compounds which are not explicitly present in compound libraries. Previous contributions have shown success in using protein 3D structures information to reach this objective in proteins [Aumentado-Armstrong, 2018; Gómez-Bombarelli et al., 2018; Mallet et al., 2019; Torng and Altman, 2019], but this is to our knowledge the first attempt to apply a similar strategy to RNAs. Next, because we also aim to leverage the specificity of the RNA structural organization, we investigate the impact of higher-order base pair interactions (beyond classical secondary structure), which has yet to be explored.

3.1.3 Contribution

`RNAmigos` brings together domain knowledge of RNA structure, currently available crystal structure data, and graph neural networks, to show that base pairing networks can be used to automatically predict ligands for RNA structures. Importantly, we propose the use of Augmented Base Pairing (ABPNs) networks, an enriched alphabet of base pairing interactions, and demonstrate that they are a necessary component for capturing binding signatures. Molecular fingerprints predicted by `RNAmigos` serve as ligand search tools across diverse ligand classes and show strong performance in two different ligand screens, as well as compared to a state of the art method, Inforna [Disney et al., 2016]. Additionally, we explore the use of an unsupervised graph representation learning scheme for boosting model performance in this low-data setting. The implications of our work are two-fold (i) we show for the first time that we can learn from non-canonical interaction data to make predictions about RNA function, and (ii) our ability to enrich for actives in compound libraries shows potential for `RNAmigos` as an upstream filtering step for more fine-grained drug discovery tools such as docking. The core implementation of `RNAmigos` is built in Pytorch [Paszke et al., 2019] and DGL [Wang et al., 2019] and is available as an open source Python 3.6 software package.

3.2 Materials and Methods

3.2.1 Model Overview

Our model (`RNAmigos`) seeks to identify possible ligands for a given coarse-grained representation of an RNA binding site (See **Figure 3.2**). More precisely, our input is an ABPNs modelling the RNA structure, from which we predict a molecular fingerprint for a potential ligand. This fingerprint can be used to search a library of compounds for active binders. We train `RNAmigos` on RNA-ligand pairs found in the RCSB PDB Data Bank (`www.rcsb.org`) [Berman et al., 2000], and use graph neural networks [Schlichtkrull et al., 2018] to learn the relationship between RNA structure and ligand binding preferences.

3.2.2 Dataset Preparation

We begin by collecting a set of RNA-small molecule complexes from the PDB Data Bank [Berman et al., 2000]. We download all crystal structures (90% identity threshold) which contain RNA and at least one ligand. This results in 2993 PDB structures. We omit ions such as magnesium (Mg+) from the set of valid ligands as they vastly outnumber organic ligands and likely require customized models. [Philips et al., 2012] We choose a maximum allowable distance between any ligand atom and any RNA atom of 10 Angstroms according to David-Eden et al. [2010] which statistically characterized ribosome antibiotic binding sites. The number of valid sites is further reduced when we remove binding sites with fewer than five RNA residues and remove binding sites containing a large proportion of protein residues, (See **Supp Fig. S1**). The final training set consists of 773 binding sites associated with 270 unique ligands.

Finally, we build an ABPN from the 3D structure of each binding site identified in the previous step. In the ABPN, each node corresponds to a residue in the binding site, and links/edges are formed between nodes if they form a backbone or base pair interaction. Node and edge annotations are taken from the BGSU RNA 3D Motif Atlas [Petrov et al., 2013b] database which maintains base pairing annotations of all PDBs with Leontis-Westhof and backbone interaction types computed by the software FR3D [Sarver et al., 2008]. In this manner, each ABPNs stores the nucleotide identity (`A`, `U`, `C`, `G`) of each of its residues as a node attribute, and each base pairing interaction corresponds to an edge with one of 13 different types (backbone + 12 base pairing geometries). The resulting graphs are on average 15.76 nodes in size. At this point, the ligand is removed from the structure so that the graph contains only RNA base-pairing information. While atomic coordinates are the current source of data, we highlight that a key feature of taking ABPNs as input is that we can eventually learn from many other sources of ABPN data which are easier to obtain than crystal structures. A promising example comes from recent developments in predicting base pairing networks from RNA sequences in high-throughput [Roll et al., 2016; Sarrazin-Gendron et al., 2019]. Our model would then be able to directly use such predictions once they are linked to a functional label (such as a ligand in this case). For full details on binding site extraction and graph construction, see Supplementary Material Section C.1.

3.2.3 Fingerprint Prediction

Given a binding site, our model predicts a set of chemical features which can be used to identify a ligand. This set of features is typically known as a *molecular fingerprint* [Cereto-Massagué et al., 2015]. Many approaches to compute fingerprints from chemical structures have been developed; all with the common aim of numerically encoding chemicals [Duvenaud et al., 2015; Glen et al., 2006; Rogers and Hahn, 2010]. Such encodings greatly facilitate searches for similar compounds in databases and screens. In this work, we use a common fingerprint implementation known as the MDL Molecular Access Keys (MACCS) fingerprint [Durant et al., 2002] which has the advantage of providing compact and interpretable entries. For a given chemical compound c, the MACCS fingerprint f_c is a 166 bit binary vector where each entry indicates the presence or absence of a chemical property. For the i^{th} chemical property, $f_c[i]$ is set to 1 if the chemical property is present and is 0 otherwise. We use the set of 166 predefined chemical properties from the Pybel [O'Boyle et al., 2008] implementation as a target vector for our model. We emphasize that the computation of the fingerprint depends only on the chemical composition of the ligand and not on the RNA binding site. The main objective of our model is to predict the set of chemical features (fingerprint) that is close to that of the co-crystallized ligand using only RNA base pairing networks. For convenience, we call the ligand co-crystallized with a given site its native ligand.

3.2.4 Model Architecture

Since a key feature of our ABPNs is the fact that we encode base pairing geometry as an edge category (or relation type) in a graph, we use a Relational Graph Convolutional Network (RGCN) [Schlichtkrull et al., 2018] as the core of the fingerprint prediction model (See **Figure 3.2**). An RGCN is a specialized neural network which acts directly on graphs, allowing us to naturally model ABPN structures. Here, a nucleotide is associated with a node and a base pair interaction represented by an edge. At a high level, the RGCN computes an encoding for each node, known as a *node embedding*. Formally, we denote a node embedding for node i as a d-dimensional real-valued vector, $z_i \in \mathbb{R}^d$. The notion of a node embedding can be understood in a similar manner to molecular fingerprints. Each entry of the vector numerically encodes a feature of the node and its neighborhood (i.e. the nucleotide).

We can choose the embeddings such that they maximize performance on some classification task (supervised; analogous to image classification), or to capture structural similarity relationships (unsupervised; analogous to dimensionality reduction, and molecular fingerprints). More formally, a supervised task is one where each training point has an associated external label (i.e. the feature we want to predict). In our case, the native ligand acts as a label for the binding site. On the other hand, an unsupervised task is one where we only have the input data but no ground truth; and the task becomes to compute the best possible classification of the data points. We will therefore additionally train a model to recognize structurally similar RNA neighborhoods via unsupervised node embedding techniques.

In this work, we propose a pipeline that combines supervised and unsupervised node embedding methods to best represent ABPN structure and maximize predictive performance. **Figure 3.2** provides an overview of our system. Given an ABPN, (**Figure 3.2 (a)**) we use an RGCN to compute an embedding for each node, to which add the identity of the corresponding nucleotide. In this manner, node embeddings represent structure and sequence identity. The

node embedding RGCN is pre-trained using an unsupervised structure encoding task (**Figure 3.2 (b)**). Since our task is to associate the entire ABPN with a molecular fingerprint, we use a pooling process (**Figure 3.2 (c)**), which aggregates node-level embeddings into a graph-level (binding site) representation. The final graph-level representation (the vector h in **Figure 3.2 (c)**), is fed through a simple neural network to output the final fingerprint. The entire network is trained to minimize the difference between the predicted fingerprint $\hat{y}$ and the native fingerprint y using a standard binary cross-entropy loss. Finally, we evaluate our predictions by using the predicted fingerprint to identify the native ligand from a compound library (**Figure 3.2 (d)**). See Supplemental Material Section C.2 for a full description of the neural network.

3.2.5 Unsupervised Pre-Training: ABPN Node Embeddings

Since RNA-small molecule binding events are relatively infrequent in the set of RNA 3D structures, the number of training points for ligand prediction (supervised learning) is limited. However, we are still able to leverage the full set of RNA 3D structures (3,972 full RNA structures vs 773 binding sites) using unsupervised pre-training, which is known to boost performance when labeled data is scarce [Erhan et al., 2010]. Recent methods have been developed for unsupervised learning on network data [Hamilton et al., 2017b; Sun et al., 2019]. As described in the preceding section, node embeddings can be trained to maximize performance on a prediction task (e.g. ligand prediction), or an unsupervised task (encoding similarity relationships, e.g. molecular fingerprints). In our case, we would train an RGCN to simply produce similar embeddings for RNA nodes with similar local structures. This would define a learning task on RNA structures for which we don't have a label (native ligand).

This process is analogous to molecular fingerprint building, where we wish to numerically encode structural similarity relationships. Once the RGCN has learned to encode the local structure of each node, the downstream task of ligand prediction becomes less prone to overfitting and more likely to learn general patterns [Erhan et al., 2010]. In the unsupervised setting, we train a model to produce embeddings for a pair nodes such that the similarity between the embeddings z_u and z_v is proportional to a user-defined similarity measure K which compares nodes u and v in the graph.

We are free to choose the pairwise node similarity function $K : (u, v) \rightarrow [0, 1]$ according to the application domain.

Here, we adapt the node similarity function proposed in `struc2vec` [Ribeiro et al., 2017] which allows us to capture local structural similarity across graphs. Other node similarity functions such as the ones used in `GraphClust` for RNA 2D structures [Heyne et al., 2012] are only able to compare nodes within the same graph and are affected by the distance between nodes which is not necessarily related to structural identity. Our similarity function addresses these limitations by comparing the counts of edge types in the local neighborhood of u and v. We provide an example of a comparison between a pair of two nodes on simplified graphs in **Figure 3.3** and show the result on a sample ABPNs in **Supplementary Figure C.3**.

Therefore, in the first training phase, our network tries to learn node embeddings on a large data set which are aware of general RNA structural patterns. Once this phase has converged, the model is then asked to predict ligand fingerprints.

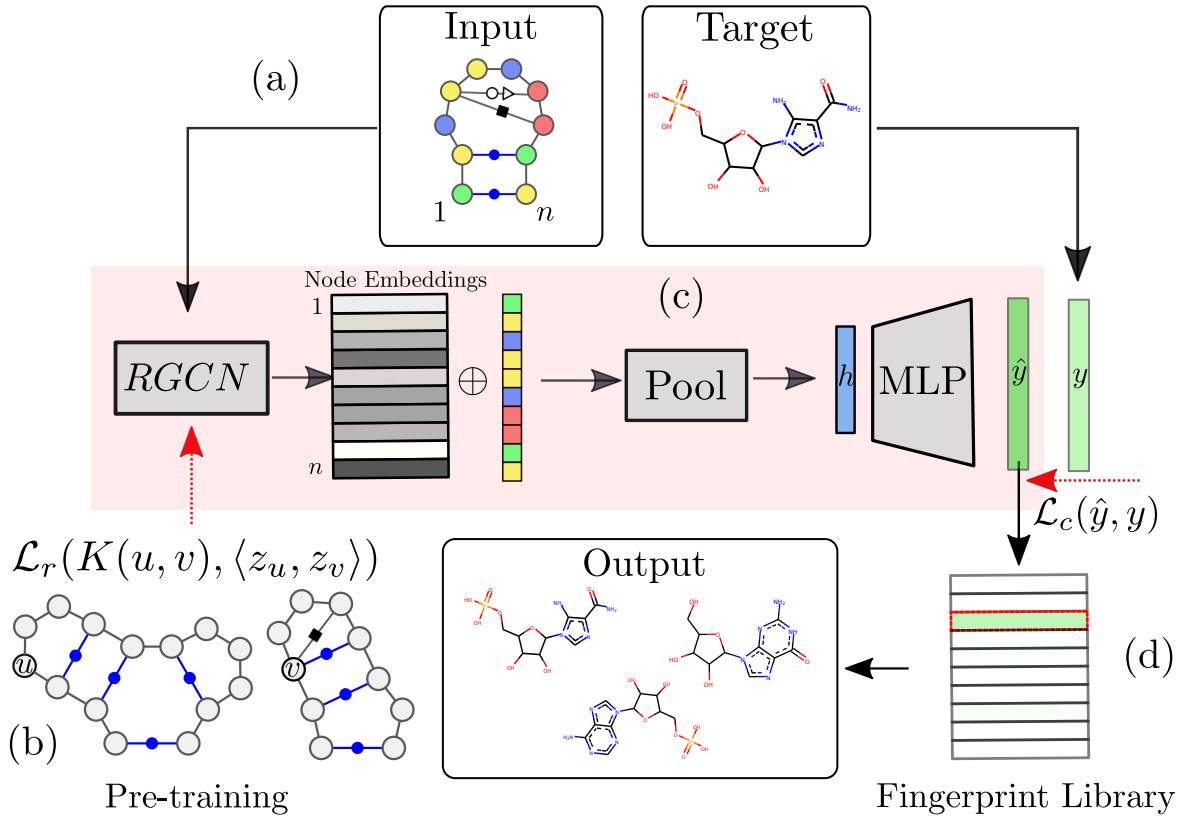


Figure 3.2 – Outline of the `RNAmigos` pipeline. A base pairing network is passed as input to `RNAmigos`. In training mode, it is paired with a native ligand (Target) from which a target fingerprint y is constructed. The embedding network (RGCN) produces a matrix of node embeddings of dimension $n \times d$ where n is the number of nodes in the graph, and d is a fixed embedding size. This is followed by a pooling step which reduces node embeddings to a single graph-level vector. Finally, the graph representation is fed through a multi-layer perceptron (MLP) to produce a predicted fingerprint $\hat{y}$ that minimizes the distance $\mathcal{L}_c$ to the native fingerprint y. The fingerprint is then used to search for similar ligands to the prediction in a ligand screen and thus enriches the probability of identifying an active compound. The RGCN network is pre-trained using an unsupervised node embedding framework which allows us to leverage structural patterns from a large dataset of RNA structures. This network is trained to generate embeddings which minimize the distance ($\mathcal{L}_r$) between kernel similarity $k(u, v)$ and embedding similarity $\langle z_u, z_v \rangle$

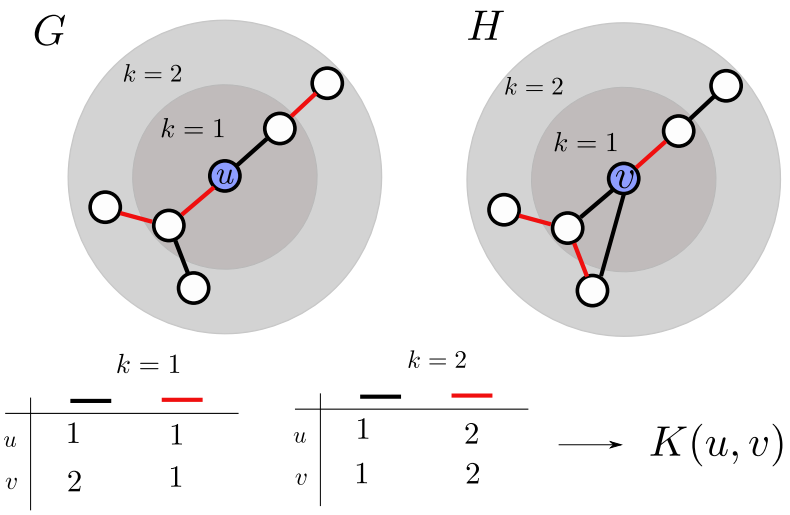


Figure 3.3 – Here, we compare the local neighborhoods of node u in graph G and node v in graph H. In this simple example, graphs only have one of two possible edge types, red and black. We compare the distributions of edge labels at each distance from the source nodes (u and v) to obtain the final similarity value $K(u, v)$.

3.2.6 Ligand Screen

Here, we propose a test to interpret the usefulness of our model by measuring its performance in a ligand screen setting. In a ligand screen, we are given a set of compounds and we seek to identify the most promising one. For validation, we know a native binder and we hide it in a set of inactive compounds, also known as decoys. The model is asked to find back the active. Given a binding site, our model produces a predicted fingerprint. We then rank all compounds of the decoy set according to distance to the predicted fingerprint. We normalize this score by the size of the set. Thus, a successful predictor will rank the native ligand as closest to its prediction (normalized rank close to 1), while a random predictor will result in an average rank of 0.5.

Considering that the distribution of RNA ligands appears to cluster to specific sub-regions (see **Supplementary Figure C.2**), this evaluation method also ensures that a classifier does not obtain a good score by simply predicting the average ligand as it would when only considering the absolute distance between the predicted and the native fingerprints.

We construct two decoy sets for our experiments. Since there are currently no experimentally validated data sets of active and inactive binders for a given RNA site (such as DUDE for protein [Mysinger et al., 2012]), our first set consists of all RNA-binding ligands in the PDB (270 ligands). The second decoy set is constructed using *DecoyFinder* [Adrià et al., 2012] on default settings, which samples a list of 36 decoys for each compound such that generic chemical properties are preserved while potentially disturbing binding potential. Of course, this test assumes that chemical dissimilarity between an active compound implies inactivity which is not always the case [Bantscheff et al., 2009]. However, the current aim of our work is simply to determine whether ABPNs retains a significant amount of information about its observed ligand

preferences, for which this test is sufficient.

3.3 Results

We report the resulting rank over the list of all RNA-small molecule pairs as well as the set of all decoys for each ligand, following the two decoy benchmark process.

Due to the limited size of our labeled data set, we performed a 10-fold cross-validation to include all training pairs in the evaluation and provide a more accurate measure of performance. All results are reported from the held-out sets in our validation, hence the model is never trained on the same binding sites that are being predicted on.

Node embeddings are computed using a 3-layer RGCN, each layer consisting of 16 dimensional inputs and outputs, a graph attention layer computing a 16 dimensional graph embedding and a fully-connected layer, which outputs a 166-dimensional vector. See **Supplementary Table C.1** for full model architecture and hyperparameters. Variations of the architecture used did not have strong effects on performance, so no extensive hyper-parameter search was conducted. We leave the exploration of other architecture choices for future work.

3.3.1 Augmented RNA Base Pairing Networks Encode Binding Preferences

Setting : The first hypothesis to test is that the proposed framework ($ABPN$) is able to retrieve some information about ligand binding. To explore this question, we compute the rank and distance metrics on ablated data. We compare this performance to three baselines:
· *random* consists of a synthetic label set where each binding site is assigned a uniformly random 166-dimensional binary vector (fingerprint).
· *swap* is designed to account for imbalances in the data (some ligands are more frequent than others): each binding site is assigned a fingerprint selected at random from the set of observed fingerprints. The overall distribution of ligand fingerprints thus remains the same but the input-output correlations are broken.
· *majority* is a constant ligand annotation computed as a majority vote over all fingerprints at each index. This is to be compared to the *swap* to check that the only thing that can be learnt on swapped data is over-representation of some ligands within the experiment.

The distributions of performance over each binding site-ligand pair are visualized for all experiments in **Figure 3.4** as a box plot. Summary statistics can be found in **Table 3.1** with accompanying standard deviations in Supp. Table S2, and Euclidean distances from the native ligand in Supp. Fig. S5. We also assessed the statistical significance of the difference of the means in a pairwise Wilcoxon rank test which is shown in **Table 3.2**.

Performance : In the RNA ligands setting, our full model achieves a rank of 0.68 and a mean-squared error (MSE) of 0.150 to the native fingerprint. The *random*, *swap* and *majority* experiments respectively yield ranks of 0.542, 0.603 and 0.603 and Mean Squared Errors (MSE) of 0.5, 0.18 and 0.18. This confirms that this model retrieves signal for the data and outperforms baselines. This conclusion is statistically significant based on a Wilcoxon p-value of at most $7e^{-18}$ between the model and the randomized results. As expected, the majority scheme is statistically equivalent to the swapped one and superior to the random one. These results are similar in the

Experiment	Ranks		L2	
	DecoyFinder	RNA	*DecoyFinder*	RNA
random	0.611	0.542	0.502	0.502
majority	0.621	0.603	0.175	0.179
swap	0.617	0.603	0.177	0.179
no-label	0.628	0.606	0.176	0.180
primary	0.624	0.592	0.181	0.186
secondary	0.631	0.605	0.178	0.182
ABPN	0.695	0.681	0.155	0.160
ABPN + unsup.	**0.735**	**0.715**	**0.145318**	**0.150189**

Table 3.1 – Mean ligand screen ranks and L2 (Euclidean) distance achieved on held-out binding sites for each experiment and decoy set.

DecoyFinder (See **Supplemental Figure C.4**) setting where the mean rank of the model is 0.69 compared to 0.62 in the majority setting. This shows that the full model successfully retrieves some signal and outperforms the baselines (Wilcoxon test results for DecoyFinder are shown in **Supplemental Table C.3**).

3.3.2 Augmented Base Pairing Networks Encode Ligand Binding Information

Next, we test the hypothesis that robust descriptors in the form of ABPNs from RNA domain knowledge are key to retrieve this signal. The question is whether the non canonical interactions encode information that lower levels of structure (secondary, primary) do not. We answer this question by performing three ablation experiments on our training set:
· *primary* encodes the binding sites as graphs that only contain node sequence and backbone interactions.
· *secondary* uses only information from the secondary structure which includes canonical pairs and backbones.
· *no-label* preserves all the interactions (and thus graph structure) in the graph (including non canonical) but does not distinguish between different edge types (i.e. edges only have one label).
In all these conditions, we find that **performance is no better than the randomized baselines**, indicating that non canonical interactions are essential for encoding specificity in ligand binding.

Indeed, the best performing model is *no-label* which has a Wilcoxon p-value of 0.55 with the *majority* experiment and of $1.7e^{-18}$ with the ABPN. This finding is in agreement with biological literature on RNA binding sites and the importance of complex structural motifs for determining functional specificity [David-Eden et al., 2010; Warner et al., 2018]. This is a major validation of the hypothesis that these are the correct representation for RNA structure for this task.

3.3.3 Unsupervised Pre-Training Boosts Performance

As explained in 'Unsupervised Pre-Training: ABPN Node Embeddings', one major limitation for this supervised task is the paucity of data. We investigate the possibility of using unsupervised learning by pre-training on an unsupervised task, and denote this experiment as *ABPN unsup.* The use of unsupervised pre-training of the node embedding network provides a significant

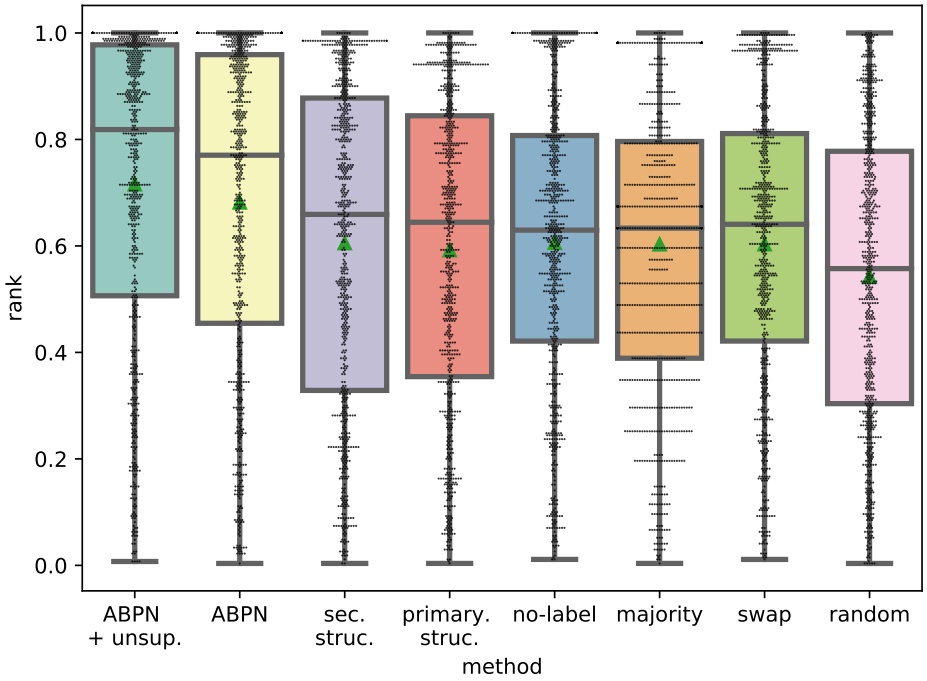


Figure 3.4 – Distribution of rank achieved on ligand screening. All points are from test set data on a 10-fold cross validation. The median is denoted with a dashed line and the mean with a green triangle. Each point is the normalized rank of one binding site's native ligand when searching for it using our network's predicted fingerprint.

performance boost over a network trained only on fingerprint reconstruction (MSE = 0.68 vs MSE=0.715), with a p-value of $2.9e^{-6}$ This is a methodological insight that can have applications for various other RNA-related tasks for which labeled data is typically scarce.

experiment 2 experiment 1	aBPN	secondary	primary	no-label	majority	swap	random
ABPN + unsup.	**2.9e-06**	**5.1e-26**	1.4e-22	2.1e-21	9.3e-25	7.2e-26	2.3e-18
ABPN	-	1.7e-11	5.6e-11	1.5e-08	4.3e-10	6.4e-12	2.0e-08
secondary		-	3.2e-01	7.7e-01	1.3e-01	2.8e-02	1.7e-01
primary			-	4.3e-01	2.7e-01	2.4e-02	3.2e-01
no-label				-	5.5e-01	1.5e-02	1.8e-01
majority					-	3.7e-01	3.3e-01
swap						-	5.5e-01

Table 3.2 – Wilcoxon rank test for all pairs of training conditions. Each entry in the table is the p-value for testing the hypothesis that the ranks resulting from a pair of experiments come from the same distribution. These are performed on the RNA decoy set. We provide the test results for the *DecoyFinder* decoy set in **Supplemental Table C.3** material and show consistent results.

3.3.4 Our Model Can Predict Diverse Ligand Classes

Next, we ask whether the positive results can be explained by a small set of ligands, or whether it is able to achieve high scores on a diverse set of ligands. To get a better view of performance, we plot the same prediction scores but averaged over ligand types (270 unique ligands) against a hierarchical clustering dendrogram of each ligand (shown in **Figure 3.5**).

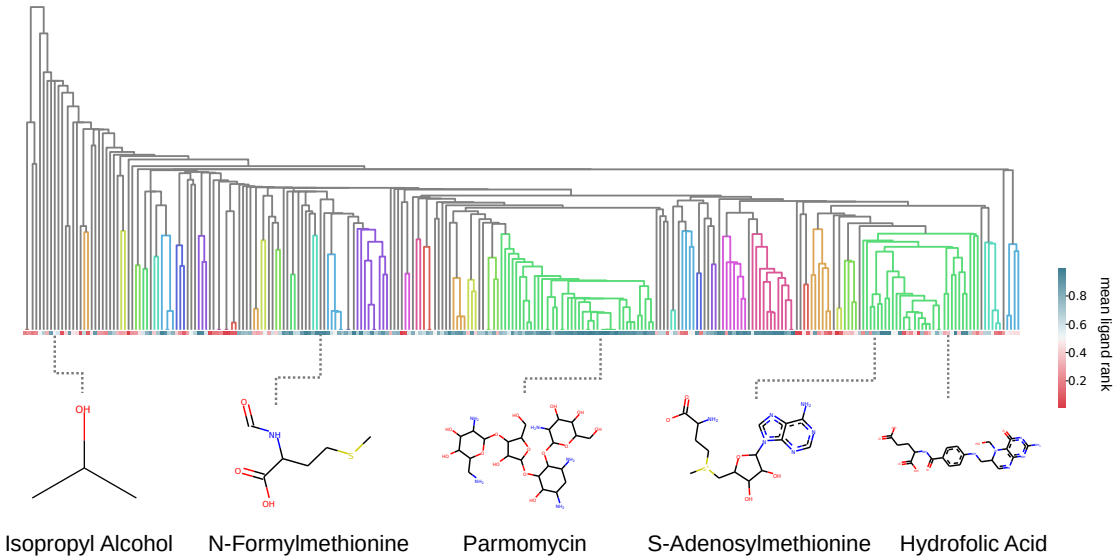


Figure 3.5 – `RNAmigos` performance by ligand class. Hierarchical clustering dendrogram of the ligands, classifying ligand families by similarity. Each cell in the horizontal grid is the average score for binding sites containing a given ligand. Ligands belonging to the same tree are grouped together by the clustering procedure. Colored-in sub-trees denote tight clusters which contain ligands within 0.25 Jaccard distance.

Colored-in subtrees indicate groups of ligands that are similar, (i.e., within 0.25 Jaccard distance of each other) which would indicate strong clustering. In this manner, we are able to assess the performance across 'classes' of similar ligands. We first observe that successful classifications are not restricted to a single class of ligands and instead show good predictions for diverse ligands. Interestingly, the class that is most consistently predicted accurately corresponds to the aminoglycosides (highlighted in the green cluster in the middle). Aminoglycosides are a class of antibiotics binding to bacterial RNA with well-defined binding sites [Walter et al., 1999], and are quite abundant in the dataset. Nucleic acid-like compounds, many of which bind riboswitches, also form a large family of binders (green) however results were less consistent than for aminoglycosides. A possible explanation for strong performance on aminoglycosides, apart from the larger number of examples obtained, is that these are typically large polysaccharide-like structures with a large number of interactions with the RNA. On the other hand, riboswitches bind much smaller molecules with a limited number of interactions. As a result, binding site requirements are much more complex and specific with aminoglycosides and the large number of interactions can only be fulfilled by a limited number of molecules. We leave this question

for future work, as with the current dataset size, we are unable to provide quantitative evidence of such phenomena. Finally, ligands clustered on the left of the dendrogram show the weakest performance. Since these groups show little branching in the dendrogram, we can conclude that they represent sparsely populated ligand classes for which we have few examples and thus, obtaining more data in these regions could improve performance.

3.3.5 Comparison with a Secondary Structure-Based Tool

Finally, we compare the performance of `RNAmigos` with the closest related tool we could find, Inforna [Disney et al., 2016]. Inforna accepts as input a RNA sequence with a secondary structure and returns a list of candidate ligands, based on sequence and structural similarities with motifs stored in a database. Although the input is not strictly identical, it is quite close to the one of `RNAmigos`. Similarly for the output, Inforna provides direct ligand information. In this benchmark, we provide Inforna secondary structures computed with Forgi [Thiel et al., 2019] directly from the PDB files, which is the most accurate input available.

For each chain in each RNA PDB associated with a ligand in the PDB database set, we query the Inforna web server and obtain a list of candidate ligands that we use to search for the native ligand (see Supp. Table S4). Importantly, in contrast to `RNAmigos` for which we directly provide the binding site, Inforna scans the whole input structure for candidate sites. To address this discrepancy, we only take the maximum score returned by Inforna for each structure.

We show the results of our benchmark in **Figure 3.6** (see Supp. Fig. S6 for corresponding distance comparisons). We were able to obtain predictions for 176 unique RNA chains corresponding to 82 unique ligands. The reduction from the RNAmigos set is mainly due to excluding PDBs which contain protein and some secondary structure extraction failures. In this context, Inforna achieves an average enrichment at the level of our random model (mean rank 0.43, and distance 0.48), and to some extent also consistent with the performance of `RNAmigos` using only secondary structure information.

We analyzed the performance across ligand classes (see **Supplemental Figure C.7**) and observed that the accuracy of the predictions appears to be stronger in well-known classes such as aminoglycosides and riboswitch ligands, but the performance decreases sharply outside these classes. This phenomenon could highlight a shortcoming of non-generalizable models, and thus a benefit of our approach. Looking at ligand classes where both tools made predictions (**Supplemental Table C.4**), we observe that `RNAmigos` outperforms Inforna in nearly all classes (Inforna outperformed `RNAmigos` on 10 of 66 ligands tested on Inforna by a margin larger than 0.1.). It suggests that the richer structural representation leveraged by `RNAmigos` is an important source of specificity. Since both tools work with differing levels of representation (2D vs augmented 2D) and at different scales (binding site vs full sequence), we stress that this benchmark does not intend to be a direct comparison but rather a demonstration that higher-level interactions are a crucial source of information.

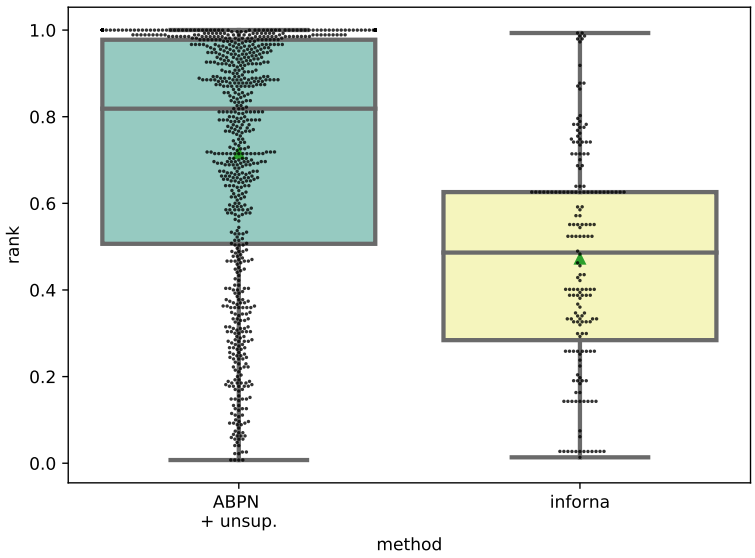


Figure 3.6 – Distribution of native ligand rank achieved on ligand screening in `RNAmigos` and Inforna.

3.4 Discussion

We have developed a unique computational platform, `RNAmigos`, to show that augmented RNA base pairing networks contain useful ligand binding information. The significance of our results is two-fold.

We show for the first time that ABPN encodes sufficient information for a classification task, and establish an initial methodological primitive for such a task. To date, the majority of computational methods which leverage ABPNs have focused on sequence to structure [Sarrazin-Gendron et al., 2019; Zirbel et al., 2015] prediction and motif identification [Petrov et al., 2013b; Reinharz et al., 2018a]. While these tasks involve some degree of learning, the relevance of higher-order interactions lies ultimately in their potential to specify function, which until now has been left unexplored. Interestingly, these findings come at a time when information of the type our model uses is becoming more widely available. Computational prediction tools such as [Sarrazin-Gendron et al., 2019; Zirbel et al., 2015] promise to yield large amounts of higher-order RNA pairwise interaction data without need for costly crystallography experiments. This opens the door to applying such data in other important biological problems such as RNA binding protein prediction [Uhl et al., 2019] and ion binding [Philips et al., 2012]. Furthermore, the promising results obtained from the unsupervised pre-training provide a methodological building block for assisting in supervised learning on complex RNA structures.

Second, our findings take a first step towards learning-based methods for systematically identifying drugs binding to RNA, and pinpoint ABPNs as essential tools for this task. The finding that only ABPN representations of binding sites was able to produce a significant signal in the task indicates that richer representations are necessary for successful classification when complex interactions are at play. Since our prediction is a fingerprint vector (chemical descriptor) and not a simple classification of ligands (i.e directly selecting a single ligand as output, or

predicting an affinity), the fingerprint itself can be used to search large ligand databases, and can eventually be applied to direct molecule generation [Gómez-Bombarelli et al., 2018]. While performance was strong across different ligand classes, it is apparent that classes for which data is more abundant received more consistently positive predictions. Therefore, as more examples of RNA-ligand complexes are characterized by experimental and computational techniques, we believe that the performance of our platform will improve. Additional data will also allow us to account for properties desired in medical applications such as synthesizability, drug-likeness [Lipinski, 2004]. Our choice of graphs for binding site representation reflects this consideration, as graphs can natively hold additional information such as evolutionary or chemical properties without requiring changes to the pipeline. Furthermore, recent advances in graph neural networks would provide the ability to model binding site flexibility [Pareja et al., 2019]. Eventually, computational predictions of ABPNs from sequence [Sarrazin-Gendron et al., 2019] combined with our methods will enable large-scale searches for binding sites.

We hope that this work will motivate further investigation of the links between ABPNs and RNA function, and eventually facilitate efforts in RNA targeted drug discovery.

Chapter 4

Vernal : A Tool for Mining Fuzzy Network Motifs in RNA

This chapter was made in collaboration with Carlos Oliver, Pericles Philippopoulos, William L Hamilton and Jérôme Waldispühl and was published in Bioinformatics in 2022 [Oliver et al., 2022].

Contents

Abstract

Motivation : RNA 3D motifs are recurrent substructures, modeled as networks of base pair interactions, which are crucial for understanding structure-function relationships. The task of automatically identifying such motifs is computationally hard, and remains a key challenge in the field of RNA structural biology and network analysis. State of the art methods solve special cases of the motif problem by constraining the structural variability in occurrences of a motif, and narrowing the substructure search space.
Results : Here, we relax these constraints by posing the motif finding problem as a graph representation learning and clustering task. This framing takes advantage of the continuous nature of graph representations to model the flexibility and variability of RNA motifs in an efficient manner. We propose a set of node similarity functions, clustering methods, and motif construction algorithms to recover flexible RNA motifs. Our tool, VERNAL can be easily customized by users to desired levels of motif flexibility, abundance and size. We show that VERNAL is able to retrieve and expand known classes of motifs, as well as to propose novel motifs.
Availability and Implementation : The source code, data and a web server are available at vernal.cs.mcgill.ca

Résumé

Motivation : Les motifs 3D de l'ARN sont des sous-structures fréquentes, modélisées par des réseaux d'interactions entre paires de bases, et cruciales pour la compréhension des liens entre structure et fonction. La détection automatique de ces motifs est computationnellement difficile et représente un challenge primordial pour le domaine de la biologie structurale de l'ARN et de l'analyse des réseaux. Les méthodes existantes résolvent des cas particuliers de la détection de motifs, en restreignant la variabilité au sein des occurrences d'un motif et en se limitant à certaines sous-structures.
Resultats : Dans ce travail, nous relâchons ces contraintes en formulant cette détection comme un problème d'apprentissage de représentation de graphes et de regroupement. Cette formulation utilise la continuité des représentations induites pour modéliser la flexibilité et la variabilité de ces motifs de manière efficace. Nous proposons un ensemble de fonctions de similarité, de méthodes de regroupement et d'algorithmes de construction de motifs pour détecter des motifs d'ARN flexibles. Notre outil, VERNAL peut facilement être ajusté par ses utilisateurs pour fixer des niveaux de flexibilité, fréquence et taille de motifs. Nous montrons que VERNAL est capable de retrouver des motifs connus et de les étendre, ainsi que d'en proposer de nouveaux.
Disponibilité : Le code source et les données, ainsi qu'un serveur web sont disponibles à l'addresse : vernal.cs.mcgill.ca

4.1 Introduction

Comparisons of RNA structures revealed the occurrence of highly similar 3D sub-units, called RNA 3D motifs, which are thought to form a basis for non-coding RNA function [Leontis et al., 2006]. These motifs are typically characterized by sets of similar base pairing patterns, which are repeated across unrelated RNA [Lescoute et al., 2005]. A complete library of RNA 3D motifs is thus a valuable source of information for evolutionary studies, discovering functional sites, and is an important component of structure prediction methods [Roll et al., 2016; Sarrazin-Gendron et al., 2019]. Moreover recent advances in small molecule graph generation with deep learning demonstrate the importance of motifs as building blocks [Jin et al., 2020a]. Efficient and automated methods to mine motifs from databases of RNA structures are essential to achieve this goal [Djelloul, 2009; Petrov et al., 2013b; Reinharz et al., 2018a].

From a methodological point of view, RNA motif mining methods can be placed in two categories: 3D-based and graph-based. 3D-based tools seek to identify families of related structures by performing alignments and clustering of *atomic coordinates*. DARTS [Abraham et al., 2008], `RNA 3D Motif Atlas` [Petrov et al., 2013b], RNA Bricks [Chojnowski et al., 2014], and RNA MCS [Ge et al., 2018] illustrate this approach. Since structural proximity is naturally defined in coordinate space, an advantage of these tools is that variability across occurrences of a motif is achieved for free. However, these methods require a decomposition of RNA into rigid sub-units to be compared to each other (e.g., comparing all internal loops to each other), which limits the scope of possible motifs to be found.

Alternatively, graph-based approaches work on discrete encodings of RNA 3D structures and rely on network analysis algorithms to extract motifs. The building blocks of such encodings are linear polymers of nucleotides (A, U, C, G) bound by backbone interactions. These chains determine first, the highly stable canonical (Watson-Crick and Wobble) and then non-canonical (all other) base base pairing geometries [Leontis and Westhof, 2001]. These pairs serve as a scaffold for the formation of the full tertiary structure. The conservation of these base pairs is essential to preserving the folding properties of the RNA and offers a robust signature for the functional classification of RNAs [Griffiths-Jones et al., 2003; Oliver et al., 2020; Zhang et al., 2016]. Using these components, any RNA 3D structure (set of atomic coordinates) can be represented as a multi-relational graph (also referred to as base pairing network), where nodes correspond to nucleotides, and edges to interactions between nucleotides. Edge types are assigned based on the classification developed by Westhof et al Leontis and Westhof [2001] which defined 12 categories of relative base pair orientations. These classes can be determined by noting the relative angles and interacting atoms of the bases involved in the pair in 3D space. In this set of 12 geometries, we can find the standard Watson-Crick (A-U, C-G, G-U) pairs, also known as "canonical base pairs", which are the most stable and abundant class. However, when interpreting 3D motifs, the remaining 11 geometries, also known as "non canonical" are necessary for understanding RNA structure at a 3D level [Leontis and Westhof, 2001].

Graph-based tools therefore typically aim to identify similarities at the base pairing level. Of course, identifying motifs requires a combinatorial search and hence such tools impose strong limitations on the search over subgraphs. Chief among these is the ability to include variability within motifs. The notion of a fuzzy motif has been very well studied in the sequence domain [D'haeseleer, 2006] where certain DNA sequences are accepted to be related while their nucleotide composition can vary. Not surprisingly, the same applies in the RNA structural domain where

well-studied motifs such as the A-minor are known to admit variability in their connectivity pattern [Nissen et al., 2001] Closely related 3D structures may be represented as quasi isomorphic (or fuzzy) graphs. However, most classic and tractable graph comparison algorithms rely on exact matching. Methods which rely on strict isomorphism would fail to identify fuzzy instances, as well as fail to discover some motifs entirely. Another limitation here is that evaluating all potential graph motifs involves searching over the set of all subgraphs which grows exponentially in the graph size. A first approach to resolve this challenge is to use already known motifs as a queries to search for new instances (RMDetect [Cruz and Westhof, 2011], RNAMotifscan [Liu et al., 2011; Popenda et al., 2010; Zhong and Zhang, 2015; Zhong et al., 2010]). However these methods require motifs as input, which are typically obtained through visual inspection such as A-minor or kink-turn motifs. There is thus a need for *de-novo* motif mining tools that look into the space of subgraphs and find the recurrent ones. To avoid enumerating all subgraphs, many motif mining tools restrict themselves to specific substructures. A first reach for *de-novo* motifs is thus conducted by Lemieux and Major [2006] working only on a single ribosomal unit, and focusing on cycle motifs, a very specific and small subgraph type. `rna3dmotif` [Djelloul, 2009] offered the first library of exact motifs only within certain known structural elements, namely the k-way junction. Another approach to this problem is metaRNAmodules [Theis et al., 2013] which enumerates all nested loops and uses RMDetect with a statistic to filter the recurrent ones. Another graph-based method, developed in 2015 is RAG-3D [Zahran et al., 2015] which uses a graph abstraction to mine motifs spanning multiple secondary structure elements, and simultaneously proposes a query-search functionality as discussed above. More recently, `CaRNAval` [Reinharz et al., 2018a; Soulé et al., 2021] attempted to expand the class of motifs by considering interactions that connect multiple secondary structure elements while maintaining isomorphic motif instances. All these tools either focus on local motifs or impose a strict isomorphism of motif occurrences.

Contributions

We leverage the state of the art in graph representation learning to build continuous embeddings of RNA substructures and identify structurally conserved yet variable motifs. We then propose two algorithms that use these graph representations to find graphs similar to a query and to identify novel motifs. By comparing with existing motif libraries, we are able to efficiently identify unknown instances of existing motifs, and propose over 1,800 densely populated motifs for further exploration.

4.2 Datasets

We extract motifs from the set of experimentally determined RNA structures [Rose et al., 2016]. To ensure that the frequency of a motif is not biased by redundant structures, we use the representative set at 4 Angstrom resolution provided by the BGSU RNA 3D Hub [Petrov et al., 2013b]. We then build an RNA network with 13 edge types for each RNA using the FR3D annotations provided by the same framework. Each edge represents either a backbone (covalent) interaction, or a base pair classified in 12 geometries according to the relative orientation of the interacting bases (nodes). This results in a total of 899 RNA graphs and 210616 nodes (nucleotides). In the learning phase, we chop these graphs in constant-sized chunks of approximately 50 nucleotides to

avoid dealing with graphs of heterogeneous sizes, as is detailed in **Supplementary Algorithm 3**. Once the model is trained we perform all motif finding operations on whole graphs. Our validation sets consist of motifs identified by `RNA 3D Motif Atlas` [Petrov et al., 2013b], `rna3dmotif` [Djelloul, 2009], and `CaRNAval` [Reinharz et al., 2018a].

4.3 Methods

We introduce VeRNAl, an algorithm to efficiently identify fuzzy recurrent network motifs in RNA. VeRNAl decomposes RNA networks into structural building blocks and then aggregates these blocks based on their co-occurrence in RNA.

The decomposition step introduces custom structural comparison functions which are used to build a space of continuous embeddings for efficient clustering (Section 4.3.2). We then combine information from the embedding space and connectivity in the graph space into a meta-graph data structure (Section 4.3.3). We leverage this data structure to retrieve graphs similar to a query (Section 4.3.4), and to streamline frequent substructure searches and thus identify *fuzzy* motifs (Section 4.3.5).

4.3.1 Problem Definition

As described above, each RNA 3D structure can be encoded into a multi relational graph. Without loss of generality, we consider that the set of all such graphs forms one large disconnected directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with about 670k nodes. We define a motif as a set of subgraphs $\mathcal{M} = \{g_1, g_2, ..\}$, drawn from $\mathbb{G}$, such that the following properties hold:

1. *Similar:* Let s be a similarity function on graphs, and $\gamma \in [0, 1]$. $\forall (g_i, g_j) \in \mathcal{M}^2$, $s(g_i, g_j) \geq \gamma$
2. *Connected:* $\forall g_i \in \mathcal{M}$, g_i is a connected subgraph.
3. *Frequent:* the number of subgraphs of $\mathcal{M}$ should be above some user-defined threshold : $|\mathcal{M}| > \delta$

The motif mining problem is simply to identify all sets of subgraphs (motifs) $\mathcal{M} \in \mathbb{G}$ that fit the above criteria. An exact solution to this problem would imply enumerating all subsets (search for subgraphs) of $\mathbb{G}$ and ensuring that these criteria are satisfied (compare graphs). In the most general case, both procedures admit exponential time algorithms [Zeng et al., 2009]. Previous works set $\gamma = 1$, so that the similarity constraint becomes another graph problem, known as the subgraph isomorphism problem [Djelloul, 2009; Reinharz et al., 2018a]. This also constraints these methods to rely only on pairwise comparisons and prevents them from detecting communities of close but different neighbors. Additionally, the search step is often limited by considering only certain substructures (e.g. hairpins, internal loops, etc). We can consider two additional properties of motifs: *maximality*, and size. A motif is said to be *maximal* if adding a node to g_i breaks the other motif constraints. We define the size of a motif as the mean number of nodes per graph in $\mathcal{M}$. Enforcing these two properties is left as an implementation choice. For example `CaRNAval` [Reinharz et al., 2018a] returns maximal exact subgraphs containing long range interactions with no size constraints. Here, we remove search constraints on secondary

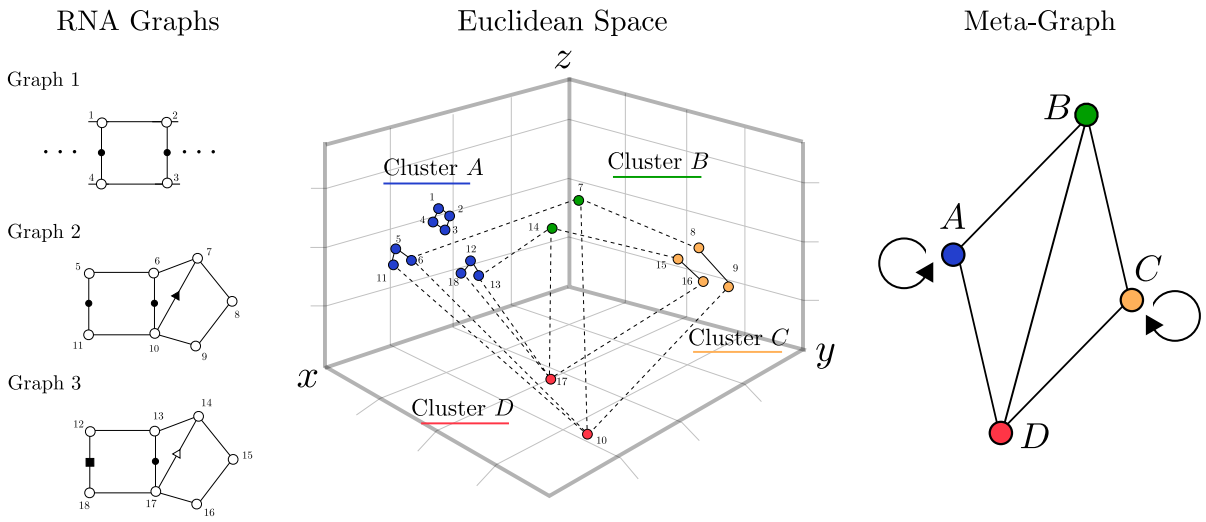


Figure 4.1 – Meta-graph creation : RNA graphs *(Left)* get aligned in the embeddings space *(Middle)* and represented as a meta-graph *(Right)*. RNA nodes are grouped in meta-nodes through clustering, which reflects structural similarity. Meta-edges are then inferred from the source graphs connectivity

structure context [Petrov et al., 2013b; Reinharz et al., 2018a]. We also allow for non-identity γ and call this property *fuzziness*.

4.3.2 Rooted Subgraph Embeddings

Maximal subgraph isomorphism algorithms rely on heuristics that are not applicable anymore for $\gamma < 1$. Allowing fuzziness calls for other efficient ways to compare graphs. We turn to recent advances in Graph Representation Learning, which provide a framework for encoding rooted subgraphs [Hamilton et al., 2017b]. A rooted subgraph g_u is the induced subgraph on the set of nodes $u' \in g$ such that $p(u, u') \leq r$ where p is the length of the shortest path between two nodes, and r is a user-defined threshold (also known as radius). A vector embedding of dimension d for a given g_u is computed by a parametric function $\phi : g_u \to \mathbb{R}^d$ (typically a Graph Neural Network). These embeddings seek to approximate a graph similarity function $s_G : g_u \times g_v \to [0, 1]$ acting directly on graphs. In an unsupervised setting, ϕ is trained to minimize the loss described in Equation 4.1. The resulting embedding space conveniently captures the desired property of fuzziness, as proximity in the embedding space (via inner product) corresponds to structurally similar nodes in the graph space.

$$\mathcal{L} = \|\langle \phi(g_u), \phi(g_{u'}) \rangle - s_G(g_u, g_{u'})\|_2^2, \tag{4.1}$$

For RNA motifs, we are only interested in considering edge type and graph structure. Notably, it is known that certain base pairing geometries (edge types) share structural similarities : a phenomenon known as isostericity (Supplementary Section D.1.2). We introduce various customized similarity functions on RNA graphs which account for key 3D geometric features such as isostericity [Stombaugh et al., 2009] and base pairing type [Leontis and Westhof, 2001].

They are based on the matching of sub-parts of the rooted subgraphs as computed using the Hungarian algorithm [Kuhn and Yaw, 1955]. We then use a Relational Graph Convolutional Network (RGCN) model [Schlichtkrull et al., 2018] as the parametric mapping. The network is implemented in Pytorch [Paszke et al., 2019] and DGL [Wang et al., 2019]. To focus performance on subgraphs that contain non canonical nodes and avoid the loss to be flooded by the canonical interactions (Watson-Crick pairs), we then scale this loss based on the presence of non canonical interactions in the neighborhood of each node being compared. Details on the similarity functions and the training of the network are included in the Supplementary Sections D.3 and D.4. We can then perform clustering in the embedding space, using the k-means algorithm [MacQueen et al., 1967]. We select the number of clusters according to the Silhouette Score and several clustering metrics (See **Figure D.4**). We denote the resulting clusters as 1-motifs (rooted subgraphs with 1 root) as they represent the aforementioned structural blocks of RNA.

4.3.3 Meta-graph

While there is no limit to the size of a real-world motif, our rooted subgraph embeddings are currently only aware of a fixed-size neighborhood, (i.e. the radius of the rooted subgraphs). For this reason, these clusters only identify motifs as large as the radius of the rooted graphs. They are centered around just one node so we denote them as 1-motifs. However, we want to extend these to k-motifs by aggregating k different nodes based on co-occurrence in the original graph.

To guide this aggregation, we introduce a meta-graph data structure $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, whose meta-nodes are composed of regions of the embedding space and whose meta-edges are based on the connectivity in the RNA graphs between those regions. Meta edges are weighted by the amount of existing edges they represent. Hence, the meta-graph simultaneously encodes structural proximity and connectivity in the graph in one object. We can see it as a coarsened version of $\mathbb{G}$ when its nodes are structurally embedded in the Euclidean space. If two meta-nodes are connected by a large meta-edge, it means that these two structural elements are often adjacent in the graph and thus, they are good candidates to be merged.

To get the meta-nodes, we embed the original nodes in $\mathbb{V}$ into $\mathbb{R}^d$. We then cluster these embeddings and use the clusters as meta-nodes : $C_i = \{u \in \mathbb{V},\ \text{cluster}(\phi(g_u)) = i\}$, where C_i is the $i-th$ node in $\mathcal{G}$. Choosing the number of clusters and their spread modulates the fuzziness of the resulting motifs. Meta-edges $E_{i,j} = \{(u_i, u_j) \in (C_i \times C_j) \cap \mathbb{E}\}$ store the edges in RNA graphs that go from one cluster to another. This process is illustrated in **Figure 4.1**. Building the meta-graph requires RGCN inference and clustering over $\mathbb{V}$, and iterating through all edges in $\mathbb{G}$. With linear-time clustering techniques (like k-Means [MacQueen et al., 1967] or Gaussian Mixture), building the meta-graph is therefore done in time $O(|\mathbb{V}| + |\mathbb{E}|)$.

This meta-graph data structure helps us merge clusters together based on their connectivity. Merging meta-nodes C_1 and C_2 results in a 2-meta-node which contains all subgraphs in C_1 that are connected to a subgraph in C_2. These subgraphs are indexed in the meta-edge $E_{1,2}$. However, inferring the connectivity of the merged meta-node is not trivial. Indeed, the neighbors of a merged meta-node are not the union of the neighbors of its constituent meta-nodes, since we retain only a subset of the graphs of each merged meta-node. This problem is illustrated in **Figure 4.2** where the merged meta-node BC is not connected to A despite A and B being connected. To address this problem, we only consider adding 1-meta-nodes to an arbitrary meta-node. This is not a limitation for our algorithm and is more compatible with our formalism. We

implement a merging algorithm presented in **Algorithm 1**.

Algorithm 1: Merging Algorithm. We merge a meta-node S with a 1-meta-node C, through a meta-edge E.

Data:

- Meta-node S
- Meta-edge E from S to C

Result: T, an expanded S along E

```
T ← ∅
foreach (v1, v2) ∈ E do
    foreach nodeset ∈ S do
        if {v1, v2} Δ nodeset ≠ ∅ then
            T ← T ∪ {nodeset ∪ {v1} ∪ {v2}}
        end
    end
end
return T
```

4.3.4 Retrieving known motifs

The first use of the meta-graph data structure is to retrieve subgraphs similar to a query subgraph. Given a query subgraph Q and a large disconnected graph $\mathbb{G}$, the task is to identify all hits : subgraphs $H \in \mathbb{G}$ which maximize similarity to the query. The idea of the algorithm is to use the 'alignment' of the RNA graphs induced by the embeddings (**Figure 4.1**) to efficiently search for similar structures. Such an algorithm can identify subgraphs that resemble known motifs but which were not identified by tools imposing strict isomorphism [Cruz and Westhof, 2011; Zhong and Zhang, 2015; Zhong et al., 2010].

Using the RGCN, we place the query graph in the embedding space, which induces a query meta-graph $\mathcal{Q}$. This meta-graph is a subgraph of $\mathcal{G}$, each query meta-node is defined by the cluster assignment of the embedding of each nucleotide of the query graph. Then for each edge in the query graph, we add a meta-edge between its corresponding meta-nodes. Let c_i be the centroid of a meta-node C_i. We can directly obtain a compatibility score between a query meta-node $q \in \mathcal{Q}$ and a hit node $h \in \mathbb{G}$: $score = \langle \phi(g_h), c_q \rangle$. We start with all nodes in a meta-node of $\mathcal{Q}$, as one-node hits. The one-node hits and their scores are added to a set $\mathcal{R}$.

To expand the match, we iterate through the edges of the $\mathcal{Q}$ and merge any two elements of $\mathcal{R}$ that fall along the current edge. We do so by using the aforementioned merging algorithm. We can think of this step as iteratively building bigger and bigger sub-graphs of $\mathbb{G}$ that match the template provided by $\mathcal{Q}$. Any merge operation increases the score of the resulting set by summing the score of the merged elements. If a hit encompasses all nodes in the query, it will

[0] Δ represents the symmetric difference of two sets. Here, it indicates if precisely one of the nodes is already in the nodeset.

have undergone the most merging operations and obtain a maximal score. However, if a hit misses one node or has a somewhat different structure, we still retrieve it with a sub-optimal but high score. This retrieval procedure is detailed in **Algorithm 2**.

Algorithm 2: Motif Instances Retrieval. We traverse the edges of the graph $\mathcal{Q}$ and the meta-graph identifying connected subgraphs which match query embeddings.

Data:

- Meta-graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$
- Original RNA graphs $\mathbb{G} = (\mathbb{V}, \mathbb{E})$
- Query multi-graph $\mathcal{Q}$

Result: $\mathcal{R}$: Motif instances candidates : a set of subgraphs and their associated scores

$\mathcal{R} \leftarrow \bigcup_{C \in \mathcal{Q}} C$
foreach E *in* $\mathcal{Q}$ **do**
  $T \leftarrow \texttt{merge}(\mathcal{R}, E)$
  $\mathcal{R} \leftarrow \mathcal{R} \cup T$
end
return $\mathcal{R}$

The algorithm remains tractable thanks to the sparsity of the meta-graph that allows efficient iteration through edges, efficient set operations to expand motifs and graph-based separation of the candidate hits. A theoretical analysis of the complexity depends heavily on both the topology of the meta graph and of the query graph and is explained further in Supplementary Section D.6. We observe that in practice, this algorithm runs in an average of 10s on a single i7-10610U core.

4.3.5 Mining new motifs

We can leverage a similar strategy to the retrieve procedure when mining motifs *de novo*. The basic intuition of our algorithm, Motif Aggregation Algorithm (MAA) is again that the set of nodes assigned to a given cluster can be considered to be a motif of cardinality 1 (a 1-motif). We can then use the meta-graph to identify clusters with connections to the current motif set to build larger motifs. Because we lack the guidance of the query, instead of merging just along one meta-edge, we merge along all meta-edges in the meta-graph and filter results based on a user-defined minimal frequency δ.

As an example, starting with a 1-motif e.g. the set of subgraphs in cluster A, we can create 2-motifs by merging every other cluster in its meta-graph neighborhood, $X \in \mathcal{N}(A)$. We then identify the new 2-motifs from their constituent meta-nodes. This process can then be iterated to discover k-motifs. This is illustrated in **Figure 4.2** and outlined in detail in **Supplementary Algorithm 6**.

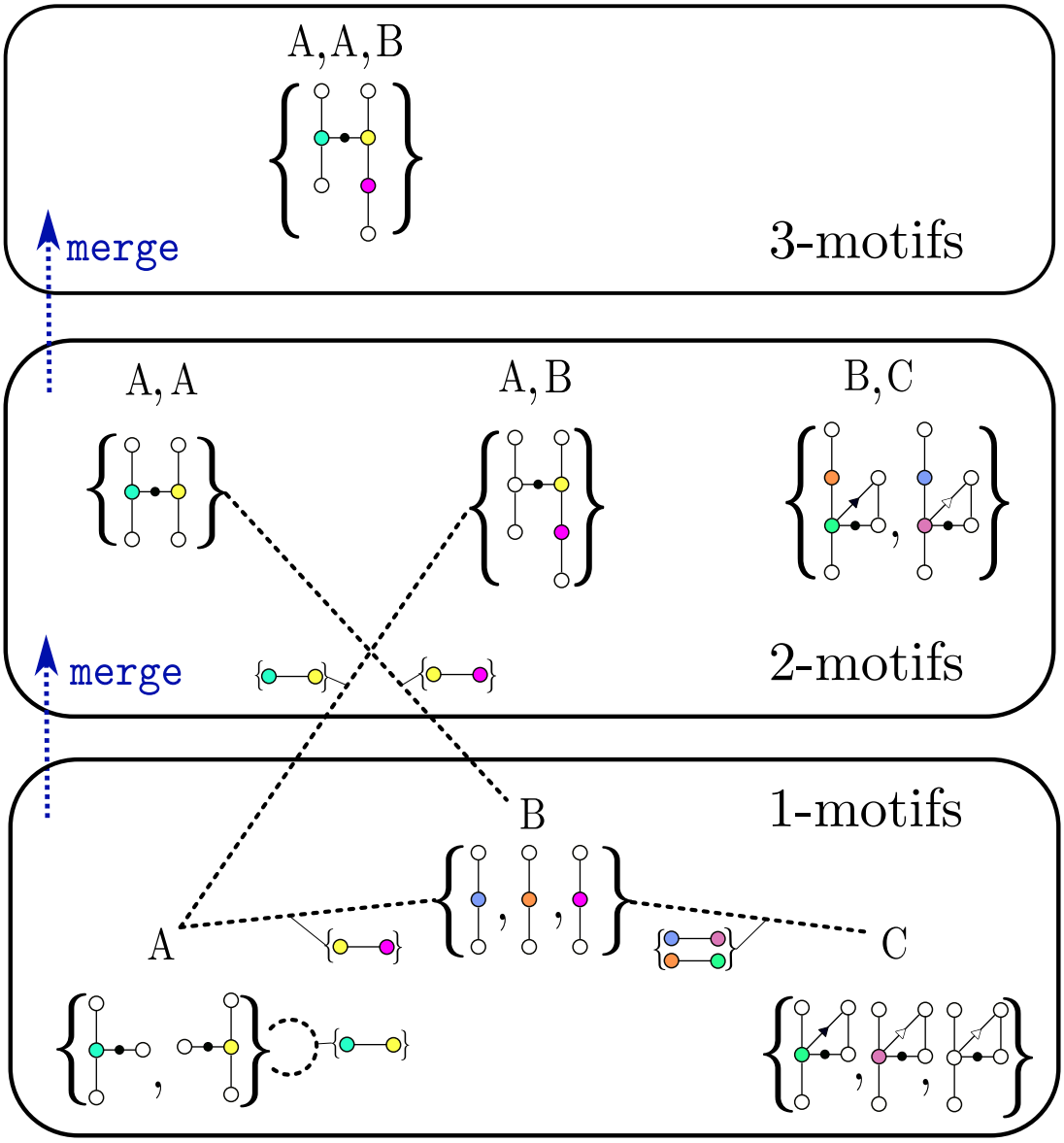


Figure 4.2 – MAA Illustration : Meta-nodes A,B and C get merged into three 2-Meta-nodes AA, AB and BC. Then new meta-edges are computed that link singletons A and B with 2-meta-nodes AB and AA respectively. A second merge follows these links and yields the 3-meta-node AAB. Node colors here are a proxy for node ID, and not tied to the cluster IDs (A, B, C).

Given the state of the meta-graph $\mathcal{G}$ at step k, identifying all instances of a single $(k+1)$ motif requires a single `merge` operation between two meta nodes, with complexity $\mathcal{O}(n)$. The number of all such possible $(k+1)$ motifs in the worst case is $\mathcal{O}\left(\binom{|\mathcal{C}|}{k}\right)$ which is the number of choices of $(k+1)$-size meta nodes. However, this is a loose bound since in practice, the number of acceptable motifs is constrained by the sparsity of $\mathcal{G}$ at $k=0$ and the minimum motif frequency δ. Empirical complexity depends strongly on hyper-parameters choices but is on average a few minutes on a single i7-10610U core.

4.4 Results

Our tool relies on graph representation methods to drastically improve the scalability of motif mining and facilitate fuzzy matching of motifs. We first evaluate the quality of our RNA-specific similarity functions and subsequent embeddings (Section 4.4.1) and show that structural information is faithfully encoded. Following this, we show that our approach can consistently retrieve existing motifs (Section 4.4.2) while also uncovering new fuzzy motifs (Section 4.4.3). Throughout the evaluation of the tools, we use Graph Edit Distance (GED) [Bunke and Riesen, 2008] as an external (and costly) oracle to select a similarity function, assess embedding quality, and motif consistency. For more details on GED definition and implementation see Supplementary Section D.2.

4.4.1 Subgraph Comparisons and Embeddings Correlate with GED

We sample 200 rooted subgraphs of radius 1 and 2 uniformly at random from $\mathbb{G}$. We recall that the radius of a graph is the maximum length shortest path between any two nodes in this graph. Next, we compute all-to-all GED on this sample, yielding 20,000 non trivial values for each radius. We then compute similarities on the same set of subgraphs using various choices of s_G and ϕ. In **Supplementary Table D.1** we summarize the resulting Pearson correlation values.

Under these metrics, the best performing method performs a matching over ordered sets of smaller graphs known as graphlets, for more details see Supplementary Section D.3. It gets an almost perfect correlation at a radius of one and 0.52 at a radius of two. Since we consider fuzzy motifs to consist of graphs with slight variations, performance on similar graphs is more relevant. On pairs of graphs with low GED to each other, we obtain higher correlations of 0.637 on the radius-two subgraphs. Next, we train a 2-layers RGCN using this similarity function and obtain a thresholded correlation value of 0.74 with the GED values. Therefore, the dot product of our embeddings approximates structural similarities. Moreover, we note that the running time of a comparison becomes negligible, as it amounts to a dot product. To simplify downstream analysis, we take advantage of the strong correlation and use only 1-hop rooted subgraphs. Full results are available in **Supplementary Table D.1**.

We complete our report of the performance assessment of the embeddings with a visual representation of the results. In **Figure 4.3**, we generate a 2D projection of the local RNA structures from the learned embedding with t-SNE [Maaten and Hinton, 2008]. We draw example subgraphs corresponding to a sample of clusters.

Visually, we observe that similar subgraphs lie in the same clusters. Additional quantitative metrics are provided in the Supplementary Section D.5. This validation provides us the structural building blocks to assemble and retrieve motifs.

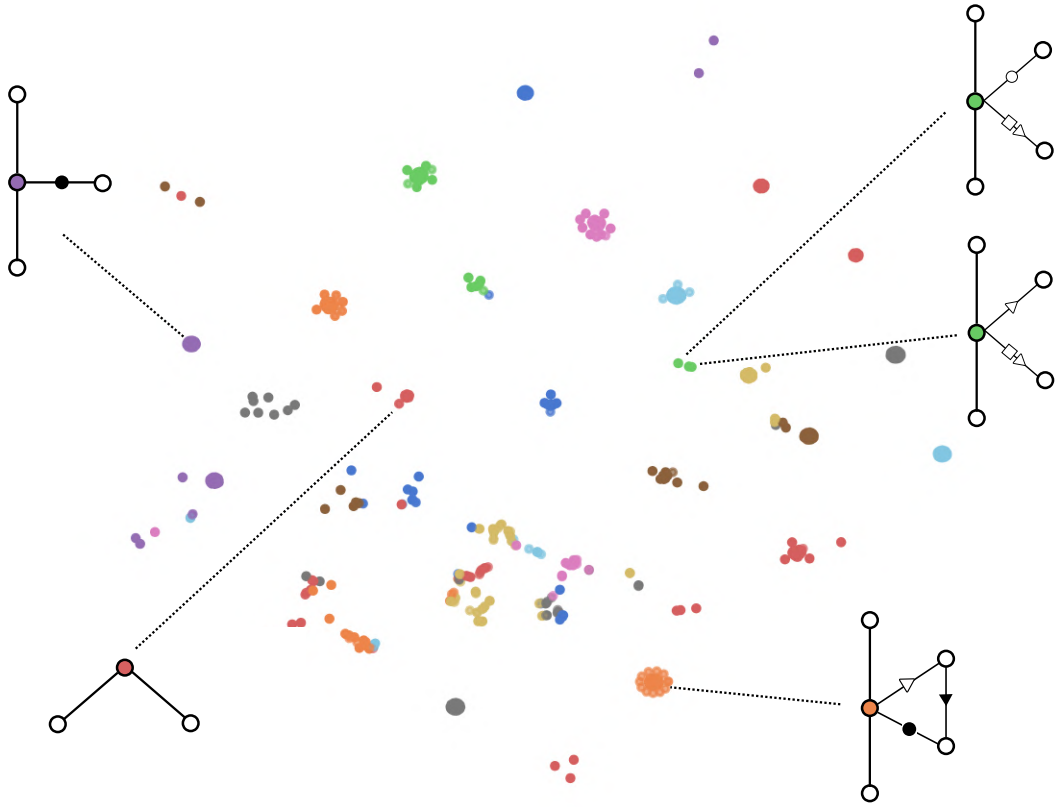


Figure 4.3 – t-SNE projection applied to embedding space. Drawn rooted subgraphs correspond to an example from the cluster connected by a dotted line. Point colors correspond to the nearest mixture model component.

4.4.2 Retrieval Algorithm Expands Known Motifs

Next, we turn to the validation of the retrieval algorithm. Given a query graph, the retrieve algorithm returns a list of subgraphs of $\mathbb{G}$, denoted as "hits", in decreasing order of compatibility to the query. We run the algorithm on our validation set of motifs (motifs identified by `RNA 3D Motif Atlas` [Petrov et al., 2013b], `rna3dmotif` [Djelloul, 2009], and `CaRNAval` [Reinharz et al., 2018a]), filtered for sparsity (more than 3 instances) and size (more than 4 nodes), resulting in 285 motifs. For a given known motif, we perform a retrieve with two types of queries: a true instance of the motif, and an instance of another randomly chosen motif (decoy). We show the resulting ranks in **Table 4.1**. The hit list contains a few good hits and a long tail of very small hits. We see that when queried with a true instance, the algorithm retrieves other instances in 97% of the cases, with the average rank in the top 5%. When queried with a decoy, the success rate drops to 83%, with an average rank at the 24th percentile of the hit list, indicating that only partial solutions were retrieved.

Method	Success Rate	Normalized rank
True query	97%	5,6%
Decoy query	83%	23%

Table 4.1 – Comparison of the performance of the retrieve algorithm when used with a query instance vs. a random one. Success rate denotes the rate at which the instance is in the hit list. The rank denotes the rank in the list, normalized by its length (lower is better).

We can go further by analyzing the structure of the retrieved hits. A first way to do so is to plot several hits with increasing ranks (**Figure 4.4**). A visual inspection of the results indicates that the retrieved graphs differ more and more as we plot hits with decreasing scores. A more quantitative way to do this is to compute the mean GED value of hits at fixed ranks compared to their respective queries. Since the motifs contain up to 15 nodes, the GED computation is not always exact and can yield high running times with incorrectly high values. We discard motifs that reached the GED timeout and end up with a set of 140 motifs, for which we present the mean GED with their hits in **Table 4.2**. For comparison, we also include the GED to a decoy corrected to match the size of the query. Sometimes, the best hits are not exactly isomorphic but happen in a more similar context than other isomorphic graphs, resulting in closer embeddings for the nodes at the border of the motif. This happens more frequently for larger graphs and explains why the GED is not a hard zero, but we see that it is very significantly shifted towards lower values. Based on both of these results we claim that our method is able to retrieve sets of subgraphs where the GED to the query correlates with the retrieval rank.

Rank	1^{st}	10^{th}	100^{th}	1000^{th}	Decoy
Mean GED	3.1 ± 0.3	3.9 ± 0.4	6.2 ± 0.6	9.2 ± 0.8	14.4 ± 0.8

Table 4.2 – Mean GED with standard errors between motifs queries and their hits at fixed ranks. We also included mean GED values to other random motifs as a control.

The average number of instances of a motif across `rna3dmotif`, `RNA 3D Motif Atlas`, and `CaRNAval` is only 22.3. Interestingly, the fact that we are able to obtain up to 100 hits with a

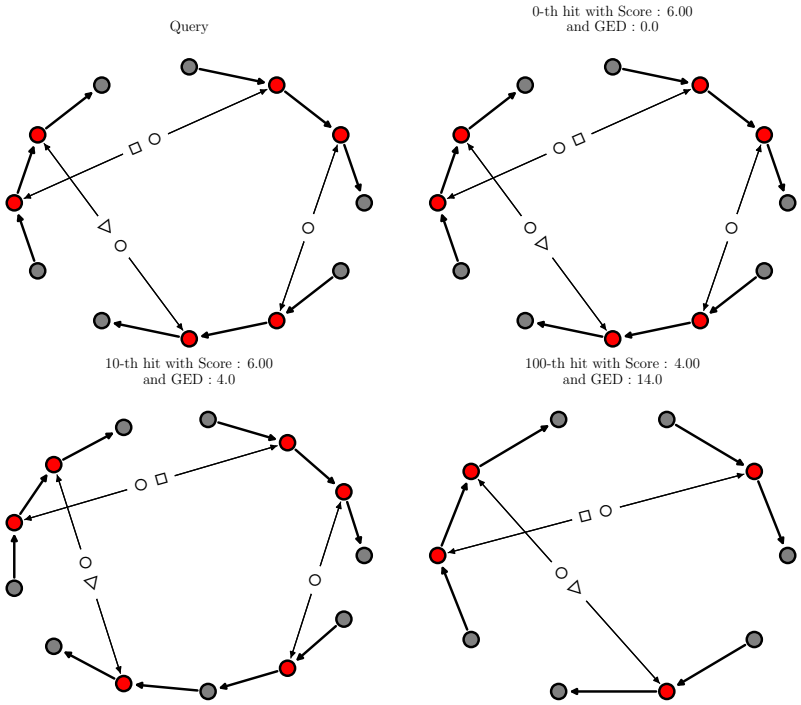
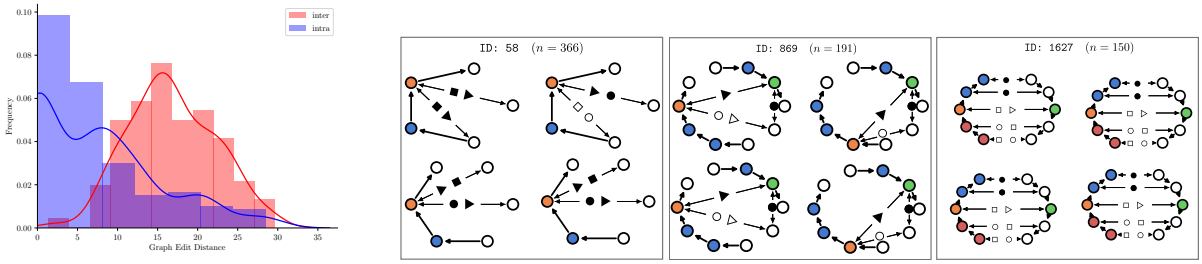


Figure 4.4 – Hit graphs with decreasing rank to the query. Red nodes indicate matches to the query.



a Distributions of GED for subgraphs sampled within the same motif (intra) and across motifs (inter).

b Four instances from three random VeRNAl motifs that did not overlap with external motifs. Each root node's color corresponds to its cluster ID.

Figure 4.5 – VeRNAl motif quality, as measured by GED, and sample novel motifs.

low GED indicates that many of these represent an ensemble of highly similar structures that are missed by existing tools. This observation suggests that our method can be used not only to assess if we find known instances of a motif, but also to identify fuzzy instances of these well known motifs.

4.4.3 MAA Identifies Novel Fuzzy Motifs

Finally, we assess the quality of the MAA procedure to identify *de novo* motifs. Of course, there are many choices of hyperparameters which are ultimately application-dependent (fuzziness, motif frequency, size, etc.). We require a minimum frequency of 100 instances per motif, as well as a maximum cluster spread of 0.4 in units of Euclidean distance. Finally, we remove a motif if more than 80% of its instances are included in a bigger motif, to enforce *maximality* of the retrieved results. We obtain a set of 3,496 motifs up to cardinality 7. **Supplementary Table D.4** shows the average number of instances and number of motifs at each cardinality.

To check for internal consistency, we compute the intra- and inter-motif GED between a random sample of 20 motifs and plot the results in **Figure 4.5a**. We obtain an intra-motif GED of 7.27 ± 7.63 and an inter-motif GED of 16.78 ± 5.42 when comparing motifs of the same size. This shows that VeRNAl finds motifs with internal consistency.

Next, we measure the degree to which our motif set agrees with existing motif databases. Since our approach is a generalization of RNA motifs (with $\gamma \leq 1$), we expect that known motifs would form a subset of VeRNAl motifs. Indeed, we find that a subset of our motifs aligns well (at least 60% overlap in number of nucleotides) with all databases, with a total of 82% of known motifs being found as VeRNAl motifs (detailed results in **Supplementary Table D.5**). In the heatmap of **Supplementary Figure D.5**, we plot the percentage of nodes of a known motif (`RNA 3D Motif Atlas` (alias: Atlas), `CaRNAval`, and `rna3dmotif`) that can be found in any of our motifs of the same size. Noticeably, our tool automatically retrieves motifs previously identified using several different constraints, methods and objectives. Indeed, `CaRNAval` focuses on motifs with long range interactions, while `Rna3Dmotif` targets motifs within loops. It shows that our framework unifies previous approaches but also expands their reach. Additionally, the VeRNAl motifs that match known motifs feature many more instances, again suggesting that we

are able to expand the set of known motif instances. Finally, despite no runtime being reported by `CaRNAval`, the authors indicated through personal communications that their analysis ran in about 300 hours. By contrast, VERNAl used about one hour of training (once) and a couple of minutes for running the clustering algorithm.

Since our motif set is significantly larger than those presented by existing methods, it is possible that the tool is uncovering novel motifs and exploring new regions of the structure space. A cursory examination of randomly sampled motifs reveals potentially interesting relationships between RNAs of varying functional roles. For example, motif `ID: 869` (drawn in **Figure 4.5b**) contains 191 occurrences, mostly from ribosomal RNA, but interestingly some instances fall in a viral ribozyme complex (PDBID: 1Y0Q). Similarly, motif `ID: 1627` features instances from ribosomes, as well as the FMN riboswitch (PDBID: 3F2X). An in-depth analysis of all individual instances is out of the scope of this contribution, but we plot some additional examples in **Supplementary Figure D.6**. Nonetheless, all the motifs identified by VERNAl can be browsed and downloaded on our web server `vernal.cs.mcgill.ca`, and are thus available to the community for further analysis.

4.5 Conclusions

We describe VERNAl, a novel pipeline for identifying fuzzy graph motifs. We develop various node structure comparison functions and approximate their feature map using an RGCN, embedding our graph dataset to a vector space for fast similarity computation between rooted subgraphs. We show that these computations correlate well with the RNA GED while being significantly faster. This enables us to find small structural building blocks of RNA and organize them into a meta-graph data structure.

Using this custom data structure, we introduce two algorithms to retrieve similar instances to a known query and to discover new motifs. We show that the retrieval procedure enables us to efficiently identify other instances of known motifs but also to find sets of subgraphs similar but not identical to a query. The motif extraction algorithm is also successful in mining sets of subgraphs with low intra-cluster GED, re-discovering and expanding known motifs as well as introducing new ones. All together, our platform VERNAl is the first tool to propose fuzzy graph motif extraction.

The nature of graph convolutions somewhat limits the type of motif that can be detected by VERNAl. Since RGCNs perform convolutions of entire neighborhoods around a node, motifs without a wide-enough conserved core can be lost, as information from outside the motif gets aggregated together with frequent nodes. Additional tuning of the similarity function radius, or more advanced message passing methods could address this limitation.

The main focus of this work is to build and validate the algorithm. Yet, a detailed exploration of the candidate motifs and the impact of the hyperparameters (fuzziness, density, size, etc.) is left for future work.

The algorithms introduced here are general and the field of subgraphs mining is still rapidly evolving. We believe VERNAl could also be applied to other sources of data such as chemical compounds, protein networks, and gene expression networks to automatically mine for novel generalized structural patterns.

Implementation

The source code is available at vernal.cs.mcgill.ca. We also provide a flexible interface and a user-friendly web server to browse and download our results.

Chapter 5

`RNAglib`: A python package for RNA 2.5D graphs

This chapter was made in collaboration with Carlos Oliver, Jonathan Broadbent, William L Hamilton and Jérôme Waldispühl and was published in Bioinformatics Application Notes in 2022 [Mallet et al., 2022b].

Contents

Abstract

RNA 3D architectures are stabilized by sophisticated networks of (non-canonical) base pair interactions, which can be conveniently encoded as multi-relational graphs and efficiently exploited by graph theoretical approaches and recent progress in machine learning techniques. `RNAglib` is a library that eases the use of this representation, by providing clean data, methods to load it in machine learning pipelines and graph-based deep learning models suited for this representation. `RNAglib` also offers other utilities to model RNA with 2.5D graphs, such as drawing tools, comparison functions or baseline performances on RNA applications.
The method is distributed as a pip package, `RNAglib`. The source code, data, and documentation is available at rnaglib.cs.mcgill.ca.

Résumé

Les structures tridimensionnelles de l'ARN sont stabilisées par des réseaux sophistiqués d'interactions (non-canoniques) entre paires de bases, qui peuvent se représenter comme des graphes multi-relationnels, à leur tour exploités efficacement par des méthodes issues de la théorie des graphes ainsi que de recents developpements en apprentissage machine. `RNAglib` est une librairie qui vise à faciliter l'adoption de cette représentation, en fournissant des données nettoyées, des méthodes pour les utiliser dans un pipeline d'apprentissage et des modèles d'apprentissage profond sur graphes, adaptés à cette représentation. `RNAglib` inclut aussi d'autres fonctionnalités utiles à cette représentation, comme des outils de dessin, des fonctions de comparaison ou des pipelines de bases pour des problèmes classiques sur l'ARN. Nous proposons l'outil comme un package pip, `RNAglib`. Le code source, la documentation et les données sont disponibles à l'adresse : rnaglib.cs.mcgill.ca.

5.1 Introduction

Recent developments in machine learning and deep learning techniques enable us to leverage the vast amount of biological data, such as sequencing data or biochemical assays, publicly released and organized by the community. This allowed breakthroughs in many areas, including the prediction of protein 3D structures with AlphaFold [Jumper et al., 2021].

These progresses allow the structural biology community to address new critical challenges, previously out of reach and far from the spotlight. This is particularly the case for RNAs. RNA is a highly structured molecule that supports many regulatory and enzymatic functions beyond its well-known messenger role [Fire et al., 1998; Makunin, 2006]. As such, it is a promising class of therapeutic drug targets [Crooke et al., 2018; Yu et al., 2020] as illustrated by novel treatments of CMV retinitus patients with AIDS [Hutcherson and Lanz, 2002] or the production of self-amplifying vaccines, which has recently seen a high-level of success in clinical trials for COVID-19 [Fuller and Berglund, 2020]. Because of the limited (but growing) amount of structural data available for RNAs, the task of designing robust machine learning methods to predict RNA 3D structures is more challenging than for proteins. However, RNA folding relies on a remarkable hierarchical organization of its structure. From our capacity to efficiently use this information will depend the success of machine learning applications. Representing objects as graphs is a strong prior knowledge and graph neural networks have shown to induce significant performance boosts in many applications.

To capture the tertiary structure of RNA in a computationally feasible manner, a growing number of algorithms make use of 2.5D graph networks - sometimes coined with the equivalent term Augmented Base-Pairing Networks (ABPN) [Djelloul and Denise, 2008; Oliver et al., 2020, 2022; Reinharz et al., 2018b; Sarrazin-Gendron et al., 2019]. These networks represent RNA molecules as topological graphs, whose nodes are nucleotides and whose edge types are structural categories of interactions between nucleotides. We have previously successfully combined the 2.5D graph representation with graph neural networks to predict small molecule binding [Oliver et al., 2020] and believe that their wider adoption is limited by the lack of dedicated software to use this representation, such as `rna-tools` [Magnus et al., 2020] for 3D representations. To our knowledge, the Python package `forgi` is the only effort in streamlining research on RNA networks. However, it focuses on coarse-grained models based on secondary structure elements instead of base-pair interactions and does not include machine learning features.

Contribution

We present a PyPi package, `RNAglib`, that aims to fill that gap by providing utilities to represent the structure of RNA as 2.5D graphs. These graphs are implemented as networkx [Hagberg et al., 2008] objects and an example of such a graph along with the different attributes it contains is shown in **Figure 5.1**. `RNAglib` provides clean data available for download along with loading, encoding and splitting routines. It also provides structural comparison functions that help unsupervised pre-training, and default models to learn RNA properties such as small-molecules or protein binding: we offer a benchmark performance for these tasks. Finally, `RNAglib` offers utility scripts to save, preprocess or plot graphs so that the manipulation of the data for research is facilitated.

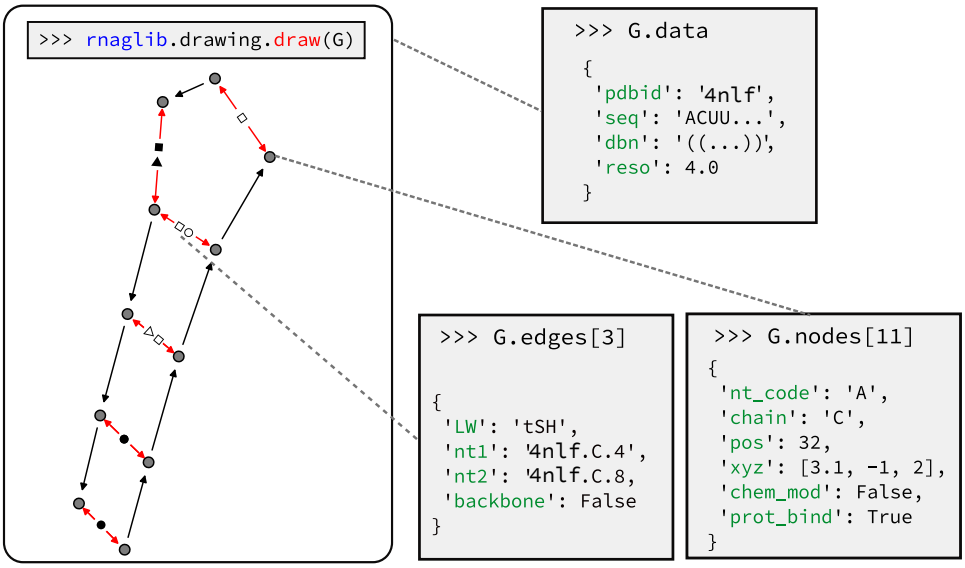


Figure 5.1 – 2.5D graph representation of the 23S ribosomal RNA Sarcin Ricin Loop (`PDBID: 4NLF`). Left panel shows the output of the drawing tool with default settings. Grey boxes are added later to depict a few of the attributes of the graph object, its nodes and its edges.

5.2 Data collection and graph processing

We collect and update (on a bi-monthly basis) a set of all RNA structures published in the RCSB Protein Data Bank (3739 graphs and 6,086,589 nucleotides) in the form of 2.5D graphs on our server. This procedure is detailed in Supplementary Section 5.6.1. These graphs contain information in each node (such as the nucleotide type or chemical modifications), edge (such as the Leontis-Westhof classification [Leontis and Westhof, 2001]) and also at the level of the whole graph, such as resolution. This data can be downloaded directly or through our python package.

Equipped with this data, the user can specify which node or edge features need to be included in the graph representation. The user can also choose a specific target for the machine learning algorithm, and we provide an automatic data splitting routine to test the trained models. Finally, pre-computations enabling fast subgraphs comparisons are run and also available for use for instance in unsupervised learning settings.

5.3 Machine Learning

In recent years, graph machine learning tasks are predominantly using graph neural networks. The first standardized pipeline we offer is the use of kernel functions for unsupervised machine learning [Hamilton et al., 2017a]. Unsupervised machine learning settings rely on functions which can compare data points and circumvent the need for annotated data during training. We have implemented several structural kernels introduced in [Oliver et al., 2022] (see Supplementary Section 5.6.2), along with dedicated data loaders to conduct the data processing specifically for this unsupervised task. `RNAglib` then enables the users to easily combine supervised learning and unsupervised phase, by adding a classification head (extra layers) on the top of the unsupervised

model. Both parts of this training scheme can be conducted simultaneously, with a loss for each component.

5.4 Utility functions

Along with these machine learning features, we also include several functions to facilitate the handling of RNA 2.5D graphs. For instance, `RNAglib` includes scripts to trim dangling nucleotides, perform statistics on the data or cut a big graph into smaller coherent ones. `RNAglib` also comes with an RNA Graph Edit Distance, the gold standard of graph comparisons, as well as drawing tools customized for 2.5D RNA graphs.

Because `RNAglib` comes with a principled way of loading and learning on graphs, we anticipate it can become a reference benchmark for RNA bioinformatics, but also for graph machine learning practitioners. We include a first baseline on predicting several node and edge-level properties, detailed in Supplementary Section 5.6.3

5.5 Conclusion

We present `RNAglib`, a set of tools to manipulate graph representations of RNA 3D structures, and use it to conduct machine learning and visualization tasks. We provide the graph machine learning community with a novel and challenging data set to develop and benchmark new methodologies. Not only is it solving a real world problem, it is also a data set whose core signal lies in the graph topology and the edge types, an original setting compared to current graph data sets. Simultaneously, we provide the structural RNA community an interface to use graphs and machine learning, hopefully helping this community to better solve RNA challenges.

5.6 Appendix

5.6.1 Data preparation

We collect and update (on a bi-monthly basis) a set of all PDB structures containing RNA on our server. We include a redundancy reduced subset of graphs based on a list of integrated functional elements curated by the RNA BGSU group (version 3.145; or latest) [Petrov et al., 2013a]. The full data contains 3739 graphs and 6,089,589 nucleotides, while the non redundant one contains 899 graphs with 210616 nucleotides. We then use the annotation software `x3dna-dssr` [Lu et al., 2015] to compute annotations for all structures. We complement these annotations by adding ion, small-molecule and protein binding using `BioPython` [Cock et al., 2009]. With this collection of annotations we construct annotated directed 2.5D graphs using the `networkx` package [Hagberg et al., 2008]. All of these graphs are stored on our servers. The successive releases are available for download. Finally, we provide tools for converting these graphs into the two most established frameworks for machine learning on graphs : PyTorch [Paszke et al., 2019] and DGL [Wang et al., 2019]. Descriptions of the data structure, attributes and examples of processing pipelines are detailed in the documentation.

5.6.2 Rooted Subgraph Kernels

To compare small rooted subgraphs is useful to produce embeddings in an unsupervised way. Indeed, we can train a neural network to approximate the implicit feature map induced by such a kernel. This strategy to obtain embeddings was validated in a general setting [Hamilton et al., 2017b] as well as in RNA [Oliver et al., 2020].

We have implemented the kernels described in [Oliver et al., 2020] and refer the reader to the paper for a detailed explanation. For convenience, we also include a shorter description here. We define kernels between pairs of rooted subgraphs, g_u and $g_{u'}$ (subgraphs with all nodes less than a fixed number of edges away from a node). We propose two main classes of kernels: ring-based, and matching-based kernels.

Ring-based kernels The first functions we consider are a weighted sum of a distance between l-hop neighborhoods, that we call rings. Rings are the set of edges at distance l from the root node u and are formally defined as $R_u^l = \{(u', w) : \delta(u, u') = l \quad \forall (u', w) \in E\}$. Let d be a normalized similarity function between two sets of edges. Let $0 < \lambda < 1$ be a decay factor to assign higher weight to rings closer to the root nodes, and N^{-1} be a normalization constant to ensure the function saturates at 1. Then we can obtain a structural similarity for the rooted subgraphs around u and v as : $k_L(u, v) := N^{-1} \sum_{l=0}^{L-1} \lambda^l d(R_u^l, R_v^l)$

The main degree of freedom is in the choice of a matching function d. The first kernel (`R_1`) simply computes the intersection over union score between the histograms f_R of edge labels found at each ring.

We have then used a matching algorithm to make a more subtle matching. Indeed, some geometrical relationships are more similar to others, a notion known as isostericity [Stombaugh et al., 2009]. We rely on the Hungarian [Kuhn and Yaw, 1955] algorithm to optimally match the edges of two rings being compared, which yields the second kernel: (`R_iso`).

Matching-based kernels The ring based decomposition encodes the prior that the surrounding of a node can be compared only in concentric rings. However, in the case of a bulge for instance, we might want to remove this constraint. To compute the `hungarian` kernel, we compute a hungarian matching over all nodes in the rooted subgraph, but we add in this comparison a cost for matching nodes that are at a different distance from the root node.

Graphlet kernels Since the degree of our graphs is strongly bounded (max degree 5), we can define a graphlet as a rooted subgraph of radius 1 and obtain a manageable number of possible graphlets. We can then replace the edges we previously compared in the methods above with graphlets. To compare these graphlets, we rely on GED computations. While the GED computation is tractable for such small graphs, it is still expensive when repeated many times. For this reason, we implement a solution caching strategy which stores the computed GED when it sees a new pair of graphlets, and looks up stored solutions when it recognizes a previously seen pair.

When replacing edges with graphlets in the ring-based kernel, we get the *R_graphlet* method, while when doing so in the matching method, we get the *graphlets* method.

5.6.3 Benchmark preliminary results

Several properties of RNA molecules can be predicted from their structure as well as their 2.5D representation. This is the main function of `RNAglib` and we have built some example pipelines on how to select the relevant nucleotide feature and use them to predict several interesting properties. Because it comes with a principled way of loading the data and using it, we anticipate it could also become a benchmark for methodological contributions on RNA function prediction.

The first two tasks we include as example are the binding of RNA with a protein partner or a small molecule. To get the data for these two tasks, we use the 'SNAP' mode of `x3dna-dssr` with default settings to extract interface information. For the protein interactions, this results in 108821 nodes interacting with a protein in 641 graphs. For the small molecule binding, this results in 722 nodes interacting with a small molecule in 151 graphs. Chemical modification is a good indicator for solvent accessibility. The default output of `x3dna-dssr` includes chemical modification boolean values for every residue. We get 1417 modified nodes in 247 graphs. All of these figures are computed on the non-redundant set, but this could be changed, for instance to include interaction of a redundant pocket with several small molecules partners. We then frame these three tasks as a binary node-level prediction (binds or does not bind), and we load a certain fraction (0.2) of decoy graphs that do not contain positive nodes. Finally, the link prediction is an edge-level task, where the network is asked to predict whether two nodes are indeed connected in the graph. We have re-used the train/test split for protein binding here as we are using the core structure of the graphs so there is no easy domain splitting.

The results presented in this section result from training a single embedding network with two layers of size 64 and the 'R_graphlets' kernel in an unsupervised way. We then used this embedding network for all downstream tasks. For the node-level tasks, we simply add an extra convolutional classification layer to map these embeddings to node classification prediction. The resulting network (concatenation of pretrained embedder and classification head) was fine-tuned independently for each task. For the link prediction we use the pretrained embeddings to create a pairwise dot product of the node embeddings. We then use true and negative edges and fit the dot product values to the presence or absence of an edge. Because all of our predictions output a binary prediction, we use the Area under the Receiving Operator Curve as a metric. This procedure is illustrated in the files : `examples/second_example.py` and `examples/third_example.py` of our git repository. Our results are presented in Table 5.1. We intend to keep a leaderboard of the performances on these tasks to stimulate the methodological research on RNA 2.5D graphs. If you manage to beat the state-of-the-art on one of those tasks using our splitting procedure and with a reproducible script, we will reference your work and update the leaderboard.

Task :	Protein Binding	Small-molecule Binding	Chemical Modification	Link Prediction
AuROC	0.63	0.60	0.75	0.93

Table 5.1 – Area under the Receiving Operator Curve of a baseline machine learning model on the different tasks of the benchmark. These should be considered as a starting point by practitioners

Chapter 6

InDeep: 3D fully convolutional neural networks to assist in silico drug design on protein-protein interactions

This chapter was made in collaboration with Luis Checa Ruano, Alexandra Moine Franel, Michael Nilges, Karen Druart, Guillaume Bouvier and Olivier Sperandio and was published in Bioinformatics, 2022 [Mallet et al., 2022a].

Contents

Abstract

Motivation: Protein-protein interactions (PPIs) are key elements in numerous biological pathways and the subject of a growing number of drug discovery projects including against infectious diseases. Designing drugs on PPI targets remains a difficult task and requires extensive efforts to qualify a given interaction as an eligible target. To this end, besides the evident need to determine the role of PPIs in disease-associated pathways and their experimental characterization as therapeutics targets, prediction of their capacity to be bound by other protein partners or modulated by future drugs is of primary importance.
Results: We present `InDeep`, a tool for predicting functional binding sites within proteins that could either host protein epitopes or future drugs. Leveraging deep learning on a curated data set of PPIs, this tool can proceed to enhanced functional binding site predictions either on experimental structures or along molecular dynamics trajectories. The benchmark of `InDeep` demonstrates that our tool outperforms state of the art ligandable binding sites predictors when assessing PPI targets but also conventional targets. This offers new opportunities to assist drug design projects on PPIs by identifying pertinent binding pockets at or in the vicinity of PPI interfaces.
Availability: The tool is available on GitLab at :
gitlab.pasteur.fr/InDeep/InDeep.

Résumé

Motivation : Les Intéractions Protéine-Protéine (IPP) représentent des composantes capitales pour de nombreuses voies biologiques et sont la cible de nombreux projets de découverte médicamenteuse par exemple contre des maladies infectieuses. La conception de médicaments à cibles IPP reste un problème délicat et nécessite un travail minutieux pour qualifier une interaction comme cible potentielle. Au-delà du besoin d'identifier les IPP qui interviennent dans les voies de chaque maladie et leur caractérisation expérimentale comme cibles thérapeutiques, il est primordial de pouvoir prévoir leur capacité à se lier à d'autres protéines ou à d'éventuels modulateurs.
Résultats : Nous proposons l'outil `InDeep` pour prédire sur une protéine les sites d'interactions fonctionnels avec des épitopes protéiques ou des médicaments. En exploitant une base de donnée spécifique avec des méthodes d'apprentissage profond, cet outil améliore la prédiction de tels sites sur des structures statiques ou le long d'une trajectoire de dynamique moléculaire. A travers un benchmark contre des méthodes existantes, nous montrons qu'`InDeep` améliore l'état de l'art sur les prédictions de site de liaison sur les cibles IPP mais aussi sur les cibles traditionnelles. Notre outil ouvre donc la porte à de nouveaux projets de découverte médicamenteuse sur IPP, en identifiant les poches de liaison dans ou à proximité des interfaces IPP.
Disponibilité : L'outil est disponible sur GitLab à l'adresse :
gitlab.pasteur.fr/InDeep/InDeep.

6.1 Introduction

Protein-protein interactions as therapeutic targets. Protein-protein interactions are central elements in numerous biological pathways. They represent increasing interests as therapeutic targets, with a growing number of published studies describing the successful modulation of PPIs using small molecules [Torchet et al., 2021]. Yet, identifying chemical probes or drugs on PPIs remains a difficult task. As opposed to more conventional drug discovery targets, such as G-protein coupled receptors (GPCRs) or enzymes and more recently protein kinases, PPIs have not evolved to bind small molecules. Therefore, the proof of their ligandability has to be made on a case by case scenario [Sperandio et al., 2010]. Indeed, the design of small molecules binding orthosterically at the interface to prevent protein interactions is not achievable for all PPIs [Lu et al., 2020]. Given their number and heterogeneity of structures, it is therefore of primary importance to have powerful tools to efficiently evaluate the feasibility of considering PPIs as targets in complement of the unavoidable biological evaluations.

In the situation of designing orthosteric inhibitors of PPIs (**iPPIs**) using small molecules, strategies like epitope mimetics can be envisaged [Ashkenazi et al., 2017]. This was successfully made against the B-cell lymphoma-2 (Bcl-2) family to combat chronic lymphocytic leukaemia [D'Aguanno and Del Bufalo, 2020]. This led to the development of Venetoclax which was approved by the FDA in 2016 as the first orthosteric PPI drug. The design of iPPIs implies to evaluate two complementary features within the interface: 1) the knowledge of an epitope binding at the interface and the presence of hotspot residues that carry out most of the binding energy of interaction [Clackson and Wells, 1995], and 2) the existence of a ligandable binding site around these hotspots that could host a small molecule.

Predicting and profiling epitope binding sites.

Several *in silico* tools can predict hot spot residues within PPIs [Krüger and Gohlke, 2010; Tuncbag et al., 2010] but they necessitate the structure of a complex and the fore knowledge of an identified protein partner. To predict protein interactions, some tools directly use the sequence information [Murakami and Mizuguchi, 2010] or evolutionary data [Cong et al., 2019]. However, leveraging the structure of the protein has shown to drastically increase performance of prediction of interface regions. Moreover, structural motifs and local arrangements of atoms can be highly conserved even across different secondary structures and different global protein folding. These local motifs are hypothesized to be the key element of partner binding. This has motivated using convolutional strategies to encode such local information about binding sites. Some of these methods focus on predicting the interaction patch on the protein: i.e. determine which residues are involved in an interaction [Dai and Bailey-Kellogg, 2021; Gainza et al., 2020]. Some methods also take as input the partner to predict the interaction patch [Dai and Bailey-Kellogg, 2021], deemed as partner-specific predictions. In that case, the prediction can be more fine-grained and also provide contact prediction: which residue interacts with which other [Sanchez-Garcia et al., 2019; Townshend et al., 2019]. All of these tools annotate the sequence by predicting residue-level information. However, knowing not only which residues are involved in the binding (sequence or surface derived info) but which types of partner residues and where they bind in the vicinity of the protein surface is highly desirable to understand the mechanisms of epitope binding or the design of future drugs mimicking these epitopes.

Predicting ligandable binding sites within PPIs. A plethora of tools is now available to predict binding sites and binding site ligandability. One can cite historical and efficient geomet-

ric based methods such as Fpocket [Guilloux et al., 2009], VolSite [Da Silva et al., 2018] and mkgridXf [Monet et al., 2019], fragment-based methods like FTMap [Kozakov et al., 2015] and more recent and powerful methods using deep learning such as DeepSite [Jiménez et al., 2017], P2rank [Krivák and Hoksza, 2018], Kalasanty [Stepniewska-Dziubinska et al., 2020], DeepSurf [Mylonas et al., 2021], or OctSurf [Liu et al., 2021], with Kalasanty being a reference in the field. These methods have demonstrated their predictive capacity to identify ligandable binding sites on conventional drug targets. Although FTMap was the first method to raise the question of PPI ligandability without really providing a ligandability score, none of the methods cited above is specific to the ligandability of PPIs. Nonetheless, interfaces of PPIs are historically described as rather flat, large and devoid from deep binding pockets [Arkin et al., 2014]. It is therefore most legitimate to anticipate a specific form of ligandability in the case of PPIs. There are numerous examples of co-crystallized orthosteric iPPIs [Torchet et al., 2021]. Using machine learning and PPI-specific data sets, we can expect to address the specificity of PPIs ligandability.

Capturing holo-likeness along MD trajectories. PPIs are known to undertake important conformational changes depending on their binding state: apo, holo with ligand or a protein partner. These conformational changes affect the shape and binding capacity of interface binding pockets [Johnson and Karanicolas, 2013]. It is therefore of primary importance to take these conformational changes into account when profiling epitope and ligandable binding sites [Kozakov et al., 2015] as those will condition binding to partners. This represents a major challenge when attempting to identify chemical probes, using *in silico* methods such as virtual screening or designing epitope mimetics, in the absence of the partner bound. Indeed, it is for example key for such methods to sample and identify so-called holo-like conformations prior to virtual screening in the context of ensemble docking [Amaro et al., 2018; Ivetac and McCammon, 2012]. Previous works have addressed the holo-like sampling challenge using MD simulations with methanol solvent. The use of a less polar solvent than water favors the opening of transient hydrophobic pockets, resulting in an improvement of docking results [Eyrisch et al., 2012]. Other methods have already been developed to monitor ligandability along molecular dynamics trajectories using geometric [Guilloux et al., 2009], or Deep Learning approaches [Kozlovskii and Popov, 2020], although none of these are specific to PPIs. Moreover, no method is available to monitor interactability patches and epitope binding sites along molecular dynamics trajectories.

Contribution. Our work builds upon our last release of iPPI-DB [Torchet et al., 2021] and of its new target-centric mode, and aims to facilitate the identification of iPPIs. Our tool `InDeep` has capitalized on iPPI-DB structural data to train predictive models relying on neural networks with a 3D fully convolutional U-Net architecture. It is a unified multi-tasking prediction tool that uses the 3D structure of proteins to predict ligandable binding for iPPIs and so-called interactability patches for epitope binding. We show that `InDeep` outperforms the state of the art of binding pockets detection methods and that our tool is especially efficient to detect iPPI and epitope binding sites. While remaining competitive on annotating the protein sequence with interactability, it also predicts the spatial location of its putative partner. Our tool also enables tracking of these druggability and interactability scores for a given detected pocket along molecular dynamics trajectories. It is integrated in a PyMol [DeLano et al., 2002] plugin (see Supplemental - Section E.7) for easy visualization of the predictions, making it a real toolbox for iPPI drug design. It is freely available at `https://gitlab.pasteur.fr/InDeep/InDeep`. Finally, the results of InDeep predictions (before post-processing) can be consulted on the iPPI-DB website for every heterodimer and iPPI-bound protein in the database at `https://ippidb.`

pasteur.fr/targetcentric/.

6.2 Methods

6.2.1 Data curation, splitting and representation

For training and assessing the models, we have used the dataset available in the iPPI-DB [Torchet et al., 2021]. The dataset relies on two subsets: one contains Hetero-Dimeric complexes (HD interactions) and the other iPPI-bound protein complexes (PL interactions) where ligands bind one of the two partners at the interface within a HD complex (orthosteric inhibitors). We have then split this dataset based on CATH [Sillitoe et al., 2021] folds to avoid any structural overlap between our train, validation and test data sets.

Equipped with these sets of co-crystallized proteins with either a protein or ligand partner, we wish to represent them in a way a neural network can learn on. We follow the volumetric CNN framework used, for instance, in [Jiménez et al., 2017; Stepniewska-Dziubinska et al., 2020]. This framework consists in treating the interaction sites as 3D images, whose color channels are functional atom types. We introduce five functional atom types: α-carbon (Cα), donor and receptor of h-bonds and positively and negatively charged atoms and hydrophobic/aromatic atoms. We then put a Gaussian function around each atom center based on its type, and interpolate its values on a regular 3D grid with 1A spacing. The details and results of these procedures are described in Supplemental (Section E.1).

6.2.2 Model architecture and learning

We now want to build a model that takes a protein structure as input and predicts iPPI ligandability (PL interaction) as well as protein interactability (HD interaction). These two machine learning tasks can benefit from the concept of multitasking, a well-documented phenomenon that means that one hybrid machine learning model that solves two tasks usually performs better than two separate ones [Goodfellow et al., 2016]. Indeed, in the multitasking setting, each task benefits from the representations learned using the other task's supervision. After several shared layers, our network is split into a PL and an HD branch. These branches are two independent sequences of layers with a sigmoid and softmax activations for PL and HD respectively. We use a U-Net [Ronneberger et al., 2015] architecture for our prediction. Our model does not use fixed-size linear layers (fully-convolutionnal network) which enables it to take any grid size as input. A visual representation of this branching scheme is available in Figure 6.1 and a detailed description of the model is available in the Supplementary Section E.2.

We then train the resulting models with batches containing a mix of PL and HD data points. For memory limitations issues, we used the accumulated gradient trick to use batches of size greater than one. Since we encode the 3D structure into voxels, the resulting representation is sparse. Moreover, the key part of the prediction lies in the space surrounding the protein surface and especially around the position of the true ligand, while having a hard zero inside the protein or very far from its surface is less relevant. To account for these two points, we use the weighted versions of cross entropy (CE) and binary cross entropy as the loss $\mathcal{L}$ to train our network. All voxels receive a small weight $\mathbf{w}^{\text{background}}$ of 0.05, then the voxels closer than 6 Å from the surface receive an additional weight $\mathbf{w}^{\text{surface}}$ of 0.35 and finally the voxels corresponding to the target

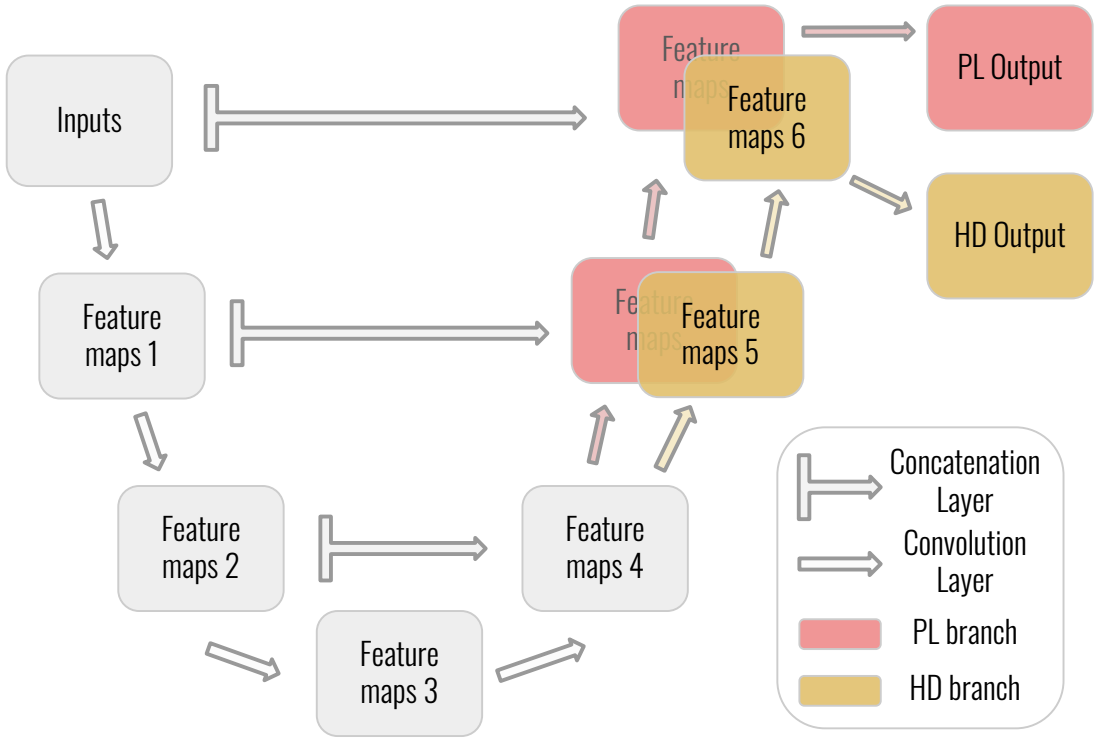


Figure 6.1 – Visual representation of InDeep's architecture.

voxels get an additional value $\mathbf{w}^{\text{partner}}$ of 1. We found this weighting scheme to stabilize the learning and optimized our network with an Adam [Kingma and Ba, 2014] optimizer.

$$\mathcal{L}(\hat{y}, y) = \sum_{(i,j,k)\in \text{Grid}} (\mathbf{w}^{\text{background}}_{i,j,k} + \mathbf{w}^{\text{surface}}_{i,j,k} + \mathbf{w}^{\text{partner}}_{i,j,k}) \text{CE}_{i,j,k}(\hat{y}, y).$$

6.2.3 Post-processing and optimization

Once equipped with scalar fields prediction, we need to segment them in contiguous regions of high values. Other approaches have simply used a mean-shift algorithm [Jiménez et al., 2017; Stepniewska-Dziubinska et al., 2020], but such algorithms can split the prediction, or discard important neighboring parts of the prediction. To address this segmenting problem, we have relied on the watershed [Beucher, 1979] algorithm. The watershed algorithm finds all basins around local minima. We build a graph whose nodes are these basins and whose edges contain the euclidean distance as well as a normalized value of the lowest saddle point joining two neighboring basins. Then, we merge the neighboring nodes in a greedy manner, by prioritizing the ones with the smallest edge. We stop the merging process when the merged nodes exceed a geometric distance threshold of 15 Å and 20 Å for PL and HD respectively. Each of the resulting group of basins is denoted as a predicted pocket and scored based on the mean values of its best 150 voxels. Finally, we filter these predicted pockets to remove the smallest or least high-scoring ones, yielding the final list of predictions.

To choose the optimal values for the hyperparameters of the network and the post-processing (number of layers, number of neurons per layer, thresholds), we conducted an hyper-parameter

optimization (HPO). The HPO metric and optimization procedure is described in detail in the Supplemental (Section E.3). We then ran the HPO for approximately 100 experiments on the validation set (Supplementary Figure E.2), which gave us our final predictive model.

6.2.4 Molecular Dynamics Trajectory Analysis

To distinguish the conformations with higher ligandability (`PL`) or interactability (`HD`) propensity, the developed model can be used on an ensemble of protein conformations generated by molecular dynamics (MD). To do so, each snapshot of a molecular dynamics simulation is treated as an input for `InDeep`. Therefore, one can specify some residues to `InDeep` along which to monitor the ligandability or interactability, resulting in a reduced grid compared to an inference on the whole protein. Then, the post-processing is simplified because the prediction size is reduced: a spatial anchor is chosen either by the user or as the point of the grid in the solvent closest to the grid center. Finally, we simply grow a volume around this anchor following a greedy nearest neighbor policy. The average value of the voxels in this volume represents a ligandability or interactability score that can be easily tracked along the MD time steps. We chose a volume of 150 $Å^3$, close to the cutoff of 100 $Å^3$ used in the study by [Gao and Skolnick, 2013] which considers that 80% of pockets occupied by ligands are encompassed by this cutoff.

6.3 Results

6.3.1 Prediction of ligandable binding sites

Benchmark of InDeep on conventional target binding sites

Rosell and Fernández-Recio have introduced a method to detect iPPI binding sites that uses FPocket [Guilloux et al., 2009] along a MD trajectory to detect transient binding sites. These binding sites are then selected to be nearby potential interfaces based on protein-protein docking results. However, the docking step requires the structure of the partner, while our method just uses the structure of the protein meant to be bound. Moreover, these steps require substantial computing time, making it less suitable for the investigation of several structures. There are several tools that aim to predict small molecule binding sites, among which Kalasanty is a reference in the field. In the absence of other iPPI dedicated tools for ligandability, we benchmarked `InDeep` against Kalasanty. The other major difference between our tool and Kalasanty, is that Kalasanty was trained on VolSite predicted cavities in ligand locations, whereas our model has been trained on ligand position directly. For fairness, we compare the ability of our tool to predict VolSite predicted cavities as well as the ligand location with Kalasanty.

We use the same dataset as the authors of Kalasanty did for validation: a distinct dataset, made by Chen *et al* [Chen et al., 2011]. The original test set is composed of 111 protein-ligand holo structures and 104 corresponding apo structures. We have filtered out a few systems that were too similar to our training set according to the TM-score metric [Zhang and Skolnick, 2004] and end up with 187 and 196 pockets evaluation for apo and holo structures respectively. Ligands coordinates were extracted from the holo structures of the Chen benchmark and VolSite was used to describe cavities for each ligand, as shown in Supplementary Figure E.3. Following Kalasanty, the number of retained predicted pockets is the number of small molecules present in the deposited PDB system.

We then used the metrics used by Kalasanty on their dataset: DCC and DVO. The DCC metric computes the distance in Angstrom between the center of mass of the predicted pocket and the one of the ground truth. We denote a prediction as successful when its DCC is below a distance threshold, and we plot the success rate at different thresholds of the different methods. We compare predictions of `InDeep` compared to the ones of Kalasanty, for the bound conformation (holo) as well as the unbound one (apo). The DCC values were computed in 3 conditions: between the computational predictions and (i) the VolSite cavities, (ii) the ligand positions and (iii) the ligand positions that have a VolSite cavity associated with them (top of Figure 6.2). The DVO metric is only computed on successful prediction at 6 Å and consists in the volume of the overlap over the volume of the union. This procedure is illustrated in Supplementary Figure E.3 and the DVO results are presented at the top of Supplementary Figure E.4.

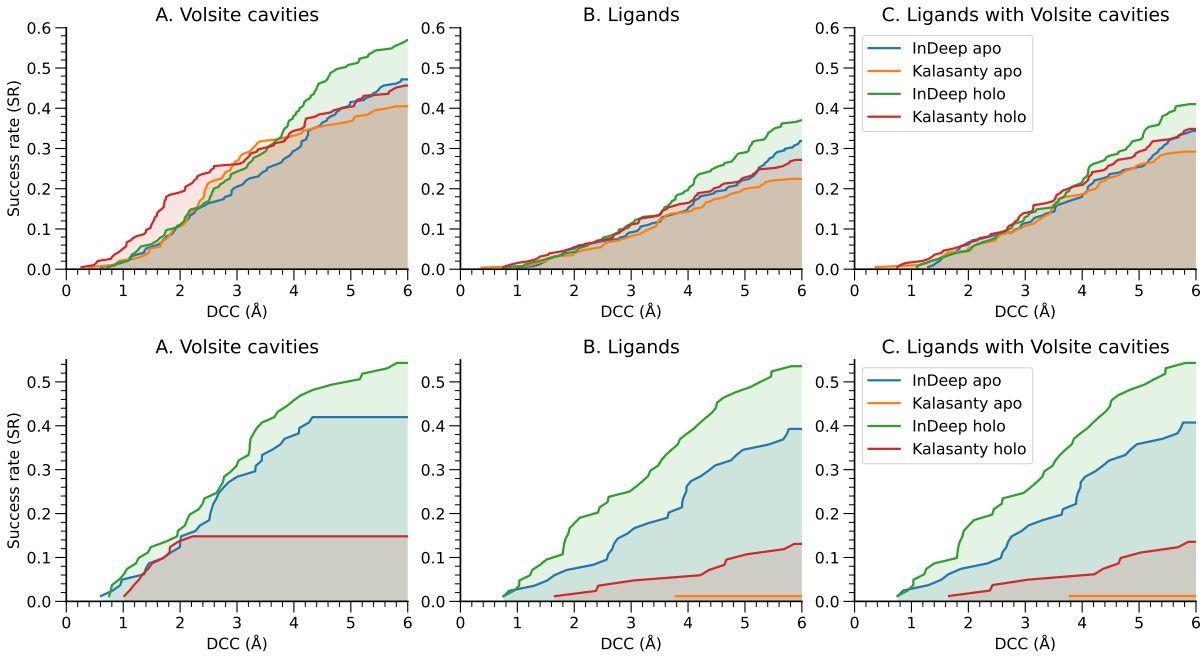


Figure 6.2 – *Top:* DCC evaluation for `InDeep` and Kalasanty on the Chen benchmark. We plot the Success Rate (SR), the fraction of systems for which we have a DCC value below a threshold, for different thresholds. *Bottom:* Performance on the test set filtered by TM-score. The plots are produced following the same procedure as the ones above on this new data set. The metrics are computed with the VolSite cavities associated with the ligand position given by the PDB (**A.**), the ligand position itself (**B.**) and the ligand position having a cavity detected by VolSite (**C.**).

First of all, we reproduce the results claimed by Kalasanty. In this benchmark dataset, `InDeep` outperforms Kalasanty on all settings except in the very low DCC range and for VolSite cavities only. At a 6 Å threshold, we have an average relative performance boost of 27% . We see that this difference is most important in the ligand setting that resembles our training procedure, but that the performance remains stable on the cavities setting. We also see that this boost of performance is observed across holo and apo predictions. We detect approximately 40% of the binding sites in the best ranked predictions. We then compute the DVO values for

the different methods among the successful predictions at 6 Å. We obtain comparable values of DVO overall. Because Volsite cavities tend to be deeper into the pocket, Kalasanty predictions tend to be at the bottom of binding sites. This explains why the average DVO value for cavities is better for Kalasanty and worse for `InDeep` (0.612 vs 0.503), and vice-versa for ligands (0.257 vs 0.320). Moreover, one should note that the total populations are not the same because the DVO is computed only on the 'successful predictions' with low DCC values. Therefore, the better performance in terms of DCC can hinder the DVO distribution performance.

Finally, we further tested InDeep on the complete sc-PDB, an annotated database of druggable binding sites from the protein data bank. We computed the DCC values to compare InDeep predictions with the Volsite cavities stored in the sc-PDB and the position of the ligand. At 6 Å we found a success rate value of 0.71 and 0.68 for the Volsite cavities and the ligand positions respectively. The full results are available in Supplementary Figure E.5. Overall, `InDeep` yields state of the art performance on the binding site prediction for general ligands, even in the unfavorable setting of predicting cavities, as opposed to actual ligand localization.

Benchmark of InDeep on iPPI binding sites

We have then repeated the same data extraction pipeline as used for the Chen data set and described in Supplementary Figure E.3 on our test set. We have applied the same TM-score filtering between the train set of Kalasanty and our test set to avoid data leakage. This represents a more suitable application of InDeep, as it was designed to identify PPI specific binding pockets. We also note that this procedure results in few (81) systems, because of the large overlap with the training set of Kalasanty. We present the DCC results in Figure 6.2 (bottom) and those for DVO in Supplementary Figure E.4 (bottom).

Despite the limited size of this filtered data set, we see that `InDeep` clearly outperforms Kalasanty. The SR is increased 5-fold and the DVO values remain reasonable. Moreover, we retain 80% of the performance when predicting on the apo form of the protein, an important feature for these binding sites that are known to be hard to detect. This shows that `InDeep` is not only a good predictor for conventional target binding sites, but that it is much more efficient than existing methods for iPPI binding site detection.

6.3.2 Predicting and profiling epitope binding sites.

Benchmark of InDeep on PPI data sets

As for `PL`, several tools exist that predict which region of a protein interact with another. Once again, we choose to compare against the state of the art and reproduce the results of PInet [Dai and Bailey-Kellogg, 2021]. We use two benchmarking data sets they propose: DBD5 [Guest et al., 2021] and EpiPred [Krawczyk et al., 2014]. DBD5 is a protein docking benchmark that offers several pairs of structures of interacting proteins. This dataset is split into a train and test set by PInet. EpiPred is a dataset centered around interactions between antigens and antibodies. For fairness, we used their protocol to annotate the data as in PInet and have rerun their method. Finally, we have used their tool in the partner-specific setting (giving the partner as input) and in a blind setting that is closer to our use case.

We then had to slightly adapt our validation pipeline. Indeed, we have found no study trying to predict the actual location of the partner in the vicinity of the protein surface. The tools

we compare against always project the predicted interactability onto the surface residues of the protein or the sequence. To project our 3D prediction onto the sequence, we use a convolution with a Gaussian kernel between our 3D prediction and the coordinates of the atoms of the protein.

We compute the Area under the Precision-Recall curve for the DBD5 test set and EpiPred. On the DBD5 test set, InDeep is widely outperformed by PInet. However, PInet was trained on a different split of the same data set, so we turned to EpiPred, which was not used for training either methods. PInet gets a value of 0.235 and 0.217 in the native and blind setting, respectively. On this data set, we get a value of 0.232, close to the performance of the partner-specific setting. We achieve a state of the art performance in our blind setting. Moreover, it should be noted that InDeep performance suffers from the extra step of sequence projection. Overall, this shows that our tool is able to accurately predict the interaction sites of a protein.

Localization of interacting partner

We complement this comparison to other tools with a validation of InDeep with metrics closer to the PL validation. We compute these metrics on our test set as well as on DBD5 and EpiPred. We note that to our knowledge, no tool exists that outputs 3D prediction of the volume occupied by a putative protein partner. However, this prediction is of great use to assess if the 3D prediction for a small molecule binding would collide into its corresponding protein partner, opening new doors for therapeutic design of iPPIs. Since for PPIs, the number of observed partners is just one, we have computed DCC and DVO values for one, three and all predicted binding sites. Because the interfaces are bigger, we also present our results up to a distance threshold of 10A. The results are presented on Figure 6.3.

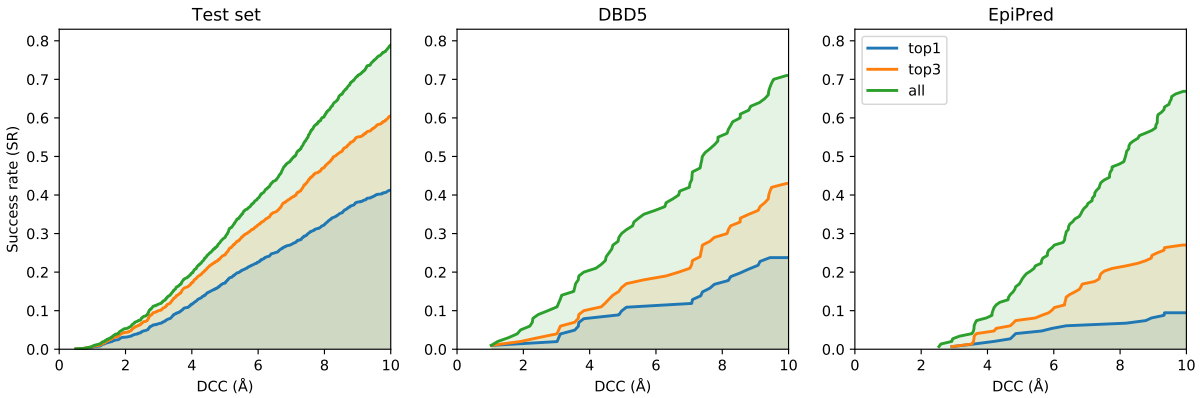


Figure 6.3 – DCC values for InDeep on the test set (*Left*), DBD5 (*Center*) and EpiPred (*Right*) data sets, when considering the best, top-3 or all pockets predicted.

At the 10 Å threshold, we have success rates of 42%, 24% and 11% with only the first prediction, and of 78%, 72% and 69% using all of them for our test set, DBD5 and EpiPred respectively. This means that our method finds the correct binding site in about 70% of the cases, but that a significant amount of times, the correct predicted volume is not ranked as the first one. This can be partially imputed to the fact that a given protein can have several partners, so the first prediction might actually be a correct one that does not correspond to the

partner at hand. However, `InDeep`'s performance on the top-3 falls in between the performance of the top-1 and keeping all pockets, which indicates that the correct binding sites is often among the best scored positions, proving once again the relevance of the tool for epitope binding site prediction.

Epitope binding site prediction: atom-typed channel validation

We now turn to the channel evaluation. We have used five channels to encode our protein environment and our prediction: α-carbon ('CA'), donor and receptor of h-bonds ('HAD'), positively and negatively charged atoms ('POS' and 'NEG') and hydrophobic/aromatic atoms ('COB').

This means that beyond prediction of the presence or absence of a protein partner, the model also predicts which protein atom type should be at a given voxel. This is close to the idea developed by LigVoxel [Skalic et al., 2019c] but for the interactability model. However, finding a quantitative metric to describe the quality of these channels is not easy. Indeed, the target now contains several little volumes (each atom type environment) that can be split across the protein partner interface. Therefore, we can not use the DCC metric easily, because the centers of mass of split volumes do not represent our objects accurately. Moreover, we cannot easily interpret the DVO values, as previous experiments only plotted the DVO for successful DCCs, which we do not have anymore. This is even more true if we consider the large size of the interface, which explains why we cannot use the same validation procedure as LigVoxel.

We have turned to a more direct method for assessing the performance. At each voxel of our prediction, we have a distribution of probability for each channel. We can aggregate these voxel distributions for all voxels around an atom of the ligand to obtain a mean distribution of channel probabilities. We also compute an atom-type specific distribution by aggregating only the voxels around atoms of each specific channel. We plot a heatmap representing the Z-scores of the observed channels distributions compared to the overall ones. We expect to see enhanced values on the diagonal and decreased ones off the diagonal. The results are presented in Supplementary Figure E.6.

We see that the hydrophobic channel (COB) performs well at localizing hydrophobic patches of protein partners (Z-Score = 1.6). It is an important result as transient protein-protein interactions, that represents most of the known PPI targets, are often mediated by hydrophobic patches at the interface whose seclusion from the solvent upon binding helps to regulate protein association. This COB channel can therefore be used as a way to suggest point mutations at the interface when dealing with hydrophobic interaction, or in the context of epitope mimicking or peptide design. The backbone channel (CA) has a more modest performance than COB, although it displays a partial enrichment. In this case, the perspective of depicting the backbone of a putative partner for a given interactability patch is also very pertinent. Indeed, for example the spatial arrangements of Cα within a α-helix are fitting very nicely within a cylinder that can be clearly identified within some PPIs mediated by such secondary structure at the interface (see case study about Bcl-2). Nevertheless, the other channels are clearly non-specific and shall be the subject of improvements in the future.

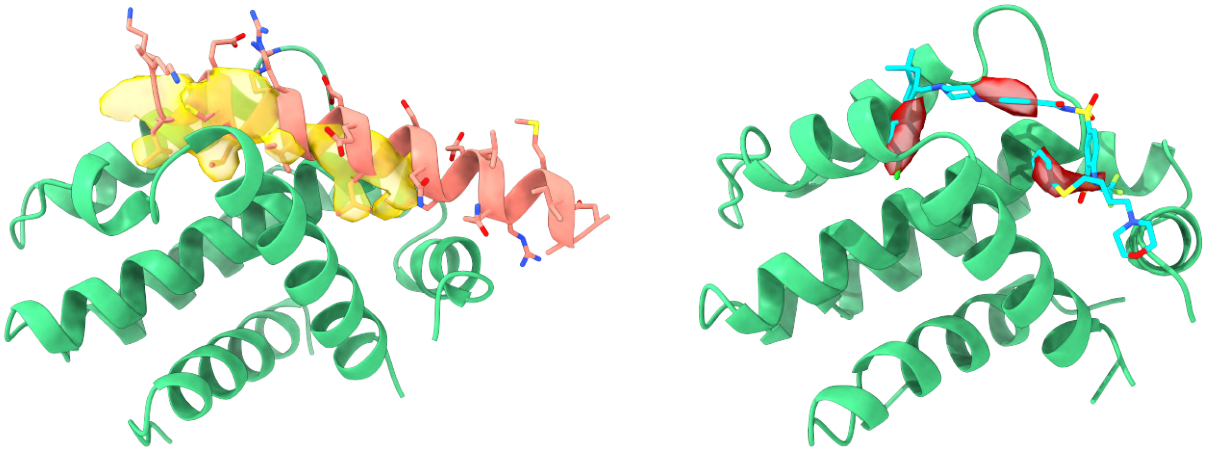


Figure 6.4 – *Left:* InDeep interactability patch prediction on Bcl-2 (pdb 2xa0). *Right:* Ligandability prediction (red surface) performed on Bcl-2 (pdb 4lvt) surface. The red surface patches of InDeep ligandability are localized around the known hot spots of the Bcl-2/Bax interaction that are mimicked by some of the ligand atoms.

6.3.3 Case study: Bcl-2 as therapeutic target

The B-cell lymphoma-2 (Bcl-2) protein is the eponymous protein of the Bcl-2 family, which is central to the regulation of apoptosis and vital for proper tissue development and cellular homeostasis [Bajwa et al., 2012]. Upon interaction with pro-apoptotic BH3 domains containing proteins, Bcl-2 inhibits cell death. In the last two decades, small molecules that disrupt this interaction by binding to Bcl-2 and other anti-apoptotic proteins of this family, have been successfully designed and clinically approved to induce apoptosis of cancer cells [Vogler et al., 2009]. Pro-apoptotic partners of Bcl-2 possess a 20 Å-long α-helical-containing BH3 domain that interact at Bcl-2 surface through an extended hydrophobic groove. A recent review underpinned that the successful development of drugs, such as Venetoclax, against this target was mainly due to the fact they manage to mimic two of the hotspot residues within the BH3 domain binding this groove [Ashkenazi et al., 2017].

This protein family is therefore an excellent case study to retrospectively evaluate the pertinence of using a tool like InDeep. The fact that this tool can predict ligandable/druggable binding pockets, interactability patches including hydrophobic and backbone atom-typed channels, and also monitor such predictions along molecular dynamics trajectories allows a retrospective analysis of feasibility of designing ligands binding to the BH3 groove of Bcl-2. Although Bcl-2 complexes were present in our training set, it is worth noting that the different InDeep predictions below have been made exclusively on the sole structure of Bcl-2 without any consideration for Bax or known co-crystallized ligands.

We can first use InDeep to predict interactability patches at the surface of Bcl-2. As can be seen on Figure 6.4 (Left panel), InDeep correctly predicts (1st ranked patch), within the BH3 groove, the location of the interactability patch with the α-helix of its protein partner Bax. Inspecting more specifically the Cα- and hydrophobic-atom-typed channels within the interactability patch Supplementary Figure E.7 (Top panel), one can observe respectively 1) a faithful depiction of the α-helix shape of the Bax epitope binding the BH3 groove (as a

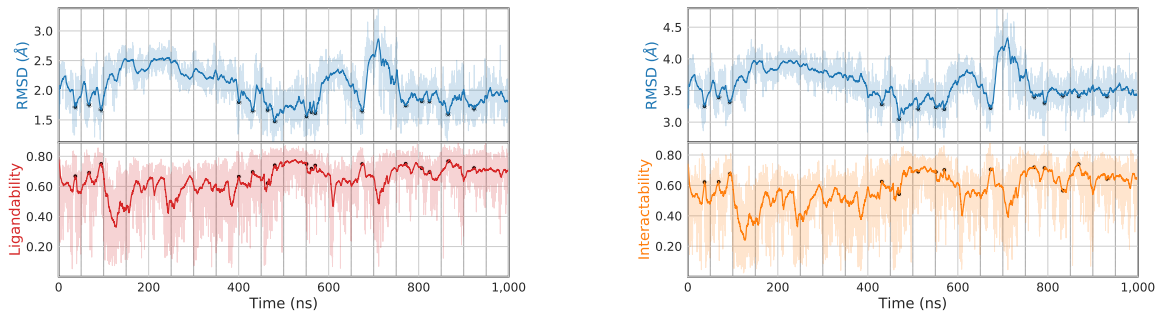


Figure 6.5 – `InDeep` predictions (red and orange) along molecular dynamics trajectory of Bcl-2. Moving averages (exponential) on 200 frames are represented with solid lines. *Left:* `InDeep` ligandability score evolution (red) compared to the minimal binding site RMSD (blue) with respect to 16 `PL` structures. *Right:* `InDeep` interactability score evolution (orange) compared to the RMSD (blue) with respect to the reference `HD` structure. Local minima on the RMSD curve and their corresponding `InDeep` predictions are highlighted as black points.

cylinder shape), and 2) a proper localization of the hydrophobic hotspots known for this system [Ashkenazi et al., 2017].

If we now use the ligandability prediction of `InDeep` on the same structure of Bcl-2 as co-crystallized with Bax (pdb: 2xa0), one can see on Supplementary Figure E.7 (Bottom panel) that `InDeep` correctly highlights the known aforementioned hotspots as the most ligandable regions of the BH3 groove of Bcl-2. In the end, one can notice on Figure 6.4 (Right panel) that the same regions that were successfully targeted by Venetoclax analogs (ex with pdb: 4lvt) and correctly highlighted by `InDeep`.

NMR and X-ray crystallography have highlighted important backbone rearrangements within Bcl-2 upon peptide and small molecule binding to the BH3 groove [Liu et al., 2003]. Early virtual screening approaches failed to identify validated small molecules inhibitors, as they did not consider protein flexibility [Scott et al., 2016]. It is therefore essential to properly sample the flexibility of the system and be able to monitor both ligandable and interactability patches on representative Bcl-2 structure conformations. To do so, a 1 µs-long molecular dynamics simulation (see MD parameters in Supplemental section E.5), starting from the apo form of Bcl-2 (pdb 1gjh), was run to monitor the ligandability and the interactability of the BH3 groove known to bind Venetoclax and other chemical analogs. The `InDeep` prediction has been focused on this region of Bcl-2, as the goal of this approach is to detect holo-like conformations i.e. favorable conformations for ligand binding and not to detect the binding region on the whole protein surface. Figure 6.5 (Left panel) shows the values of the `InDeep` ligandability score of each frame (red line) and the minimal binding site RMSD value (blue line) with respect to 16 ligand-bound `PL` structures (see binding site definition and the list of PL structures in Supplemental section E.6). It can be noted from the fluctuation profiles of the RMSD and the ligandabilty scores that local minima in the RMSD value (highlighted as black points) correspond to ligandability peaks. Therefore, holo-like conformations (with low RMSD against a `PL`) tend to be correctly predicted as ligandable by `InDeep`. A similar trend was observed by other groups on conventional targets [Kozlovskii and Popov, 2020]. Likewise, the interactability was monitored along the same simulation and the RMSD of the binding site residues was computed against a

holo HD structure of Bcl-2 bound to Bax (pdb 2xa0) Figure 6.5 (Right panel). Similarly to the ligandability model, holo-like conformations have a higher score of InDeep interactability than conformations distant from the HD conformations.

These predictions collectively show that a proper usage of InDeep early in the drug discovery initiative against Bcl-2/Bax would have highlighted the hydrophobic hotspots residues within the BH3 groove of Bcl-2 and the most ligandable binding regions of these spots to assist the design of selective BH3 mimetics. Moreover, InDeep would have been efficient at profiling holo-like conformations prior to virtual screening campaigns even when starting from apo structures, and in the absence of Bax bound to Bcl-2, within a molecular dynamics trajectory. Finally, as holo-like conformations are not always accessible with regular water-solvated MD simulations [Eyrisch and Helms, 2009], InDeep can be easily combined to other sampling protocols in order to profile the resulting pocket conformations.

6.4 Discussion

We have introduced InDeep, a unified prediction tool for structure-based drug design targeting protein interfaces. We show that this tool is competitive in detecting the residues that interact with a protein partner. We go beyond this sequence prediction by predicting the localization in space of a putative partner using atom-typed channels signal that helps understand how such protein interaction can take place. 70% of the observed binding sites are present in one of our predictions and 35% in the top three ones. Moreover, InDeep clearly outperforms the state of the art of binding pockets detection for iPPI binding sites but even for conventional targets. Combining those two predictions for a newly solved structure, one can investigate binding sites for ligands that would potentially disrupt a given PPI. Given such a detected binding site, our tool also enables tracking ligandability or interactability scores along a molecular dynamics trajectory, which opens the door for a refined ligandability assessment as well as conformation selection for virtual screening, or unravel epitope binding-prone conformations. Finally, we illustrate these functionalities in a retrospective drug discovery use case on Bcl-2. InDeep is integrated in a PyMol [DeLano et al., 2002] plugin for easy visualization of the predictions (see Section E.7).

Despite several promising results, the therapeutic use of iPPIs remains a minority. We hope this dedicated tool can help enhance their use as well as spark a development of other methods following this line of work. InDeep uses 3D-FC U-Net because the grid-like prediction enables actual localization and profiling of protein partners or future drugs. However, this method, as many others, is sensitive to rotation, a possible extension would be to use equivariant networks [Weiler et al., 2018a] to bake the rotation invariance in the network.

Future developments also include actual drug discovery project use of the tool as well as the implementation of a web-server for easier access to the predictions.

Chapter 7

`quicksom`: Self-Organizing Maps on GPUs for Clustering of Molecular Dynamics Trajectories

This chapter was made in collaboration with Michael Nilges and Guillaume Bouvier and was published in Bioinformatics Application Notes in 2021 [Mallet et al., 2021].

Contents

Abstract

We implement the Self-Organizing Maps (SOMs) algorithm running efficiently on GPUs, and also provide several clustering methods of the resulting maps. We provide scripts and a use case to cluster macro-molecular conformations generated by molecular dynamics (MD) simulations.
Availability and Implementation: The method is available on GitHub and distributed as a pip package.

Résumé

Nous implémentons l'algorithme des cartes auto-organisatrices avec un support GPUs, et proposons plusieurs méthodes de clustering sur les cartes obtenues. Nous fournissons également des scripts et des exemples pour regrouper des conformations de macro-molécules générées par dynamique moléculaire.
Disponibilité : La méthode est disponible sur GitHub et en tant que package pip.

7.1 Introduction

We proposed in a former paper [Bouvier et al., 2015] a Self-Organizing Map (SOM) based algorithm to cluster macro-molecular conformations generated by molecular dynamics (MD) simulations. Alternative methods exist but they either rely on pairwise distance computation [González-Alemán et al., 2019] or on including additional prior information [Olsson and Noé, 2019] whereas SOMs are simple linear clustering algorithms. Due to the expansion of the usage of graphics processing units (GPUs) to perform MD, the number of conformations from trajectories that need to be analyzed exploded. Therefore, our previous, CPU based, implementation of the SOM reached its limit. In the current paper, we propose a fast and efficient GPU implementation of SOM, `quicksom`. This is highly useful for the analysis of long MD trajectories, but can also be used for the clustering of other massive and high dimensional data.

Additionally, we added a set of clustering tools that can be used on the maps produced by the methods. These tools serve to further summarize the inputs. They rely on either automatic clustering or a manual tool with a graphical interface. The efficiency of our tool is demonstrated through a case study on a long MD trajectory.

7.2 Efficient Self Organizing Maps on GPU

Our implementation of Self-Organizing Maps (SOMs)[Kohonen, 1982] is based on PyTorch [Paszke et al., 2019]. This alone speeds up operations and in addition allows us to use GPUs to make the computations even faster. We timed our method against the former application note [Bouvier et al., 2015] and SOMPY[Moosavi et al., 2014], the main implementation of SOMs in Python that is a parallel CPU-based tool.

We used our method on some 2-dimensional toy data as well as on a molecular dynamics trajectory resulting in 168-dimensional vectors. We show the time necessary to perform the training loop on 100k of these vectors in Table 7.1. We include a run on CPU for comparison as well as a run on several cores for SOMPY.

Method	Toy Data	MD Trajectory
Old method [Bouvier et al., 2015]	87 s	536 s
SOMPY [Moosavi et al., 2014]	5.9 s	6.9 s
SOMPY (20 cores)	0.85 s	12.4 s
`quicksom` (CPU)	13 s	101 s
`quicksom`	1.6 s	3.3 s

Table 7.1 – Mean time necessary to process 100k points in the training loop

As expected, the new method is much faster than the old one, especially when run on the GPU, with a 160-fold speedup. We have comparable run times to the SOMPY implementation for the synthetic data set, and two-fold speedup on the higher dimensional MD data. This was unexpectedly fast for this CPU-based method. However, the SOMPY implementation does not support custom batch sizes, so the whole dataset is passed at once, which does not allow flexible training and biases the training for large data sets. We believe this to be a major limitation

because tuning the optimization for large data sets was revealed to be key, in particular for the analysis of MD trajectories.

7.3 Clustering

We introduce several tools to cluster our data beyond a simple SOM cell affectation. The idea is to merge neighboring cells that represent several centroids inside the same cluster.

7.3.1 Automated Approaches

The SOM yields a lattice graph whose nodes are the centroids and whose connectivity is given by the SOM grid. Edges are weighted by the euclidean distance of the centroids they connect. We then compute the matrix of graph distances of nodes. To detect nodes that belong to a given community and to cluster these nodes, we compute the topological distance matrix, and we can thus use algorithms such as the Agglomerative Clustering algorithm. The U-matrix is an informative 2D representation of the SOM that can depict efficiently its topology. A U-Matrix is defined as the matrix whose value at each grid point is the mean of the weight of its edges. To deal with toric connectivity, we rearrange the map by flattening it following the graph shortest path from the global minimum, and pad it to get a square matrix. Using this formalism, we can turn our SOM into an image and use any segmentation algorithm to group centroids together. We default to the graph approach but implement other algorithms that the user can choose.

7.3.2 Manual Approaches

These automatic methods are always prone to failure because of the variety of possible maps resulting from the variety of possible data at hand as well as the choice of hyperparameters for the maps. Therefore we also include a manual clustering option with a GUI, available as a command-line tool. The user can click on a point on the map and expand a region around it by hand, to select the relevant zones of the map they produced. We believe that this user-defined and application-specific clustering is a good work-around in case automatic clustering fails.

7.4 Molecular Dynamics Clustering

The tool can be applied to efficiently cluster Molecular Dynamics (MD) trajectories. This is useful to create a library of representative structures. To do so, we represent each frame by the concatenation of each atom's coordinates and train a SOM to cluster these frames. We included a script to take a MD trajectory in the CHARMM [Brooks et al., 2009] **dcd** format as input and output a **npy** file that can be handled by our SOM implementation. This script also allows the user to select a set of atoms of interest for the SOM analysis. We also included utilities to select the frames that fall into a given cluster for visualization.

The method was applied on the trajectory analyzed in our previous implementation [Bouvier et al., 2015]: 15 μs molecular dynamics at 330 K of a simplified sequence of a 56-residue α/β subdomain of the protein G [Guarnera et al., 2009] starting from an extended conformation. The analysis was performed on the C-α coordinates yielding 750 000 vectors of dimension 168. We include the results in **Figure 7.1**. We can see that the unfolded protein conformations correspond

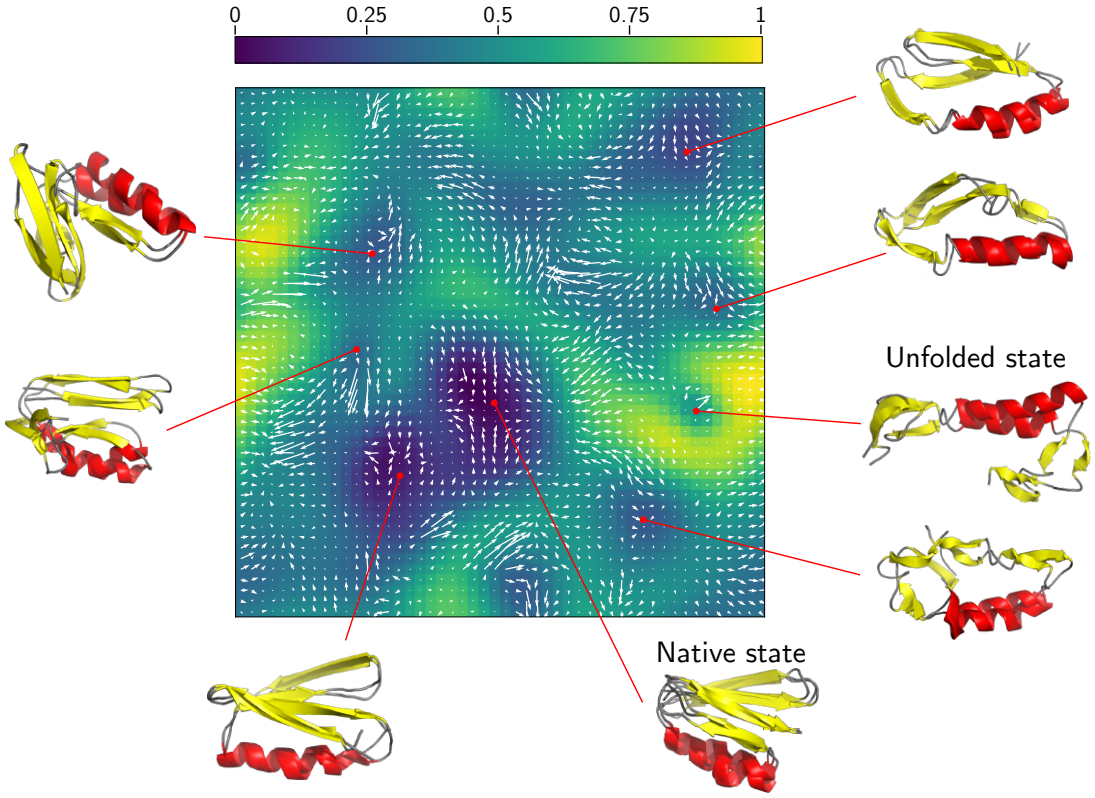


Figure 7.1 – Resulting U-Matrix of the clustering of a MD trajectory. The range of values is normalized. Darker color implies closer cells that represent a data cluster. White arrows represent the flow defined as the sum of the transition steps of the MD from each cell. Some structures were represented with a pointer to the cell they are mapped to by the algorithm.

to a sparse region of the map while the most stable scaffolds fall into dense ones. We also included a representation of the transition steps as a flow map. This flow goes from the least populated and stable states to the more stable ones. It also enables visualization of the paths preferentially followed by the trajectory.

Implementation and availability

We have packaged our project into a pip package, `quicksom`, for easy setup and command line usage. The source code is available at `https://github.com/bougui505/quicksom`. A full description for installation and usage is available as a README.

Chapter 8

OptiMol : Optimization of binding affinities in chemical space for drug discovery

This chapter was made in collaboration with Jacques Boitreaud, Carlos Oliver and Jerome Waldispuhl and was published in the Journal of Chemical Information and Modelling in 2020 [Boitreaud et al., 2020].

Contents

Abstract

Ligand-based drug design has recently benefited from the development of deep generative models. These models enable extensive explorations of the chemical space, and provide a platform for molecular optimization. However, the vast majority of current methods do not leverage the structure of the binding target, which potentiates the binding of small molecules and plays a key role in the interaction.

We propose an optimization pipeline that leverages complementary structure-based and ligand-based methods. Instead of performing docking on a fixed chemical library, we iteratively select promising compounds in the full chemical space using a ligand-centered generative model. Molecular docking is then used as an oracle to guide compound optimization. This allows to iteratively generate compounds that fit the target structure better and better, without prior knowledge about bio-actives.

For this purpose, we introduce a new graph to selfies Variational Autoencoder (VAE) which benefits from an 18-fold faster decoding than the graph to graph state-of-the-art, while achieving similar performance. We then successfully optimize the generation of molecules towards high docking scores, enabling a ten-fold enrichment of high-scoring compounds found with a fixed computational cost.
Code is available at : csb.cs.mcgill.ca/optimol

Résumé

La découverte médicamenteuse centrée sur les ligands a bénéficié récemment de l'introduction de méthodes génératives par apprentissage profond. Ces modèles permettent une exploration accrue de l'espace chimique et le recours à des méthodes d'optimisation continue. Cependant, la plupart de ces méthodes n'exploitent pas la structure de leur cible thérapeutique, qui est pourtant centrale dans l'interaction cible-ligand.

Nous proposons un pipeline d'optimisation qui utilise à la fois les approches centrées sur les ligands et sur les cibles. Plutôt que de faire du docking sur une chimiothèque fixée, nous proposons itérativement des candidats issus de l'ensemble de l'espace chimique à l'aide un modèle génératif. A chacune de ces itérations, nous utilisons le docking moléculaire pour guider cette génération. Nous pouvons ainsi générer des populations de composés chimiques avec des scores de docking croissants au cours du processus, sans connaissances préalables sur d'éventuelles affinités chimiques.

Nous introduisons un modèle génératif muni d'un décodeur 18 fois plus rapide que l'état de l'art, tout en maintenant des performances équivalentes. Nous démontrons aussi que le processus itératif d'optimisation est efficace pour générer des molécules à bon score de docking, avec des populations enrichies dix fois en composés à haut score pour un coût computationnel fixé.

Le code est disponible à l'adresse : csb.cs.mcgill.ca/optimol

8.1 Introduction

8.1.1 Molecular optimization

Molecular optimization, also known as inverse design, consists in designing compounds with desired drug-like properties and biological activity. Direct molecular optimization was first introduced in Gómez-Bombarelli et al. [2018]. Subsequently, a string of papers addressed this problem with a variety of approaches as explained in reviews [Sanchez-Lengeling and Aspuru-Guzik, 2018; Schwalbe-Koda and Gómez-Bombarelli, 2019]. Formally, we are looking for compounds $\mathbf{x}$ that maximize a function $\mathbf{f}(\mathbf{x})$. This function is often a chemical property that makes a compound more drug-like such as QED or solubility, or a more complex property, such as bio-activity. Several properties of $\mathbf{f}$ affect its optimization: differentiability, dimension of the output space , evaluation cost and smoothness.

The drug-like chemical space contains an estimated $\sim 10^{60}$ compounds [Bohacek et al., 1996]. The size of the chemical space and the difficulty to accurately estimate the objective $\mathbf{f}$ without *in-vitro* tests make it impractical to look for a single candidate. Moreover, $\mathbf{f}$ is often a simple surrogate for a complex, phenotypical endpoint. Hence the current approach to molecular optimization is to search for ensembles of compounds with enhanced estimated properties and then conduct *in-vitro* tests. Formally, we are looking for a non-trivial distribution q that augments $\mathbb{E}_{\mathbf{x}\sim q}(\mathbf{f}(\mathbf{x}))$, instead of the global maxima of $\mathbf{f}$. This induces a trade-off between maximizing the objective function and sampling diverse compounds.

8.1.2 Binding affinity estimation

A drug's activity is induced by its affinity to a target. There are three avenues for obtaining binding affinity estimates :

1. Experimental bio-assays consist in *in-vitro* quantitative assessment of the interaction between a compound and a target. This data does not require computing and is reliable, but often scarce.

2. Quantitative Structure Activity Relationship (QSAR) models are machine learning models trained on experimental bio-assays to derive structure-activity rules. They rely on the assumption that molecules with similar structure are likely to exhibit similar bio-activity, which does not hold true in the whole chemical space, especially near activity cliffs [Husby et al., 2015]. Moreover, as any other machine learning algorithm, they have an applicability domain. This means that their accuracy for a compound $\mathbf{x}$ depends on the similarity between their training set and $\mathbf{x}$.

3. Molecular docking softwares search for ligand conformations that minimize the binding energy with a given protein pocket. Although the estimates of the binding affinity they provide are often noisy, the top-scoring compounds are known to be enriched in active molecules [Huang et al., 2006]. Thus they are widely used to select the most promising compounds in a library[Shoichet et al., 2002]. Docking is computationally very intensive ($\sim$ 10 CPU minutes / compound with Vina[Trott and Olson, 2010]), making it crucial to carefully choose the compounds to dock.

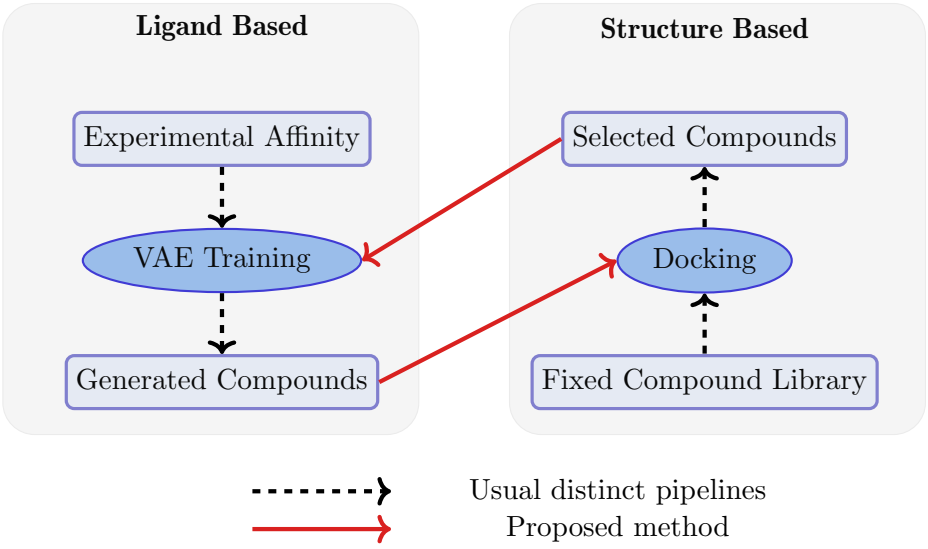


Figure 8.1 – Closing the loop : In ligand-based approaches (left), new actives are sampled by a generative model which is trained to generate compounds based on a fixed dataset with high experimental affinities. In structure based approaches (right), a fixed library is screened for actives via docking. We propose to dock the compounds produced by the generative model, and to use the results of the docking to fine-tune the ligand-based generative model.

8.1.3 Binding affinity optimization

In practice, the earliest molecular optimization task consists in finding compounds with high affinity to a given target. In the structure-based approach, an ensemble of putative candidates is obtained by screening fixed chemical libraries using docking. Computational resources currently limit the library size to $\sim 10^6 - 10^9$ compounds, which is a very small fraction of the drug-like chemical space.

In the ligand-based approach, QSAR models are used as a surrogate for binding affinity, leveraging known actives. Therefore, it is ill-suited to finding compounds with new binding modes. It also requires the target to have enough known actives.

Active learning and automated synthesis have been identified as promising research directions to accelerate the drug discovery process [Schneider, 2018; Winter et al., 2019b]. By allowing a model to iteratively query the most informative data, and learn from experimental answers to these queries, these methods enable a guided and data-efficient exploration of the activity landscape in the chemical space. Inspired by such closed-loop strategies, we propose to iteratively search the chemical space for promising leads, guided by a structure-based assessment of their activity (Figure 8.1). By combining molecular docking and a latent-variable generative model, our framework finds regions of high affinity to a target in the chemical space.

8.1.4 Contributions

We introduce a new Variational Autoencoder architecture that retains state-of-the-art results and enables faster molecule generation. Next, we develop `OptiMol`, a pipeline to generate molecules with optimized docking scores, taking into account the computational cost of molecular

docking. We show `OptiMol` generates a population with ten times more high-scoring compounds than uniform sampling on the human Dopamine Receptor D3. Finally, we show `OptiMol` can handle the optimization of composite objective functions.

8.2 Related work

8.2.1 Molecule representation

Molecules can be represented in several ways, trading off between the accuracy of the depiction and its computational advantages. In decreasing order of richness of representation, molecules are represented as ensembles of 3D, static 3D, molecular graphs, SMILES, Selfies [Krenn et al., 2019] or fingerprints. Each of these representations has drawbacks. Using 3D objects with no preferred orientation in machine learning pipelines is under research with promising results [Hoffmann et al., 2019; Hoffmann and Noé, 2019] but not yet established. Molecular graphs are efficient to encode information but deconvolution and generation is harder [Assouel et al., 2018; Jin et al., 2018a]. Finally, string representations were used first as they benefited from advances in natural language processing. SMILES were extensively used but when generating SMILES, some sequences are invalid. This problem was recently solved by the introduction of Selfies[Krenn et al., 2019], which yield all valid sequences, but are sensitive to small modifications. Another approach to get molecular representations uses Variational Autoencoders (VAE) [Kingma and Welling, 2013]. These methods learn a continuous encoding of the chemical space by reconstructing or translating these representations [Gómez-Bombarelli et al., 2018; Jin et al., 2018a; Krenn et al., 2019; Kuzminykh et al., 2018; Liu et al., 2018; Skalic et al., 2019a; Winter et al., 2019a]. The latent space can be interpreted as a flattened manifold, of which the structure and geometry mostly reflects chemical similarity[Winter et al., 2019a]. Turning the chemical space into a euclidean space enables one to continuously navigate in the space, which opens the door to classical optimization methods.

8.2.2 Molecular optimization

Latent space optimization Once molecules are embedded in a continuous space, classical optimization schemes become possible. Bayesian optimization was used first [Dai et al., 2018; Gómez-Bombarelli et al., 2018; Jin et al., 2018a; Kusner et al., 2017], followed by constrained Bayesian optimization [Griffiths and Hernández-Lobato, 2017; Korovina et al., 2019] and swarm optimization [Winter et al., 2019b]. Another approach is to approximate $\mathbf{f}$ with a differentiable function and to conduct gradient ascent in the latent space. A potential advantage of this method is that the function mapping from latent space to $\mathbf{f}$ can be learned jointly with the generative model, contributing to shaping the latent space [Liu et al., 2018].

Molecular translation Matching Molecular Pairs Analysis was recently used to train a translational VAE to turn a molecule into a similar one with better target property [Jin et al., 2018b] : this model learns to take an optimization step, given a starting point. Damani et al. [2019] show that molecules can be optimized for a target property by recursively taking such steps in the chemical space. This step-wise approach is limited to lead optimization, but seems well suited to handle activity cliffs.

Guided generation using Reinforcement Learning By decomposing the molecule generation as a sequence of actions, we can build a generative model using a probabilistic RL agent. Guided generation consists in biasing the generation towards compounds that optimize $\mathbf{f}$, easily including non differentiable objectives [Olivecrona et al., 2017; Popova et al., 2018]. An adversarial term can be added to ensure that we keep generating realistic compounds [Guimaraes et al., 2017; You et al., 2018]. One caveat in using such methods is that for multidimensional objective functions, rewards become sparse making the training harder [You et al., 2018]. Another issue is that the number of evaluations of $\mathbf{f}$ is not optimized, limiting the use of costly oracles.

Other generative models Jin et al. [2020b] use a fragment based approach where they automatically learn a library of activity-inducing fragments and then generate compounds combining them. Prykhodko et al. [2019] train a Generative Adversarial Network (GAN) to mimic the active compounds distribution in the latent space. A related approach was proposed in Gupta and Zou [2019], and performs iterative fine-tuning of a generative model on its most successful outputs. Other approaches of this kind with better statistical grounding exist [Brookes and Listgarten, 2018; Brookes et al., 2019], but we are unaware of their application to small molecule optimization. They are applicable both to models with and without a latent space structure. Finally, Skalic et al. [2019a] condition a generative model by a 3D shape. By coupling it with a GAN that generates shapes conditioned on a target protein structure, they generate ligands that fit an input protein structure[Skalic et al., 2019b]. This is a way to generate compounds with augmented affinities that side-steps the direct optimization.

8.2.3 Binding affinity optimization

To generate compounds with high binding affinities, we can also use one of the three aforementioned sources of binding affinity estimates. In Prykhodko et al. [2019], authors train a GAN to sample from a distribution resembling the one of known actives in latent space, and evaluate the samples' affinities using a QSAR model. Alternatively, a QSAR model can be used to bias the generation. This is the approach used in the RL framework [Olivecrona et al., 2017; Popova et al., 2018]. This method gives good results, but is based on a QSAR model that can be inaccurate: the authors constrain the model to avoid drifting away from its validity region. Finally, one can resort to a docking program that uses physics-based molecular mechanics force fields to compute binding affinities. However the computational cost of molecular docking makes the optimization challenging.

8.3 Methods

8.3.1 Graph to Selfies Variational Autoencoder

We propose a translational graph to Selfies VAE that achieves comparable performance to state of the art models, while benefiting from the following design choices :

- Using the molecular graph as input and a graph convolution encoder solves the issue of data augmentation. It is also better suited for learning a chemically organized latent space, since chemically similar molecules have very similar graphs, while their SMILES representations may change more due to syntax rules. Finally, graph convolution embeddings and circular

fingerprints were shown to enhance molecular property predictions [Duvenaud et al., 2015; Yang et al., 2019].

- Decoding to a molecular graph results in more complex architectures than decoding to sequences, thereby increasing the computation time. The main weakness of string-based decoders is that due to the SMILES syntax, decoded sequences can result in invalid molecules. By using recently published Selfies [Krenn et al., 2019], we circumvent this issue and generate 100% valid molecules.

Model architecture details as well as a mathematical framing of VAE are available in Supplementary Section **F.1**. The data sets used for training are detailed in the Results Section. The training regime and hyper-parameters chosen for the architecture and optimization procedure are detailed in Supplementary Section **F.2**.

8.3.2 Docking on Dopamine Receptor D3

We use AutoDock MGLtools to prepare the ligand and the pockets and Autodock Vina [Trott and Olson, 2010] to estimate compounds binding affinities to human Dopamine Receptor D3. We use the PDB structure of the DRD3 receptor provided in the DUD-E[Mysinger et al., 2012] data set, and keep the same binding site coordinates. We compute the docking score as the average of the ten best poses found for each ligand as it has lower variance and we found it to yield the best enrichments.

8.3.3 Query efficient optimization

We turn to binding affinity optimization using this generative model and include docking scores in the function $\mathbf{f}$, making it non differentiable. In addition, this oracle is now costly and the sparse rewards induced by the RL frameworks are not tractable, which calls for specific optimization methods.

Bayesian Optimization

Bayesian Optimization uses Gaussian processes to approximate $\mathbf{f}$ in a query-efficient way, by learning on queries that maximize expected improvement. To sample these points, a rigid (not learnt nor adaptive) sampling is used. This does not scale well to high dimensional spaces or large batch sizes as the grid evaluation becomes intractable (this amounts to finding an estimate on all molecules and only picking the most promising candidates for docking). This choice limits scalability when sampling tens of thousands of compounds in the chemical space.

Conditioning by Adaptive Sampling

Conditioning by Adaptive Sampling (CbAS) is a recently published alternative method. This method trains a generative model that seeks to maximize an objective function. Starting from a prior generative model, it progressively shifts its distribution to maximize the expectation of a function of the samples. We efficiently use queries thanks to an importance sampling scheme coupled with reachable objectives for the model. The alternating phases of tuning and sampling also enable a computationally efficient implementation. In DbAS[Brookes and Listgarten, 2018],

the model obtained at each iteration plays the role of the prior model, resulting in zeroed importance weights. This technique can drift away from the prior. We have used a hybrid technique that clamps the importance weights, still putting more emphasis on the samples which are likely under the prior but learning on other ones too. For a more detailed explanation of these algorithms, see Supplemental Section **F.4**.

8.3.4 OptiMol

OptiMol combines the graph to Selfies VAE, a docking program and the clamped version of CbAS. First, our prior is trained on a molecular data set and mimics its distribution. Then we iteratively take samples, re-weight them with docking and fine-tune the model (Figure 8.2).

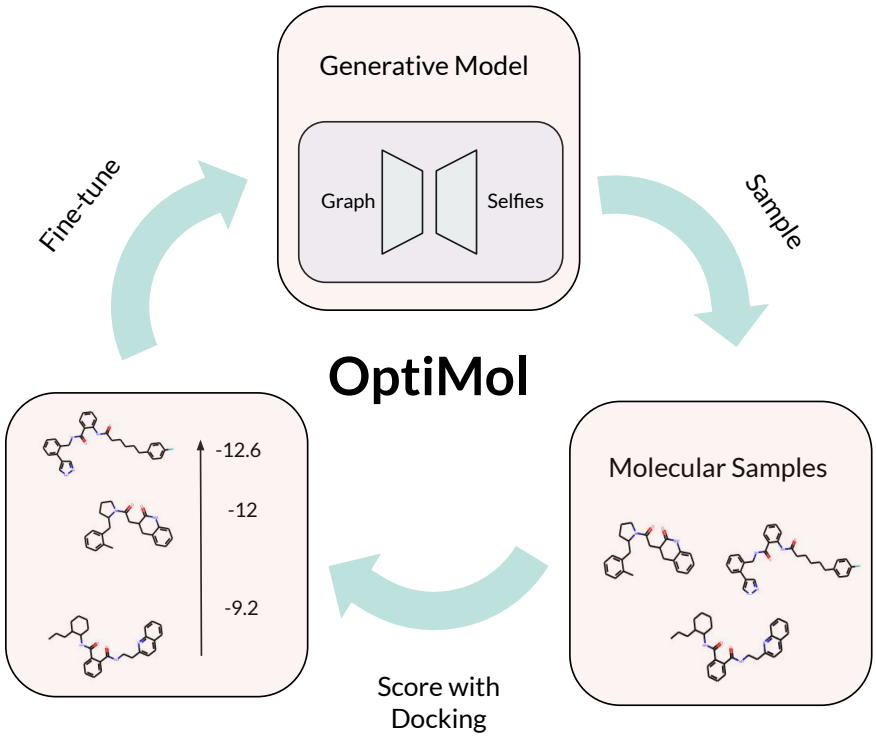


Figure 8.2 – OptiMol workflow: A prior generative model is trained on a broad chemical space and generates samples. We use docking to score these samples, and fine-tune the generative model on the samples weighted by their docking scores

To our knowledge, this is the first use of CbAS with a computationally costly oracle. Coupling CbAS with docking requires a dedicated implementation. We have implemented a parallel version of the code that can leverage multi-node architecture computational clusters, using several hundreds of CPU cores for the docking phases, and running the sampling and training phases on GPU nodes. Using this implementation, we trained OptiMol for 30 iterations over batches of 1000 samples in approximately 15 hours using 200 cores. We note that parallelism over more CPUs would result in linear speed up, up to several thousands of CPUs. The implementation is available at `http://csb.cs.mcgill.ca/optimol`. Given a dataset, users can easily train a new

prior model. `OptiMol` can also easily be trained on any target structure, given its PDB file.

8.4 Results

8.4.1 Graph to Selfies VAE

To compare to other architectures, we trained on the Moses[Polykovskiy et al., 2018] benchmark set, which contains $\sim$ 1.5M clean leads from the ZINC database [Sterling and Irwin, 2015]. Moses is a standard benchmark for molecular generative models. We achieve comparable results on the Moses benchmark metrics (Table 8.1, see Supplementary Table F.1 for full comparison). Our model reaches state of the art performances while benefiting from design choices in our application. We note that our model has 3.2M trainable parameters, compared to 5.1M for the default JTVAE implementation [Jin et al., 2018a]. Our model also runs approximately 18 times faster for sampling and training.

Model	Valid	Unique 10k	IntDiv	IntDiv2	Filters	Novelty	Sampling time
JTVAE	1.0	0.9992	0.8512	0.8453	**0.9778**	**0.9153**	153 ms
graph2selfies	1.0	**0.9998**	**0.8560**	**0.8496**	0.9557	0.9097	**8.3** ms

Table 8.1 – *Left :* Moses metrics for samples generated by JTVAE and graph2selfies. *Right :* Sampling time is the average time needed to produce one molecule from a trained model.

To test whether our latent space is suitable for optimization tasks, we run Bayesian Optimization on the toy task of enhancing the composite logP score[1], as it is the standard benchmark. We compare to latent spaces introduced in previous works [Dai et al., 2018; Jin et al., 2018a; Kusner et al., 2017]. As all baselines, we train a prior generative model on the 250k compounds dataset, we use the open-source BO implementation by Kusner et al. [2017] and report our results on Table 8.3. We note that picking only the 3 best compounds was subject to high variance across runs because of the size of the chemical space. We got compounds with comparable scores to the ones found by the state of the art. We conclude that our architecture has a similar performance as the current state of the art with a significantly reduced computational load.

8.4.2 Docking on Human Dopamine Receptor 3

To evaluate the ability of docking to find active compounds for DRD3, we dock 380 actives and 20 000 decoys from the ExCAPE database [Sun et al., 2017a]. The actives were picked by clustering the ExCAPE actives by similarity and picking the centroids. Full docking scores distributions are shown in Supplementary Figure F.3. We sort the list of compounds by docking score values. The log-ROC curve (Figure 8.3) shows the fraction of actives found at a given decoys fraction threshold, with a focus on the top-scoring compounds in the ranked list. When compared to a random permutation of the list, it shows that docking manages to separate actives and inactives, with significantly more actives in the top of the list.

[1] $clogP(m) = logP(m) - SA(m) - cycle(m)$ where *cycle(m)* is 0 if the molecule has no cycle bigger than 6 atoms, otherwise the size difference between the biggest cycle in the molecule and a 6-atoms cycle

We then compute Enrichment Factors (EF) at different thresholds and present results in Table 8.2. We also include the 1% decoys metric, which represents the enrichment factor computed after retrieving top 1% of the decoys in the ranked list. Among the 1% top-scoring compounds, we find a ratio of actives 8.168 times higher than in the whole list. In DUDE[Mysinger et al., 2012], the authors compute such metrics for 102 diverse proteins. It is worth noting that they obtain an enrichment factor of 4 at 1% decoys (EF1) for DRD3, which is quite low compared to other targets, as several get an EF1 higher than 40. This suggests that DRD3 is a rather difficult target for docking softwares.

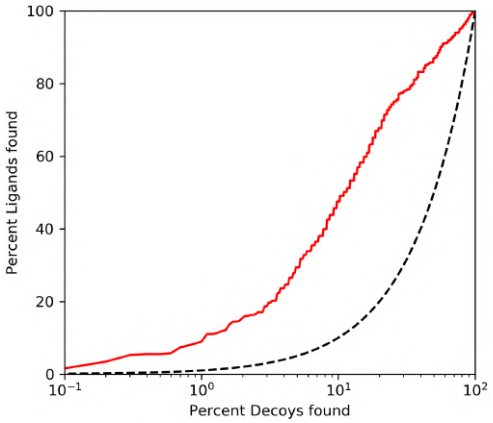


Figure 8.3 – Log-ROC curves for ExCAPE database DRD3 actives and decoys *(red)* and random sampling *(dashed line)*

EF	Threshold
13.373	0.1%
11.122	0.5%
8.168	1%
7.229	2%
8.919	1% decoys

Table 8.2 – Enrichment Factor (EF) in active compounds at different thresholds.

This experiment illustrates that our docking oracle is able to enrich a list of samples in actives. Hence, the top-scoring compounds found by docking are significantly more likely to bind DRD3. As a control for lead generation experiments, we also dock 10k ZINC compounds [Sterling and Irwin, 2015]. As expected, their docking scores distribution is significantly shifted from the actives distribution, and only a negligible fraction of them have a score better than -10 kcal.mol^{-1} (Supplementary Figure F.3).

8.4.3 OptiMol training and benchmarking

Optimization of clogP

We benchmark OptiMol on the toy task of generating compounds with high clogP scores. We use the prior generative model of the 250k dataset and run 20 iterations of CbAS, coupled with a clogP oracle. We plot the clogP distribution of the 1000 training samples at each iteration in Figure 8.4, and compare it to a 20-step run of Bayesian optimization, with 500 new samples per step (Implementation details in Supplementary Section **F.4.1**). It is worth noting that Bayesian optimization hardly scales to 1000 samples per step as it does not have a generative model. We also compare to a reinforcement learning [You et al., 2018] approach that does not use a latent space to the benchmark, even though it is not query-efficient and thus not adapted to our end goal of optimizing docking scores.

Figure 8.4 shows `OptiMol` gets a much stronger and steadier shift of the distribution towards high clogP than BO. Under the top-3 compounds metric used in previous works, We obtain scores (normalized, as in Table 8.3) of 11.52, 11.28 and 11.09 respectively, outperforming the Reinforcement learning approach by You et al. [2018]. Top-scoring compounds are shown in Supplementary Figure F.4.

Model	1st	2nd	3rd
GVAE [Kusner et al., 2017]	2.94	2.89	2.8
SD-VAE [Dai et al., 2018]	4.04	3.5	2.96
G2S - BO	4.81	4.6	4.27
JTVAE [Jin et al., 2018a]	5.3	4.93	4.49
GCPN [You et al., 2018]	7.98	7.85	7.8
`OptiMol`	**11.52**	**11.28**	**11.09**

Table 8.3 – Top 3 clogP scores found by each method (All scores are normalized using the 250k dataset scores, baseline results are copied from previous works).

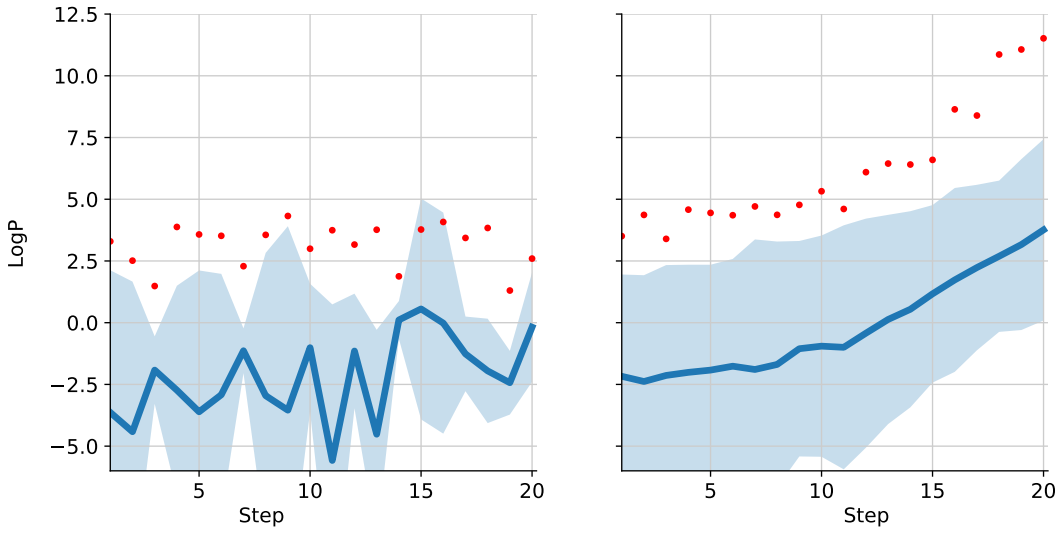


Figure 8.4 – Composite logP optimization results : the blue curve plots the mean and standard deviation of the clogP at each iteration and the red dots show the maximum score.
Left: Bayesian Optimization of clogP with 10k initial samples and 500 new samples per step, using implementation from Kusner et al. [2017]. *Right:* CbAS for clogP optimization, with 1k samples per step.

Visual inspection of the top-scoring compounds found by `OptiMol` and GCPN [You et al., 2018] suggests that these methods consistently outperform BO because they are able to shift away from the prior chemical space, contrary to BO. This is both a strength and a weakness of `OptiMol`, as the clogP objective drives the optimization towards regions of the chemical space that are not desirable in practice. However, since `OptiMol` is trained step by step, it is possible

to control this shift from the prior by early-stopping (See Supplementary Figure F.4 for top compounds at different steps). We believe this flexibility makes our tool well-suited to real-life cases of goal-directed optimization.

Optimization of docking scores

We now present the main result of our work : we show OptiMol generates a diversity of molecules with high docking scores.

We train the prior using a drug-like subset of the ZINC database [Sterling and Irwin, 2015], following the filtering procedure described by Skalic et al. [2019b]: LogP ≤ 5, molecular weight 250-500 Da, removing molecules containing radicals and SMILEs strings of length greater than 60 characters. We train OptiMol and report the distribution of docking scores of samples at each step, as well as the number of novel samples (never sampled by the generative model in the previous steps) on Figure 8.5. Step-by-step, the docking score distribution of the samples shifts towards lower scores (lower is best). However, the number of novel samples in the training batch starts to decrease after 14 steps, indicating the chemical space of the generative model is shrinking. This is not desirable in lead generation, since we aim at learning a generative model with sufficient ability to explore, even though focusing more on a promising region could lead to a better distribution of docking scores. For this reason, we use the last generative model with non redundant samples.

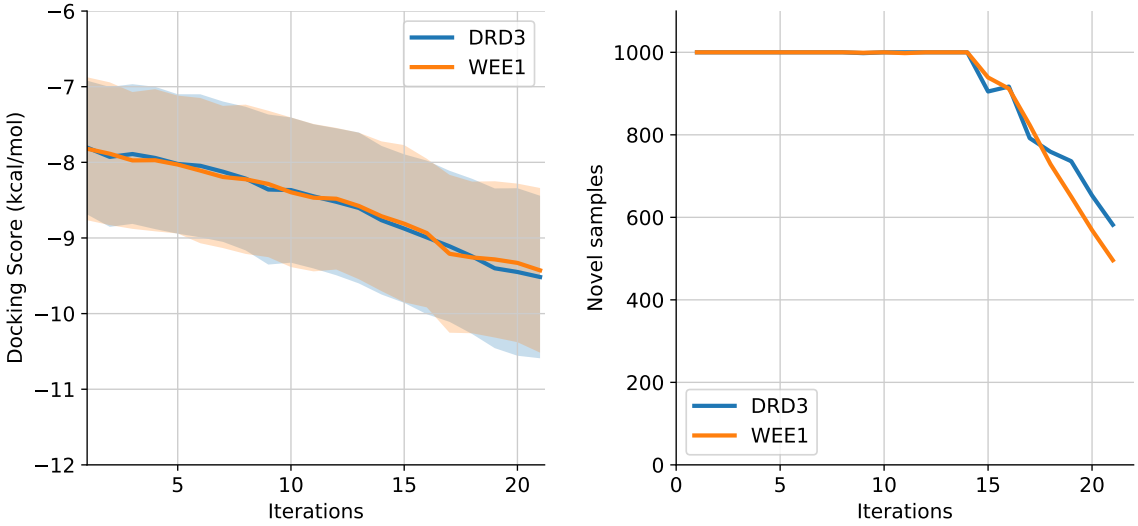


Figure 8.5 – OptiMol training results on DRD3 (blue) and WEE1 (coral). *Left*: The average and standard deviation of the docking scores obtained at each iteration. *Right*: The number of compounds never sampled before at each iteration.

Skalic et al. [2019b] also try to generate actives using a target structure. They use a generative model and condition it using the shape of the target. This does not require additional training for a query target structure. This is not possible with OptiMol, as a generative model has to be retrained with a new docking oracle that docks samples on the new target. On the other

hand, OptiMol oracle and optimization is user-defined, whereas their model is not flexible. We acknowledge that docking optimization is not the direct goal of Skalic et al. [2019b], but to the best of our knowledge, it is the only other approach that learns a distribution of compounds with enhanced docking scores.

We assess OptiMol samples under QSAR and docking metrics, following the evaluation procedure of Skalic et al. [2019b]. We use their web server implementation to generate 4k compounds for our target, and generate 4k compounds using OptiMol. As a decoy population, we uniformly sample 10k random samples from the ZINC training set. We then train a QSAR model as in Skalic et al. [2019b] (Supplementary Section **F.5.2**) on Dopamine Receptor 3. We rank the aforementioned samples by QSAR scores, and plot the composition of the ranked list by deciles (Supplementary Figure F.5). We find that the top half of the list is significantly enriched in samples from OptiMol, however the first decile still contains more samples from Skalic et al. [2019b]. Our results could be boosted by finetuning the docking procedure so that it better correlates with the experimental actives. Moreover, the point of using a structure-based method is to find new hits that may not belong to the validity domain of the QSAR model.
We then compare the docking scores for our samples to those of Skalic et al. [2019b]. The distributions of docking scores for OptiMol samples, Skalic et al. [2019b] samples and random samples from the ZINC training set are shown in Figure 8.6. We find that the scores distribution of Skalic et al. [2019b] is not significantly shifted from the ZINC prior. OptiMol outperforms Skalic et al. [2019b] under the ROC-AUC metric they use (ability to distinguish a model sample from a random ZINC sample) by a large margin (0.47 vs 0.76 for OptiMol). We conclude that OptiMol successfully generates compounds with enhanced docking scores, and significantly outperforms related methods at this task.

Diversity and optimization trade-off

Both the results on docking and on clogP highlight the trade-off between optimization and diversity. Indeed, pushing the optimization too far results in highly optimized compounds but undesirable behaviors of the generative model. Each user can choose when to stop optimizing. As a heuristic, we pick the model just before diversity in the samples starts to decrease. All the following figures and plots will use this heuristic.

8.4.4 OptiMol performance

Sampling from OptiMol model

A challenge in hit finding is to obtain a sufficient diversity of promising compounds. We therefore assess whether OptiMol samples form a chemical space that is large and diverse enough to be practically useful. To do so, we generate 100k samples using the previously trained OptiMol model. The docking scores distributions of the first 2500 and last 2500 samples are similar (respective means -8.59 and -8.56, p-value = 0.045, F.8), indicating that the quality of samples does not decrease during sampling. We also assess the diversity of these samples, using the mean pairwise Tanimoto distance as a diversity metric. This distance is a bit-to-bit distance over fingerprints, ranging from zero to one. Table 8.4 shows that the samples produced by OptiMol remain chemically diverse, and, most importantly, that the 10% top-scoring samples are as diverse as the whole population sampled from OptiMol. Random samples from OptiMol,

sorted by docking scores, are shown in Supplementary Figure F.9. These molecules span over multiple scaffolds but some of them include macro-cycles or halogens, undesirable features for drugs. Finally, as OptiMol shifts away from the prior during training, we study the druglikeness (QED) and synthetic accessibility (SA) of the learnt chemical subspace. Indeed, both properties are relevant to the lead generation problem, and synthetic accessibility of compounds proposed by generative models is an open problem. We find that OptiMol samples have a lower QED than the ZINC compounds in the prior, but that the SA scores do not shift (Table 8.4).

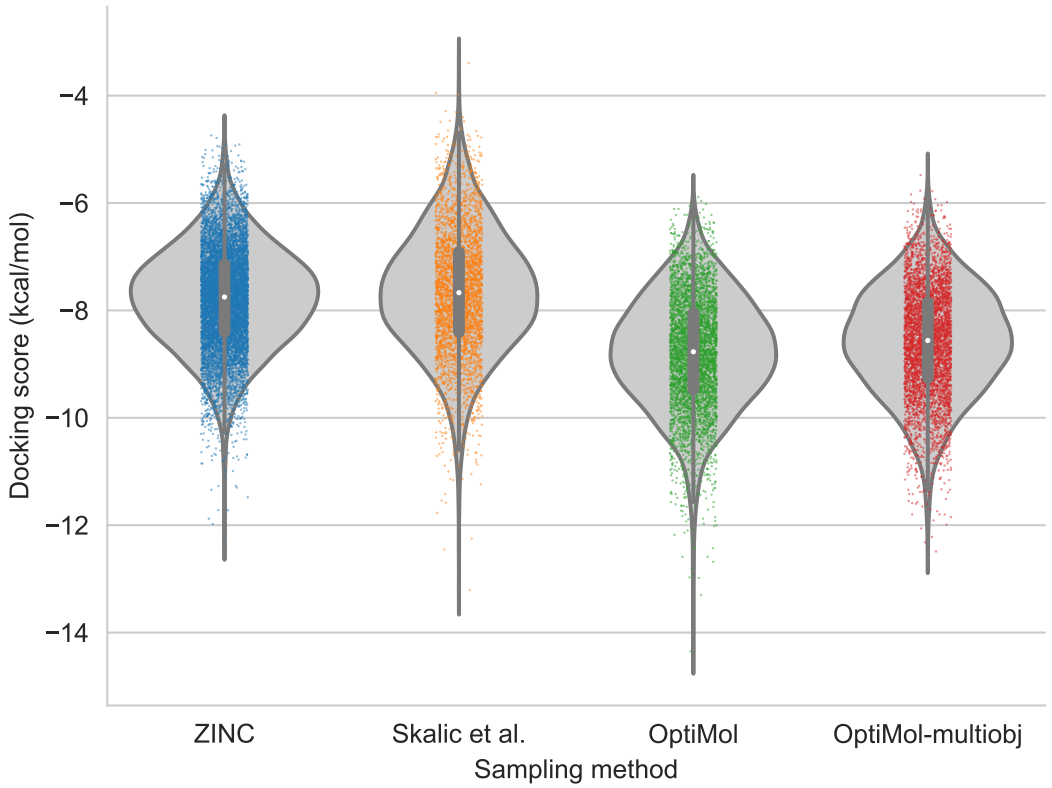


Figure 8.6 – Distributions of docking scores on DRD3 for 2k random ZINC compounds, samples obtained with Skalic et al., OptiMol and OptiMol-multiobjective generative models

Flexibility and robustness

We have designed our tool to be as flexible as possible regarding the target structure and the objective function used. The user can easily change the docking target, simply by using a command-line option. OptiMol can also be used with any objective function $\mathbf{f}$. Detailed instructions about how to plug-in such user-defined scoring functions are given in the documentation. We assessed the impact of such modifications by running the docking score optimization for another target, WEE1, a nuclear kinase. The results are presented in Figure 8.5. The optimization results are on par with DRD3, showing the tool's flexibility and robustness.

We turn to assessing the robustness of `OptiMol`. We start by monitoring how the results change when tuning hyperparameters. We find our results to be stable to different architecture choices detailed in Supplementary Sections **F.2** and **F.5.3**. We explicitly assessed the robustness of the optimization by doing three independent runs of `OptiMol` on DRD3. Optimization trajectories are shown in Supplementary Figure F.6. Despite a limited statistical power, the results indicate that the convergence of the optimization is stable and robust.

Multi-objective optimization

A key strength of `OptiMol` is its flexibility regarding the oracle scoring function $\mathbf{f}$, which allows for multi-objective optimization with minimal changes. To address the aforementioned decrease in QED, we included it in the objective function, asking the model to only get molecules with a QED above 0.4 and a SA below 5. These thresholds were found to account for most molecules in the filtered ZINC prior. The training dynamics over iterations are shown in the Supplemental Figure F.7 and the results are presented in Table 8.4.

	Docking		QED		SA		Diversity	
	Overall	10%	Overall	10%	Overall	10%	Overall	10%
ZINC	-7.78	-9.46	0.74	0.657	2.92	2.73	0.89	0.88
`OptiMol`	**-8.78**	**-10.67**	0.62	0.51	2.91	2.98	**0.86**	**0.86**
Multiobjective	-8.58	-10.42	**0.69**	**0.57**	**2.84**	**2.91**	**0.86**	0.85

Table 8.4 – Mean of Docking, QED, SA and Diversity score for samples generated with `OptiMol` and the multi-objective `OptiMol` implementation. The values for the ZINC prior are shown for comparison. 'Overall' means the mean was computed over the whole distribution and '10%' means it was computed only over the 10% best scoring compounds. The multi-objective model mitigates the drop in QED values, indicating the model focuses on more lead-like molecules.

We find that the average QED and SA of `OptiMol`-multiobj samples are closer to the ones of the prior. On the right side of Supplementary Figure F.9, we plot samples drawn for this multi-objective model. These samples do not include any macro-cycles and rarely halogens, mitigating the flaws of the first model. `OptiMol`-multiobj successfully constrains the samples to a chemical subspace of acceptable QED without affecting the docking results (Figure 8.6, Table 8.4). This shows `OptiMol` can handle challenging composite objectives.

Use of `OptiMol` to boost virtual screening

These results illustrate the potential of our method to speed-up lead discovery without making compromises on the exploration of the chemical space. Indeed, in our experimental setting where ZINC is used as the prior chemical space, only 1.08% of the compounds have a docking score better than -10 kcal/mol (estimated on 10k random samples from our ZINC dataset). By contrast, at the cost of docking 15k compounds to train `OptiMol`, we are able to sample from a distribution where 12.18% of the samples have docking scores better than -10. If we were to dock 100k random molecules from the prior chemical space, we therefore would expect to find 1080 compounds with a score better than -10. Training `OptiMol` required 15k docking queries, but the expected number of hits with a score better than -10 if we then dock 85k samples is 10360,

almost 10 times higher. By finding promising regions in the chemical space, the generative model learned by OptiMol thus maximizes the efficacy of a virtual screening with fixed computational cost.

Conclusion

We introduced a new small-molecules generative model using graphs and Selfies and which achieves state of the art representation with lower computational cost. By combining our efficient generative model with CbAS, we are able to optimize costly and composite scoring functions. We generate compounds with improved docking scores, bringing optimization methods closer to the ultimate goal of biological activity.

OptiMol has applications for ligand-based pipelines by generating compounds with enhanced activity towards a target. This would allow for target optimization without having to rely on previously available affinity data, which is often the case with novel targets such as the ones involved with the recent SARS-CoV-2. Structure-based pipelines can also be improved by generating focused libraries for docking and saving computations on low affinity compounds, as is demonstrated by the tenfold increase in the ratio of hits at fixed computational cost. Finally OptiMol can be seen as a step towards a unified drug discovery pipeline using multiple objectives such as combining the efficient prediction of putative binders with essential pharmacokinetic profiling.

Extensions of this work could include investigating the use of different priors with specific properties or with previous knowledge about actives. Another next step is the application of the method to the simultaneous optimization of several targets, or the maximization of affinity to a target while minimizing off-target affinities.

Chapter 9

Conclusion

Drug discovery is still a long and tedious process. Even after identifying a target and finding its structure, few tools manage to assist the quest for drugs. This is illustrated by the Covid situation, where the structure of the Spike protein was published just a few weeks after the pandemic outbreak but no candidate drugs exist yet. To enhance drug design, several questions about the targets, potential drugs and their interaction need to be answered. Inspired by the breakthrough of machine learning in other fields and motivated by the rising amount of structural data available, it is natural to try and answer these questions with machine learning. To do so, we need to adapt machine learning to structured data, to find the right way to model biomolecules for machine learning and to apply this approach to assist drug discovery. In this PhD, we make progress towards this general goal by making steps in all three directions.

In chapter 2, we develop a machine learning method to deal with the structure induced by the Reverse Complement symmetry and show improved performance. In chapters 3,4 and 5, we model RNA with 2.5D graphs and use this representation with geometric deep learning. We show that it again improves performance over existing methods and opens new doors for RNA biology and drug design. Finally, we develop tools readily usable for drug discovery scientists. We offer a tool to investigate the druggability of PPI sites in chapter 6, one to efficiently cluster MD trajectories in chapter 7, and one to generate molecular compounds with high scores of docking in chapter 8.

Geometric deep learning is useful for structural biology and drug design Machine learning has been used to assist structural biology and drug design for fifteen years, improving performance over other approaches in a variety of tasks. Throughout our work, we have consistently found machine learning to perform very well. This is a positive indicator that our general goal is reasonable.

Geometric deep learning methods were developed without consideration of biological data. Moreover, their development follow machine learning in competing for accuracy in artificial benchmarks such as discriminating bathtubs from airplanes represented as surfaces [Wu et al., 2015]. Their applicability to questions from structural biology is thus an open question.

Prior to this PhD, there was almost no biological application of these methods. Our results indicate that structural biology applications are fruitful and that respecting the structural data prior enhances results. Several other papers in the recent literature confirm these findings.

Moreover, our first chapter makes use of a general theoretical result about equivariant networks. This illustrates how a general theory can be applied to obtain theoretical results for specific regularities of the data. To summarize, the current fast-paced methodological developments of geometric deep learning help to model structural biological data.

Dedicated methods are needed for structural biology We established that expanding the allowable representations is beneficial for learning on structural data. This is especially true in cases where the established modeling of the molecular structures falls into one of these representations. It was illustrated in our work with the representation of RNA as 2.5D graphs, that is both established within the biologists community and now manageable with graph neural networks. We show in our work that in this ideal case, we can offer new efficient pipelines.

However, this is not always the case as there is no consensus on the representation to use for biomolecules. Therefore, we must rely on empirical performance to choose a representation rather than another. Defining a performance assumes that we have biologically relevant benchmarks, which was recently pioneered by [Townshend et al., 2021] but not yet adopted by the community. Moreover there is no canonical way to model a protein as a surface or as a graph. Since the empirical performance relies on the successive steps of modeling and learning, the benchmark does more than just comparing learning methods. Finally, the geometric deep learning field is more mathematically involved than more classical methods, which creates a negative bias against adoption of better performing but less accessible methods.

For these reasons, it is often not trivial to know which geometric deep learning model to use for a given structural biology application. This has resulted in the parallel design of tens of dedicated methods trying to bend existing geometric deep learning methods into a solution for each application. One could also argue that each application might have a different optimal modelling. In our work, we have found geometric methods to work well on discrete structures, whether it be finite groups or graphs with categorical edges. We have struggled to use equivariant networks for the continuous group of rotations and have found volumetric CNNs to outperform them for learning on protein structure. We believe however that coupling the methodological developments with their application would result in much better performance. For instance, one could imagine a neural network taking as input several representations of the protein structure and using the several distances induced to learn in the most efficient way. We will investigate this direction in future work.

Validation on biological tasks is challenging We mentioned how we ultimately rely on empirical performance to choose and develop our tools. The first step of designing a benchmark is to choose several tasks of interest that practitioners should try to solve, which was proposed by [Townshend et al., 2021]. However, there is a second step of carefully choosing measures of performance on each of those tasks. This involves using the right metrics as well as avoiding data leakage between the learning and testing sets. Indeed, the data deposited in the PDB is highly redundant and several slightly modified copies of the same object can appear. A model trained on one of these copies is expected to perform optimistically well on another.

Such caveats were pointed out recently in [Volkov et al., 2022]. The authors try to learn affinity values of protein-ligand complexes and show that models trained only using the ligand perform as well as ones trained on the complex. This is a nonsensical result that sheds light on biases in the validation. These biases have been undetected for years and several papers were published over this shaky validation. In our work, we advocate for a careful design of the splits based on a low sequence identity and different structure classes such as CATH classes. However, we acknowledge that these controls are flawed and two proteins from different CATH classes can have almost identical local structural features. Finding automated ways to challenge our models and detect artificially high performances - for instance with the use of counterfactual

explanations - is another promising research direction.

Tool adoption depends on deployment as much as performance The empirical performance we discussed are all based on *a posteriori* usage of data. However, the ultimate goal of this extracted knowledge is to be used on new unknown instances of problems, such as the design of drugs for a new target. To mimic this use case, several papers use temporal data splitting that amounts to predicting the recent past from the more ancient past. Thus, when presented with some new data like a new target, it makes sense to use the best tool according to metric performance on such a split. However, it is far from being the sole ground for this choice. Among classical other factors are interpretability of the model or its computational efficiency.

We underline the importance of the deployment of produced tools. In this fast-paced field, the ability to use a tool quickly and without disproportionate efforts is as important as performance. From our experience, it is far from standard to be able to use published tools. This hurts both the adoption of the tools and the trust one puts in results that cannot be reproduced. In our thesis, we have systematically released public, cleaned, commented versions of our code. We have additionally implemented two web servers, one graphical interface and released three python packages. We advocate for deployment to become a required publishing standard and hope that scientific journals will gradually impose reproducibility requirements and incentivize user-friendly tool packaging.

Next steps on the long way to go The final challenge for machine learning methods to actually help drug design is to identify bottleneck tasks and pain points in an already well-established pipeline. There is no demand for well packaged, fully validated geometric deep learning models that solve a trivial or anecdotal task. This is all the more true considering that, for obvious reasons, the drug discovery field is particularly risk-averse. Thus, new tool adoption is especially challenging and disrupting the pharmaceutical pipeline is arduous. This thesis modestly contributes to a wide community effort that believes there is an opportunity to disrupt it with machine learning.

A reasonable way to push in this direction is to address all questions independently, conducting specialized research. However, we believe in an integrative approach including methodological and fundamental contributions that are motivated by application's insights. It necessarily relies on interdisciplinary collaboration. Such approach has permitted the elaboration of AlphaFold [Jumper et al., 2021], arguably the most impactful scientific application of machine learning methods, that happens to address another key structural biology question. This is also the research philosophy we have tried to follow throughout this PhD and hope it will appeal to evermore researchers in the future.

Bibliography

M. Abraham, O. Dror, R. Nussinov and H. J. Wolfson. Analysis and classification of rna tertiary structures. *RNA*, 14(11):2274–2289, 2008. 61

M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess and E. Lindahl. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1:19–25, 2015. 16

C. M. Adrià, S. Garcia-Vallvé and G. Pujadas. DecoyFinder, a tool for finding decoy molecules. *Journal of Cheminformatics*, 4(S1), May 2012. doi: 10.1186/1758-2946-4-s1-p2. URL https://doi.org/10.1186/1758-2946-4-s1-p2. 50

F. Alber, F. Förster, D. Korkin, M. Topf and A. Sali. Integrating diverse data for structure determination of macromolecular assemblies. *Annu. Rev. Biochem.*, 77:443–477, 2008. 16

B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter et al. Molecular biology of the cell. *Scandinavian Journal of Rheumatology*, 32(2):125–125, 2003. ix

B. Alipanahi, A. Delong, M. T. Weirauch and B. J. Frey. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015. 13, 28

R. E. Amaro, J. Baudry, J. Chodera, O. Demir, J. A. McCammon, Y. Miao and J. C. Smith. Ensemble Docking in Drug Discovery. *Biophysical Journal*, 114(10):2271–2278, 5 2018. ISSN 00063495. doi: 10.1016/j.bpj.2018.02.038. 18, 88

B. Anderson, T. S. Hy and R. Kondor. Cormorant: Covariant molecular neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 9, 16, 27

D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers and J. Davis. Scape: Shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, jul 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073207. URL https://doi.org/10.1145/1073204.1073207. XIII, 10

M. R. Arkin, Y. Tang and J. A. Wells. Small-molecule inhibitors of protein-protein interactions: Progressing toward the reality. *Chemistry and Biology*, 21(9):1102–1114, 2014. doi: 10.1016/j.chembiol.2014.09.001. 88

A. Ashkenazi, W. J. Fairbrother, J. D. Leverson and A. J. Souers. From basic apoptosis discoveries to advanced selective bcl-2 family inhibitors. *Nature reviews drug discovery*, 16(4): 273–284, 2017. 87, 96, 97

R. Assouel, M. Ahmed, M. H. Segler, A. Saffari and Y. Bengio. Defactor: Differentiable edge factorization-based probabilistic graph generation. *arXiv preprint arXiv:1811.09766*, 2018. 109

K. Atz, F. Grisoni and G. Schneider. Geometric deep learning on molecular representations. *Nature Machine Intelligence*, pages 1–10, 2021. 16

T. Aumentado-Armstrong. Latent molecular optimization for targeted therapeutic design. *arXiv preprint arXiv:1809.02032*, 2018. 45, xiv

N. Bajwa, C. Liao and Z. Nikolovska-Coleska. Inhibitors of the anti-apoptotic Bcl-2 proteins: a patent review. *Expert Opinion on Therapeutic Patents*, 22(1):37–55, 1 2012. doi: 10.1517/13543776.2012.644274. 96

M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. Wilson and E. Bakshy. Botorch: Programmable bayesian optimization in pytorch. arxiv e-prints. *arXiv preprint arXiv:1910.06403*, 2019. lxxvii

P. J. Ballester and J. B. Mitchell. A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. *Bioinformatics*, 26(9):1169–1175, 2010. 18

M. Bantscheff, A. Scholten and A. J. Heck. Revealing promiscuous drug–target interactions by chemical proteomics. *Drug discovery today*, 14(21-22):1021–1029, 2009. 50

E. Bengio, M. Jain, M. Korablyov, D. Precup and Y. Bengio. Flow network based generative models for non-iterative diverse candidate generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27381–27394. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/e614f646836aaed9f89ce58e837e2310-Paper.pdf. 21

H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov and P. E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000. XVII, 14, 46, xii, xiii

S. Beucher. Use of watersheds in contour detection. In *Proceedings of the International Workshop on Image Processing*. CCETT, 1979. 90

R. S. Bohacek, C. McMartin and W. C. Guida. The art and practice of structure-based drug design: a molecular modeling perspective. *Medicinal research reviews*, 16(1):3–50, 1996. 15, 107, xiii

J. Boitreaud, V. Mallet, C. Oliver and J. Waldispuhl. Optimol: optimization of binding affinities in chemical space for drug discovery. *Journal of Chemical Information and Modeling*, 60(12): 5658–5666, 2020. 24, 105

D. Boscaini, J. Masci, E. Rodolà and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *Advances in neural information processing systems*, 29, 2016. 11

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. vi

G. Bouvier, N. Desdouits, M. Ferber, A. Blondel and M. Nilges. An automatic tool to analyze and cluster macromolecular conformations based on self-organizing maps. *Bioinformatics*, 31 (9):1490–1492, 2015. 101, 102

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas et al. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax. 23

F. Broccatelli and N. Brown. Best of both worlds: On the complementarity of ligand-based and structure-based virtual screening. *Journal of Chemical Information and Modeling*, 54(6): 1634–1641, May 2014. doi: 10.1021/ci5001604. URL https://doi.org/10.1021/ci5001604. 19, xiv

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. XIII, 7, 12

M. M. Bronstein, J. Bruna, T. Cohen and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. Technical Report 2104.13478, arXiv, 2021. 7, 27

D. H. Brookes and J. Listgarten. Design by adaptive sampling. *CoRR*, abs/1810.03714, 2018. URL http://arxiv.org/abs/1810.03714. 110, 111

D. H. Brookes, H. Park and J. Listgarten. Conditioning by adaptive sampling for robust design. *arXiv preprint arXiv:1901.10060*, 2019. 110, lxxviii

B. R. Brooks, C. L. Brooks III, A. D. Mackerell Jr, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels et al. Charmm: the biomolecular simulation program. *Journal of computational chemistry*, 30(10):1545–1614, 2009. 16, 102

C. Brouard, H. Shen, K. Dührkop, F. d'Alché Buc, S. Böcker and J. Rousu. Fast metabolite identification with input output kernel regression. *Bioinformatics*, 32(12):i28–i36, 2016. 14

R. C. Brown and G. Lunter. An equivariant Bayesian convolutional network predicts recombination hotspots and accurately resolves binding motifs. *Bioinformatics*, 35(13):2177–2184, 2019. 28, 31

J. Bruna, W. Zaremba, A. Szlam and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 11

H. Bunke and K. Riesen. Graph classification based on dissimilarity space embedding. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 996–1007. Springer, 2008. 69, xlviii

D. A. Case, T. E. Cheatham III, T. Darden, H. Gohlke, R. Luo, K. M. Merz Jr, A. Onufriev, C. Simmerling, B. Wang et al. The amber biomolecular simulation programs. *Journal of computational chemistry*, 26(16):1668–1688, 2005. 16

A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé and G. Pujadas. Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58–63, 2015. 47

K. Chen, M. J. Mizianty, J. Gao and L. Kurgan. A critical comparative assessment of predictions of protein-binding sites for biologically relevant organic compounds. *Structure*, 19(5):613–621, 2011. 91

Y.-C. Chen, R. Tolbert, A. M. Aronov, G. McGaughey, W. P. Walters and L. Meireles. Prediction of protein pairs sharing common active ligands using protein sequence, structure, and ligand similarity. *Journal of chemical information and modeling*, 56(9):1734–1745, 2016. xiv

A. Cherkasov, E. N. Muratov, D. Fourches, A. Varnek, I. I. Baskin, M. Cronin, J. Dearden, P. Gramatica, Y. C. Martin et al. Qsar modeling: where have you been? where are you going to? *Journal of medicinal chemistry*, 57(12):4977–5010, 2014. 19

G. Chojnowski, T. Waleń and J. M. Bujnicki. Rna bricks—a database of rna 3d motifs and their interactions. *Nucleic Acids Research*, 42(D1):D123–D131, 2014. 61

F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015. 39

T. Clackson and J. A. Wells. A hot spot of binding energy in a hormone-receptor interface. *Science*, 267(5196):383–386, 1995. doi: 10.1126/science.7529940. 87

J. Clauwaert and W. Waegeman. Novel transformer networks for improved sequence labeling in genomics. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2021. 28

D.-A. Clevert, T. Unterthiner and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. lxvi

P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff et al. ython: Freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 2009. doi: 10.1093/bioinformatics/btp163. 81

T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016. 8, 27, 28

T. Cohen, M. Weiler, B. Kicanaoglu and M. Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2019a. 27

T. S. Cohen and M. Welling. Steerable CNNs. In *International Conference on Learning Representations (ICLR)*, 2017. 9, 27, 28, 29

T. S. Cohen, M. Geiger, J. Köhler and M. Welling. Spherical CNNs. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018. 9, 27

T. S. Cohen, M. Geiger and M. Weiler. A General Theory of Equivariant CNNs on Homogeneous Spaces. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019b. 9, 27, 31, xix

F. Collins, E. Lander, J. Rogers, R. Waterston and I. Conso. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945, 2004. 14, x

Q. Cong, I. Anishchenko, S. Ovchinnikov and D. Baker. Protein interaction networks revealed by proteome coevolution. *Science*, 365(6449):185–189, 2019. 87

S. T. Crooke, J. L. Witztum, C. F. Bennett and B. F. Baker. Rna-targeted therapeutics. *Cell metabolism*, 27(4):714–739, 2018. 79

J. A. Cruz and E. Westhof. Sequence-based identification of 3d structural modules in RNA with RMDetect. *Nature methods*, 8(6):513, 2011. 43, 44, 62, 66

F. Da Silva, J. Desaphy and D. Rognan. Ichem: A versatile toolkit for detecting, comparing, and predicting protein–ligand interactions. *ChemMedChem*, 13(6):507–510, 2018. 88

S. D'Aguanno and D. Del Bufalo. Inhibition of Anti-Apoptotic Bcl-2 Proteins in Preclinical and Clinical Studies: Current Overview in Cancer. *Cells*, 9(5), 2020. doi: 10.3390/cells9051287. 87

B. Dai and C. Bailey-Kellogg. Protein interaction interface region prediction by geometric deep learning. *Bioinformatics*, 2021. 87, 93

H. Dai, Y. Tian, B. Dai, S. Skiena and L. Song. Syntax-directed variational autoencoder for structured data. *CoRR*, abs/1802.08786, 2018. URL http://arxiv.org/abs/1802.08786. 16, 109, 113, 115

F. Damani, V. Sresht and S. Ra. Black box recursive translations for molecular optimization, 2019. 109

H. David-Eden, A. S. Mankin and Y. Mandel-Gutfreund. Structural signatures of antibiotic binding sites on the ribosome. *Nucleic acids research*, 38(18):5982–5994, 2010. 43, 46, 52

M. Defferrard, X. Bresson and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016. 11

W. L. DeLano et al. Pymol: An open-source molecular graphics tool. *CCP4 Newsletter on protein crystallography*, 40(1):82–92, 2002. 88, 98

J. Desaphy, K. Azdimousa, E. Kellenberger and D. Rognan. Comparison and druggability prediction of protein–ligand binding sites from pharmacophore-annotated cavity shapes. *J. Chem. Inf. Model.*, 52(8):2287–2299, aug 2012. doi: 10.1021/ci300184x. 17

P. D'haeseleer. What are dna sequence motifs? *Nature biotechnology*, 24(4):423–425, 2006. 61

S. Dieleman, J. De Fauw and K. Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International conference on machine learning*, pages 1889–1898. PMLR, 2016. 9

T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, page 1–15, Berlin, Heidelberg, 2000. Springer-Verlag. 37

M. D. Disney, A. M. Winkelsas, S. P. Velagapudi, M. Southern, M. Fallahi and J. L. Childs-Disney. Inforna 2.0: A platform for the sequence-based design of small molecules targeting structured rnas. *ACS chemical biology*, 11(6):1720–1728, 2016. 45, 55

M. Djelloul. *Algorithmes de graphes pour la recherche de motifs récurrents dans les structures tertiaires d'ARN*. PhD thesis, Université Paris Sud-Paris XI, 2009. 61, 62, 63, 71, lx

M. Djelloul and A. Denise. Automated motif extraction and classification in rna tertiary structures. *RNA*, 14:2489–2497, 12 2008. ISSN 14699001. doi: 10.1261/rna.1061108. URL http://www.rnajournal.org/cgi/doi/10.1261/rna.1061108. 79

A. Donlic and A. E. Hargrove. Targeting RNA in mammalian systems with small molecules. *Wiley Interdisciplinary Reviews: RNA*, 9(4):e1477, 2018. 43

J. L. Durant, B. A. Leland, D. R. Henry and J. G. Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and computer sciences*, 42(6):1273–1280, 2002. 16, 47, viii

D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015. 12, 16, 19, 47, 111

N. Dym and H. Maron. On the Universality of Rotation Equivariant Point Cloud Networks. *arXiv preprint arXiv:2010.02449*, 2020. 9, 27, 34

D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11 (Feb):625–660, 2010. 48

P. Ertl. Cheminformatics analysis of organic substituents: identification of the most common substituents, calculation of substituent properties, and automatic identification of drug-like bioisosteric groups. *Journal of chemical information and computer sciences*, 43(2):374–380, 2003. 15, xiii

C. Esteves. Theoretical aspects of group equivariant neural networks. *arXiv preprint arXiv:2004.05154*, 2020. 9, 27

R. Evans, M. O'Neill, A. Pritzel, N. Antropova, A. W. Senior, T. Green, A. Žídek, R. Bates, S. Blackwell et al. Protein complex prediction with alphafold-multimer. *BioRxiv*, 2021. xii

S. Eyrisch and V. Helms. What induces pocket openings on protein surface patches involved in protein–protein interactions? *Journal of computer-aided molecular design*, 23(2):73–86, 2009. 98

S. Eyrisch, J. L. Medina-Franco and V. Helms. Transient pockets on xiap-bir2: toward the characterization of putative binding sites of small-molecule xiap inhibitors. *Journal of molecular modeling*, 18(5):2031–2042, 2012. 88

I. Ezkurdia, D. Juan, J. M. Rodriguez, A. Frankish, M. Diekhans, J. Harrow, J. Vazquez, A. Valencia and M. L. Tress. Multiple evidence strands suggest that there may be as few as 19 000 human protein-coding genes. *Human molecular genetics*, 23(22):5866–5878, 2014. 14

A. Fire, S. Xu, M. K. Montgomery, S. A. Kostas, S. E. Driver and C. C. Mello. Potent and specific genetic interference by double-stranded rna in caenorhabditis elegans. *Nature*, 391: 806–811, 2 1998. doi: 10.1038/35888. 79

A. Fout, J. Byrd, B. Shariat and A. Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017. 15

S. M. Freier, R. Kierzek, J. A. Jaeger, N. Sugimoto, M. H. Caruthers, T. Neilson and D. H. Turner. Improved free-energy parameters for predictions of RNA duplex stability. *Proceedings of the National Academy of Sciences*, 83(24):9373–9377, 1986. 43

F. Fuchs, D. Worrall, V. Fischer and M. Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1970–1981. Curran Associates, Inc., 2020. 9, 15, 36

K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982. 11

D. H. Fuller and P. Berglund. Amplifying rna vaccine development. *New England Journal of Medicine*, 2020. doi: 10.1056/nejmcibr2009737. 79

P. Gainza, F. Sverrisson, F. Monti, E. Rodola, D. Boscaini, M. Bronstein and B. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020. 15, 87

Y. Gal et al. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016. vi

M. Gao and J. Skolnick. A Comprehensive Survey of Small-Molecule Binding Pockets in Proteins. *PLoS Computational Biology*, 9(10), 2013. ISSN 1553734X. doi: 10.1371/journal.pcbi.1003302. 91, xiv

W. Gao and C. W. Coley. The synthesizability of molecules proposed by generative models. *Journal of chemical information and modeling*, 60(12):5714–5723, 2020. 16

W. Gao, R. Mercado and C. W. Coley. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. *arXiv preprint arXiv:2110.06389*, 2021. 16

X. Gao, B. Xiao, D. Tao and X. Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010. xlviii

A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012. xiii

P. Ge, S. Islam, C. Zhong and S. Zhang. De novo discovery of structural motifs in rna 3d structures through clustering. *Nucleic Acids Research*, 46(9):4783–4793, 2018. 61

M. Geiger, T. Smidt, A. M., B. K. Miller, W. Boomsma, B. Dice, K. Lapchevskyi, M. Weiler, M. Tyszkiewicz et al. Euclidean neural networks: e3nn, Apr. 2022. URL https://doi.org/10.5281/zenodo.6459381. 9

M. Ghandi, D. Lee, M. Mohammad-Noori and M. A. Beer. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput Biol*, 10(7):e1003711, 2014. 28

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. 12

R. C. Glen, A. Bender, C. H. Arnby, L. Carlsson, S. Boyer and J. Smith. Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to adme. *IDrugs*, 9 (3):199, 2006. 16, 47

V. Gligorijević, P. D. Renfrew, T. Kosciolek, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):1–14, 2021. 13

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 et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276, 2018. 16, 21, 44, 45, 57, 107, 109, lxxvii

R. González-Alemán, D. Hernández-Castillo, A. Rodríguez-Serradet, J. Caballero, E. W. Hernández-Rodríguez and L. Montero-Cabrera. Bitclust: Fast geometrical clustering of long molecular dynamics simulations. *Journal of Chemical Information and Modeling*, 60(2):444–448, 2019. 101

I. Goodfellow, Y. Bengio and A. Courville. *Deep learning*. MIT press, 2016. 89

S. Graham, D. Epstein and N. Rajpoot. Dense steerable filter cnns for exploiting rotational symmetry in histology images. *IEEE Transactions on Medical Imaging*, 39(12):4124–4136, 2020. 9, 27

R.-R. Griffiths and J. M. Hernández-Lobato. Constrained bayesian optimization for automatic chemical design, 2017. 109, lxxvii

S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna and S. R. Eddy. Rfam: an rna family database. *Nucleic Acids Research*, 31(1):439–441, 2003. 61

A. R. Gruber, R. Lorenz, S. H. Bernhart, R. Neuböck and I. L. Hofacker. The vienna rna websuite. *Nucleic acids research*, 36(suppl_2):W70–W74, 2008. 14

R. Grünberg, M. Nilges and J. Leckner. Flexibility and conformational entropy in protein-protein binding. *Structure*, 14(4):683–693, 2006. 14

E. Guarnera, R. Pellarin and A. Caflisch. How does a simplified-sequence protein fold? *Biophysical journal*, 97(6):1737–1746, 2009. 102

J. D. Guest, T. Vreven, J. Zhou, I. Moal, J. R. Jeliazkov, J. J. Gray, Z. Weng and B. G. Pierce. An expanded benchmark for antibody-antigen docking and affinity prediction reveals insights into antibody recognition determinants. *Structure*, 29(6):606–621, 2021. 93

V. L. Guilloux, P. Schmidtke and P. Tuffery. Fpocket: An open source platform for ligand pocket detection. *BMC Bioinformatics 2009 10:1*, 10(1):1–11, 6 2009. doi: 10.1186/1471-2105-10-168. 17, 88, 91

G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias and A. Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017. 110

A. Gupta and J. Zou. Feedback gan for dna optimizes protein functions. *Nature Machine Intelligence*, 1(2):105–111, 2019. 110

A. A. Hagberg, D. A. Schult and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15, Pasadena, CA USA, 2008. 79, 81

W. Hamilton, Z. Ying and J. Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017a. 80

W. L. Hamilton, R. Ying and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017b. 48, 64, 82, xlvii

P. C. Hawkins, A. G. Skillman and A. Nicholls. Comparison of shape-matching and docking as virtual screening tools. *Journal of medicinal chemistry*, 50(1):74–82, 2007. xiv

M. Henaff, J. Bruna and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 11

S. Heyne, F. Costa, D. Rose and R. Backofen. Graphclust: alignment-free structural clustering of local RNA secondary structures. *Bioinformatics*, 28(12):i224–i232, 2012. 48

B. Hoffmann, M. Zaslavskiy, J.-P. Vert and V. Stoven. A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3d: application to ligand prediction. *BMC Bioinformatics*, 11(1), Feb. 2010. doi: 10.1186/1471-2105-11-99. URL https://doi.org/10.1186/1471-2105-11-99. 18

J. Hoffmann, L. Maestrati, Y. Sawada, J. Tang, J. M. Sellier and Y. Bengio. Data-driven approach to encoding and decoding 3-d crystal structures. *arXiv preprint arXiv:1909.00949*, 2019. 109

M. Hoffmann and F. Noé. Generating valid euclidean distance matrices. *arXiv preprint arXiv:1910.03131*, 2019. 109

P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Physical review*, 136(3B):B864, 1964. 18

E. Hoogeboom, J. W. T. Peters, T. S. Cohen and M. Welling. Hexaconv. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 27

J. A. Howe, H. Wang, T. O. Fischmann, C. J. Balibar, L. Xiao, A. M. Galgoci, J. C. Malinverni, T. Mayhood, A. Villafania et al. Selective small-molecule inhibition of an RNA structural element. *Nature*, 526(7575):672, 2015. 43

J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. d. Groot, H. Grubmüller and A. D. MacKerell. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nature Methods 2016 14:1*, 14(1):71–73, 11 2016. doi: 10.1038/nmeth.4067. lxxi

N. Huang, B. K. Shoichet and J. J. Irwin. Benchmarking sets for molecular docking. *Journal of medicinal chemistry*, 49(23):6789–6801, 2006. 107

S. J. Hubbard, J. M. Thornton et al. naccess. *Computer Program, Department of Biochemistry and Molecular Biology, University College London*, 2(1), 1993. lxxi

J. P. Hughes, S. Rees, S. B. Kalindjian and K. L. Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011. 14, 19, xiv

W. Humphrey, A. Dalke and K. Schulten. Vmd: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996. 16

J. Husby, G. Bottegoni, I. Kufareva, R. Abagyan and A. Cavalli. Structure-based predictions of activity cliffs. *Journal of Chemical Information and Modeling*, 55(5):1062–1076, 2015. doi: 10.1021/ci500742b. 107

S. L. Hutcherson and R. Lanz. A randomized controlled clinical trial of intravitreous fomivirsen for treatment of newly diagnosed peripheral cytomegalovirus retinitis in patients with aids. *American Journal of Ophthalmology*, 133:467–474, 2002. doi: 10.1016/S0002-9394(02)01327-2. 79

M. Ilse, J. Tomczak and M. Welling. Attention-based deep multiple instance learning. In *International conference on machine learning*, pages 2127–2136. PMLR, 2018. 17, 19

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 32, lxvi

A. Ivetac and J. A. McCammon. A molecular dynamics ensemble-based approach for the mapping of druggable binding sites. *Methods in Molecular Biology*, 819:3–12, 2012. doi: 10.1007/978-1-61779-465-0{_}1. 88

Y. Ji, Z. Zhou, H. Liu and R. V. Davuluri. DNABERT: pre-trained bidirectional encoder representations from transformers model for DNA-language in genome. *Bioinformatics*, Feb. 2021. 28

D. Jiang, Z. Wu, C.-Y. Hsieh, G. Chen, B. Liao, Z. Wang, C. Shen, D. Cao, J. Wu et al. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of cheminformatics*, 13(1):1–23, 2021. 16

J. Jiménez, S. Doerr, G. Martínez-Rosell, A. S. Rose and G. De Fabritiis. Deepsite: protein-binding site predictor using 3d-convolutional neural networks. *Bioinformatics*, 33(19):3036–3042, 2017. 14, 18, 88, 89, 90

J. Jiménez, M. Skalic, G. Martinez-Rosell and G. De Fabritiis. K deep: protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks. *Journal of chemical information and modeling*, 58(2):287–296, 2018. 18, 44

W. Jin, R. Barzilay and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation, 2018a. 16, 109, 113, 115

W. Jin, K. Yang, R. Barzilay and T. S. Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *CoRR*, abs/1812.01070, 2018b. URL http://arxiv.org/abs/1812.01070. 109

W. Jin, R. Barzilay and T. Jaakkola. Hierarchical generation of molecular graphs using structural motifs. *arXiv preprint arXiv:2002.03230*, 2020a. 61

W. Jin, R. Barzilay and T. Jaakkola. Multi-objective molecule generation using interpretable substructures, 2020b. 16, 110

D. K. Johnson and J. Karanicolas. Druggable protein interaction sites are more predisposed to surface pocket formation than the rest of the protein surface. *PLoS computational biology*, 9 (3):e1002951, 2013. 88

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021. 79, 123, xii

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., 2017. lxxx

S. Kearnes, K. McCloskey, M. Berndl, V. Pande and P. Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016. 12, 16

O. T. Kim, K. Yura and N. Go. Amino acid residue doublet propensity in the protein–rna interface and its application to rna interface prediction. *Nucleic acids research*, 34(22):6450–6460, 2006. 14

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 90

D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013. 109, lxxiv

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 12

D. B. Kitchen, H. Decornez, J. R. Furr and J. Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nature reviews Drug discovery*, 3(11):935–949, 2004. 44

E. Kligun and Y. Mandel-Gutfreund. Conformational readout of RNA by small ligands. *RNA biology*, 10(6):981–989, 2013. 43

T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982. 101

R. Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint arXiv:1803.01588*, 2018. 9, 34

R. Kondor and S. Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. PMLR, 2018. 9, 27

R. Kondor, Z. Lin and S. Trivedi. Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. *Advances in Neural Information Processing Systems*, 31, 2018. 9

K. Korovina, S. Xu, K. Kandasamy, W. Neiswanger, B. Póczos, J. Schneider and E. P. Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. *ArXiv*, abs/1908.01425, 2019. 109

D. Kozakov, L. E. Grove, D. R. Hall, T. Bohnuud, S. E. Mottarella, L. Luo, B. Xia, D. Beglov and S. Vajda. The FTMap family of web servers for determining and characterizing ligand-binding hot spots of proteins. *Nature Protocols 2015 10:5*, 10(5):733–755, 4 2015. doi: 10.1038/nprot.2015.043. 88

I. Kozlovskii and P. Popov. Spatiotemporal identification of druggable binding sites using deep learning. *Communications biology*, 3(1):1–12, 2020. 17, 18, 88, 97

K. Krawczyk, X. Liu, T. Baker, J. Shi and C. M. Deane. Improving b-cell epitope prediction and its application to global antibody-antigen docking. *Bioinformatics*, 30(16):2288–2294, 2014. 93

M. Krenn, F. Häse, A. Nigam, P. Friederich and A. Aspuru-Guzik. SELFIES: a robust representation of semantically constrained graphs with an example application in chemistry. *CoRR*, abs/1905.13741, 2019. URL http://arxiv.org/abs/1905.13741. 16, 109, 111

R. Krivák and D. Hoksza. P2Rank: machine learning based tool for rapid and accurate prediction of ligand binding sites from protein structure. *Journal of Cheminformatics 2018 10:1*, 10(1):1–12, 8 2018. doi: 10.1186/S13321-018-0285-8. 88

A. Krizhevsky, I. Sutskever and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 11

D. M. Krüger and H. Gohlke. DrugScorePPI webserver: Fast and accurate in silico alanine scanning for scoring protein-protein interactions. *Nucleic Acids Research*, 38(SUPPL. 2):1–7, 2010. doi: 10.1093/nar/gkq471. 87

H. W. Kuhn and B. Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955. 65, 82, li

K. Kundert, J. E. Lucas, K. E. Watters, C. Fellmann, A. H. Ng, B. M. Heineike, C. M. Fitzsimmons, B. L. Oakes, J. Qu et al. Controlling crispr-cas9 with ligand-activated and ligand-deactivated sgrnas. *Nature communications*, 10(1):2127, 2019. 43

M. J. Kusner, B. Paige and J. M. Hernández-Lobato. Grammar variational autoencoder, 2017. XVII, 16, 109, 113, 115

D. Kuzminykh, D. Polykovskiy, A. Kadurin, A. Zhebrak, I. Baskov, S. Nikolenko, R. Shayakhmetov and A. Zhavoronkov. 3d molecular representations based on the wave transform for convolutional neural networks. *Molecular pharmaceutics*, 15(10):4378–4385, 2018. 109

M. W. Lafarge, E. J. Bekkers, J. P. Pluim, R. Duits and M. Veta. Roto-translation equivariant convolutional networks: Application to histopathology image analysis. *Medical Image Analysis*, 68:101849, 2021. 9, 27

P. T. Lang, S. R. Brozell, S. Mukherjee, E. F. Pettersen, E. C. Meng, V. Thomas, R. C. Rizzo, D. A. Case, T. L. James et al. Dock 6: Combining techniques to model RNA–small molecule complexes. *Rna*, 2009. 18

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 11

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 11

D. Lee, D. U. Gorkin, M. Baker, B. J. Strober, A. L. Asoni, A. S. McCallion and M. A. Beer. A method to predict the impact of regulatory variants from DNA sequence. *Nature genetics*, 47(8):955, 2015a. 28

J. Lee, X. Cheng, J. M. Swails, M. S. Yeom, P. K. Eastman, J. A. Lemkul, S. Wei, J. Buckner, J. C. Jeong et al. CHARMM-GUI Input Generator for NAMD, GROMACS, AMBER, OpenMM, and CHARMM/OpenMM Simulations Using the CHARMM36 Additive Force Field. *Journal of Chemical Theory and Computation*, 12(1):405–413, 1 2015b. doi: 10.1021/ACS.JCTC.5B00935. lxxi

S. Lemieux and F. Major. Automated extraction and classification of rna tertiary structure cyclic motifs. *Nucleic Acids Research*, 34(8):2340–2346, 2006. 62

N. B. Leontis and E. Westhof. Conserved geometrical base-pairing patterns in RNA. *Quarterly reviews of biophysics*, 31(4):399–455, 1998. 43

N. B. Leontis and E. Westhof. Geometric nomenclature and classification of rna base pairs. *RNA*, 7(4):499–512, 2001. 43, 61, 64, 80

N. B. Leontis and E. Westhof. Analysis of RNAmotifs. *Current opinion in structural biology*, 13(3):300–308, 2003. 43

N. B. Leontis, J. Stombaugh and E. Westhof. The non-watson–crick base pairs and their associated isostericity matrices. *Nucleic acids research*, 30(16):3497–3531, 2002. 14

N. B. Leontis, A. Lescoute and E. Westhof. The building blocks and motifs of rna architecture. *Current opinion in structural biology*, 16(3):279–287, 2006. 43, 61

A. Lescoute, N. B. Leontis, C. Massire and E. Westhof. Recurrent structural rna motifs, isostericity matrices and sequence alignments. *Nucleic Acids Research*, 33(8):2395–2409, 2005. 61

J. J. Levy, A. J. Titus, C. L. Petersen, Y. Chen, L. A. Salas and B. C. Christensen. MethylNet: an automated and modular deep learning approach for DNA methylation analysis. *BMC bioinformatics*, 21:108, 2020. 28

Q. Liang, P. W. Bible, Y. Liu, B. Zou and L. Wei. DeepMicrobes: taxonomic classification for metagenomics with deep learning. *NAR genom. bioinform.*, 2:lqaa009, Mar. 2020. 27

W. Liang. Segmenting DNA sequence into words based on statistical language model. *Nature Precedings*, pages 1–1, 2012. 34

E. Lieberman-Aiden, N. L. Van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo et al. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *science*, 326(5950):289–293, 2009. 13

C. A. Lipinski. Lead-and drug-like compounds: the rule-of-five revolution. *Drug Discovery Today: Technologies*, 1(4):337–341, 2004. 57

Q. Liu, M. Allamanis, M. Brockschmidt and A. Gaunt. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems 31*, pages 7795–7804, 2018. 109

Q. Liu, P. S. Wang, C. Zhu, B. B. Gaines, T. Zhu, J. Bi and M. Song. OctSurf: Efficient hierarchical voxel-based molecular surface representation for protein-ligand affinity prediction. *Journal of Molecular Graphics and Modelling*, 105:107865, 6 2021. doi: 10.1016/J.JMGM.2021.107865. 88

X. Liu, S. Dai, Y. Zhu, P. Marrack and J. W. Kappler. The structure of a bcl-xl/bim fragment complex: implications for bim function. *Immunity*, 19(3):341–352, 2003. 97

Y.-C. Liu, C.-H. Yang, K.-T. Chen, J.-R. Wang, M.-L. Cheng, J.-C. Chung, H.-T. Chiu and C. L. Lu. R3d-blast: a search tool for similar rna 3d substructures. *Nucleic acids research*, 39(suppl_2):W45–W49, 2011. 62

R. Lorenz, S. H. Bernhart, C. H. Zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler and I. L. Hofacker. ViennaRNA package 2.0. *Algorithms for molecular biology*, 6(1):26, 2011. 43

H. Lu, Q. Zhou, J. He, Z. Jiang, C. Peng, R. Tong and J. Shi. Recent advances in the development of protein–protein interactions modulators: mechanisms and clinical trials. *Signal transduction and targeted therapy*, 5(1):1–23, 2020. 87

X.-J. Lu, H. J. Bussemaker and W. K. Olson. Dssr: an integrated software tool for dissecting the spatial structure of rna. *Nucleic acids research*, 43(21):e142–e142, 2015. 81

J. Luo, W. Wei, J. Waldispühl and N. Moitessier. Challenges and current status of computational methods for docking small molecules to nucleic acids. *European journal of medicinal chemistry*, 168:414–425, 2019. 18, 44, 45

P. D. Lyne. Structure-based virtual screening: an overview. *Drug discovery today*, 7(20):1047–1055, 2002. xiv

L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. XVIII, 69, xxxi

J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 65

M. Magnus, M. Antczak, T. Zok, J. Wiedemann, P. Lukasiak, Y. Cao, J. M. Bujnicki, E. Westhof, M. Szachniuk et al. Rna-puzzles toolkit: a computational resource of rna 3d structure benchmark datasets, structure manipulation, and evaluation tools. *Nucleic acids research*, 2020. 79

J. M. I. Makunin. Non-coding rna. *Hum Mol Genet*, 15(Spec No 1):R17–29, 2006. 79

V. Mallet and J.-P. Vert. Reverse-complement equivariant networks for dna sequences. *Advances in Neural Information Processing Systems*, 34, 2021. 21, 25

V. Mallet, C. G. Oliver, N. Moitessier and J. Waldispuhl. Leveraging binding-site structure for drug discovery with point-cloud methods. *arXiv preprint arXiv:1905.12033*, 2019. 45

V. Mallet, M. Nilges and G. Bouvier. quicksom: Self-organizing maps on gpus for clustering of molecular dynamics trajectories. *Bioinformatics*, 37(14):2064–2065, 2021. 23, 99

V. Mallet, L. Checa Ruano, A. Moine Franel, M. Nilges, K. Druart, G. Bouvier and O. Sperandio. Indeep: 3d fully convolutional neural networks to assist in silico drug design on protein–protein interactions. *Bioinformatics*, 38(5):1261–1268, 2022a. 23, 85

V. Mallet, C. Oliver, J. Broadbent, W. L. Hamilton and J. Waldispühl. Rnaglib: a python package for rna 2.5 d graphs. *Bioinformatics*, 38(5):1458–1459, 2022b. 23, 77

J. Masci, D. Boscaini, M. Bronstein and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 11

D. Maticzka, S. J. Lange, F. Costa and R. Backofen. Graphprot: Modeling binding preferences of rna-binding proteins. *Genome Biology*, 15:R17, 1 2014. doi: 10.1186/gb-2014-15-1-r17. URL http://genomebiology.biomedcentral.com/articles/10.1186/gb-2014-15-1-r17. 14

A. T. McNutt, P. Francoeur, R. Aggarwal, T. Masuda, R. Meli, M. Ragoza, J. Sunseri and D. R. Koes. Gnina 1.0: molecular docking with deep learning. *Journal of cheminformatics*, 13(1): 1–20, 2021. 18

R. Menegaux and J.-P. Vert. Continuous embeddings of DNA sequencing reads and application to metagenomics. *Journal of Computational Biology*, 26(6):509–518, 2019. 27, 34

S. Mir, Y. Alhroub, S. Anyango, D. R. Armstrong, J. M. Berrisford, A. R. Clark, M. J. Conroy, J. M. Dana, M. Deshpande et al. PDBe: Towards reusable data delivery infrastructure at protein data bank in Europe. *Nucleic Acids Research*, 46(D1):D486–D492, 2018. doi: 10.1093/nar/gkx1070. lxiv

L. Möller, L. Guerci, C. Isert, K. Atz and G. Schneider. Translating from proteins to ribonucleic acids for ligand-binding site detection. *Molecular Informatics*, 2022. 14, 18

D. Monet, N. Desdouits, M. Nilges and A. Blondel. mkgridXf: Consistent Identification of Plausible Binding Sites Despite the Elusive Nature of Cavities and Grooves in Protein Dynamics. *Journal of Chemical Information and Modeling*, 59(8):3506–3518, 8 2019. doi: 10.1021/ACS.JCIM.9B00103. 88

F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017. 11

V. Moosavi, S. Packmann and I. Vallés. Sompy: A python library for self organizing map (som), 2014. GitHub.[Online]. Available: https://github. com/sevamoo/SOMPY. 101

Y. Murakami and K. Mizuguchi. Applying the naïve bayes classifier with kernel density estimation to the prediction of protein–protein interaction sites. *Bioinformatics*, 26(15):1841–1848, 2010. 87

S. K. Mylonas, A. Axenopoulos and P. Daras. DeepSurf: a surface-based deep learning approach for the prediction of ligand binding sites on proteins. *Bioinformatics*, 37(12):1681–1690, 7 2021. doi: 10.1093/BIOINFORMATICS/BTAB009. 18, 88

M. M. Mysinger, M. Carchia, J. J. Irwin and B. K. Shoichet. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, 55(14):6582–6594, 2012. 50, 111, 114

P. Nissen, J. A. Ippolito, N. Ban, P. B. Moore and T. A. Steitz. Rna tertiary interactions in the large ribosomal subunit: the a-minor motif. *Proceedings of the National Academy of Sciences*, 98(9):4899–4903, 2001. 62

N. M. O'Boyle, C. Morley and G. R. Hutchison. Pybel: a python wrapper for the openbabel cheminformatics toolkit. *Chemistry Central Journal*, 2(1):5, 2008. 47

P. J. O'Brien. High-content analysis in toxicology: screening substances for human toxicity potential, elucidating subcellular mechanisms and in vivo use as translational safety biomarkers. *Basic & clinical pharmacology & toxicology*, 115(1):4–17, 2014. 19

M. Olivecrona, T. Blaschke, O. Engkvist and H. Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017. 21, 110, lxxx

C. Oliver, V. Mallet, R. S. Gendron, V. Reinharz, W. L. Hamilton, N. Moitessier and J. Waldispühl. Augmented base pairing networks encode rna-small molecule binding preferences. *Nucleic acids research*, 48(14):7690–7699, 2020. 22, 41, 61, 79, 82

C. Oliver, V. Mallet, P. Philippopoulos, W. L. Hamilton and J. Waldispühl. Vernal: a tool for mining fuzzy network motifs in rna. *Bioinformatics*, 38(4):970–976, 2022. 22, 59, 79, 80

S. Olsson and F. Noé. Dynamic graphical models of molecular kinetics. *Proceedings of the National Academy of Sciences*, 116(30):15001–15006, 2019. 101

K. Onimaru, O. Nishimura and S. Kuraku. Predicting gene regulatory regions with a convolutional neural network for processing double-strand genome sequence information. *PloS one*, 15(7):e0235748, 2020. 28, 31

M. Oubounyt, Z. Louadi, H. Tayara and K. T. Chong. DeePromoter: Robust promoter predictor using deep learning. *Frontiers in genetics*, 10:286, 2019. 28

T. Ozaki and A. Nakagawara. Role of p53 in cell death and human cancers. *Cancers*, 3 (1):994–1013, Mar. 2011. doi: 10.3390/cancers3010994. URL https://doi.org/10.3390/cancers3010994. XVII, xii

A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler and C. E. Leisersen. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. *arXiv preprint arXiv:1902.10191*, 2019. 57

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein et al. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 39, 45, 65, 81, 101, liii, lxxvi

S. Pérot, O. Sperandio, M. A. Miteva, A.-C. Camproux and B. O. Villoutreix. Druggable pockets and binding site centric chemical space: a paradigm shift in drug discovery. *Drug discovery today*, 15(15-16):656–667, 2010. 17

M. Pertea. The human transcriptome: an unfinished story. *Genes*, 3(3):344–360, 2012. x

A. I. Petrov, C. L. Zirbel and N. B. Leontis. Automated classification of rna 3d motifs and the rna 3d motif atlas. *RNA*, 19:1327–1340, 10 2013a. doi: 10.1261/rna.039438.113. URL http://www.rnajournal.org/cgi/doi/10.1261/rna.039438.113. 81

A. I. Petrov, C. L. Zirbel and N. B. Leontis. Automated classification of rna 3d motifs and the rna 3d motif atlas. *RNA*, 19(10):1327–1340, 2013b. 43, 44, 46, 56, 61, 62, 63, 64, 71, lx

E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng and T. E. Ferrin. Ucsf chimera?a visualization system for exploratory research and analysis. *Journal of computational chemistry*, 25(13):1605–1612, 2004. XIV, 44

P. Pfeffer and H. Gohlke. DrugScoreRNA knowledge-based scoring function to predict RNA-ligand interactions. *Journal of chemical information and modeling*, 47(5):1868–1876, 2007. 45

A. Philips, K. Milanowska, G. Lach, M. Boniecki, K. Rother and J. M. Bujnicki. MetalionRNA: computational predictor of metal-binding sites in RNA structures. *Bioinformatics*, 28(2): 198–205, 2012. 46, 56

A. Philips, K. Milanowska, G. Łach and J. M. Bujnicki. Ligandrna: computational predictor of RNA–ligand interactions. *RNA*, 2013. 45

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*, 27(8):675–679, 2013. 15, xiii

D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy et al. Molecular sets (MOSES): A benchmarking platform for molecular generation models. *CoRR*, abs/1811.12823, 2018. XXII, 113, lxxvi

M. Popenda, M. Szachniuk, M. Blazewicz, S. Wasik, E. K. Burke, J. Blazewicz and R. W. Adamiak. Rna frabase 2.0: an advanced web-accessible database with the capacity to search the three-dimensional fragments within rna structures. *BMC bioinformatics*, 11(1):1–12, 2010. 62

M. Popova, O. Isayev and A. Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018. 110

E. B. Porter, J. T. Polaski, M. M. Morck and R. T. Batey. Recurrent RNA motifs as scaffolds for genetically encodable small-molecule biosensors. *Nature chemical biology*, 13(3):295, 2017. 43

O. Prykhodko, S. V. Johansson, P.-C. Kotsias, J. Arús-Pous, E. J. Bjerrum, O. Engkvist and H. Chen. A de novo molecular generation method using latent vector based generative adversarial network. *Journal of Cheminformatics*, 11(1):74, 2019. 21, 110

D. Quang and X. Xie. Factornet: a deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data. *Methods*, 166:40–47, 2019. 28

J. Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003. xxxiii

S. Rauch, K. A. Jones and B. C. Dickinson. Small molecule-inducible rna-targeting systems for temporal control of rna regulation. *ACS central science*, 6(11):1987–1996, 2020. 43

V. Reinharz, A. Soulé, E. Westhof, J. Waldispühl and A. Denise. Mining for recurrent long-range interactions in rna structures reveals embedded hierarchies in network families. *Nucleic Acids Research*, 46(8):3841–3851, 2018a. 44, 56, 61, 62, 63, 64, 71, lx

V. Reinharz, A. Soulé, E. Westhof, J. Waldispühl and A. Denise. Mining for recurrent long-range interactions in rna structures reveals embedded hierarchies in network families. *Nucleic Acids Research*, 46:3841–3851, 5 2018b. ISSN 13624962. doi: 10.1093/nar/gky197. URL http://carnaval.lri.fr. 79

L. F. Ribeiro, P. H. Saverese and D. R. Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394, 2017. 48, l

D. Rogers and M. Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010. 16, 47

J. Roll, C. L. Zirbel, B. Sweeney, A. I. Petrov and N. Leontis. Jar3d webserver: Scoring and aligning rna loop sequences to known 3d motifs. *Nucleic Acids Research*, 44(W1):W320–W327, 2016. 46, 61

O. Ronneberger, P. Fischer and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 89, lxvi

P. W. Rose, A. Prlić, A. Altunkaya, C. Bi, A. R. Bradley, C. H. Christie, L. D. Costanzo, J. M. Duarte, S. Dutta et al. The rcsb protein data bank: integrative view of protein, gene and 3d structural information. *Nucleic Acids Research*, page gkw1000, 2016. 62

M. Rosell and J. Fernández-Recio. Docking-based identification of small-molecule binding sites at protein-protein interfaces. *Computational and structural biotechnology journal*, 18:3750–3761, 2020. 91

R. Sanchez-Garcia, C. O. S. Sorzano, J. M. Carazo and J. Segura. Bipspi: a method for the prediction of partner-specific protein–protein interfaces. *Bioinformatics*, 35(3):470–477, 2019. 87

B. Sanchez-Lengeling and A. Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018. 107

M. C. Sanguinetti and M. Tristani-Firouzi. herg potassium channels and cardiac arrhythmia. *Nature*, 440(7083):463–469, 2006. 19

R. Sarrazin-Gendron, V. Reinharz, C. G. Oliver, N. Moitessier and J. Waldispühl. Automated, customizable and efficient identification of 3d base pair modules with bayespairing. *Nucleic Acids Research*, 47(7):3321–3332, 2019. 44, 46, 56, 57, 61, 79

M. Sarver, C. L. Zirbel, J. Stombaugh, A. Mokdad and N. B. Leontis. Fr3d: finding local and composite recurrent structural motifs in RNA 3d structures. *Journal of mathematical biology*, 56(1-2):215–252, 2008. 46

V. G. Satorras, E. Hoogeboom and M. Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332. PMLR, 2021. 9, 15

A. L. Satz, A. Brunschweiger, M. E. Flanagan, A. Gloger, N. J. V. Hansen, L. Kuai, V. B. K. Kunig, X. Lu, D. Madsen et al. DNA-encoded chemical libraries. *Nature Reviews Methods Primers*, 2(1), Jan. 2022. doi: 10.1038/s43586-021-00084-5. URL https://doi.org/10.1038/s43586-021-00084-5. xiv

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018. 12, 46, 47, 65, xxxii, liii

G. Schneider. Automating drug discovery. *Nature reviews drug discovery*, 17(2):97, 2018. 108

K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017. 16

K. T. Schütt, O. T. Unke and M. Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. *arXiv preprint arXiv:2102.03150*, 2021. 27

D. Schwalbe-Koda and R. Gómez-Bombarelli. Generative models for automatic chemical design. *arXiv preprint arXiv:1907.01632*, 2019. 107

D. E. Scott, A. R. Bayly, C. Abell and J. Skidmore. Small molecules, big targets: Drug discovery faces the protein-protein interaction challenge, 2016. 97

R. Sender, S. Fuchs and R. Milo. Revised estimates for the number of human and bacteria cells in the body. *PLoS biology*, 14(8):e1002533, 2016. ix

A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020. xii

J.-P. Serre. *Linear representations of finite groups*, volume 42. Springer, 1977. 7

N. Sharp, S. Attaiki, K. Crane and M. Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022. 11

N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pages 488–495, 2009. l

B. K. Shoichet, S. L. McGovern, B. Wei and J. J. Irwin. Lead discovery using molecular docking. *Current opinion in chemical biology*, 6(4):439–446, 2002. 18, 107

A. Shrikumar, P. Greenside and A. Kundaje. Reverse-complement parameter sharing improves deep learning models for genomics. *bioRxiv*, page 103663, 2017. 28, 31, 32, 34, 36, xxi

I. Sillitoe, N. Bordin, N. Dawson, V. P. Waman, P. Ashford, H. M. Scholes, C. S. Pang, L. Woodridge, C. Rauer et al. Cath: increased structural coverage of functional space. *Nucleic acids research*, 49(D1):D266–D273, 2021. 89, lxiv

M. Simonovsky and J. Meyers. Deeplytough: learning structural comparison of protein binding sites. *Journal of chemical information and modeling*, 60(4):2356–2366, 2020. 15

M. Skalic, J. Jiménez, D. Sabbadin and G. De Fabritiis. Shape-based generative modeling for de novo drug design. *Journal of Chemical Information and Modeling*, 59(3):1205–1214, 2019a. doi: 10.1021/acs.jcim.8b00706. URL https://doi.org/10.1021/acs.jcim.8b00706. PMID: 30762364. 109, 110

M. Skalic, D. Sabbadin, B. Sattarov, S. Sciabola and G. De Fabritiis. From target to drug: Generative modeling for the multimodal structure-based ligand design. *Molecular Pharmaceutics*, 16(10):4282–4291, 2019b. doi: 10.1021/acs.molpharmaceut.9b00634. URL https://doi.org/10.1021/acs.molpharmaceut.9b00634. PMID: 31437001. XVII, XIX, 21, 110, 116, 117, 118, lxxx, lxxxi

M. Skalic, A. Varela-Rial, J. Jiménez, G. Martínez-Rosell and G. De Fabritiis. Ligvoxel: inpainting binding pockets using 3d-convolutional neural networks. *Bioinformatics*, 35(2):243–250, 2019c. 14, 95

J. Smith and A. L. Willis. Aspirin selectively inhibits prostaglandin production in human platelets. *Nature New Biology*, 231(25):235–237, 1971. xiv

A. Soulé, V. Reinharz, R. Sarrazin-Gendron, A. Denise and J. Waldispühl. Finding recurrent rna structural networks with fast maximal common subgraphs of edge-colored graphs. *PLoS computational biology*, 17(5):e1008990, 2021. 62

O. Sperandio, C. H. Reynès, A.-C. Camproux and B. O. Villoutreix. Rationalizing the chemical space of protein–protein interaction inhibitors. *Drug discovery today*, 15(5-6):220–229, 2010. 87

D. V. D. Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark and H. J. C. Berendsen. GROMACS: Fast, flexible, and free. *Journal of Computational Chemistry*, 26(16):1701–1718, 12 2005. doi: 10.1002/JCC.20291. lxxi

H. Stärk, O.-E. Ganea, L. Pattanaik, R. Barzilay and T. Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. *arXiv preprint arXiv:2202.05146*, 2022. 15, 18

M. M. Stepniewska-Dziubinska, P. Zielenkiewicz and P. Siedlecki. Improving detection of protein-ligand binding sites with 3d segmentation. *Scientific reports*, 10(1):1–9, 2020. 14, 18, 88, 89, 90

T. Sterling and J. J. Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015. 113, 114, 116, xiii

J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4): 688–702.e13, Feb. 2020. doi: 10.1016/j.cell.2020.01.021. URL https://doi.org/10.1016/j.cell.2020.01.021. 17

J. Stombaugh, C. L. Zirbel, E. Westhof and N. B. Leontis. Frequency and isostericity of rna base pairs. *Nucleic Acids Research*, 37(7):2294–2312, 2009. 14, 64, 82, xxxii, xlvi, xlviii

G. D. Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, 2000. 28

H. Su, Z. Peng and J. Yang. Recognition of small molecule–RNA binding sites using RNA sequence and structure. *Bioinformatics*, 37(1):36–42, Jan. 2021. doi: 10.1093/bioinformatics/btaa1092. URL https://doi.org/10.1093/bioinformatics/btaa1092. 18

F.-Y. Sun, J. Hoffmann and J. Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019. 48

J. Sun, N. Jeliazkova, V. Chupakhin, J.-F. Golib-Dzib, O. Engkvist, L. Carlsson, J. Wegner, H. Ceulemans, I. Georgiev et al. Excape-db: an integrated large scale dataset facilitating big data analysis in chemogenomics. *Journal of cheminformatics*, 9(1):17, 2017a. 113, lxxx

L.-Z. Sun, D. Zhang and S.-J. Chen. Theory and modeling of RNA structure and interactions with metal ions and small molecules. *Annual review of biophysics*, 46:227–246, 2017b. 45

F. Sverrisson, J. Feydy, B. E. Correia and M. M. Bronstein. Fast end-to-end learning on protein surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15272–15281, 2021. 15

A. Tampuu, Z. Bzhalava, J. Dillner and R. Vicente. ViraMiner: Deep learning on raw DNA sequences for identifying viral genomes in human samples. *PloS one*, 14:e0222271, 2019. 27, 28

C. Theis, C. Höner zu Siederdissen, I. L. Hofacker and J. Gorodkin. Automated identification of rna 3d modules with discriminative power in rna structural alignments. *Nucleic acids research*, 41(22):9999–10009, 2013. 62

B. C. Thiel, I. K. Beckmann, P. Kerpedjiev and I. L. Hofacker. 3d based on 2d: Calculating helix angles and stacking patterns using forgi 2.0, an rna python library centered on secondary structure elements. *F1000Research*, 8, 2019. 55

J. R. Thomas and P. J. Hergenrother. Targeting rna with small molecules. *Chemical reviews*, 108(4):1171–1224, 2008. 43

N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff and P. Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018. 9, 16, 19, 27, 34

I. Tinoco Jr and C. Bustamante. How RNAfolds. *Journal of molecular biology*, 293(2):271–281, 1999. 43

R. Torchet, K. Druart, L. C. Ruano, A. Moine-Franel, H. Borges, O. Doppelt-Azeroual, B. Brancotte, F. Mareuil, M. Nilges et al. The iPPI-DB initiative: a community-centered database of protein–protein interaction modulators. *Bioinformatics*, 1 2021. doi: 10.1093/bioinformatics/btaa1091. 18, 87, 88, 89

W. Torng and R. B. Altman. Graph convolutional neural networks for predicting drug-target interactions. *Journal of Chemical Information and Modeling*, 59(10):4131–4149, 2019. 15, 45

R. Townshend, R. Bedi, P. Suriana and R. Dror. End-to-end learning on 3d protein structure for interface prediction. *Advances in Neural Information Processing Systems*, 32:15642–15651, 2019. 15, 87

R. J. L. Townshend, M. Vögele, P. A. Suriana, A. Derry, A. Powers, Y. Laloudakis, S. Balachandar, B. Jing, B. M. Anderson et al. Atom3d: Tasks on molecules in three dimensions. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. 13, 15, 122

O. Trott and A. J. Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010. 18, 107, 111

N. Tuncbag, O. Keskin and A. Gursoy. HotPoint: hot spot prediction server for protein interfaces. *Nucleic Acids Research*, 38(Web Server issue):W402, 5 2010. doi: 10.1093/NAR/GKQ323. 87

M. Uhl, F. Heyl, R. Backofen et al. Graphprot2: A novel deep learning-based method for predicting binding sites of RNA-binding proteins. *bioRxiv*, page 850024, 2019. 14, 56

E. van der Pol, D. Worrall, H. van Hoof, F. Oliehoek and M. Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020. 9

L. T. Vassilev, B. T. Vu, B. Graves, D. Carvajal, F. Podlaski, Z. Filipovic, N. Kong, U. Kammlott, C. Lukacs et al. In vivo activation of the p53 pathway by small-molecule antagonists of mdm2. *Science*, 303(5659):844–848, 2004. XVII, xii

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. xxxii

K. Vervier, P. Mahé, M. Tournoud, J.-B. Veyrieras and J.-P. Vert. Large-scale machine learning for metagenomics sequence classification. *Bioinformatics*, 32:1023–1032, 2016. 27

M. Vogler, D. Dinsdale, M. J. Dyer and G. M. Cohen. Bcl-2 inhibitors: small molecules with a big impact on cancer therapy. *Cell Death & Differentiation*, 16(3):360–367, 2009. 96

M. Volkov, J.-A. Turk, N. Drizard, N. Martin, B. Hoffmann, Y. Gaston-Mathé and D. Rognan. On the frustration to predict binding affinities from protein–ligand structures with deep neural networks. *Journal of Medicinal Chemistry*, 2022. 122

T. E. Wagner, J. R. Becraft, K. Bodner, B. Teague, X. Zhang, A. Woo, E. Porter, B. Alburquerque, B. Dobosh et al. Small-molecule-based regulation of RNA-delivered circuits in mammalian cells. *Nature chemical biology*, 14(11):1043, 2018. 43

F. Walter, Q. Vicens and E. Westhof. Aminoglycoside–RNAinteractions. *Current opinion in chemical biology*, 3(6):694–704, 1999. 54

K. Wang, Y. Jian, H. Wang, C. Zeng and Y. Zhao. Rbind: computational network method to predict RNAbinding sites. *Bioinformatics*, 34(18):3131–3136, 2018. 18, 44

L. Wang, Y. Wu, Y. Deng, B. Kim, L. Pierce, G. Krilov, D. Lupyan, S. Robinson, M. K. Dahlgren et al. Accurate and reliable prediction of relative ligand binding potency in prospective drug discovery by way of a modern free-energy calculation protocol and force field. *Journal of the American Chemical Society*, 137(7):2695–2703, 2015. 18

M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019. 45, 65, 81, liii, lxxvi

K. D. Warner, C. E. Hajdin and K. M. Weeks. Principles for targeting RNA with drug-like small molecules. *Nature Reviews Drug Discovery*, 17(8):547, 2018. 14, 43, 52

M. Weiler and G. Cesa. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019. 9, 34

M. Weiler, M. Geiger, M. Welling, W. Boomsma and T. S. Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, pages 10381–10392, 2018a. 9, 27, 98, xix

M. Weiler, F. A. Hamprecht and M. Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018b. 9

D. Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988. 16

C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006. lxvi

M. Winkels and T. S. Cohen. Pulmonary nodule detection in ct scans with equivariant cnns. *Medical image analysis*, 55:15–26, 2019. 9, 27

R. Winter, F. Montanari, F. Noe and D.-A. Clevert. Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chem. Sci.*, 10:1692–1701, 2019a. doi: 10.1039/C8SC04175J. URL http://dx.doi.org/10.1039/C8SC04175J. 21, 109

R. Winter, F. Montanari, A. Steffen, H. Briem, F. Noe and D.-A. Clevert. Efficient Multi-Objective Molecular Optimization in a Continuous Latent Space, April 2019b. 21, 108, 109

D. E. Worrall, S. J. Garbin, D. Turmukhambetov and G. J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017. 9

F. Wu, Q. Zhang, D. Radev, Y. Wang, X. Jin, Y. Jiang, Z. Niu and S. Z. Li. Pre-training of deep protein models with molecular dynamics simulations for drug binding. *arXiv preprint arXiv:2204.08663*, 2022. 17

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 121

Z. Yan and J. Wang. Spa-ln: a scoring function of ligand–nucleic acid interactions via optimizing both specificity and affinity. *Nucleic acids research*, 45(12):e110–e110, 2017. 45

K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley et al. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, 2019. doi: 10.1021/acs.jcim.9b00237. 111

J. You, B. Liu, Z. Ying, V. Pande and J. Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in neural information processing systems*, pages 6410–6421, 2018. XIX, 21, 110, 114, 115, lxxviii, lxxix

A. M. Yu, Y. H. Choi and M. J. Tu. Rna drugs and rna targets for small molecules: Principles, progress, and challenges. *Pharmacological Reviews*, 72:862–898, 10 2020. doi: 10.1124/pr.120.019554. URL https://doi.org/10.1124/pr.120.019554. 79

M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontañón, P. Pham, A. Ravula, Q. Wang et al. Big Bird: Transformers for Longer Sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 28

M. Zahran, C. Sevim Bayrak, S. Elmetwaly and T. Schlick. Rag-3d: a search tool for rna 3d substructures. *Nucleic acids research*, 43(19):9474–9488, 2015. 62

D. V. Zankov, M. Matveieva, A. V. Nikonenko, R. I. Nugmanov, I. I. Baskin, A. Varnek, P. Polishchuk and T. I. Madzhidov. Qsar modeling based on conformation ensembles using a multi-instance learning approach. *Journal of Chemical Information and Modeling*, 61(10):4913–4923, 2021. 17

H. Zeng, M. D. Edwards, G. Liu and D. K. Gifford. Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics*, 32(12):i121–i127, 2016. 28

Z. Zeng, A. K. H. Tung, J. Wang, J. Feng and L. Zhou. Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.*, 2:25–36, 2009. 63

H. Zhang, C.-L. Hung, M. Liu, X. Hu and Y.-Y. Lin. NCNet: Deep learning network models for predicting function of non-coding DNA. *Frontiers in genetics*, 10:432, 2019. 28

S. Zhang, J. Zhou, H. Hu, H. Gong, L. Chen, C. Cheng and J. Zeng. A deep learning framework for modeling structural features of rna-binding protein targets. *Nucleic Acids Research*, 44 (4):e32–e32, 2016. 61

Y. Zhang and J. Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004. 91

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 et al. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. *Nature Biotechnology*, 37(9):1038–1040, Sept. 2019. doi: 10.1038/s41587-019-0224-x. URL https://doi.org/10.1038/s41587-019-0224-x. 17

C. Zhong and S. Zhang. Rnamotifscanx: a graph alignment approach for rna structural motif identification. *RNA*, 21(3):333–346, 2015. 62, 66

C. Zhong, H. Tang and S. Zhang. Rnamotifscan: automatic identification of rna structural motifs using secondary structural alignment. *Nucleic acids research*, 38(18):e176–e176, 2010. 62, 66

H. Zhou, A. Shrikumar and A. Kundaje. Towards a better understanding of reverse-complement equivariance for deep learning models in regulatory genomics. *bioRxiv*, 2020.11.04.368803, 2020. 34, 37, 38

J. Zhou and O. G. Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931–934, 2015. 28

C. L. Zirbel, J. Roll, B. A. Sweeney, A. I. Petrov, M. Pirrung and N. B. Leontis. Identifying novel sequence variants of RNA 3d motifs. *Nucleic acids research*, 43(15):7504–7520, 2015. 56

Appendix A

A gentle introduction to the general concepts of the thesis

A.1 Machine learning

In this part, we will first motivate the use of machine learning as the continuation of a century-old process of explaining reality through models based on empirical observations. We will then explain how machine learning effectively approximates relationships, minimizing the error of a model by tuning its parameters with optimization algorithms. Then we will bring up the concept of generalization of a model derived from a limited dataset on new data, based on the theory of statistics. Finally, we explain that the relationships that we learn can only tie together mathematical objects, calling for the proper representations of real-world concepts into these mathematical objects.

A.1.1 Machine learning stems from Natural Sciences

Natural science aims at understanding the world around us better. To that end, mathematics provides abstract tools to build rational models to support and explain the functioning of reality. However, experiments are the essential foundation of the development of useful models under the principle of realism. Generations of scientists have kept designing new general laws inspired by the limitation of former ones and subsequently validated these laws against experiments, leading to more and more refined models of our reality. This approach has relied on the introduction of natural constants, such as the gravitational constant. Beautifully, very few constants naturally mix to explain most phenomenons arising around us. However, the values of these fundamental numerical constants are not justified by mathematical ground reasons and thus, these values were simply tuned to make our models explain reality in the best possible manner. For instance, Newton's famous law of universal gravitation states that : $\mathcal{F} = \mathcal{G}\ \frac{m_1.m_2}{d^2}$, where F denotes the force of the attraction of two objects of respective masses m_1 and m_2, at a distance d from each other, and $\mathcal{G}$ is a constant named *the gravitational constant*. Its value was measured to be $\mathcal{G} = 6,674 \times 10^{-11} m^3.kg^{-1}.s^{-2}$

This approach of using mathematical tools and deriving models from first-principles has enabled breakthroughs for centuries but failed for intricate problems such as biological ones. For several problems, one needs to use **modeling**. Modeling relies on partial resolution of equations, classical solutions such as exponential decay or even intuition, yielding a parametric mathematical equation. The parameters of these equations are considered as new constants, such as the half-life of a molecule, without a holistic understanding of their nature or their specific values. These values are set according to their ability to **match experimental data**. For instance, first order chemical reactions or radioactive disintegration follow exponential decays. Oxygenated water(H_2O_2) decomposes in water following such a law. Starting from an initial concentration C_0, its concentration through time evolves as $C(t) = C_0 2^{-\frac{t}{\tau}}$, where τ is the half-life of H_2O_2 in water. This value was computed based on experimental data.

Machine learning was introduced as a set of algorithms able to get better performance based on acquiring experience. Empirical measurements can be considered experience and this approach of adjusting a model's parameter based on experimental data consists in statistical machine learning. In modern machine learning and deep learning, the size of the data has grown to extents where the most efficient models now often bypass the manual crafting of parameter-efficient models that encode human intuition. Instead, the parameters of the model lose the status of universal constant and are thought to approximate an underlying but unknown model.

This way of reasoning raises questions about the trust one can put in a model one does not understand, but the results are so good and easy to produce that it has overwhelmed many research fields. Equipped with these models, one could imagine using them as a proxy of previously unreachable truth to build mechanistic and more reliable models. For instance, professional Go or Chess players routinely use the prediction of machine learning models to learn how to play themselves.

A.1.2 Statistical machine learning approximates relationships

A first way to formally introduce machine learning is function approximation. Let $h : [0, 1] \to \mathbb{R}$ be a quadratic function : $h(x) = ax^2 + bx + c, \quad (a, b, c) \in \mathbb{R}^3$ and let us imagine that we have a tabular representation of this function on a regularly spaced grid with n steps of 0.1 (i.e. we know h(0), h(0.1)...). Finally, let us imagine that we want to approximate this unknown function on these grid points with a linear function.

One approach to do so resembles machine learning. Let us define the **parametric function** $f_\theta(x) = \theta_1 x + \theta_0$,with parameters $\theta = (\theta_0, \theta_1) \in \mathbb{R}^2$. This just says that f_θ is a linear function with unknown slope and intercept. Define as the point-wise error the square of the difference between a predicted value and the true value. Define as the **loss** $\mathcal{L}$, the mean of this point-wise error on each grid value : $\mathcal{L}(\theta) = \frac{1}{n} \sum_{x \in \{0, 0.1 ... 1\}} [h(x) - f_\theta(x)]^2$. This loss represents the total error of our parametric function to approximate the function h. To get the best approximation, one want to **minimize** this loss with respect to the parameters of $f_{(\theta_0, \theta_1)}$. In machine learning, this minimization is often done using variations on the gradient descent algorithm, that consists in iteratively adjusting the parameters in the opposite direction of the gradient : $\theta^{(t+1)} = \theta^{(t)} - \frac{\partial \mathcal{L}}{\partial \theta}$. This general procedure of minimizing the error of a parametric model enables us to automatically find the best parameters and is referred to as ***learning***. This is illustrated in Figure A.1.

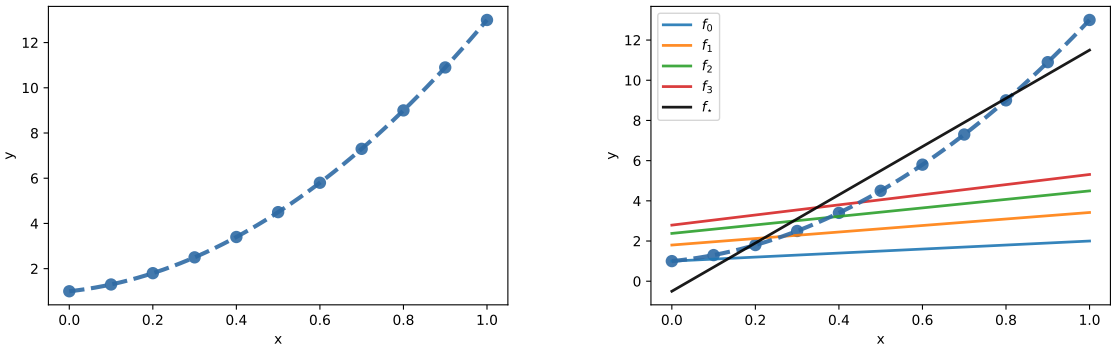


Figure A.1 – The dashed line represents the unknown function and the blue points are our observed data points. *Left :* The function $h(x) = 10 * x^2 + 2 * x + 1$ observed on a regular grid. *Right :* Visualization of the iterations of optimization ; the model is randomly initialized and converges towards the optimal solution.

In current machine learning, the sum of square error and the gradient descent algorithm described above are routinely used (along more complex ones). However, the approximated functions and the models used are much more complex. The other big difference is that this

unknown function is not observed on a grid but through empirical observations described as *statistical samples*.

A.1.3 Real world data as statistical samples

Informally, the theory of statistics assesses the relevance of extrapolating from a finite number of examples to a general knowledge. Intuitively, if we give a treatment to three rats and no treatment to three others, we will be more likely to observe a difference by chance than if we had given it to two populations of thousands of rats. Statistics formalize this intuition and provide tools to compute the probability that the treatment has an effect. A random variable is a quantity that depends on a random event, for instance the value of a dice. The **distribution** of a random variable consists of the frequencies of each observed value if we had infinitely many representative examples. For instance, a normal dice results in a uniform distribution : where all values of a dice appear with a frequency of one sixth. A **statistical sample** denotes realizations of these distributions, for instance rolling a dice once. Properties from the distribution can be estimated from samples, for instance one can compute the frequency of ones or the mean value of our dice rolls. Given a certain number of samples and a statistical model, we can assess the probability that properties derived from the finite sampling reveal something about the underlying distribution.

The statistical learning framework relies on this theory to infer general relationships from finite data. In the example setting above, our data was observed on a fixed, regular grid. However, in the usual setting, the data at hand is considered as independent samples from a hidden, unknown, underlying distribution : $(x, h(x)) \sim \mathcal{D}$. In this formulation, the loss as formulated above (mean error over the whole data) can be seen as an estimate of the loss over the distribution. This lays the ground for stochastic optimization [Bottou, 2010] that uses sub-samples of m data points to compute an easier, unbiased estimate of this expected loss :

$$\begin{aligned}\mathcal{L}(\theta) &= \frac{1}{n} \sum_{x_{1,2,..n} \sim \mathcal{D}} (h(x) - f_\theta(x))^2 \\ &\sim \mathbb{E}_{x \in \mathcal{D}}[h(x) - f_\theta(x))^2] \\ &\sim \frac{1}{m} \sum_{x_{1,2,..m} \in \mathcal{D}} (h(x) - f_\theta(x))^2\end{aligned}$$

This trick of computing the loss only on reduced sub-samples of the data enabled scaling machine learning to huge data sets.

Moreover, this framework enables thinking about noise of the observations. First of all, the samples only give us a partial glimpse of the distributions we observe. Two different data collections will give two different models, as illustrated in Figure A.2. Moreover, the function values are also very often assumed to be noisy, for instance because of labeling errors. The statistical framework can be used to model these sources of uncertainty and to quantify the trust one can put in a model's prediction [Gal et al., 2016].

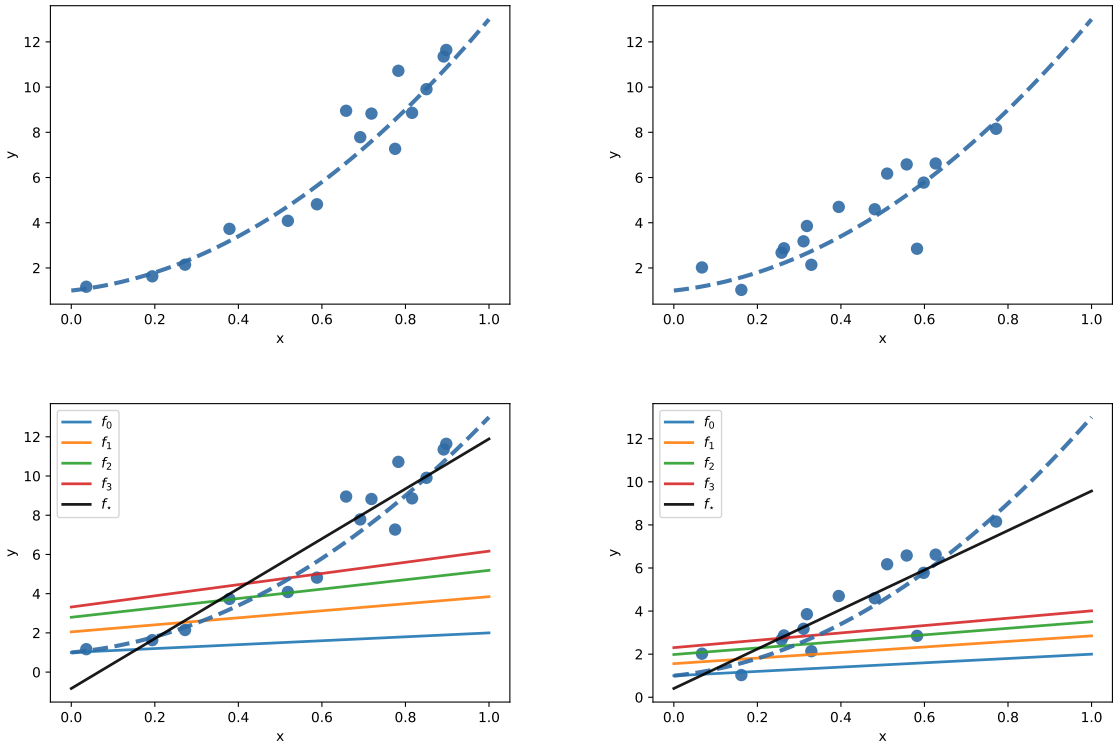


Figure A.2 – Two different samples from the same data and noise distributions and the same underlying relationship give two different observed data sets. This results in different estimated functions through machine learning.

A.1.4 Data needs to be represented in a computer

Now that we have formally described our data set using statistics, we can be stuck with objects that do not really fit in a computer. For instance, we can gather data about the solubility of molecules, but to apply a mathematical function over molecules, we need another step of representing them into our computer. If this representation is too coarse, it will be impossible to learn a meaningful relationship - for instance if we represent the molecules only with their weight. However, to extract relationships from a high-dimensional representation, one has to have enough data.

For a long time, machine learning relied on **feature vectors** : fixed-size, numerical representation that encodes a representation of an object. For instance small molecules can be depicted as "fingerprints" [Durant et al., 2002] : a fixed-size list of numbers that each denote the presence or absence of certain molecular groups. Similarly, the presence of some words, grammar or punctuation could be counted to represent emails or surfaces of rooms to represent apartments. Then, a model can be learnt from this feature vector, for instance to detect spam or predict the price of the apartments.

When trying to model the relationships between two concepts, we might already know some aspects of this relationship, for instance large apartments tend to cost more than small apartments, so we might constrain our parametric function to be increasing. We also know that spam mails will more often ask your bank details so we could include this in our mail representation. Choosing the right representation for our data is often about injecting the right **prior information**.

In recent years, the amount of available data and computing power has skyrocketed, leading to an era of also learning the representation function, coined as **representation learning**. Composing functions together led to a formalism of layers - composing with one more function is seen as adding a layer - in turn leading to the term **deep learning**. Deep learning often operates directly on the raw data, such as the pixel of images.

The more data we have, the more we can expect the model to learn itself about the priors and the less important they become. Using feature vectors based on expert knowledge means injecting a lot of prior information, whereas deep learning methods use little prior domain knowledge and learn from scratch - and from more data. However, without using domain knowledge, some first-principles constraints can be leveraged to avoid nonsensical behavior of the learnt function. For instance, arbitrary data representation decisions should not impact the output of the model. Some models were designed to respect such first principles when the data is endowed with some mathematical properties. These methods will be the focus of Section 1.2.

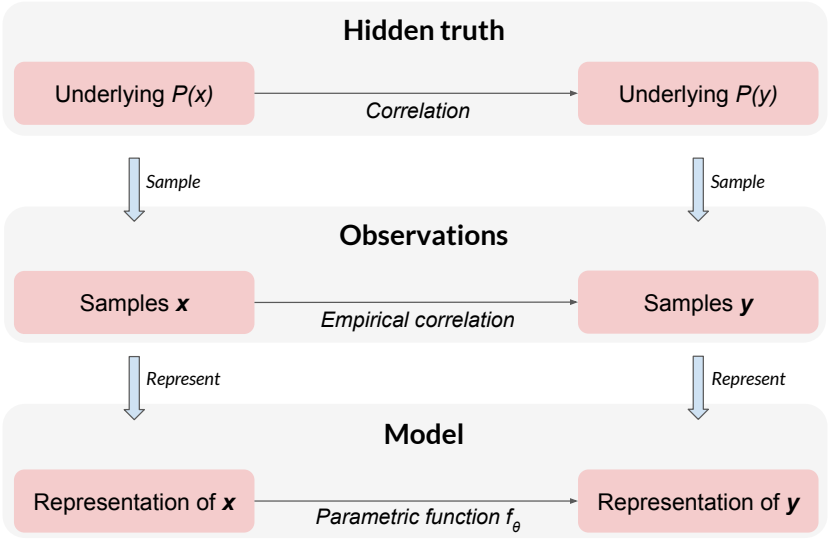


Figure A.3 – The statistical machine learning framework. A hidden relationship ties two random variables with unknown distributions. To model it, we rely on a dataset of examples considered to be samples from their joint distribution. Once we represent these samples in a computer, we can perform machine learning to approximate the relationship between the two variables.

A.2 Structural bioinformatics

Molecular biology is complex, because it describes the underpinning of arguably the most complex systems : living systems. An actual explanation of molecular biology is out of the scope of this introduction. We will restrict ourselves to a short introduction describing what cells are and what are their constituting parts. We will go on by introducing the structural biology approach that uses the 3D structure of molecules to explain the functioning of cells. Finally we show how computational and in-silico approaches help structural biology. For a more detailed explanation and references for a lot of claims in these paragraphs, we recommend the classic book of Alberts et al. [2003].

A.2.1 Living systems have a hierarchical organization with cells as the main unit

Molecular biology starts at the atomic scales based on structured biochemical objects and builds a bottom-up model of how the living systems work. Atoms are organized in molecules that assemble to form organelles that in turn form **cells**. Some organisms are made of a single cell, most famously bacteria. Other living systems are composed of many cells ($\sim 10^{13}$ for a human [Sender et al., 2016]) that are often organized in tissues : a set of cells that have the same function, muscles for instance. These tissues form organs that form organisms organized in populations and ecosystems. This hierarchical organization spans many fields of biology.

Cells are the main functioning unit of living systems and represent a key scale of study. Cells are approximately 10 microns large - 10^5 times the inter-atomic distance - and are delimited by a lipid membrane called the cell membrane. As mentioned above, cells contain subparts deemed

as organelles. Important organelles include their membranes, cytoplasm (the fluid encompassed by the membrane), mitochondria (that produce the energy of the cell). Eukaryotic cells also have a nucleus, while some unicellular organisms do not (prokaryotic cells). Nuclei is a container for the genomic information that enables cells to replicate and function.

A.2.2 Cell functioning relies on a genetic information flow

The genomic information necessary to replicate itself is contained in sequence polymers called DeoxyriboNucleic Acid (DNA). DNA is used to produce another sequential polymer called RiboNucleic Acid (RNA). RNA is in turn read to produce proteins and this last information flow cannot be reversed. This theory is known as the central dogma of molecular biology. DNA, RNA and proteins can be thought as the main building blocks of living systems. They share a polymeric nature, with a conserved *backbone* that polymerizes and some variability in groups attached to it.

In **DNA**, the monomers are called nucleic acids and their variable groups are called bases. There are four possible bases represented by an alphabet of four letters, {`A`, `C`, `T`, `G`}. DNA is organized in only a few polymers, named chromosomes. These are very long in human cells, about 10^7 bases per chromosome. These long molecules assemble pair-wise, forming a double-stranded helix stable structure. This molecule can get copied in a process called DNA **replication**. This enables cells to divide in two, with a copy of DNA in each resulting cell, in turn enabling cell replication.

RNA have a similar backbone with a chemical modification (Deoxyribo- vs Ribo-Nucleic Acids) and similar bases represented as letters from the alphabet {`A`, `C`, `U`, `G`}. RNA is produced from DNA in a process called **transcription**. It corresponds to a one-to-one mapping where DNA nucleotides are turned into RNA nucleotides. However this process produces much shorter polymers ($\sim 10^3$ nucleotides) that represent sub-parts of the DNA polymers. Overall, most of DNA is transformed into RNA ($\sim$90% in humans) [Pertea, 2012].

Proteins are built from other monomers called amino acids, and whose variable groups are called side-chains. There are 20 main amino-acids that are decoded from RNA bases triplets following a deterministic, redundant code known as the genetic code. This process is called **translation** and happens on only a small fraction of RNA in humans ($\sim$2% [Collins et al., 2004]) and a larger one in bacteria. This flow of genetic information is depicted in Figure A.4.

A.2.3 Cell function results from structure

We have explained that cells replicate by copying their DNA and how DNA is turned into RNA and proteins. Let us now explain how these phenomena happen and more generally how cells fulfill their functions.

Most cellular functions rely on molecular processes - chemical reactions and interactions. A single chemical reaction or cellular event can trigger a biochemical cascade of reactions, known as **signaling pathways**, that can alter the function of the whole cell. These signaling pathways involve RNA and protein molecules as well as several bio-molecules bound together, denoted as complexes. For instance, the aforementioned process of turning RNA into proteins involves a complex called the ribosome, which involves almost one hundred protein and RNA molecules. Signaling pathways usually also involve small molecules that are not bio-polymers, as well as changes in metabolite concentration, electric potential or other physical quantities.

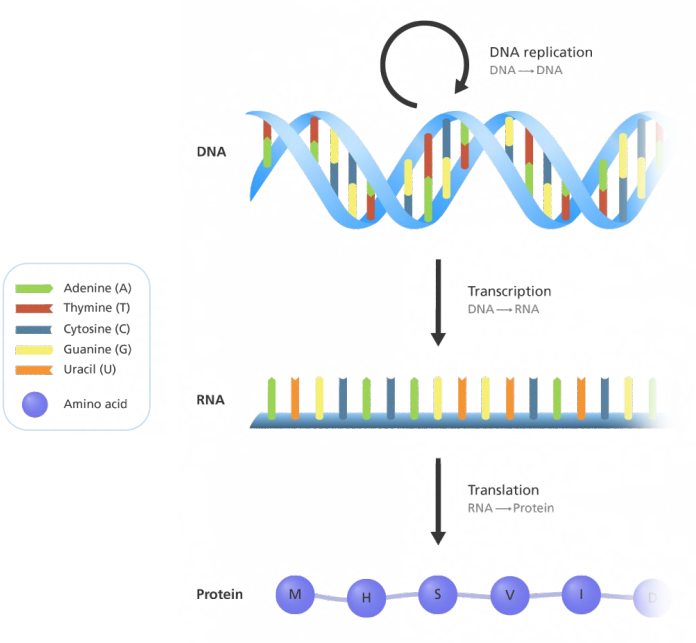


Figure A.4 – The central dogma of biology, DNA is turned into RNA that is turned into proteins. Edited from *Genome Research Limited*.

These intricate interactions depend on their 3D shape, or **molecular structure**, because the equations of physics and chemistry explicitly use the relative positions of atoms. An example protein structure is given in Figure A.5. Moreover, beyond this static view of structure, these shapes change through time, and their successive static states are known as **conformations**. Based on statistical physics reasoning, molecular properties derive from the ensemble of conformations it takes, motivating methods that use the structure of different conformations.

Some fields of biology only implicitly take this dependence in structure into account, by making wet-lab experiments that bypass a molecular understanding of the molecular processes. To reach this understanding, one needs to find - or *resolve* - the structure of the objects and how these structures interact with one another. This approach is denoted as structural biology.

A.2.4 Structural bioinformatics help understanding living systems

Several experimental measurements contribute to structural biology, some of them in an indirect way, such as mass spectrometry that helps determine the mass of a molecule. The most classical experimental measurement of structural biology is **X-ray crystallography**, that requires a crystal of the molecule one wants the structure of, but enables an atomic resolution of this structure. **Nuclear Magnetic Resonance** (NMR) experiments reveal some inter-atomic distances in an indirect way, which can in turn help resolving the structure of bio-molecules such as proteins. The last major experiment is **Cryogenic Electron Microscopy** (cryoEM) that is the closest technique to a traditional picture using electrons instead of light on a frozen sample. Other techniques include small angle X-ray scattering or laser spectroscopy.

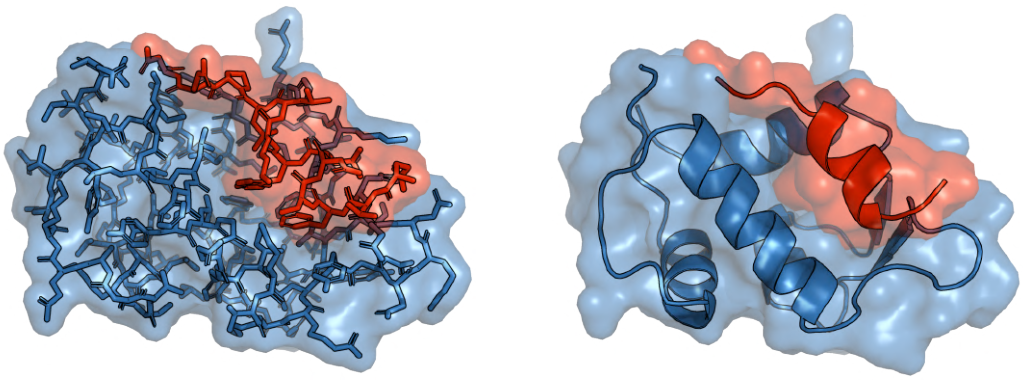


Figure A.5 – Structure of MDM2 (in blue) bound to the activation domain of p53 (in red) - a protein involved in 50% of cancers. On the left, all heavy atoms are represented while a schematic classical view is shown on the right. Upon DNA damage, p53 stops the cell replication and eventually induces cell death. MDM2 is a protein that degrades p53, preventing this interaction was shown to have an anti-cancer effect [Ozaki and Nakagawara, 2011; Vassilev et al., 2004].

In-silico methods are a complementary way to address the structural biology challenges of resolving the structure. These methods have offered an automated way to process the aforementioned experimental data to resolve the structure. The results of this joint effort has led to the creation of the Protein Data Bank (PDB) [Berman et al., 2000] with about 200 000 structures (see Figure A.6).

Some algorithms, deemed as folding algorithms, also try to predict the structure directly from the sequence of a bio-polymer. A notoriety peak for these methods was reached with the release of the successive versions of AlphaFold that reached experimental precision in the folding of a single sequence [Jumper et al., 2021; Senior et al., 2020] or multimeric structures [Evans et al., 2021]. In addition to these static tools, **Molecular Dynamics** (MD) are a class of algorithms that try to sample conformations of a given molecule. All of these experimental and computational tools give us access to the structures of biomolecules.

A plethora of computational tools then try to predict functional aspects of these molecules based on their structure, such as their physico-chemical properties or their interactions with other macro-molecules or with small molecules. However, the physical equation governing the behavior of such big objects is not tractable. Heuristics were developed to approximate these results, intrinsically based on their accordance with experimental data. In the last twenty years, along with the rise of machine learning, more and more data-driven approaches were developed in the field of structural bioinformatics. The recent establishment of geometric deep learning helps respect the mathematical properties that derive from the structured nature of this data. Moreover, the dynamic nature of bio-molecules has been underused because of the extra computational burden and represents a promising research direction. Finding how to best use machine learning methods on structured biological objects is one of the focuses of this dissertation.

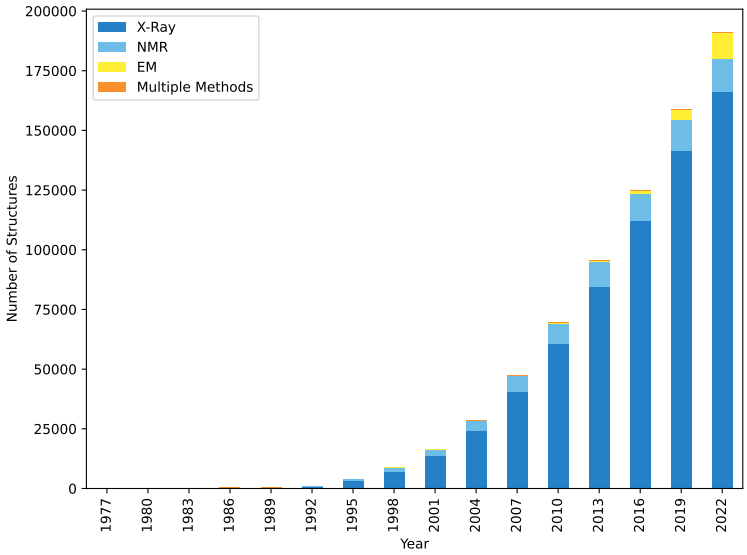


Figure A.6 – Cumulative number of structures deposited in the PDB [Berman et al., 2000], split by method.

A.3 Small Molecules and Drug Discovery

Let us now introduce the field of computational small-molecule drug discovery, in light of previous sections. This field tries to find active molecular compounds that can be administered to humans to cure a disease. This is a challenge because the number of possible compounds makes this problem like finding a needle in a haystack. Traditional pipelines exist, but are slow and costly. An approach to address the drug discovery problem is based on structural biology. The recent developments in machine learning for structural biology could thus help find novel solutions for drug discovery.

A.3.1 The size of the chemical space calls for simulation methods

It is of paramount importance to try and grasp the size of the admissible chemical space in which one should look for therapeutic drugs, as well as the complexity of the task of just correcting the malfunctioning pathway when altering the immensely complex cell functioning. Designing a drug takes approximately 15 years and 2 billions dollars. Based on some basic rules, an estimation of the number of compounds that could represent a potential drug is a whooping 10^{23}~10^{60} [Bohacek et al., 1996; Ertl, 2003; Polishchuk et al., 2013]. The lowest estimates approximately correspond to the number of drops of water in the ocean.

The most stringent limitation to this **huge chemical space** is its synthesizability, because modern chemistry can only produce a fraction of these compounds. The largest databases of commercially available compounds are the ZINC database (~10^8 compounds) [Sterling and Irwin, 2015] or CHEM-BL (~10^6 compounds) [Gaulton et al., 2012]. The larger Enamine database, filled with compounds that we know how to synthesize, contains ~10^{10} compounds. These num-

bers represent only a fraction of the potential drug-like space, which is a comforting reality for bioinformaticians who cannot deal with the whole space. However, there is a need for the exploration of a wider proportion of this space for two main reasons. The first one is that by limiting ourselves to this fraction, we most probably miss out on the best possible drugs with the least side-effects. The second one is more practical : some chemical families are patented and a wider and wider part of this available chemical space is now uninteresting to pharmaceutical companies as they would not be able to produce the retrieved compounds without paying patent rights. In short, the chemical space we need to address contains between 10^{23} and 10^{60} compounds. However, most pharmaceutical assays can now explore 10^6 compounds in assessing the binding to a given target. In some precise cases, new techniques manage to explore 10^9 compounds generated in a combinatorial fashion [Satz et al., 2022] but cannot test a given list of compounds. Then the toxicity assays conducted on mice and humans will investigate the effect of at most tens of candidates. Boiling down this huge chemical space to a **short list of compounds** that can be sent to wet-lab assay is the focus of in silico drug discovery. Continuing on our analogy, there are approximately 10^6 drops of water in a bucket. We are searching for a drop in the oceans but can only test the contents of a bucket. Drug design aims at finding the most promising bucket.

A.3.2 Molecular interactions are the foundations of drug action

Insights from structural biology are key to drug discovery. Indeed, any disease is a symptom to a set of dysfunctions at the molecular level. Drugs are small molecules or compounds that interact with a bio-molecule and affect its functioning. Therapeutic drugs therefore try to fix the malfunctioning aspect at the molecular level, for instance by blocking an interaction between a pathogen and its host. The specific way a given drug alters the cellular process is called the Mechanism Of Action (MOA) and is known for a lot of drugs, such as aspirin [Smith and Willis, 1971].

The bio-molecules a compound is supposed to bind to are called its **targets**. Identifying the malfunctioning pathways, deducing a therapeutic target and then looking for a small molecule that could bind to this target is known as the target-centric or structure-based approach for drug discovery [Chen et al., 2016; Gao and Skolnick, 2013; Hughes et al., 2011]. The other major approach to drug discovery is known as ligand-centric approach, that relies on experimental assays for activity and ligand comparison. It leverages the assumption that similar compounds have similar activities. The search for potential drugs is done by refining iteratively an initial diverse population of compounds. It does not however offer an MOA for its potential success [Lyne, 2002]. Usually both ligand-based and target-based approaches are used concurrently, in an iterative process until a population of drugs that have a good enough binding affinity is reached. Finally, some studies compared these approaches and proposed synergistic ways to combine these approaches [Broccatelli and Brown, 2014; Hawkins et al., 2007], an avenue increasingly explored by more recent papers [Aumentado-Armstrong, 2018].

A.3.3 Machine learning to assist drug discovery

Machine learning has already been very successfully applied to ligand-based drug discovery, but has the potential to also revolutionize target-based approaches. The intricate relationships that govern structural biology and drug discovery are out of the scope of exact methods, but can be

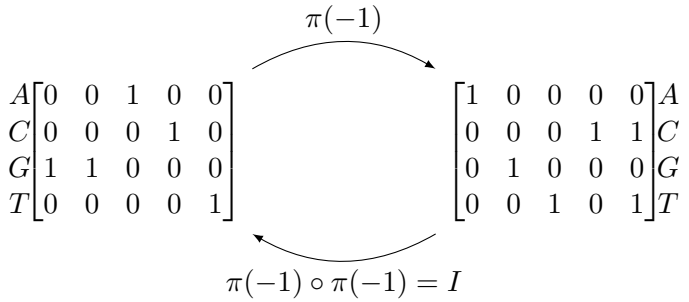
approximated with machine learning models. This requires careful design for several reasons. The data at hand is hard to obtain, gather and understand. Decades of research have tried to understand these relationships one at a time and for now we cannot afford to disregard it and use models obtained de-novo. Moreover, current drug discovery pipelines involve several steps and finding which ones are the most promising to apply and the most urgent to solve needs interdisciplinary expertise. Finally, because of its structural nature, this data has an intrinsic set of regularities that mainstream machine learning models do not respect. Therefore, we believe that the opportunity of helping the drug discovery with machine learning must take into account prior knowledge about the problems and the data at hand with care. We try to explore this research direction in the following dissertation.

Appendix B

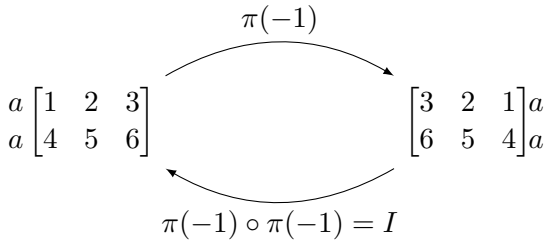
Reverse-Complement Equivariant Networks for DNA Sequences - Appendix

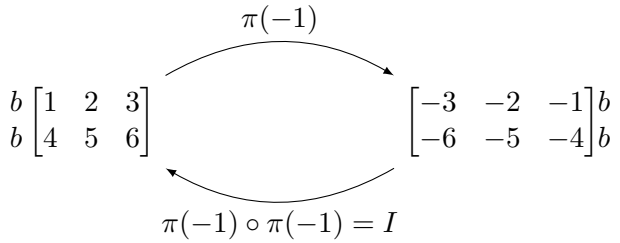
B.1 Illustration of group actions

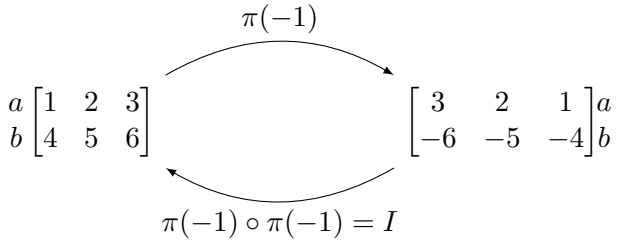
This section is intended to provide a visual, more intuitive understanding of the different group actions on the tensors of our network. We begin with a visualization of the group action for the input space. We exemplify it over the sequence `GGACT`, whose reverse complement is `AGTCC`. The sequence is one hot encoded as explained in the main text and the group action over $\mathbb{Z}_2$ consist in flipping the tensor along the spatial axis and swapping the channels pairwise.



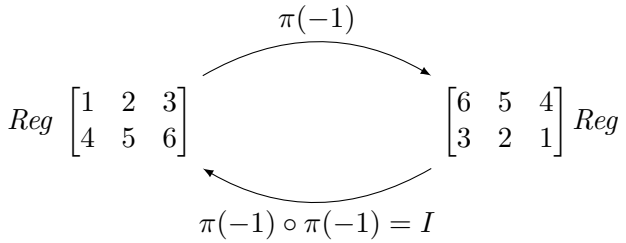
Now we illustrate the actions of other representations, on an example tensor $\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$ with two channels (of type a or b) and three positions; this could typically be the representation of an input sequence of length 3 in an intermediate layer of dimention 2. Choosing the canonical representations of type $(I, 2, 0)$, $(I, 0, 2)$ and $(I, 1, 1)$ respectively, we get the following group actions (for clarity we add the channel type, a or b, near each matrix row):

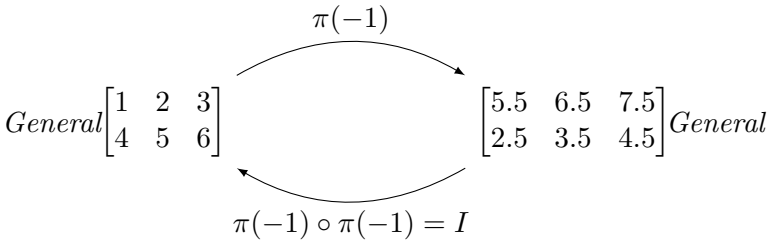






Finally, when using different values for P, we can get other group actions. As mentioned in the main text, by choosing $(P_{reg}, 1, 1)$, where $P_{reg} = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$, we get the regular representation that flips the input channel. We also provide an example of the group action for a general P matrix, by choosing $(P_{general}, 1, 1)$, where $P_{general} = \begin{bmatrix}1 & 3\\ 1 & -1\end{bmatrix}$, we get a representation on the fibers $\rho_{general} = \begin{bmatrix}-0.5 & 1.5\\ 0.5 & 0.5\end{bmatrix}$





Over the course of these examples, we have limited ourselves to the case where the input tensor had only three nucleotides and two channels, but this is coincidental. The representation with arbitrary P can mix an arbitrary number of channels together with the group action.

B.2 Proof of Theorem 1

Proof. The irreducible representations (irreps) of the 2-elements group $\mathbb{Z}_2$ are the 1-dimensional trivial and sign representations, given respectively by $\rho_1(s) = 1$ and $\rho_1(s) = s$. Any representation ρ_n can be decomposed as a direct sum of irreps, and since each irrep is 1-dimensional this means that there exists an invertible matrix P such that $P\rho_n(s)P^{-1}$ is diagonal, with diagonal terms either equal to 1 or equal to s. If we denote by a_n (resp. b_n) the number of diagonal terms equal to 1(resp. s), then Theorem 1 follows. □

B.3 Proof of Theorem 2

Proof. Cohen et al. [2019b, Theorem 3.3] gives a general result about linear equivariant mapping. We first show that this result can be applied here, to show that these linear mappings are exactly the ones written as (2.2) and (2.3). For sake of clarity, we then provide a fully self-contained proof of the same result.

Let us first show that (2.2) and (2.3) correspond to a particular case of Cohen et al. [2019b, Theorem 3.3]. Under the notations of Cohen et al. [2019b], our group is $G = \mathbb{Z} \rtimes \mathbb{Z}_2$, a locally compact, semi-direct product group. We choose $H = H_1 = H_2 = \mathbb{Z}_2$, making the coset space $G/H = \mathbb{Z}$. Since our group is a semi direct product group, we have $h_1(x, s) = s$. The spaces F_n that we have considered are signals in $\mathbb{R}^D$ over the coset space, acted upon by the representation induced by ρ. Equivalently, they are sections of the associated vector bundle for the trivial case of a product group. Therefore, these F_n exactly coincide with the setting of Cohen et al. [2019b, Theorem 3.3] and $\{\phi : F_n \to F_{n+1} | \pi_{n+1}\phi = \phi\pi_n\}$ is exactly $\mathcal{H}$. Then, by Cohen et al. [2019b, Theorem 3.3], $\phi : F_n \to F_{n+1}$ is equivariant if and only if it can be written as a convolution:

$$\forall (f, x) \in F_n \times \mathbb{Z}\,, \quad \phi(f)(x) = \sum_{y \in \mathbb{Z}} \kappa(y - x) f(y)\,, \tag{2}$$

where the kernel $\kappa : \mathbb{Z} \to \mathbb{R}^{D_{n+1} \times D_n}$ satisfies:

$$\forall x \in \mathbb{Z}\, s \in \mathbb{Z}_2,\,, \quad \kappa(sx) = \rho_{n+1}(s)\kappa(x)\rho_n(s^{-1})\,. \tag{B.1}$$

Using that for $s \in \mathbb{Z}_2$, $s^{-1} = s$, and the triviality of this equation for $s = 1$, we get that (B.1) is equivalent to (2.3)

For sake of clarity and completeness, we now provide a more explicit and self-contained proof for (2.2) and (2.3), that follows the one of Weiler et al. [2018a, Theorem 2] in our specific setting. We first notice that any linear mapping $\phi; F_n \to F_{n+1}$ can be written as

$$\forall (f, x) \in F_n \times \mathbb{Z}\,, \quad \phi(f)(x) = \sum_{y \in \mathbb{Z}} k(x, y) f(y)\,,$$

for some function $k : \mathbb{Z}^2 \to \mathbb{R}^{d_{n+1} \times d_n}$. For any $g = ts \in G$, the action of G on F_{n+1} gives:

$$\begin{aligned} \forall (f, x) \in F_n \times \mathbb{Z}\,, \quad \pi_{n+1}(g)\phi(f)(x) &= \rho_{n+1}(s)\phi(f)(s(x-t)) \\ &= \rho_{n+1}(s) \sum_{y \in \mathbb{Z}} k(s(x-t), y) f(y)\,. \end{aligned} \tag{B.2}$$

Similarly, the action of G on F_n followed by ϕ gives:

$$\begin{aligned} \forall (f, x) \in F_n \times \mathbb{Z}\,, \quad \phi(\pi_n(g)f)(x) &= \sum_{y \in \mathbb{Z}} k(x, y)\pi_n(g)f(y) \\ &= \sum_{y \in \mathbb{Z}} k(x, y)\rho_n(s)f(s(y-t)) \\ &= \sum_{y \in \mathbb{Z}} k(x, sy+t)\rho_n(s)f(y) \end{aligned} \tag{B.3}$$

where we made the change of variable $y \mapsto sy + t$ to get the last equality. ϕ is equivariant if and only if, for any $g \in G$, $\phi \circ \pi_n(g) = \pi_{n+1}(g) \circ \phi$, which from (B.2) and (B.3) is equivalent to:

$$\forall (f, x) \in F_n \times \mathbb{Z}\,, \quad \rho_{n+1}(s) \sum_{y \in \mathbb{Z}} k(s(x-t), y) f(y) = \sum_{y \in \mathbb{Z}} k(x, sy+t)\rho_n(s)f(y)\,. \tag{B.4}$$

For any $y_0 \in \mathbb{Z}$ and $v \in \mathbb{R}^{D_n}$, let us apply this equality to the function $f \in F_n$ given by $f(y_0) = v$ and $f(y) = 0$ for $y \neq y_0$:

$$\forall (x, y_0, v) \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{R}^{D_n}\,, \quad \rho_{n+1}(s)k(s(x-t), y_0)v = k(x, sy_0 + t)\rho_n(s)v\,.$$

Since this must hold for any $v \in \mathbb{R}^{D_n}$ this necessarily implies:

$$\forall (x, y_0) \in \mathbb{Z}^2\,, \quad \rho_{n+1}(s)k(s(x-t), y_0) = k(x, sy_0 + t)\rho_n(s)\,.$$

With the change of variable $y = s(y_0 - t)$, this is equivalent to:

$$\forall (x, y) \in \mathbb{Z}^2\,, \quad \rho_{n+1}(s)k(s(x-t), s(y-t)) = k(x, y)\rho_n(s)\,,$$

which itself is equivalent to

$$\forall (x, y) \in \mathbb{Z}^2\,, \quad k(s(x-t), s(y-t)) = \rho_{n+1}(s)k(x, y)\rho_n(s)\,, \tag{B.5}$$

where we used the fact that $\rho_{n+1}(s)^2 = \rho_{n+1}(s^2) = I$ for any $s \in \mathbb{Z}_2$. This must hold in particular for $s = 1$ and $t = x$, which gives:

$$\forall (x, y) \in \mathbb{Z}^2\,, \quad k(0, y-x) = k(x, y)\,,$$

i.e., k is necessarily translation invariant in the sense that there must exist a function $\kappa : \mathbb{Z} \to \mathbb{R}^{D_{n+1} \times D_n}$ such that

$$\forall (x, y) \in \mathbb{Z}^2\,, \quad k(x, y) = \kappa(y - x)\,.$$

From (B.5) we see that κ must satisfy

$$\forall (x, y) \in \mathbb{Z}^2\,, \quad \kappa(s(y-x)) = \rho_{n+1}(s)\kappa(y-x)\rho_n(s)\,,$$

which boils down to the following constraint, after observing that the constraint is always true for $s = 1$ and is therefore only nontrivial for $s = -1$:

$$\forall x \in \mathbb{Z}\,, \quad \kappa(-x) = \rho_{n+1}(-1)\kappa(x)\rho_n(-1)\,. \tag{B.6}$$

At this point, we have therefore shown that an equivariant linear function must have an expansion of the form

$$\forall (f,x) \in F_n \times \mathbb{Z}\,, \quad \phi(f)(x) = \sum_{y\in\mathbb{Z}} \kappa(y-x)f(y)\,,$$

where κ must satisfy (B.6). Conversely, such a linear layer trivially satisfies (B.4), and is therefore equivariant. This proves (2.2) and (2.3).

To prove (2.4), we simply rewrite (2.3) using Theorem 1:

$$\forall x \in \mathbb{Z}\,, \quad \kappa(-x) = P_{n+1}\mathrm{Diag}(I_{a_{n+1}}, -I_{b_{n+1}})P_{n+1}^{-1}\kappa(x)P_n\mathrm{Diag}(I_{a_n}, -I_{b_n})P_n^{-1}\,. \tag{B.7}$$

Thus writing the matrix $K = P_{n+1}^{-1}\kappa(x)P_n$ by blocs of sizes $a_{n+1}\times a_n$, $a_{n+1}\times b_n$, $b_{n+1}\times a_n$ and $b_{n+1}\times b_n$, we have :

$$\begin{aligned}(B.7) &\iff K(-x) = \mathrm{Diag}(I_{a_{n+1}}, -I_{b_{n+1}})K(x)\mathrm{Diag}(I_{a_n}, -I_{b_n}) \\ &\iff \begin{bmatrix}\alpha(-x) & \beta(-x)\\ \gamma(-x) & \delta(-x)\end{bmatrix} = \begin{bmatrix}\alpha(x) & -\beta(x)\\ -\gamma(x) & \delta(x)\end{bmatrix}\end{aligned}$$

This gives us the equivalence (2.3) $\iff$ (B.7) $\iff$ (2.4). □

B.4 Resolution of the constraint for other basis

To go from an arbitrary representation (P, a, b) to another, we can write an odd/even kernel and change of basis. One may also solve the constraints (2.3) for specific representations, and save the need of multiplication by P_{n+1} and P_n^{-1} in (2.4). In this section, we solve the constraint in other basis, to go from one kind of representation (irrep or regular) to another. We just substitute the correct representation and see what constrained kernel it gives. The irrep and regular representations are in a basis such that they write as :

$$\rho_{irrep} = \begin{bmatrix} I_a & 0 \\ 0 & -I_b\end{bmatrix}, \quad \rho_{reg} = \begin{bmatrix} 0 & 0 & \dots & 1 \\ \vdots & & & \vdots \\ 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0\end{bmatrix}.$$

We get the following table of constraints :

F_n \ F_{n+1}	'irrep'	'regular'
'irrep'	$\begin{bmatrix}\alpha(-x) & \beta(-x)\\ \gamma(-x) & \delta(-x)\end{bmatrix} = \begin{bmatrix}\alpha(x) & -\beta(x)\\ -\gamma(x) & \delta(x)\end{bmatrix}$	$\big[\kappa_{j,a}(-x), \kappa_{j,b}(-x)\big] = \big[\kappa_{n-j,a}(x), -\kappa_{n-j,b}(x)\big]$
'regular'	$\begin{bmatrix}\kappa_{a,j}(-x)\\ \kappa_{b,j}(-x)\end{bmatrix} = \begin{bmatrix}\kappa_{a,n-j}(x)\\ -\kappa_{b,n-j}(x)\end{bmatrix}$	$\kappa_{i,j}(-x) = -\kappa_{n-i,n-j}(x)$ [Shrikumar et al., 2017]

B.5 Proof of Theorem 3

With a slight abuse of notations, in this section we denote the matrix $\rho(-1)$ simply by $\rho \in \mathbb{R}^{D\times D}$, and for any $\theta : \mathbb{R} \to \mathbb{R}$ we define $\tilde{\theta}(x) := \theta(x) - \theta(0)$. We start with three technical lemmas, before proving Theorem 3.

Lemma 4. *Let $h : \mathbb{R} \to \mathbb{R}$ be a continuous function with left and right derivatives at 0. If there exists $A \in \mathbb{R}$ with $|A| > 1$ such that*

$$\forall x \in \mathbb{R}\,, \quad h(x) = Ah(A^{-1}x)\,, \tag{B.8}$$

then h is a leaky ReLu function, i.e., there exists $(\alpha_-, \alpha_+) \in \mathbb{R}^2$ such that

$$\forall x \in \mathbb{R}\,, \quad h(x) = \begin{cases} \alpha_- x & \text{if } x \leq 0\,, \\ \alpha_+ x & \text{if } x \geq 0\,. \end{cases}$$

In addition, if $A < -1$, then $\alpha_- = \alpha_+$, i.e., h is linear.

Proof. Equation (B.8) implies $h(0) = 0$ and

$$\forall x \in \mathbb{R}^*\,, \quad \frac{h(x)}{x} = \frac{h(A^{-1}x)}{A^{-1}x}\,,$$

which by simple induction gives more generally:

$$\forall (x, n) \in \mathbb{R}^* \times \mathbb{N}\,, \quad \frac{h(x)}{x} = \frac{h(A^{-n}x)}{A^{-n}x}\,. \tag{B.9}$$

The right-hand side of (B.9) for $n = 2k$ converges to $h'_{sign(x)}(0)$ when $k \to +\infty$, which by unicity of the limit must be equal to the left-hand side. As a result, for any $x \in \mathbb{R}$, $h(x) = h'_{sign(x)}(0)x$, i.e., h is a leaky ReLu function with $\alpha_s = h'_s(0)$ for $s \in \{-, +\}$. If in addition $A < -1$, then (B.9) for $n = 2k + 1$ converges to $h'_{-sign(x)}(0)$ when $k \to +\infty$. By unicity of the limit, this implies $h'_-(0) = h'_+(0)$, i.e., $\alpha_- = \alpha_+$. □

Lemma 5. *Under the assumptions of Theorem 3, if $\bar{\theta}_F$ is equivariant and if there exists $(i, j) \in [1, D]^2$ such that $\rho_{ij} \notin \{-1, 0, 1\}$, then necessarily $\tilde{\theta}$ is a leaky ReLu function.*

Proof. For any (i, j), applying the equivariance constraint $\theta(\rho x)_i = \rho\theta(x)_i$ to the vector $x = ae_j$, for any $a \in \mathbb{R}$, gives the equation:

$$\forall a \in \mathbb{R}\,, \quad \theta(a\rho_{ij}) = \rho_{ij}\theta(a) + (\sum_{k\neq j} \rho_{ik})\theta(0)\,.$$

If $|\rho_{ij}| > 1$, we can rewrite it as

$$\forall a \in \mathbb{R}\,, \quad \theta(a) = \rho_{ij}\theta(a\rho_{ij}^{-1}) + (\sum_{k\neq j} \rho_{ik})\theta(0)\,,$$

and if $0 < |\rho_{ij}| < 1$ we can rewrite it as

$$\forall a \in \mathbb{R}\,, \quad \theta(a) = \rho_{ij}^{-1}\theta(a\rho_{ij}) - \rho_{ij}^{-1}(\sum_{k\neq j} \rho_{ik})\theta(0)\,.$$

In both cases, this is an equation of the form

$$\forall a \in \mathbb{R}\,, \quad \theta(a) = A\theta(A^{-1}a) + B\,,$$

where $|A| > 1$. Subtracting to this equation the same equation written for $a = 0$ gives

$$\forall a \in \mathbb{R}\,, \quad \tilde{\theta}(a) = A\tilde{\theta}(A^{-1}a)\,. \tag{B.10}$$

By Lemma 4, $\tilde{\theta}$ is a leaky ReLu function.

□

Lemma 6. *Under the assumptions of Theorem 3, if $\bar{\theta}_F$ is equivariant and if there exists at least one row in ρ with at least two nonzero entry, then necessarily θ is an affine function.*

Proof. Let us suppose that ρ contains at least a row i with two nonzero entries, say $\rho_{ij} \neq 0$ and $\rho_{ik} \neq 0$. Then taking $x = x_j e_j + x_k e_k$ with $x_j, x_k \in \mathbb{R}$, the equivariance constraint for the i-th dimension gives

$$\forall x_j, x_k \in \mathbb{R}\,, \quad \theta(\rho_{ij}x_j + \rho_{ik}x_k) = \rho_{ij}\theta(x_j) + \rho_{ik}\theta(x_k) + C\theta(0)\,,$$

with $C = \sum_{p \notin \{j,k\}} \rho_{ip}$. Subtracting to this equation the same equation written for $x_j = x_k = 0$ allows us to remove the constant term and get

$$\forall x_j, x_k \in \mathbb{R}\,, \quad \tilde{\theta}(\rho_{ij}x_j + \rho_{ik}x_k) = \rho_{ij}\tilde{\theta}(x_j) + \rho_{ik}\tilde{\theta}(x_k)\,. \tag{B.11}$$

We now prove that $\tilde{\theta}$ is necessarily a leaky ReLu function, i.e., that there exist $(\alpha_+, \alpha_-) \in \mathbb{R}^2$ such that $\tilde{\theta}(x) = \alpha_{sign(x)}x$, with potentially $\alpha_+ \neq \alpha_-$. By Lemma 5 this is true if $|\rho_{ij}| \neq 1$ or $|\rho_{ik}| \neq 1$, so we focus on the case $|\rho_{ij}| = |\rho_{ik}| = 1$, which we decompose in two subcases. First, if $\rho_{ij} = \rho_{ik} = s$ with $s \in \{-1, 1\}$, then taking $x_j = x_k = a$ in (B.11) gives $\tilde{\theta}(2sa) = 2s\tilde{\theta}(a)$, for any $a \in \mathbb{R}$. Second, if $\rho_{ij} = -\rho_{ik} = 1$ (resp. $\rho_{ij} = -\rho_{ik} = 1$), then taking $x_j = 2a$ and $x_k = a$ (resp. $x_j = a$ and $x_k = 2a$) gives $\tilde{\theta}(2a) = 2\tilde{\theta}(a)$. In both subcases, by Lemma 4, $\tilde{\theta}$ must be a leaky ReLu function.

Knowing that $\tilde{\theta}$ is a leaky ReLu function with coefficients α_+ and α_-, in order to prove that θ is necessarily an affine function (i.e., that $\tilde{\theta}$ is linear), we need to show that $\alpha_+ = \alpha_-$. For that purpose, let us first suppose that ρ_{ij} and ρ_{ik} are both positive or both negative. Then there exists a pair $(x_j, x_k) \in \mathbb{R}^2$ such that $x_j > 0$, $x_k < 0$ and $\rho_{ij}x_j + \rho_{ik}x_k < 0$. Similarly, if ρ_{ij} and ρ_{ik} are of different signs, say without loss of generality $\rho_{ij} < 0$ and $\rho_{ik} > 0$, then any pair $(x_j, x_k) \in \mathbb{R}^2$ such that $x_j > 0$, $x_k < 0$ satisfies $\rho_{ij}x_j + \rho_{ik}x_k < 0$. In both cases, using the fact that $\tilde{\theta}$ is linear on $\mathbb{R}_+$ and on $\mathbb{R}_-$, (B.11) gives

$$\begin{aligned} & \alpha_-(\rho_{ij}x_j + \rho_{ik}x_k) = \alpha_+\rho_{ij}x_j + \alpha_-\rho_{ik}x_k\,, \\ \iff & \alpha_-\rho_{ij}x_j = \alpha_+\rho_{ij}x_j \\ \iff & \alpha_- = \alpha_+\,. \end{aligned}$$

□

We are now ready to prove Theorem 3.

Proof of Theorem 3. To characterize the functions θ and representations ρ such that $\bar{\theta}_F$ is equivariant, we proceed by a disjunction of cases on θ, depending on whether it is affine.

If θ is affine, say $\theta(x) = \alpha x + \beta$, then $\bar{\theta}_F$ is equivariant if and only if, for any $x \in \mathbb{R}^D$, $\bar{\theta}_{\mathbb{R}^D}(\rho x) = \rho \bar{\theta}_{\mathbb{R}^D}(x)$. This is equivalent to

$$\begin{aligned}
&\forall (i,x) \in [1,d] \times \mathbb{R}^D\,, \quad \sum_{j=1}^{D} \rho_{i,j}\theta(x_j) = \theta\left(\sum_{j=1}^{D} \rho_{i,j} x_j\right) \\
&\iff \forall (i,x) \in [1,d] \times \mathbb{R}^D\,, \quad \sum_{j=1}^{D} \rho_{i,j}(\alpha x_j + \beta) = \alpha\left(\sum_{j=1}^{D} \rho_{i,j} x_j\right) + \beta \\
&\iff \forall i \in [1,d]\,, \quad \beta\left(\sum_{j=1}^{D} \rho_{i,j} - 1\right) = 0\,.
\end{aligned}$$

This shows that if θ is affine, then $\bar{\theta}_F$ is equivariant if and only $\beta = 0$, i.e., θ is linear (case 1 of Theorem 3), or $\rho\mathbf{1} = \mathbf{1}$ (case 2 of Theorem 3).

If θ is not affine and $\bar{\theta}_F$ is equivariant, then by Lemma 6 we know that ρ can have at most one nonzero entry per row. Since ρ is invertible, it must have at least one nonzero entry per row, so we conclude that it contains exactly one nonzero entry per row, hence a total of D nonzero entries. Being invertible, it must also contain at least one nonzero entry per column, so we conclude that it also contains exactly one nonzero entry per column. Using the fact that $\rho^2 = I$, we can further clarify how nonzero entries must be organized:

- For a nonzero entry $\rho_{ii} \neq 0$ on the diagonal, we must have $\rho_{ii}^2 = 1$, i.e., $\rho_{ii} \in \{-1, +1\}$.
- For an off-diagonal nonzero entry $\rho_{ij} \neq 0$ with $i \neq j$, we must have $\rho_{ij}\rho_{ji} = 1$, i.e., $\rho_{ji} = \rho_{ij}^{-1}$.

Splitting the nonzero entries by sign, this implies that there exists a permutation matrix Π such that

$$\hat{\rho} := \Pi^{-1}\rho(-1)\Pi = \bigoplus_{i=1}^{a} \begin{pmatrix} 0 & \lambda_i \\ \lambda_i^{-1} & 0 \end{pmatrix} \oplus \bigoplus_{i=1}^{b} \begin{pmatrix} 0 & -\mu_j \\ -\mu_j^{-1} & 0 \end{pmatrix} \oplus (1)^{\oplus c} \oplus (-1)^{\oplus d}\,, \tag{B.12}$$

for some $(a,b,c,d) \in \mathbb{N}^4$ such that $a+b+c+d = D$ and $(\lambda, \mu) \in \mathbb{R}_+^a \times \mathbb{R}_+^b$. For any $i \in [1, D]$, let us now denote by $\tau(i)$ the column corresponding to the nonzero entry of the i-th row of $\hat{\rho}$, i.e., the only index such that $\hat{\rho}_{i\tau(i)} \neq 0$. Then the action of $\hat{\rho}$ on a vector $v \in \mathbb{R}^D$ has the simple form $[\hat{\rho}v]_i = \hat{\rho}_{i\tau(i)} v_{\tau(i)}$. By writing the equivariance property $\rho \circ \bar{\theta}_F = \bar{\theta}_F \circ \rho$ coordinate by coordinate, we can therefore say that $\bar{\theta}_F$ is equivariant if and only if:

$$\forall (i,x) \in [1, D] \times \mathbb{R}\,, \quad \theta(\hat{\rho}_{i\tau(i)} x) = \hat{\rho}_{i\tau(i)}\theta(x)\,. \tag{B.13}$$

Let us now consider two possible cases:

- If there exists $i \in [1, D]$ such that $|\hat{\rho}_{i\tau(i)}| \neq 1$, then by Lemma 5 $\tilde{\theta}$ is a leaky ReLu function, i.e., there exist $(\alpha_+, \alpha_-, \beta) \in \mathbb{R}^3$ such that $\forall x \in \mathbb{R}, \theta(x) = \alpha_{sign(x)}x + \beta$. In that case, by (B.13), $\bar{\theta}_F$ is equivariant if and only if:

$$\begin{aligned}
&\forall (i, x) \in [1, D] \times \mathbb{R}\,, \quad \alpha_{sign(\hat{\rho}_{i\tau(i)}x)}\hat{\rho}_{i\tau(i)}x + \beta = \hat{\rho}_{i\tau(i)}\left(\alpha_{sign(x)}x + \beta\right)\,, \\
\iff &\forall i \in [1, D]\,, \quad \begin{cases} \alpha_{sign(\hat{\rho}_{i\tau(i)})} &= \alpha_+\,, \\ \alpha_{sign(-\hat{\rho}_{i\tau(i)})} &= \alpha_-\,, \\ \beta = \hat{\rho}_{i\tau(i)}\beta\,, \end{cases} \\
\iff &\begin{cases} \forall i \in [1, D]\,, & \alpha_{sign(\hat{\rho}_{i\tau(i)})} = \alpha_+\,, \\ \beta = 0\,, \end{cases}
\end{aligned} \tag{B.14}$$

  where the first equivalence comes from identifying the coefficients of the linear equation in x on $\mathbb{R}_-$ and $\mathbb{R}_+$, and the second equivalence comes from the observation that the two conditions in α in the first equivalence are themselves equivalent to each other, so we can keep only one of them, and that the condition on β is equivalent to $\beta = 0$ since we assume the existence of an $i \in [1, D]$ such that $\hat{\rho}_{i\tau(i)} \neq 1$. Since we assume that θ is not affine, we can not have $\alpha_- = \alpha_+$, which by (B.14) rules out the possibility of having negative entries in $\hat{\rho}$, i.e., necessarily $b = d = 0$ in (B.12). If that is not the case, then the condition on α in (B.14) is automatically met for all $i \in [1, D]$, so we have that $\bar{\theta}_F$ is equivariant if and only if $\beta = 0$, i.e., if and only if θ is a leaky ReLu function. This is the second statement in Case 3 of Theorem 3, when we further notice that when $b = 0$ the only entry in $\hat{\rho}$ that can have been different from -1 and 1 is a λ_i in (B.12).

- If for all $i \in [1, D]$, $|\hat{\rho}_{i\tau(i)}| = 1$, then (B.12) simplifies as

$$\hat{\rho} = \bigoplus_{i=1}^{a} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \oplus \bigoplus_{i=1}^{b} \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \oplus (1)^{\oplus c} \oplus (-1)^{\oplus d}\,.$$

  In that case, the equivariance condition (B.13) is particularly simple, and true for any θ for positive values. For each i such that $\hat{\rho}_{i\tau(i)} = -1$ it reads $\forall x \in \mathbb{R}, -\theta(x) = \theta(-x)$, and is therefore true if and only if θ is odd. Noticing that the latter constraint occurs if and only if $b + d > 0$ finally leads to the first and third statements in Case 3 of Theorem 3.

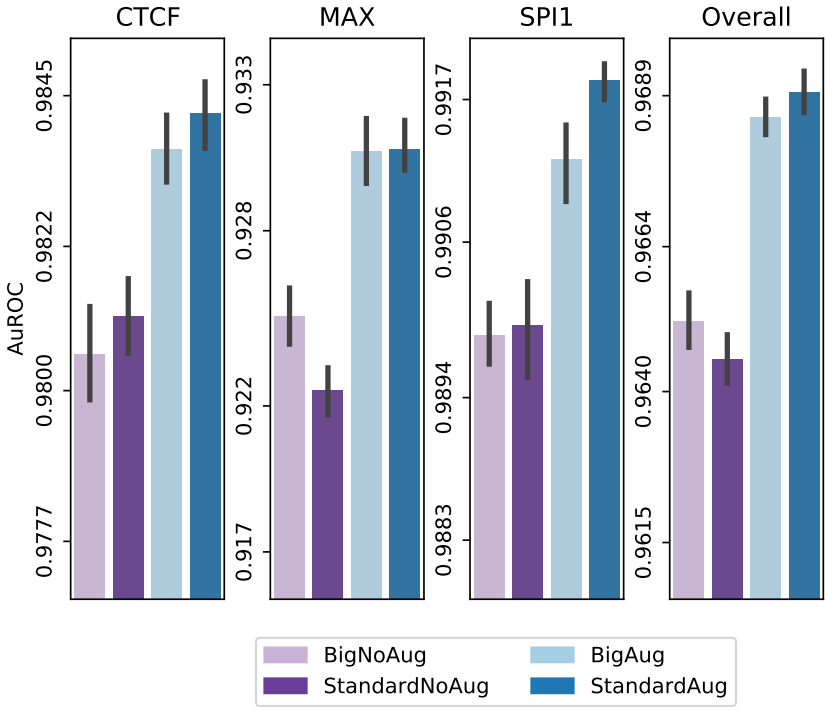
□

B.6 Additional result

B.7 Effect of data augmentation and size for non-equivariant models

Given a non-equivariant model, a simple way to let it "learn" to be equivariant is to train it with data augmentation, where for each sequence in the training set we add its reverse complement to the training set. This doubles the size of the training set, which increases the training time. If we compare such a non-equivariant model with an equivariant model with the same number of channels in each layers, then it has about twice the same number of free parameters to train,

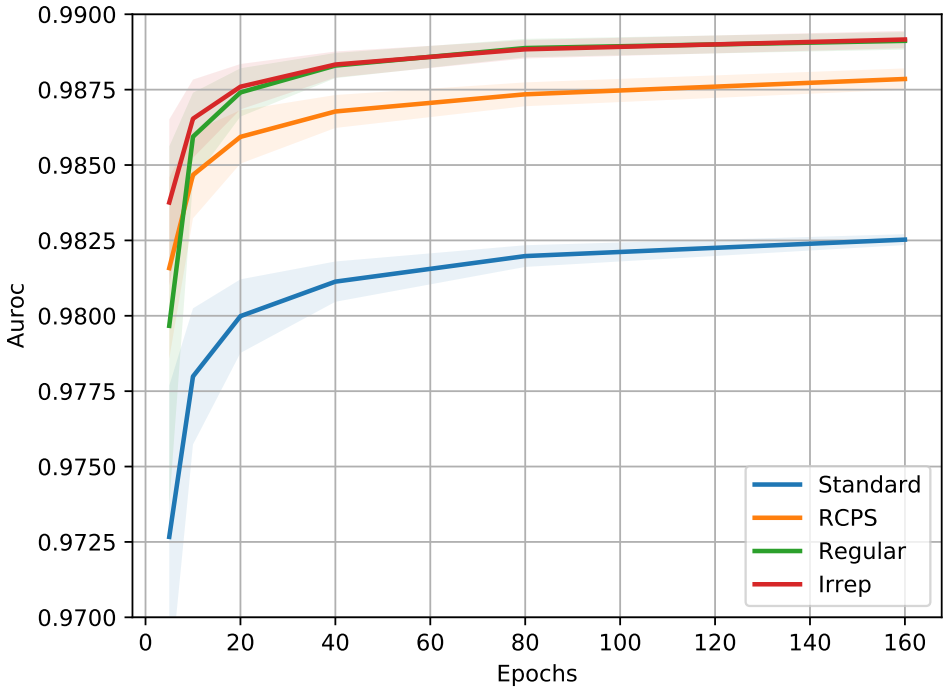
and we therefore call it "big"; as an alternative, one may want to restrict the number of channels in each layer to enforce the same number of parameters as the equivariant model. To assess the benefits of data augmentation and number of channels, we plot in Figure B.1 the performance of a standard, non-equivariant model with or without data augmentation, and with the same number of channels or half of it, on the binary classification tasks. We see that the number of channels has no significant impact on the performance, but that data augmentation has a significant positive impact. In the main text, we therefore restrict ourselves to the standard model with data augmentation as a non-equivariant baseline model.



Supplementary Figure B.1 – Binary task performance of a standard, non-equivariant model trained with ("Aug") or without ("NoAug") data augmentation, and with more ("Big") or less ("Standard") channels.

B.8 Comparison of learning curves

Because equivariant models are supposed to converge faster, we looked into the learning curves of our models, i.e., how the test performance increases as a function of the number of epochs during training. However, we do not see a major difference in the learning dynamics between the equivariant and non equivariant models.



Supplementary Figure B.2 – AuROC performance of the four different models on the three binary classification problems CTCF, MAX and SPI1, as well as their average over the course of learning.

Appendix C

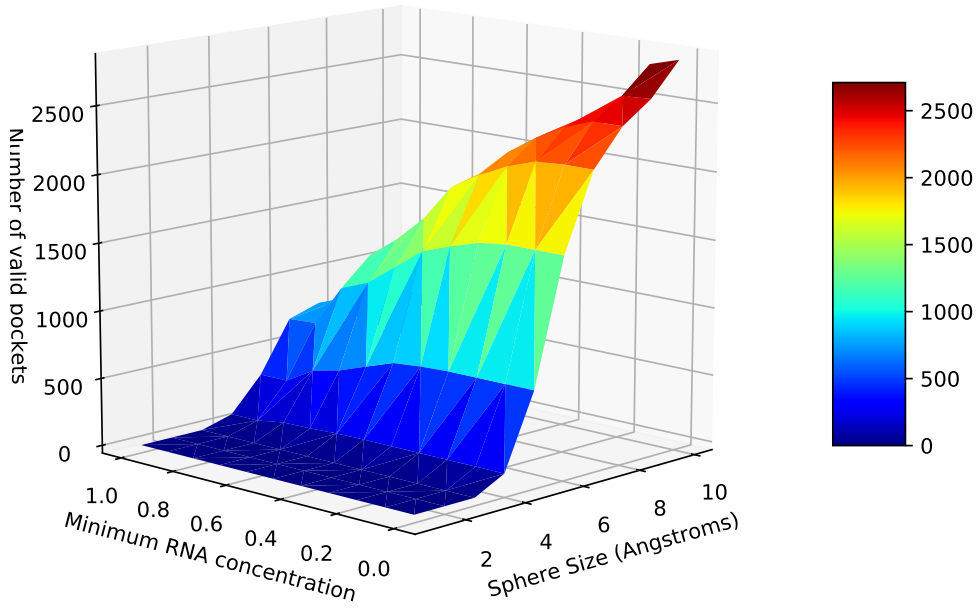
Augmented base pairing networks encode RNA-small molecule binding preferences - Appendix

Contents

C.1 Data Preparation

C.1.1 Binding Site extraction

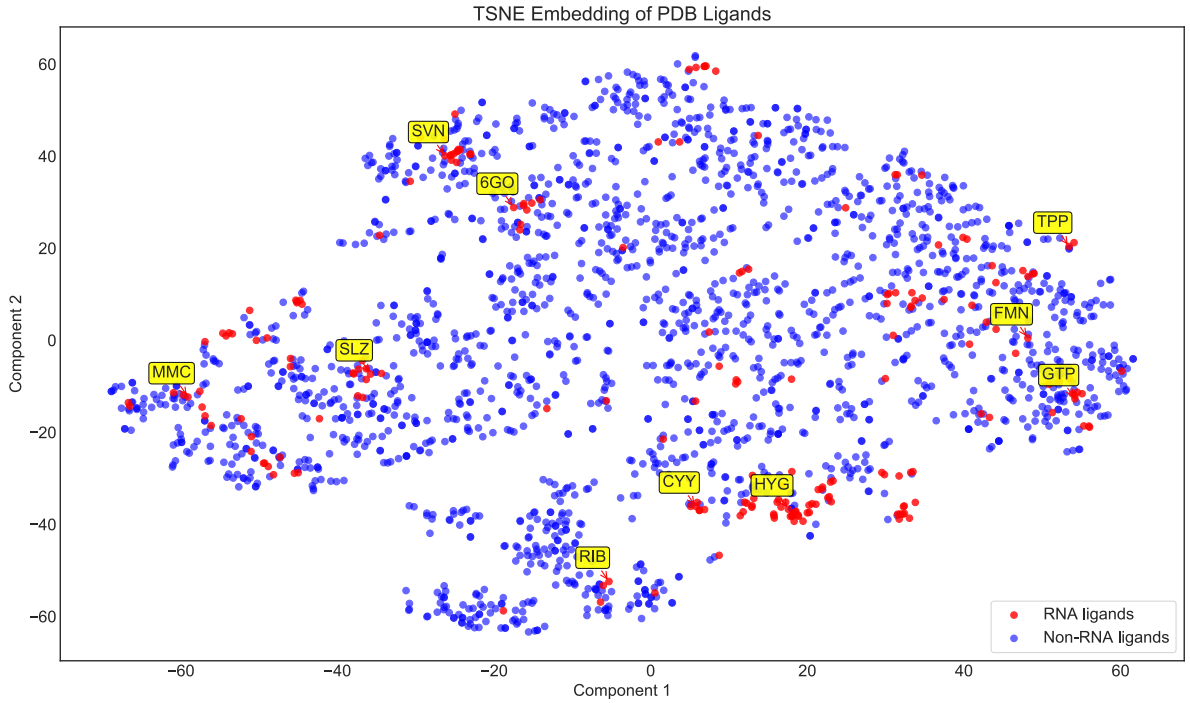
A crucial step in our learning pipeline is the construction of ABPNs from crystal structure data. Once all crystal structures are acquired, we consider spheres of varying radii around the ligand to define a binding site. We studied two parameter choices which affect the number and quality of the extracted binding sites: radius and protein vs. RNA content. As the radius increases, we obtain a larger number of binding sites. However, since the crystal structures often contain proteins, we increase the probability that the binding site will be dominated by protein residues. We therefore compute the ratio of RNA to Protein residues in the binding site. The resulting counts for binding sites are shown in **Fig. C.1**. From this data we choose a minimum RNA concentration of 0.6 for our training model. If a PDB contains multiple binding events of the same ligand, we keep only one at random to reduce redundancies. At this stage, we have identified a set of atomic coordinates which correspond to binding sites. Since we apply a hard distance cutoff in the crystal structure, the resulting graphs often have chain discontinuities. To address this issue, we add all 1-neighbour breadth first nodes to the original graph, as well as remove any disconnected components with fewer than 4 nodes.



Supplementary Figure C.1 – Number of binding sites retrieved versus distance threshold and RNA concentration threshold

C.1.2 Fingerprint representation

We use the MACCS fingerprints to represent the chemical space of small molecules. A projection of this representation as well as the space occupied by RNA ligands is depicted in **Fig. C.2**



Supplementary Figure C.2 – Two dimensional TSNE [Maaten and Hinton, 2008] embeddings of chemical fingerprints sampled from the PDB database (RNA and protein binding). RNA ligands are highlighted in red and protein ligands in blue. We label a few interesting ligands such as 'KAN' and 'FMN' which correspond to well-known RNA binding classes known as aminoglycosides and riboswitch-binding amines respectively.

C.2 RGCN

We use a Relational Graph Convolutional Network (RGCN) [Schlichtkrull et al., 2018] as the core of the fingerprint prediction model. An RGCN is a function that associates real vectors of size d for each node of a graph, known as node embeddings. Given initial node embeddings h_u^0 for each node u, an activation function σ and a graph structure that induces $\mathcal{R}$ edge-types and neighboring structure for each node $\mathcal{N}_u^r$, we can then use learnable matrices W that yield other node embeddings, according to the formula :

$$h_u^{l+1} = \sigma\left(W_{r_0}^l h_u^l + \sum_{r\in\mathcal{R}}\sum_{v\in\mathcal{N}_u^r}\frac{1}{c_{u,r}}W_r^l h_v^l\right)$$

Successive embeddings for each node are obtained by repeatedly using this process, until each node is attributed a final embedding $h_L(u)$. For this work, we consider the base pairing types to be distinct edge-types. We believe this is a fair approximation given the results of isostericity comparisons showing that computing the geometric discrepancy between all pairs of edge types yields close to a diagonal matrix [Stombaugh et al., 2009].

Once node embeddings are computed, we concatenate the resulting embedding matrix with a one-hot encoding of the input graph's nucleotides (`A, U, C, G`). Next, graph-level representation is obtained by applying a widely used trainable Graph Attention Pooling layer [Veličković et al., 2017], to map the node embeddings to a single vector $e_f \in \mathbb{R}^d$. Finally, we feed e_f through a Multi Layer Perceptron which yields probabilities for each index of the fingerprint $\hat{y}$.

$$\begin{aligned} e_f &= \text{GAT}(\text{Graph, final node embeddings}) \\ \hat{y} &= \text{MLP}(e_f) \end{aligned}$$

We supervise this process using the binary cross entropy $\mathcal{L}_{fp}$ between the predicted fingerprint and the observed one y over all dimensions i, and train the model by minimizing this loss over the training data.

$$\mathcal{L}_{fp} = \sum_{i=0}^{k}[y_i \cdot \log \hat{y}_i + (1-y_i)\cdot\log(1-\hat{y}_i)]$$

C.3 Unsupervised Pre-Training

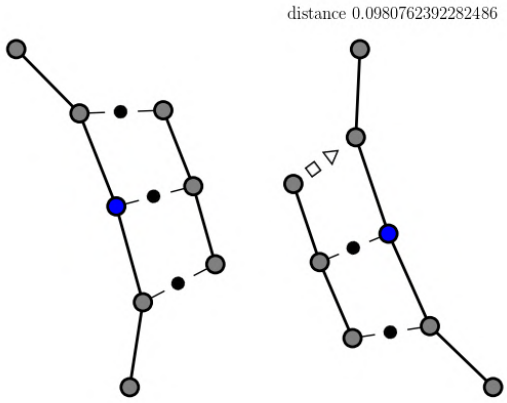
As described in the main text, we wish to define similarity functions K that take two nodes u and v and return a number close to one if they have similar neighborhoods and close to zero for dissimilar ones. In this paper we construct such a measure K from another measure d, that is a function which compares the sets of edges at a distance l, from u and v, denoted R_u^l, R_v^l. We then aggregate the results of the comparison of these sets for an increasing distance, according to the formula :

$$k_L(u,v) := N^{-1}\sum_{l=0}^{L-1}\lambda^l d(R_u^l, R_v^l)$$

The λ^l is a decay term which allows us to attend more to structural information close to the root nodes and we set $\lambda = 0.5$. We use N as a normalization constant ($\mathrm{N} = \frac{1-\lambda}{1-\lambda^L}$) to ensure the sum saturates at 1. d is defined to be a simple overlap measure on the histograms of base-pairing edge types f_R, and f'_R (i.e $f_R(i)$ stores the number of times edge type i is observed).

$$d(R, R') := \frac{|f_R \cap f'_R|}{|f_R \cup f'_R|}$$

To compensate for the over-representation of a few edge types such as backbones and Watson-Crick edges, we scale the d value with the commonly-used Inverse-Document Frequency (IDF) factor [Ramos et al., 2003]. We show in **Fig. C.3** an example of a pair of nodes that obtained a high similarity score K after embeddings were computed.



Supplementary Figure C.3 – Example of pair of nodes given similar embeddings $\phi(u), \phi(v)$. The central pair of nodes which were used to make the comparison are colored in blue.

We then use the same RGCN function on the graph as before to annotate the nodes and get for each node, its vector embedding $h_L(u)$ that depends on the parameters W. Finally, as in the supervised setting, we define a loss function $\mathcal{L}_{rep}$ to minimize, that makes our network learn to approximate the dissimilarity function :

$$\mathcal{L}_{rep} = \|K(u, v) - cosine(h_L(u), h_L(v))\|_2^2$$

$$\mathrm{rank}_{\mathcal{C}}(y, \hat{y}) = 1 - \frac{\rho_{y,\hat{y},\mathcal{C}}}{|\mathcal{C}|} \tag{C.1}$$

C.4 Model Architecture and Hyperparameters

Hyperparameter	**Value**
RGCN Layers Dimensions	16, 16, 16
RGCN Number of Relations	13
RGCN Basis Sharing	None
RGCN Activation	ReLu
RGCN Dropout Probability	0.5
GAT Layer	Default
Fully Connected Dimensions	16 166

Table C.1 – Hyperparameter choices for learning pipeline. RGCN parameters are identical for the unsupervised pre-training and the fingerprint prediction networks.

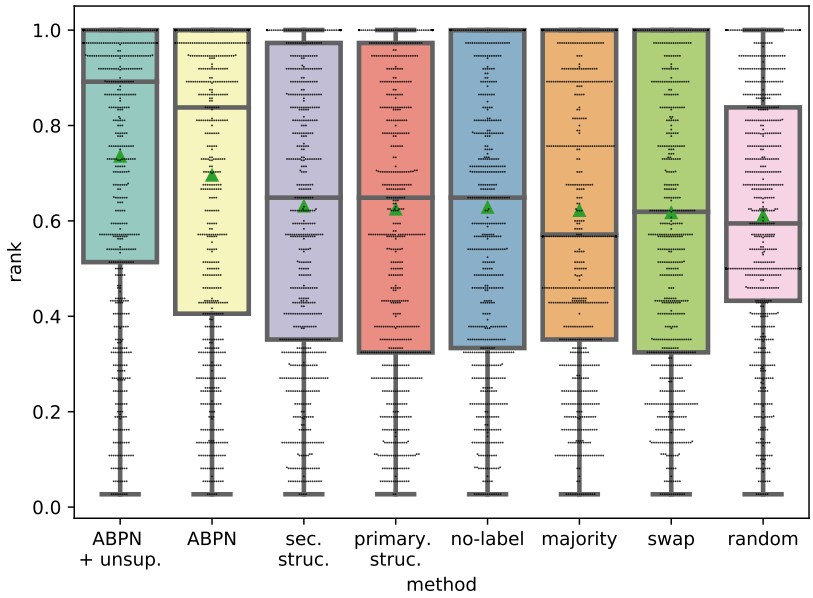
C.5 Results

Experiment	**Ranks**		**L2**	
	DecoyFinder	RNA	*DecoyFinder*	RNA
random	0.265880	0.276721	0.0384392	0.038299
majority	0.320012	0.269375	0.073969	0.074892
swap	0.319836	0.269233	0.071212	0.071308
no-label	0.317259	0.272816	0.072830	0.073768
primary	0.323843	0.064917	0.181	0.066853
secondary	0.318527	0.299428	0.074738	0.076667
ABPN	0.322124	0.301635	0.091479	0.092328
ABPN + unsup.	0.303712	0.294309	0.093006	0.095090

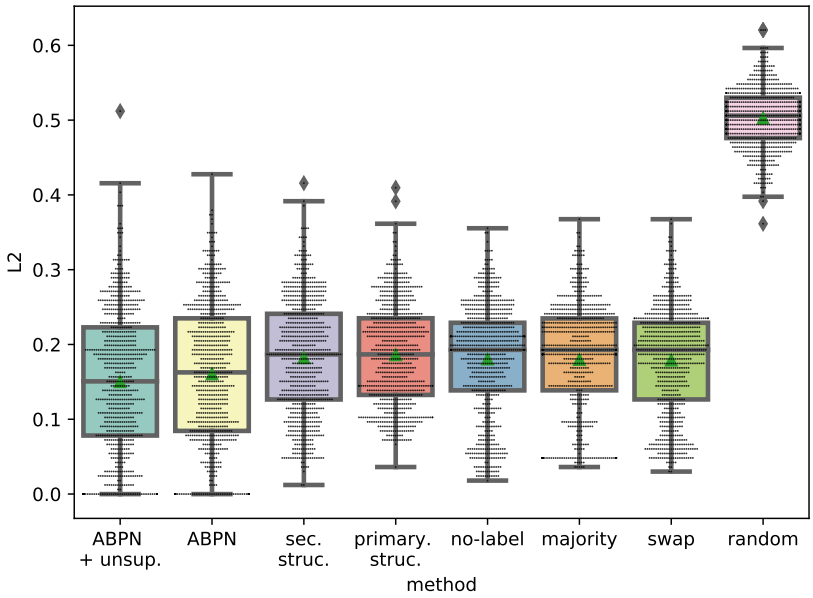
Table C.2 – Standard deviation on ligand screen ranks and L2 distance achieved on held-out binding sites for each condition on both decoy sets.

method_2 method_1	ABPN	secondary	primary	no-label	majority	swap	random
ABPN+unsup -	2.9-06	5.0e-26	1.4-22	2.0e-21	9.3e-25	7.1-26	2.3e-18
ABPN	-	1.6e-11	5.6e-11	1.4e-08	4.2e-10	6.3e-12	2.0e-08
secondary		-	3.2e-01	7.6e-01	1.2e-01	2.8e-02	1.7e-01
primary			-	4.2e-01	2.7e-01	2.3e-02	3.1e-01
no-label				-	5.5e-01	1.5e-02	1.7e-01
majority					-	3.6e-01	3.3e-01
swap						-	5.4e-01

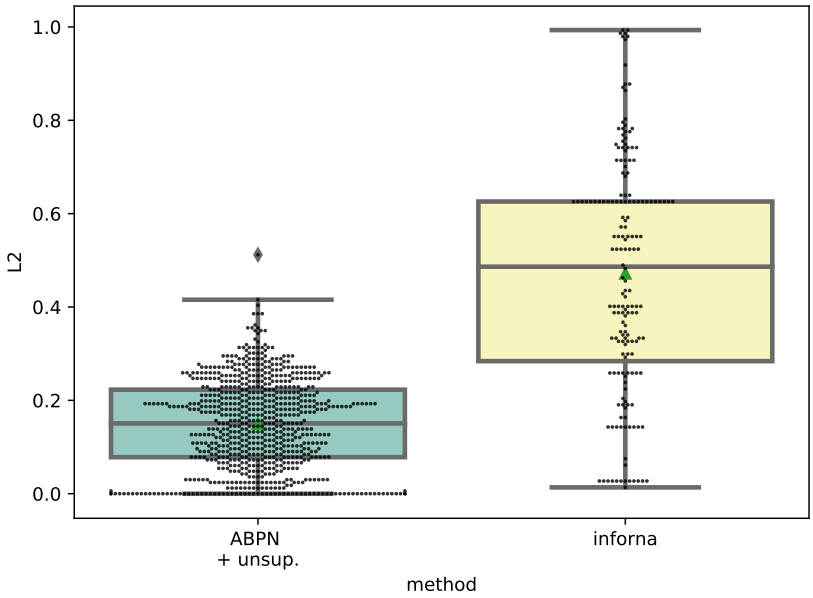
Table C.3 – Pairwise Wilcoxon test for the DecoyFinder decoy set over the ligand ranks.



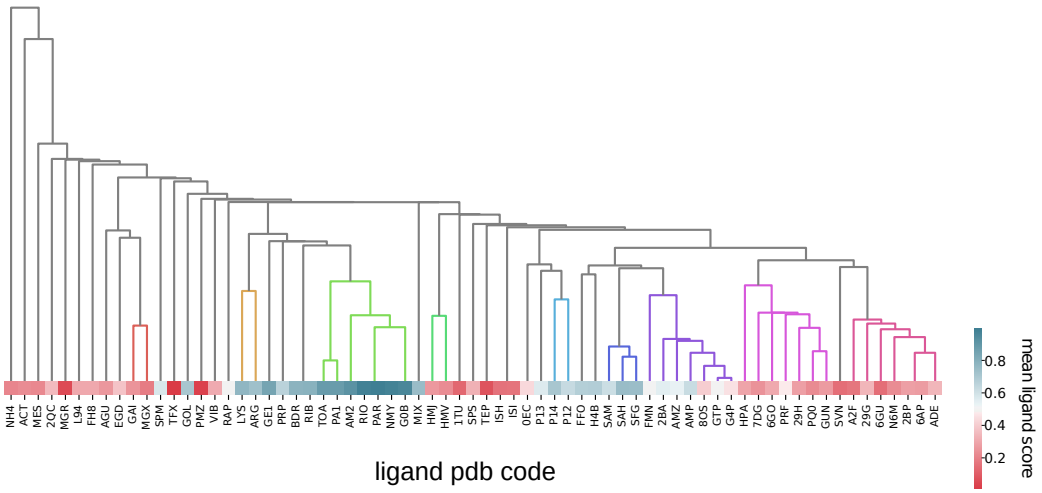
Supplementary Figure C.4 – Ranks achieved by RNAmigos against the DecoyFinder screen.



Supplementary Figure C.5 – L2 distance from the native ligand achieved with RNAmigos.



Supplementary Figure C.6 – L2 distance from the native ligand achieved with Informa



Supplementary Figure C.7 – Performance per ligand type with Informa software. A dendrogram is drawn to illustrate families of similar ligands.

Table C.4 – Details for each ligand in the dataset. PDB codes and full names are in the first two columns, followed by the number of occurrences. The `inforna` and `RNAmigos` columns contain the score achieved on average by each tool on the given ligand.

Ligand	Name	Count	Inforna	RNAmigos
SPD	spermidine	11		0.74
NMY	neomycin	11	0.95	0.94
FME	n-formylmethionine	11		0.95
SPM	spermine	11	0.57	**0.82**
SAM	s-adenosylmethionine	11	0.60	**0.82**
PAR	paromomycin	11	0.97	**0.99**
LYS	lysine	10	0.75	**0.91**
GAI	guanidine	10	0.24	**0.65**
FMN	flavin	9	**0.48**	0.38
VAL	valine	9		0.81
PRF	7-deaza-7-aminomethyl-guanine	8	0.47	**0.78**
HPA	hypoxanthine	8	0.28	**0.94**
2BA	(2r,3r,3as,5r,7ar,9r,10r,10as,12r,14ar)-2,9-bi...	8	0.54	0.54
ADE	adenine	8	0.33	**0.89**
ACY	acetic	8		0.88
EDO	1,2-ethanediol	8		0.40
GLY	glycine	8		0.76
ARG	arginine	8	**0.72**	0.69
EOH	ethanol	7		0.60
PGE	triethylene	7		0.67
PPU	puromycin-5'-monophosphate	7		0.51
GOL	glycerol	7	0.69	**0.88**
PUT	1,4-diaminobutane	7		0.94
C2E	9,9'-[(2r,3r,3as,5s,7ar,9r,10r,10as,12s,14ar)-...	7		0.79
GUN	guanine	7	0.26	**0.97**
PEG	di(hydroxyethyl)ether	7		0.80
GET	geneticin	6		0.83
SPS	sparsomycin	6	0.32	0.34
SRY	streptomycin	6		0.58
TRP	tryptophan	6		0.88
LLL	(2r,3r,4r,5r)-2-((1s,2s,3r,4s,6r)-4,6-diamino-...	6		0.70
GNG	2'-deoxy-guanosine	5		0.98
TPP	thiamine	5		0.16
GLP	glucosamine	5		0.70
SAH	s-adenosyl-l-homocysteine	5	0.73	**0.78**
5GP	guanosine-5'-monophosphate	5		0.23
CLM	chloramphenicol	5		0.45
NH4	ammonium	5	0.21	0.53

Continued on next page

Table C.4 – Details for each ligand in the dataset. PDB codes and full names are in the first two columns, followed by the number of occurrences. The `inforna` and `RNAmigos` columns contain the score achieved on average by each tool on the given ligand.

Ligand	Name	Count	Inforna	RNAmigos
MES	2-(n-morpholino)-ethanesulfonic	5	0.21	0.21
ACT	acetate	5	0.23	0.42
ERY	erythromycin	5		0.42
HYG	hygromycin	5		0.81
PRP	alpha-phosphoribosylpyrophosphoric	5	**0.64**	0.35
S9L	2-[2-(2-hydroxyethoxy)ethoxy]ethyl	4		0.79
T1C	tigecycline	4		0.37
BLS	blasticidin	4		0.31
PRO	proline	4		0.97
G4P	guanosine-5',3'-tetraphosphate	4	0.45	**0.76**
AMZ	aminoimidazole	4	**0.53**	0.49
SIS	(1s,2s,3r,4s,6r)-4,6-diamino-3-{[(2s,3r)-3-ami...	4		0.89
3HE	4-{(2r)-2-[(1s,3s,5s)-3,5-dimethyl-2-oxocycloh...	4		0.81
AM2	apramycin	3	0.89	**0.95**
PO4	phosphate	3		0.10
TAC	tetracycline	3		0.35
VIR	virginiamycin	3		0.48
1PE	pentaethylene	3		0.92
EKJ	4-[(3-{2-[(2-methoxyethyl)amino]-2-oxoethyl}-1...	3		0.91
HGR	hygromycin	3		0.68
ANM	anisomycin	3		0.76
SCM	spectinomycin	3		0.76
8UZ	tc007	3		0.97
6HS	(1s,2s,3r,4s,6r)-4,6-diamino-3-{[(2s,3r)-3-ami...	3		0.88
XXX	(2r,3s,4r,5r,6r)-6-((1r,2r,3s,4r,6s)-4,6-diami...	3		0.98
CLY	clindamycin	3		0.54
NEG	negamycin	3		0.70
DOL	5-(2-diethylamino-ethanesulfonyl)-21-hydroxy-1...	3		0.04
B6M	(1r,2s,3s,4r,6r)-4,6-diamino-2-{[3-o-(2,6-diam...	3		1.00
ZLD	n-{[(5s)-3-(3-fluoro-4-morpholin-4-ylphenyl)-2...	3		0.52
IDG	o-2,6-diamino-2,6-dideoxy-beta-l-idopyranose	3		0.93
GLN	glutamine	3		0.99
AG2	agmatine	2		0.70
3CO	cobalt	2		0.25
6GU	6-chloroguanine	2	0.15	**0.97**
8OS	5'-o-[(s)-hydroxy(4-methyl-1h-imidazol-5-yl)ph...	2	**0.40**	0.06
EGD	n-ethylguanidine	2	0.37	**0.64**
UAM	amicoumacin	2		0.79

Continued on next page

Table C.4 – Details for each ligand in the dataset. PDB codes and full names are in the first two columns, followed by the number of occurrences. The `inforna` and `RNAmigos` columns contain the score achieved on average by each tool on the given ligand.

Ligand	Name	Count	Inforna	RNAmigos
RIO	ribostamycin	2	**0.96**	0.99
4BW	2-amino-9-[(2r,3r,3as,5r,7ar,9r,10r,10as,12r,1...	2		0.61
KSG	(1s,2r,3s,4r,5s,6s)-2,3,4,5,6-pentahydroxycycl...	2		0.82
SPK	spermine	2		0.50
6AP	9h-purine-2,6-diamine	2	0.27	**0.82**
SE4	selenate	2		0.06
GTP	guanosine-5'-triphosphate	2	**0.51**	0.34
ACA	6-aminohexanoic	2		0.98
747	(5z)-5-[(3,5-difluoro-4-hydroxyphenyl)methylid...	2		0.59
6GO	6-o-methylguanine	2	0.30	**0.94**
HMT	(3beta)-o~3~-[(2r)-2,6-dihydroxy-2-(2-methoxy-...	2		0.19
MLI	malonate	2		0.47
BDG	o-2,6-diamino-2,6-dideoxy-alpha-d-glucopyranose	2		0.94
CYY	2-deoxystreptamine	2		0.92
G6P	alpha-d-glucose-6-phosphate	2		0.84
PHA	phenylalaninal	2		0.87
TOC	2,3,6-trideoxy-2,6-diamino	2		0.91
PCY	pactamycin	2		0.68
SIN	succinic	2		0.89
PHE	phenylalanine	2		0.95
VIF	flopristin	2		0.13
TOA	3-deoxy-3-amino	2	0.86	**0.91**
TFX	2-[4-(dimethylamino)phenyl]-3,6-dimethyl-1,3-b...	2	0.01	**0.70**
BDR	beta-d-ribofuranosyl	2	0.77	0.77
8AN	3'-amino-3'-deoxyadenosine	2		0.23
38E	(5z)-5-(3,5-difluoro-4-hydroxybenzylidene)-2,3...	2		0.96
TEP	theophylline	2	0.08	0.39
EUS	n-[(1r,2s,3s,4r,5s)-5-amino-4-{[(2s,3r)-3-amin...	2		0.71
TRS	2-amino-2-hydroxymethyl-propane-1,3-diol	2		0.81
CPT	cisplatin	2		0.68
EDE	edeine	2		0.71
AMP	adenosine	2	**0.61**	0.33
SUC	sucrose	2		0.63
1TU	4-(3,5-difluoro-4-hydroxybenzyl)-1,2-dimethyl-...	2	0.13	**0.98**
DAI	(3as,9as)-2-pentyl-4-hydroxymethyl-3a,4,9,9a-t...	2		0.25
LC2	n-[(1s,2r,3e,5e,7s,9e,11e,13s,15r,19r)-7,13-di...	2		0.82
SPE	thermine	2		0.61
AB9	(2r)-4-amino-n-{(1r,2s,3r,4r,5s)-5-amino-2-{2-...	2		0.89

Continued on next page

Table C.4 – Details for each ligand in the dataset. PDB codes and full names are in the first two columns, followed by the number of occurrences. The `inforna` and `RNAmigos` columns contain the score achieved on average by each tool on the given ligand.

Ligand	Name	Count	Inforna	RNAmigos
TOY	tobramycin	2		0.98
MGX	1-methylguanidine	1	0.19	**0.93**
6MN	2-amino-2-deoxy-6-o-phosphono-alpha-d-mannopyr...	1		0.86
GE2	3,5-diamino-cyclohexanol	1		0.98
2QB	5-(azidomethyl)-2-methylpyrimidin-4-amine	1		0.25
GCP	phosphomethylphosphonic	1		0.45
GE1	3,4-dideoxy-2,6-amino-alpha-d	1	**0.82**	0.72
IPA	isopropyl	1		0.53
NMZ	(2s)-4-amino-n-{(1r,2s,3r,4r,5s)-5-amino-3-{[3...	1		0.87
ZZR	3,6-diamino-1,5-dihydro[1,2,4]triazolo[4,3-b][...	1		0.93
GMP	guanosine	1		0.98
GZ4	7,8-dimethyl-2,4-dioxo-10-(3-phenylpropyl)-1,2...	1		0.79
3LK	bc-3205	1		0.36
3TS	(2s,3s,4r,5r,6r)-2-(aminomethyl)-5-azanyl-6-[(...	1		0.92
P12	4-[amino(imino)methyl]-1-[2-(3-ammoniopropoxy)...	1	**0.62**	0.41
ZZS	1,3,5-triazine-2,4-diamine	1		0.94
EZP	n-[(1r,2r)-1,3-dihydroxy-1-(4-nitrophenyl)prop...	1		0.46
RPO	(1r,2r,3s,4r,6s)-4,6-diamino-2-{[3-o-(2,6-diam...	1		0.96
GZ7	10-(6-carboxyhexyl)-8-(cyclopentylamino)-2,4-d...	1		0.98
ACE	acetyl	1		0.07
RIB	ribose	1	0.77	**0.84**
3K8	(14ar)-2,3,6-trimethoxy-11,12,13,14,14a,15-hex...	1		0.35
JS5	(2s,3s,4r,5r,6r)-5-amino-2-(aminomethyl)-6-((2...	1		0.91
SLZ	l-thialysine	1		0.78
THF	5-hydroxymethylene-6-hydrofolic	1		0.36
51B	2-[(3s)-1-{[2-(methylamino)pyrimidin-5-yl]meth...	1		0.07
218	1-[(4-amino-2-methylpyrimidin-5-yl)methyl]-3-(...	1		0.45
JS4	(2s,3s,4r,5r,6r)-5-amino-2-(aminomethyl)-6-((2...	1		0.93
62B	lefamulin	1		0.51
H4B	5,6,7,8-tetrahydrobiopterin	1	0.66	**0.70**
SLD	(3z)-n-[(4e)-5-(4-{(5s)-5-[(acetylamino)methyl...	1		0.68
EZM	n-[(1r,2r)-1,3-dihydroxy-1-(4-nitrophenyl)prop...	1		0.44
6NO	avilamycin	1		0.10
3J2	nagilactone	1		0.28
HEZ	hexane-1,6-diol	1		0.98
SJP	(2r,3r)-4-amino-n-[(1r,2s,3r,4r,5s)-5-amino-4-...	1		0.87
JS6	(1r,2r,3s,4r,6s)-4,6-diamino-2-{[3-o-(2,6-diam...	1		0.92
3L2	(4s,5r,10e,12z,16r,16as,17s,18r,19ar,23ar)-4-h...	1		0.11

Continued on next page

Table C.4 – Details for each ligand in the dataset. PDB codes and full names are in the first two columns, followed by the number of occurrences. The `inforna` and `RNAmigos` columns contain the score achieved on average by each tool on the given ligand.

Ligand	Name	Count	Inforna	RNAmigos
ISH	(7r)-7-[(dimethylamino)methyl]-1-[3-(dimethyla...	1	0.16	0.22
P14	n-[2-(2-{[(4-{[amino(imino)methyl]amino}butyl)...	1	0.69	0.66
G34	(3as,4r,5s,6s,8r,9r,9ar,10r)-5-hydroxy-4,6,9,1...	1		0.17
P13	n-[2-(3-aminopropoxy)-5-(1h-indol-5-yl)benzyl]...	1	0.55	**0.61**
KAN	kanamycin	1		1.00
7DG	7-deazaguanine	1	0.24	0.27
ON0	(1r,2r,3s,4r,6s)-4,6-diamino-2-{[3-o-(2,6-diam...	1		0.97
AB6	(2r)-4-amino-n-((1r,2s,3r,4r,5s)-5-amino-4-[(2...	1		0.89
GZG	4-{benzyl[2-(7,8-dimethyl-2,4-dioxo-3,4-dihydr...	1		0.53
G80	(3as,4r,5s,6s,8r,9r,9ar,10r)-5-hydroxy-4,6,9,1...	1		0.50
T8B	thermorubin	1		0.66
2TB	1,3-diamino-4,5,6-trihydroxy-cyclohexane	1		0.89
3AW	6-methyl-1,3,5-triazine-2,4-diamine	1		0.95
2BP	9h-purin-2-amine	1	0.27	**1.00**
EZG	n-[(1r,2r)-1,3-dihydroxy-1-(4-nitrophenyl)prop...	1		0.50
ARF	formamide	1		0.32
5CR	n-acetyl-l-phenylalanine	1		0.99
3KF	(2s,3r,4s,4ar)-2,3,4,7-tetrahydroxy-3,4,4a,5-t...	1		0.50
ZBA	12,13-epoxytrichothec-9-ene-3,4,8,15-tetrol-4,...	1		0.24
N6M	n-methyl-9h-purin-6-amine	1	0.23	**0.99**
4M2	3'-deoxy-3'-{[(2e)-3-(4-{[(4z)-6-o-(6-deoxy-3,...	1		0.23
N30	(1r,2r,3s,4r,6s)-4,6-diamino-2-[(5-amino-5-deo...	1		0.99
2QC	1-[4-(1,2,3-thiadiazol-4-yl)phenyl]methanamine	1	0.35	0.44
2HP	dihydrogenphosphate	1		0.65
0EC	6,7-dimethoxy-2-(piperazin-1-yl)quinazolin-4-a...	1	0.43	0.31
RBF	riboflavin	1		0.96
EKM	1-methyl-4-[(1e)-3-(3-methyl-1,3-benzothiazol-...	1		0.88
MMC	methyl	1		0.18
EVN	(2r,3r,4r,6s)-6-{[(2r,3ar,4r,4'r,5's,6s,6'r,7s...	1		0.03
SFG	sinefungin	1	0.73	**0.99**
EEM	[(3s)-3-amino-4-hydroxy-4-oxo-butyl]-[[(2s,3s,...	1		0.23
CIR	citrulline	1		0.70
AKN	(2s)-n-[(1r,2s,3s,4r,5s)-4-[(2r,3r,4s,5s,6r)-6...	1		0.95
34G	emetine	1		0.53
MT9	(3r,4s,5s,7r,9e,11s,12r)-12-ethyl-11-hydroxy-3...	1		0.91
RAP	rapamycin	1	0.50	**0.66**
S81	(1r,2r,3s,4r,6s)-4,6-diamino-2,3-dihydroxycycl...	1		0.94
ATP	adenosine-5'-triphosphate	1		0.08

Continued on next page

Table C.4 – Details for each ligand in the dataset. PDB codes and full names are in the first two columns, followed by the number of occurrences. The `inforna` and `RNAmigos` columns contain the score achieved on average by each tool on the given ligand.

Ligand	Name	Count	Inforna	RNAmigos
GND	2-amino-5-guanidino-pentanoic	1		0.95
PRL	proflavin	1		0.56
G0B	(1s,2r,3s,4r,6s)-4,6-bis{[amino(iminio)methyl]...	1	**0.94**	0.90
PMZ	1-[10-(3-dimethylamino-propyl)-10h-phenothiazi...	1	0.03	0.05
TPS	thiamin	1		0.11
PDI	phosphoric	1		0.60
A2F	2-fluoroadenine	1	0.17	**0.94**
CNY	13,15-diamino-2-(aminomethyl)-3,4,9,12-tetrahy...	1		0.95
LEU	leucine	1		0.97
3QB	lincomycin	1		0.71
7AL	chlorolissoclimide	1		0.33
29G	pyrimido[4,5-d]pyrimidine-2,4-diamine	1	0.33	**0.96**
D2X	3-[(4-hydroxy-2-methylpyrimidin-5-yl)methyl]-5...	1		0.36
HN8	haemanthamine	1		0.65
95H	˜{n}-[(1˜{r},2˜{r})-1-[(2˜{r},3˜{r},4˜{s},5˜{r...	1		0.26
MYC	3,5,7-trihydroxy-2-(3,4,5-trihydroxyphenyl)-4h...	1		0.53
AU3	gold	1		0.21
DKM	5-[(3s,4s)-3-(dimethylamino)-4-hydroxypyrrolid...	1		0.74
D2C	(2s,4s,4ar,5as,6s,11r,11as,12r,12ar)-7-chloro-...	1		0.37
6UQ	(2r,3s,4r,6s)-4-hydroxy-6-{[(2r,3ar,4r,4'r,5's...	1		0.09
3KD	(1s,2s,12bs,12cs)-2,4,5,7,12b,12c-hexahydro-1h...	1		0.36
B1Z	adenosylcobalamin	1		0.11
OLZ	o-(2-aminoethyl)-l-serine	1		0.81
ISI	(7s)-7-[(dimethylamino)methyl]-1-[3-(dimethyla...	1	0.16	0.18
AGU	aminoguanidine	1	0.25	0.82
FFO	n-[4-({[(6s)-2-amino-5-formyl-4-oxo-3,4,5,6,7,...	1	**0.66**	0.30
HRG	l-homoarginine	1		0.72
3J6	(3beta,7alpha)-3,7,15-trihydroxy-12,13-epoxytr...	1		0.41
CTC	7-chlorotetracycline	1		0.48
DX4	2-amino-1,9-dihydro-6h-purine-6-thione	1		0.94
L94	n'-{(z)-amino[4-(amino{[3-(dimethylammonio)pro...	1	0.30	0.27
NME	methylamine	1		0.88
PA1	2-amino-2-deoxy-alpha-d-glucopyranose	1	0.85	0.72
B12	cobalamin	1		0.01
L8H	4-methoxynaphthalen-2-amine	1		0.47
BFT	s-benzoylthiamine	1		0.26
ROS	n,n'-tetramethyl-rosamine	1		0.30
MIX	1,4-dihydroxy-5,8-bis({2-[(2-hydroxyethyl)amin...	1	0.73	0.78

Continued on next page

Table C.4 – Details for each ligand in the dataset. PDB codes and full names are in the first two columns, followed by the number of occurrences. The `inforna` and `RNAmigos` columns contain the score achieved on average by each tool on the given ligand.

Ligand	**Name**	**Count**	`Inforna`	`RNAmigos`
NEB	2-deoxy-d-streptamine	1		0.81
M5Z	(1r,2r,3s,4r,6s)-4,6-diamino-2-{[3-o-(2,6-diam...	1		0.94
PQ0	2-amino-4-oxo-4,7-dihydro-3h-pyrrolo[2,3-d]pyr...	1	0.22	0.94
TOB	1,3-diamino-5,6-dihydroxycyclohexane	1		0.84
V71	(1r,2r,3s,4r,6s)-4,6-diamino-2,3-dihydroxycycl...	1		0.94
3AY	pyrimidine-2,4,6-triamine	1		0.98
6O1	evernimicin	1		0.03
SVN	thieno[2,3-b]pyrazin-7-amine	1	0.15	0.94
29H	2-aminopyrimido[4,5-d]pyrimidin-4(3h)-one	1	0.27	0.98
6YG	2-[(3~{s})-1-[(2-methoxypyrimidin-5-yl)methyl]...	1		0.59
GE3	5-methyl-4-methylamino-tetrahydro-pyran-2,3,5-...	1		0.73
MGR	malachite	1	0.05	0.34
IEL	n~6~-[(1z)-ethanimidoyl]-l-lysine	1		0.67
VIB	3-(4-amino-2-methyl-pyrimidin-5-ylmethyl)-5-(2...	1	0.30	**0.87**
BME	beta-mercaptoethanol	1		0.72
N33	(2s,3r,4r,5s,6r)-3-amino-4-({[(2s,3r,4r,5s,6r)...	1		0.99
ZIT	azithromycin	1		0.37
DGP	2'-deoxyguanosine-5'-monophosphate	1		0.94
3V6	bactobolin	1		0.49
IR3	iridium	1		0.20
EMK	(2r,3s,4r,5r,8r,10r,11r,12s,13s,14r)-2-ethyl-3...	1		0.04
G19	(2s,3ar,4r,5s,6s,8r,9r,9ar,10r)-2,5-dihydroxy-...	1		0.17
MUL	tiamulin	1		0.15
3H3	4-{(2r,5s,6e)-2-hydroxy-5-methyl-7-[(2r,3s,4e,...	1		0.63
7MB	agelastatin	1		0.97
917	n-({(5s)-2-oxo-3-[4-(1,3-thiazol-5-yl)phenyl]-...	1		0.36
RS3	1-deoxy-1-[8-(dimethylamino)-7-methyl-2,4-diox...	1		0.96

Appendix D

VERNAL : A Tool for Mining Fuzzy Network Motifs in RNA - Appendix

Contents

D.1 RNA data

D.1.1 Chopping algorithm

We present here the algorithm used to chop RNA into fixed maximal size pieces. The idea of the algorithm is to recursively cut the RNA in halves up until the maximum size is reached. To minimally disrupt the structure, we cut the structure in the orthogonal direction to the principal axis of variation, according to Principal Component Analysis. This is detailed in **Algorithm 3**.

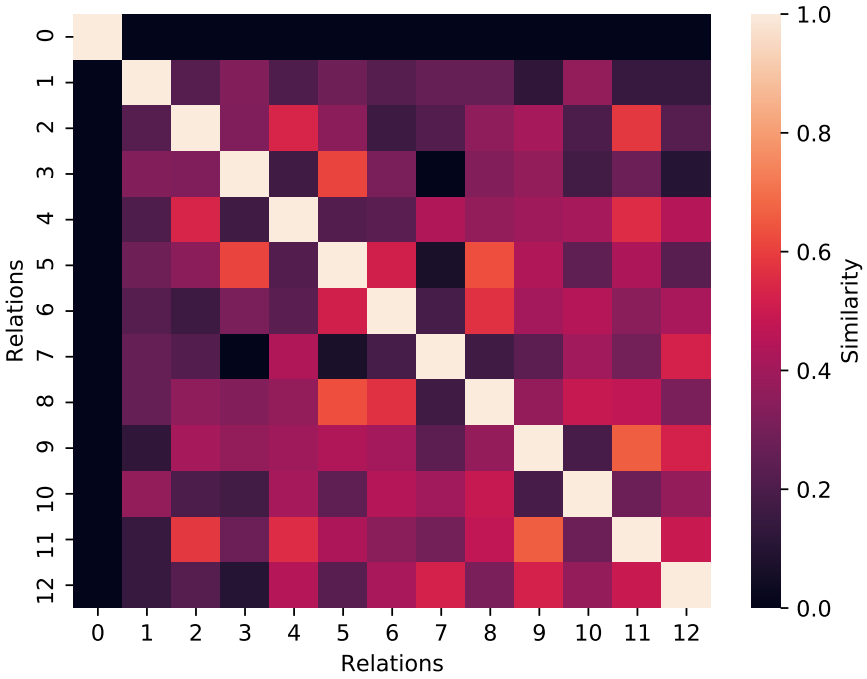
Algorithm 3: Chopper algorithm

Data: Full RNA Graph g, Maximum number of nodes N
Result: List of sub-structures of maximum size N.

```
if |g| ≤ N then
    return
end
g
else
    g_a, g_b ← Split g in halves based on the PCA
    axes
    return chopper(g_a)
    return chopper(g_b)
end
```

D.1.2 Isostericity

We represent RNA with graphs whose nodes are nucleotides and edges represent an interaction between those nucleotides. Let $\mathcal{R}$ denote the set of these interactions, this set contains backbone covalent interactions, 'cWW' interactions that are denoted as canonical base pairs and 11 other interactions that represent other geometries. Isostericity denotes the closeness between relation types at the 3-D level (base pairing geometries). Stombaugh [Stombaugh et al., 2009] computed the geometric discrepancy between all pairs of relation types (Shown in **Figure D.1**). We include a representation of the notion of isostericity between edge types in **Figure D.1**.



Supplementary Figure D.1 – Isostericity matrix between relation types.

Graphs in RNA are directed but reciprocal, i.e. if nodes A and B are linked by an edge A=(cSH)=>B, then automatically B and A are linked with an edge : B=(cHS)=>A. The aforementioned isostericity relationship is symmetric, which means that for instance $\forall r \in \mathcal{R}, iso(cHS, r) = iso(cSH, r)$. This would make our directed graphs as expressive as undirected graphs. A first solution is to simply make our graphs undirected and use the isostericity directly. However, relevant information is encoded in the direction of the edges, for instance the direction of the backbone. Therefore, we expand the isostericity values by adding the following rules :

$$\begin{cases} iso(b53, b35) = 0.2 \\ \forall r \in \mathcal{R} \setminus \{b53, b35\},\ iso(b53, r) = 0 \\ \forall r \in \mathcal{R},\ iso(r, r) = 1. \end{cases}$$

For all the other cases, we use the value reported in the isostericity matrix.

D.2 Graph Edit Distance

D.2.1 Model Selection with Graph Edit Distance

The choice of similarity function s_G is application specific. One can use a function which maximizes performance on a downstream supervised learning task, or one can choose a similarity function which best encodes structural identity [Hamilton et al., 2017b]. Since supervised learning data for RNA 3D structures is scarce, we opt for the latter and propose the Graph Edit

Distance (GED) (or its similarity analog $exp[-GED]$) between rooted subgraphs, as this is the widely accepted yet computationally intensive gold standard for structure comparison [Gao et al., 2010]. Hence, we choose the s_G which most closely correlates with GED. Interestingly, GED is a generalization of the subgraph isomorphism problem [Bunke and Riesen, 2008] which is at the core of previous RNA motif works such as `CaRNAval` and `rna3dmotif`.

D.2.2 Rooted GED

In a nutshell, the GED between two graphs G, H is the minimum cost set of modifications that can be made to G in order to make it isomorphic to H. This naturally encodes a notion of similarity since similar pairs will require few and inexpensive modifications, and vice versa. The Graph Edit Distance (GED) between two graphs g and h is thus defined as follows:

$$GED(G,H) = \min_{(e_1,\ldots,e_k)\in\Upsilon(G,H)} \sum_{i=1}^{k} c(o_i). \tag{D.1}$$

where Υ is the set of all edit sequences which transform G into H. Edit operations include: node/edge matching, deletion, and insertion. $c(o)$ is the cost of performing edit operation o and c is known as the cost function. Since we will be decomposing our graphs as rooted subgraphs, we define a slight modification to the GED formulation which compares two graphs given that their respective roots must be matched to each other. This algorithm is detailed in **Algorithm 4**.

D.2.3 RNA GED

We have adapted this algorithm to RNA data. We use the isostericity matrix [Stombaugh et al., 2009] for edge substitutions, and do not apply a penalty to node substitutions. Let $\mathcal{E}(.)$ be a function that returns the edge label for a given edge, and ISO the isostericity function which returns the similarity between edge types. We define an RNA cost function over pair of edges p and q as follows :

$$c(p \to q) = \text{ISO}(\mathcal{E}(p), \mathcal{E}(q))$$

$$c(p \to \emptyset) = \begin{cases} \alpha & \text{backbone} \\ \beta & \text{canonical} \\ \theta & \text{non-canonical} \end{cases}$$

For our experiments, we set $\alpha = 1$, $\beta = 2$, $\theta = 3$ to emphasize the differences in non-canonical interactions between graphs. We propose a simple modification to allow for comparison of rooted graphs (**Algorithm 4**), and use the general version of GED to validate the ultimate full subgraph-level quality of our identified motifs.

We include in **Figure D.2** an example of GED values for two pairs of graphlets, illustrating how similar graphs get lower values of distance.

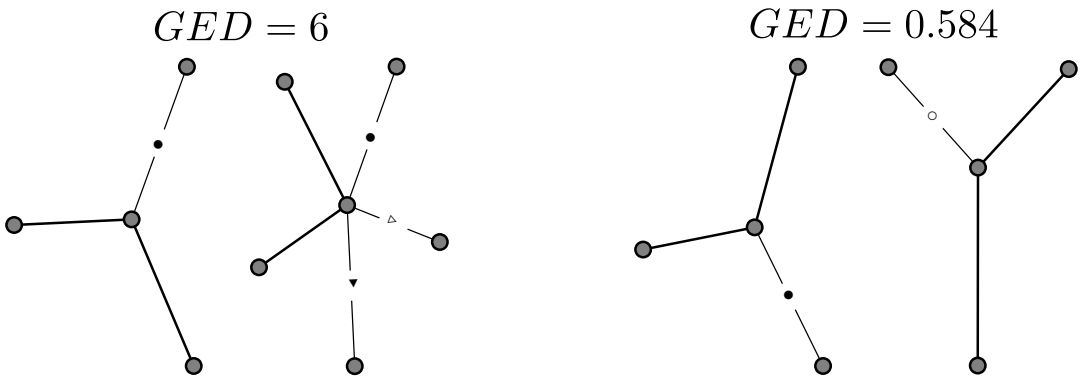
Algorithm 4: Rooted A* GED

Data:

- Pair of graphs G, H, (WLOG let G be the smaller of the two graphs.)
- cost function c
- heuristic h.
- $r_G \in \mathcal{N}(G)$ root in first graph
- $r_H \in \mathcal{N}(G)$ root in second graph

Result: Minimum cost rooted distance and alignment between two graphs.

$OPEN \leftarrow priorityQueue()$
$V_G \leftarrow G.nodes()$
$V_H \leftarrow H.nodes()$
$v \leftarrow$ first node in G $OPEN.add((r_G, r'_H), c(r_G, r_H) + h(v_0, v')$ **while** $OPEN$ **do**
  $v_{min} \leftarrow OPEN.pop()$
  Let $\mathcal{M}_k \leftarrow$ be partial mapping $\{(v_1, v'_1), .., (v_k, v'_k)\}$
  if $|\mathcal{M}_k| = |V_G|$ **then**
    Mapping complete
    return $\mathcal{M}_k$
  end
  Add nodes at next depth
  foreach $u \in V_H \setminus v_{min}$ **do**
    $OPEN.add(v_{min} \cup (v_{k+1}, u), c(v_{k+1}, u) + h(v_{k+1}, u))$
  end
end



Supplementary Figure D.2 – Examples of similar and dissimilar pairs according to the GED A* algorithm.

D.3 Rooted Subgraph Comparisons

Here, we define a similarity function between a pair of rooted subgraphs, g_u and $g_{u'}$. We propose two main classes of s_G: ring-based, and matching-based similarity functions.

D.3.1 Edge ring similarity functions

The first s_G functions we consider are a weighted sum of a distance between l-hop neighborhoods (aka rings) of each subgraph. This formulation is inspired by the function proposed in [Ribeiro et al., 2017]. We let $R_u^l = \{(u', w) : \delta(u, u') = l \quad \forall (u', w) \in E\}$ be the set of edges at distance l from the root node u. Let d be a normalized similarity function between two sets of edges. One such function can be formulated as a matching operation between the edges of each ring. Let $\mathbf{X}$ is a binary matrix describing a matching from one ring to another and $\mathbf{S}$ a scoring function corresponding to this matching, d writes as :

$$d(R_u, R_v) := \min - \sum_{e \in R_u} \sum_{e' \in R_v} \mathbf{S}_{\mathcal{L}(e), \mathcal{L}(e')} \mathbf{X}_{e,e'}$$

Let $0 < \lambda < 1$ be a decay factor to assign higher weight to rings closer to the root nodes, and N^{-1} be a normalization constant to ensure the function saturates at 1. Then we can obtain a structural similarity for the rooted subgraphs around u and v as :

$$k_L(u, v) := 1 - N^{-1} \sum_{l=0}^{L-1} \lambda^l d(R_u^l, R_v^l)$$

The first function (`R_1`) simply uses a delta function to compare to different edges. This assignment problem thus reduces to computing the intersection over union score between the histograms f_R of edge labels found at each ring. However, this function treats all edge types equally and ignores the isostericity relationships. The second function (`R_iso`) has a matching value of 1 for backbone edges matched with backbone edges, 0 for backbone matched with any non covalent bond and the isostericity value for the similarity value of two non covalent bonds.

D.3.2 Matching-based similarity functions

Here, s_G operates on the output of a function $f : g_u \to \Omega$ which decomposes a rooted subgraph into a set of objects Ω (e.g. sets/rings of nodes, edges, or smaller subgraphs such as graphlets [Shervashidze et al., 2009]). These objects can then be assigned structural and positional compatibilities. We let $\mathbf{C}_{\omega,\omega'}$ be the structural compatibility between objects ω, ω', for example, edge isostericity. Next, $\mathbf{D}_{\omega,\omega'}$ assigns a cost on pairs of objects depending on the relative path distance to their respective root nodes. We propose various similarity functions, based on optimal matching of these objects with the most general form being:

$$s_G(g_u, g_{u'}) := \min_{\mathbf{X}} \sum_{\omega \in \Omega} \sum_{\omega' \in \Omega'} (\alpha \mathbf{C}_{\omega,\omega'} + \beta \mathbf{D}_{\omega,\omega'}) \mathbf{X}_{\omega,\omega'} \tag{D.2}$$

where $\mathbf{X}$ is a binary matrix describing a matching from the elements of Ω to Ω', α and β are user-defined weights for emphasizing positional vs structural compatibility. We solve for the optimal matching between two sets of structural objects using the Hungarian algorithm [Kuhn and Yaw, 1955].

Since the degree of our graphs is strongly bounded (max degree 5), we can define a graphlet as a rooted subgraph of radius 1 and obtain a manageable number of possible graphlets. This lets us define an f which produces structurally rich objects. Moreover the rooted aspect and the small size of those graphs make the GED computation tractable.

While the GED computation is tractable for such small graphs, it is still expensive when repeated many times. For this reason, we implement a solution caching strategy which stores the computed GED when it sees a new pair of graphlets, and looks up stored solutions when it recognizes a previously seen pair **Supplementary Algorithm 5**.

We can now define $\mathbf{S}$ from Equation D.2 as $\mathbf{S}_{ij} = \exp[-\gamma \mathrm{GED}(g_i, g_j)]$. We apply an exponential to the distance to bring the distances to the range [0,1], and convert them to a similarity. An optional scaling parameter γ is included to control the similarity penalty on more dissimilar graphs. We also note that the construction of $\mathbf{S}$ can be parallelized but we leave the implementation for future work.

D.3.3 Additional settings

We have experimented with several additional parameters. We tried including an Inverse Document Frequency (IDF) weighting to account for the higher frequency of non canonical interaction. This amounted to scaling all comparison values by the product of the IDF term they involved.

We also tried adding a renormalization scheme to give higher values to matches of long rings. In particular, we want to express that having a match of 9 out of 10 elements is stronger than having a match of 2 out of 3. Let S be the raw matching score, $\mathbb{S}$ the normalized one and L be the length of the sequences, we have tried two normalization settings, the "sqrt" and "log" ones :

$$\text{sqrt} : \mathbb{S} = \left[\frac{S}{L}\right]^{\frac{5}{\sqrt{L}}}$$
$$\log : \mathbb{S} = \left[\frac{S}{L}\right]^{\frac{1}{1+\log L}}$$

D.3.4 Graphlets hashing and distributions

We build a hash function which maps isomorphic graphlets to the same output, while assigning different outputs to non-isomorphic ones, allowing us to look up graphlet GED values. This is done by building a sparse representation of an explicit Weisfeiler-Lehman isomorphism kernel, with a twist that edge labels are included in the neighborhood aggregation step. The resulting hash consists of counts over the whole graphlet of hashed observed sequences of edge labels. We enforce the edge label hashing function to be permutation invariant by sorting the observed label sequence. In this manner, isomorphic graphs are given identical hash values regardless of node ordering. Our hashing procedure outlined in **Supplemental Algorithm 5** also allows us to study the distribution of graphlets composing RNA networks **Supplemental Fig D.3**, where we can observe a characteristic power law distribution.

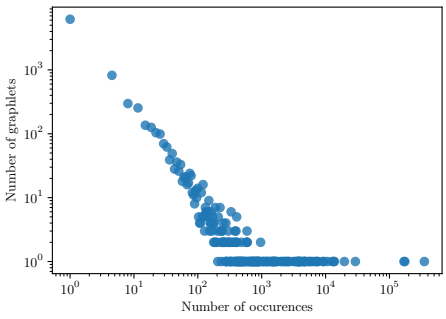
Algorithm 5: Weisfeiler-Lehman Edge Graphlet Hashing

Data:

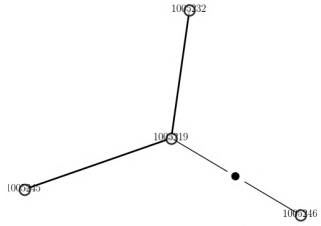
- Graphlet g,
- Maximum depth K
- HASH, function from strings to integers
- $\mathcal{L}$ function returning the label for an edge

Result: Hash code for graphlet h

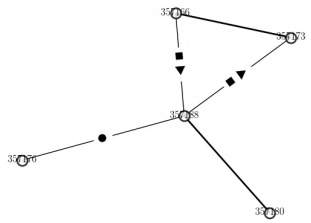
$h \leftarrow counter()$
foreach $k \in \{1, .., K\}$ **do**
  foreach $u \in g_N$ **do**
    $l_u^k \leftarrow \text{HASH}(\{\mathcal{L}(u, v) \oplus l_v^{k-1} \quad \forall v \in \mathcal{N}(u)\})$
  end
  $h \leftarrow h \cup counter(\{l_u \quad \forall u \in g\})$
end
return h



a Graphlet frequency distribution



b Most frequent graphlet



c Example of a rare graphlet.

Supplementary Figure D.3 – Graphlet distribution and examples.

D.4 Model Training

D.4.1 Mathematical framing

In order to identify related groups of rooted subgraphs using only the similarity function we would have to perform and store N^2 operations (where N is the total number of nodes in $\mathbb{G}$. When working with an order of 10^7 nodes, this quickly becomes prohibitive. Once nodes in each graph are embedded into a vector space, searches and comparisons are much cheaper.

We therefore approximate the s_G function over all pairs of rooted subgraphs using node embeddings. We use a Relational Graph Convolutional Network (RGCN) model [Schlichtkrull et al., 2018] as parametric node embedding function $\phi(u) \rightarrow \mathbb{R}^d$ which maps nodes to a vector space of dimensionality d. The network is implemented in Pytorch [Paszke et al., 2019] and DGL [Wang et al., 2019]. It is trained to minimize :

$$\mathcal{L} = \|\langle \phi(g_u), \phi(g_{u'}) \rangle - s_G(g_u, g_{u'})\|_2^2, \tag{D.3}$$

To focus performance on subgraphs that contain non canonical nodes and avoid the loss to be flooded by the canonical interactions (Watson Crick pairs), we then scale this loss based on the presence of non canonical interactions in the neighborhood of each node being compared. Given the rate of non canonical interactions r and $\mathbb{1}_u$ an indicator function that denotes the presence of non-canonical interactions in the neighborhood of node u, the scaling $\mathbf{S}_{u,v}$ of the $g_u, g_{u'}$ term writes as :

$$\mathbf{S}_{g_u, g_{u'}} = (1 + \frac{\mathbb{1}_{g_u}}{r})(1 + \frac{\mathbb{1}_{g_{u'}}}{r}), \quad \mathcal{L}_{scaled} = \mathbf{S} \odot \mathcal{L} \tag{D.4}$$

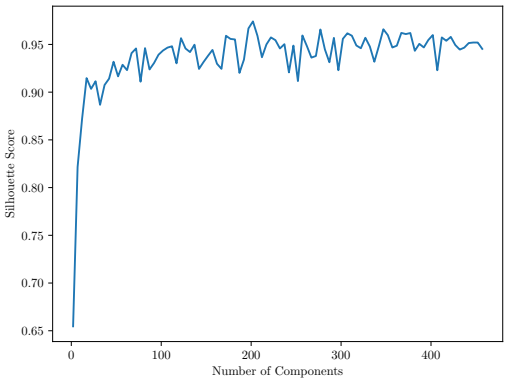
D.4.2 Architecture and hyperparameters choices

We use as many layers as the number of hops in the similarity function so that both functions have access to the same support in the subgraphs. The embeddings resulting from this message passing are then fed to a Linear Layer. The dimension per layers that we used were $[16, 32, 32]$, we used the default instantiation of DGL with ReLus and self-loops. We tried using Dropout but chose to not include it in our model in the end. The model was then trained using Adam on sampling pairs on the order of k*N, where N is the number of chunks and $k \sim 5$. Empirically, we had convergence for values around $k = 3$.

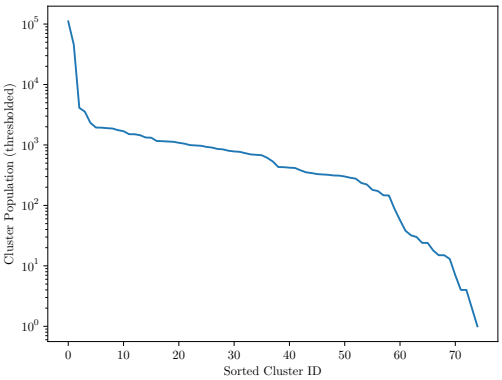
We can then perform clustering in the embedding space using any linear clustering algorithm. We denote the resulting clusters as 1-motifs as they represent the aforementioned structural blocks of RNA.

D.5 Metrics on the structural clusters

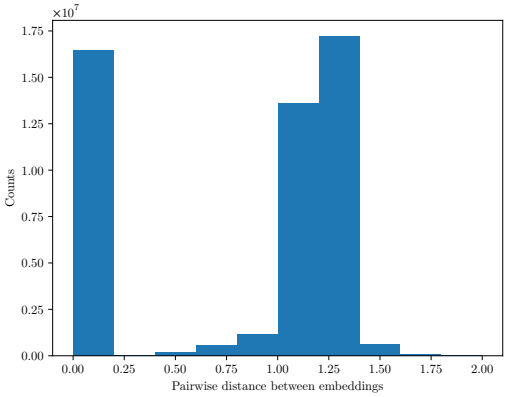
We present in **Figure D.4** some metrics on the clustering of our structural embeddings. The silhouette scores suggest that the number of clusters necessary to obtain satisfying scores is around 60. However seeding the method with more centroids can yield better results, even if they end up collapsed in the end. We used 200 centroids that got collapsed into 60 distinct centroids.



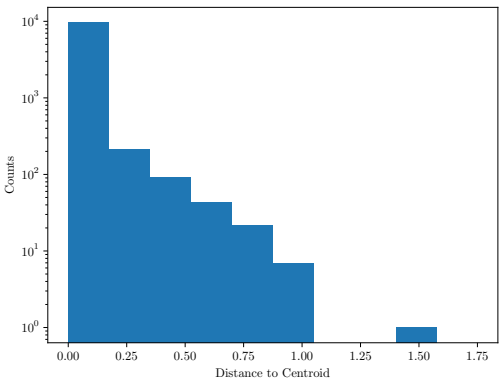
a Silhouette score for different number of initial centroids.



b Nodes per cluster ids.



c Distance between randomly selected pairs of embeddings.



d Distance between nodes embeddings and their centroid

Supplementary Figure D.4 – Metrics when clustering with minibatch K-means, with $k = 262$

The cluster population has an interesting distribution, with the two first clusters heavily populated (stem nodes), then an interesting population of graphlets and finally very rare ones. Despite the curse of dimensionality, we see that the nodes are much closer to their centroids than they are one from another. These metrics suggest that the structures present in RNA do fall into well separated clusters.

D.6 Retrieve complexity

In this section, we want to investigate the complexity of the retrieve algorithm. This complexity depends highly on both the topology of the meta-graph, the query graph and the individual RNA-graphs.

Let N_p be the number of edges in each of the p parallel edges of the query graph, and E_p be the corresponding meta-edge. We consider the edges of the query graph in the order of the parallel edges, let $p(t) = \min_k, \sum_k N_k > t$, the parallel edge considered at time t.

At each step t, the complexity bound is going to depend on the number of candidate motifs inside each RNA graph at time t-1 as well as the number of possible additional edges to insert into those candidates. If we denote as $M_{g,t}$ the number of candidates in graph g at time t, and $N_{g,t}$ the number of edges in graph g that belong to the meta-edge $E_{p(t),g}$, the complexity writes as $O(\sum_t \sum_{g\in\mathcal{G}} E_{p(t),g} M_{g,t})$ The term $E_{p(t),g}$ mostly acts as a sparsity term, as it would not exceed ten but can very often be zero if the graph does not include such an edge. Therefore, we introduce the notation $\mathcal{G}_p$, the set of RNA graphs that contain an edge in E_p, to omit this term. We also introduce $t_{g,t} = \sum_{l<t} E_{p(l),g}$ the number of query edges explored present in graph g at time t.

Let us now try to address the second term $M_{g,t}$. We divide our algorithm into each of the parallel edges and dive into the evolution of this term. $M_{g,t}$ represents the number of combinations of nodes in a given graph that are currently considered as a candidate motif. Every different p is going to launch a combinatorial explosion that results from the numerous possibilities of combining nodes. For RNA graphs, it is mostly a problem for stems that are ubiquitous and all share the same structure. Starting from all stem nodes, after one merging step we have to add all possible combinations of adjacent stem nodes. We give a loose bound of this number that is practical for our application, but note that after sufficient merging, there are just the full stems as candidates, showing that this bound becomes loose.

We rely on the fact that we can delete a partial solution if it is completely contained in another one because its expansion can only result in lower scores. Thus after a given motif gets expanded we can remove it from $\mathcal{M}$. Therefore, we can bound this number by counting the number of children a given element can produce. Using set structure enables fast neighborhood checking operations but also ensures we do not add the same object twice. For stems that are the worst case scenario, each connected component of length k can yield a maximum of $k+2$ children but any of these children is added twice because the edge has two ends. Therefore after k expansions,we have $M_{g,t+k} < M_{g,t}\frac{(k+2)!}{2^k}$. This results in a final loose complexity bound of : $O(\sum_t \sum_{g\in\mathcal{G}_{p(t)}} \frac{(t_{g,t}+2)!}{2^{t_{g,t}}})$.

As mentioned before this is not a good bound when k grows because the stems get completed. Therefore, there are a lot of ways to select 3 adjacent stem nodes of a 6 stem, but only one to get all 6. In practice, during each cycle, the number $M_{g,t}$ grows and decreases, making the

algorithm tractable, starting back from reasonable numbers at each cycle. Empirically, running the retrieve algorithm on a single core rarely exceeds a few seconds.

D.7 MAA

Algorithm 6: Motif **A**ggregation **A**lgorithm (MAA). At each step, k, the algorithm iterates through edges (m, m') of the meta-graph, applying Algorithm 1 to construct a $k+1$ motif μ. The updated meta-connectivity is stored as new meta-edges.

Data:

- Meta-Graph $\mathcal{G}$,
- Minimum density δ (number of motif instances)
- Number of steps K

Result: List of meta-graphs

$\mathcal{M} \leftarrow list()$
$\mathcal{E} \leftarrow \mathcal{G}.edges()$
foreach $k \in \{1, .., K\}$ **do**
  $\mathcal{E}' \leftarrow \varnothing$
  $\mathcal{M}[k] \leftarrow list()$
  while $\mathcal{E}$ **do**
    $m, m' \leftarrow \mathcal{E}.pop()$
    $\mu \leftarrow$ `merge`$(m.subgraphs, (m, m'))$
    if $|\mu| > \delta$ **then**
      $\mathcal{M}[t].append(\mu)$
      `/* Connect new node to adjacent clusters */`
      foreach $c' \in \mathcal{N}(\mu)$ **do**
        $\mathcal{E}'.add((\mu, c'))$
      end
    end
    $\mathcal{E} \leftarrow \mathcal{E}'$
  end
end
return $\mathcal{M}$

D.8 Full results for the similarity function validations

We include in this section the full results we got for a grid search validation in the absence of a better way to guide our intuition **Table D.2, D.3**, as well as a condensed version in **Table D.1**.

method	**depth**	**decay**	**normalization**	**r**	**r_th**	**time (s)**
edge hist.	1	0.5	sqrt	0.69	0.80	<**0.001**
edge hist. + iso	1	0.5	sqrt	0.74	0.80	<**0.001**
edge hungarian + iso	1	–	sqrt	0.80	0.91	<**0.001**
graphlets hist.	1	any	None	**1.0**	**1.0**	0.029
graphlet hungarian	1	–	None	**1.0**	**1.0**	0.030
edge hungarian + iso	2	–	sqrt	**0.64**	**0.75**	<**0.001**
graphlets hist.	2	0.8	sqrt	0.63	0.68	0.102
graphlet hungarian	2	–	sqrt	0.63	0.70	0.433
1hop RGCN	1	–	–	0.826	0.927	<**0.001**
2hop RGCN	2	–	–	0.596	0.577	<**0.001**

Table D.1 – Correlation with the GED for different kernels and embedding settings. For each experiment (row) we report the Pearson correlation coefficient **r**. Correlations are computed pairwise on a subset of 200 nodes, as described in the main text. We include the result on this whole data as **r**, as well as a subset which include only pairs of nodes with a GED below 6 (**r_th**)

Method	Decay	IDF	Normalization	Correlation	Thresholded Correlation
graphlet	NaN	NaN	None	0.999982	1.000000
R_graphlets	0.8	NaN	None	0.999982	1.000000
R_graphlets	0.5	NaN	None	0.999982	1.000000
R_graphlets	0.3	NaN	None	0.999982	1.000000
graphlet	NaN	NaN	sqrt	0.907263	0.945552
R_graphlets	0.8	NaN	sqrt	0.907263	0.945552
R_graphlets	0.5	NaN	sqrt	0.907263	0.945552
R_graphlets	0.3	NaN	sqrt	0.907263	0.945552
hungarian	NaN	0.0	None	0.796640	0.910618
hungarian	NaN	0.0	sqrt	0.784811	0.916345
hungarian	NaN	1.0	sqrt	0.777268	0.916563
hungarian	NaN	1.0	None	0.756889	0.862990
R_iso	0.5	1.0	sqrt	0.737533	0.798435
R_iso	0.8	1.0	sqrt	0.737533	0.798435
R_iso	0.3	1.0	sqrt	0.737533	0.798435
R_iso	0.3	0.0	sqrt	0.731762	0.799086
R_iso	0.5	0.0	sqrt	0.731762	0.799086
R_iso	0.8	0.0	sqrt	0.731762	0.799086
R_iso	0.3	1.0	None	0.715601	0.817003
R_iso	0.8	1.0	None	0.715601	0.817003
R_iso	0.5	1.0	None	0.715601	0.817003
R_iso	0.5	0.0	None	0.701981	0.829820
R_iso	0.3	0.0	None	0.701981	0.829820
R_iso	0.8	0.0	None	0.701981	0.829820
R_1	0.5	1.0	sqrt	0.689727	0.798827
R_1	0.5	1.0	None	0.689727	0.798827
R_1	0.3	1.0	sqrt	0.689727	0.798827
R_1	0.3	1.0	None	0.689727	0.798827
R_1	0.8	1.0	sqrt	0.689727	0.798827
R_1	0.8	1.0	None	0.689727	0.798827
R_1	0.3	0.0	sqrt	0.683472	0.838047
R_1	0.3	0.0	None	0.683472	0.838047
R_1	0.5	0.0	sqrt	0.683472	0.838047
R_1	0.8	0.0	None	0.683472	0.838047
R_1	0.5	0.0	None	0.683472	0.838047
R_1	0.8	0.0	sqrt	0.683472	0.838047

Table D.2 – One hop rooted subgraph correlation to GED

Method	Decay	IDF	Normalization	Correlation	Thresholded Correlation
hungarian	NaN	1.0	sqrt	0.640183	0.745018
graphlet	NaN	NaN	sqrt	0.627242	0.701199
R_graphlets	0.8	NaN	sqrt	0.625675	0.683274
hungarian	NaN	0.0	sqrt	0.606648	0.689525
hungarian	NaN	1.0	None	0.589180	0.624824
R_graphlets	0.5	NaN	sqrt	0.588992	0.598480
R_graphlets	0.8	NaN	None	0.570733	0.584230
R_iso	0.8	1.0	sqrt	0.565505	0.546167
R_iso	0.8	0.0	sqrt	0.558772	0.541444
R_graphlets	0.3	NaN	sqrt	0.557002	0.512982
graphlet	NaN	NaN	None	0.554304	0.595573
hungarian	NaN	0.0	None	0.552988	0.664127
R_graphlets	0.5	NaN	None	0.548846	0.519385
R_iso	0.8	1.0	None	0.538128	0.532596
R_iso	0.5	1.0	sqrt	0.529356	0.499034
R_graphlets	0.3	NaN	None	0.527732	0.455359
R_iso	0.5	0.0	sqrt	0.524007	0.492044
R_iso	0.8	0.0	None	0.512851	0.516201
R_iso	0.5	1.0	None	0.506934	0.479951
R_1	0.8	1.0	sqrt	0.500248	0.506578
R_1	0.8	1.0	None	0.500248	0.506578
R_iso	0.3	1.0	sqrt	0.491474	0.439565
R_iso	0.3	0.0	sqrt	0.487652	0.432580
R_iso	0.5	0.0	None	0.485761	0.463078
R_1	0.8	0.0	sqrt	0.481469	0.506201
R_1	0.8	0.0	None	0.481469	0.506201
R_iso	0.3	1.0	None	0.474367	0.420955
R_1	0.5	1.0	None	0.469777	0.453362
R_1	0.5	1.0	sqrt	0.469777	0.453362
R_iso	0.3	0.0	None	0.456932	0.406642
R_1	0.5	0.0	sqrt	0.452601	0.450131
R_1	0.5	0.0	None	0.452601	0.450131
R_1	0.3	1.0	sqrt	0.438522	0.395088
R_1	0.3	1.0	None	0.438522	0.395088
R_1	0.3	0.0	sqrt	0.422765	0.391077
R_1	0.3	0.0	None	0.422765	0.391077

Table D.3 – Two hop rooted subgraph correlation to GED.

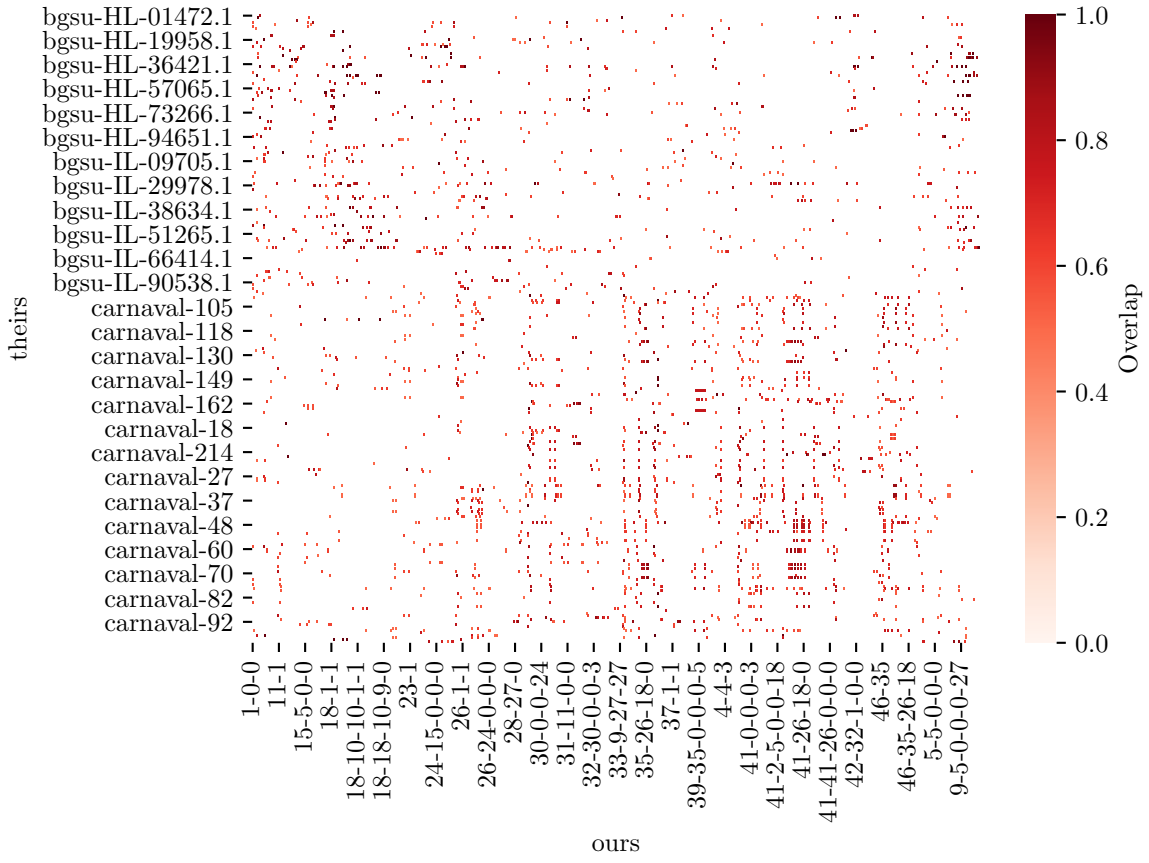
D.9 MAA supplemental results

Motif Size	Number of Motifs	Mean Number of Instances
1	48	2041.90
2	101	830.22
3	163	1112.97
4	260	1090.64
5	460	1264.23
6	823	1318.79
7	1641	1308.76

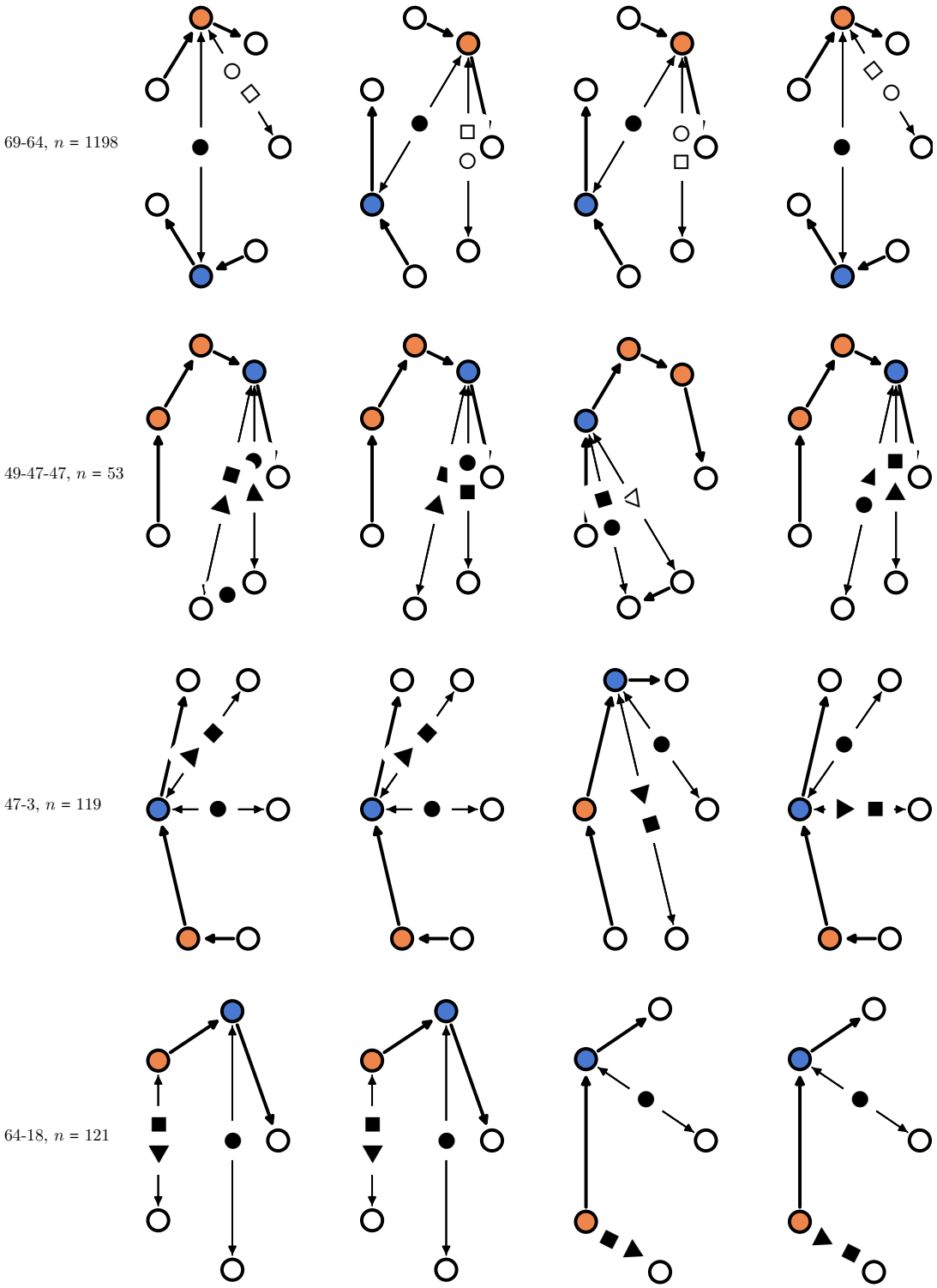
Table D.4 – Number of motifs and of instances per motif for each size, as found by the MAA algorithm

Dataset	Covered	Missed
BGSU Petrov et al. [2013b]	112	14
RNA3DMotif Djelloul [2009]	2	0
CaRNAval Reinharz et al. [2018a]	147	10

Table D.5 – Nearly all motifs identified by three published RNA motif tools are a subset of the motifs found by VeRNAl.



Supplementary Figure D.5 – Agreement with existing motif libraries. Each cell value ranges from 0 to 1, where 1 indicates that all the nodes of an instance of a known motif are contained in a VeRNAl motif.



Supplementary Figure D.6 – More example instances of motifs.

Appendix E

InDeep: 3D fully convolutional neural networks to assist in silico drug design on protein-protein interactions - Appendix

Contents

E.1 Data

E.1.1 Data collection and curation

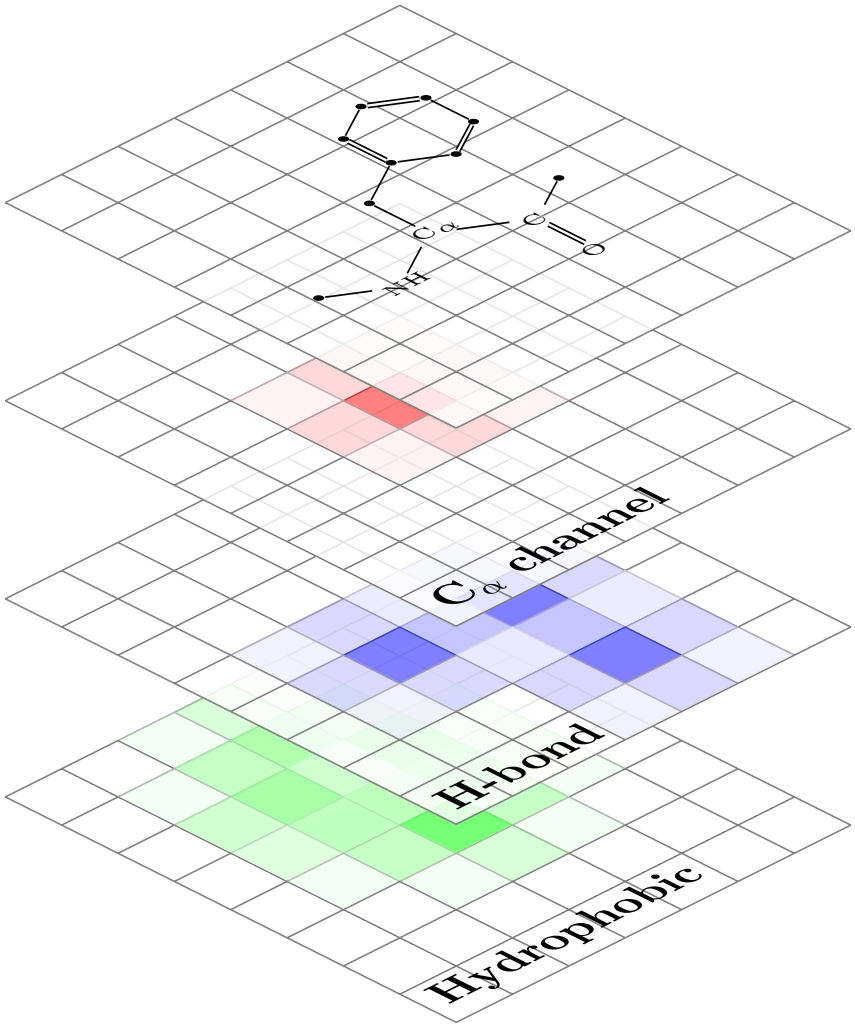
For training and assessing the models, we have used the dataset that is presently accessible on the website of our database iPPI-DB (https://ippidb.pasteur.fr/targetcentric/). These data were retrieved from the PDBe database [Mir et al., 2018] using a filtering procedure and different quality control checks. The procedure operates directly on a JSON file containing all available information within PDBe to select PDB structures accordingly. For Xray structures and CryoEM structures, resolutions should be below 3.5 Å. For Xray structures R_{Free}-R_{Factor} were below 0.07. For CryoEM structures, FSC was below 0.143. Once the selected PDB structures were downloaded, all PL structures were superimposed onto the corresponding HD complexes (sharing an identical UNIPROT ID) in order to identify orthosteric compounds competing for the same interface. Ligands were selected as to be not more than 6 Å away from the interface and contain at least five heavy atoms. Heavy atoms within compounds must be one the following (C, N, O, S, P, I, B, Br, Cl, F). Finally, we removed structures with alternative locations for residues at the interface within HD or PL complexes. Thus, we built a dataset of 2290 PL complexes and 6736 HD complexes for the training and the assessment of the models.

E.1.2 Data splitting

To avoid data leakage, we split our protein structures data based on CATH [Sillitoe et al., 2021] folds. To do so, we have used the pre-computed annotations available for each protein domain and annotated our interaction domains with those CATH classifications. We have then put only specific CATH in the train, validation and train split. We have excluded the following CATH classes from the train split : 1-25-10-10, 1-10-530-10, 1-10-245-10, 1-20-120-560, 1-50-10-20, 2-30-29-30, 2-30-42-10, 2-130-10-10, 2-60-120-620, 2-30-30-50, 3-40-390-10, 3-30-40-10, 3-90-70-10, 3-10-110-10, 3-90-540-10.

E.1.3 Data representation

The volumetric CNN framework consists in treating the interaction sites as 3D images. First we introduce functional atom types : α-carbon (Cα), donor and receptor of h-bonds and positively and negatively charged atoms and hydrophobic/aromatic atoms. At each voxel, we associate a vector that represents the probability of presence of these functional atom types. This vector is computed by considering the input protein as a sum of Gaussian densities centered around each atom in the dimension corresponding to each atom type. We then perform an interpolation of this Gaussian sum on a regular grid, which yields a vector at each grid point (Supplementary Figure E.1). To encode the protein partner, we start with the same encoding and add a 'void' channel so that on each grid point the encoding vector sum to one and represents a probability distribution. To represent iPPIs, we use only one channel for all atom types. This encoding is dependent on the arbitrary basis in which the protein is represented. To limit this arbitrary choice, we applied rotational data augmentation during the training of the network.



Supplementary Figure E.1 – Visual representation of the grid embedding process. The different atom-types are encoded with a Gaussian density around the atom center in atom-types grids. Then these grids are stacked on top of one another.

E.2 Model

We use a U-Net [Ronneberger et al., 2015], with blocks containing 3D convolutions, Batch Normalization [Ioffe and Szegedy, 2015] and ELU activations [Clevert et al., 2015]. In each block, the first convolution goes from `in_channels` to `(in_channels+out_channels)/2` and the second goes to `out_channels`. Following the classical structure of a U-Net model, at each level, we use one such block to decrease the spatial extension of the block by a factor two, while doubling the channels. We start with eight channels and use a depth of four. We use a twist on the classical U-Net, because we branch our network into a `PL` and `HD` one at the last deconvolution block of the decoder. Then each branch gets convoluted one last time and is activated by a sigmoid or a softmax.

E.3 Hyper-parameters optimization (HPO)

To choose the optimal values for the hyperparameters of the network, we conducted an hyper-parameter optimization (HPO). The principle is to choose the values for these hyper-parameters (number of layers, number of neurons per layer) automatically by considering the training as a function evaluation. The function needs to output a scalar value that we want to minimize. Then, many discrete optimization algorithms such as Gaussian processes [Williams and Rasmussen, 2006] can be used to find optimal parameters with regards to the minimization of this metric. We designed a custom metric to take into account several aspects of the prediction, such as the distance between a prediction and its supervision but also their overlap or the size of the prediction. This metric and optimization procedure is described in details below. We then ran the HPO for approximately 100 experiments, minimizing this metric on the validation set. This gave us our final predictive model.

E.3.1 HPO Metric

The HPO metric we chose to optimize need to account for :

- The accurate position of at least one predicted volume
- A good ranking of our correct predicted volume
- A good shape correspondence between the predicted volume and the actual partner

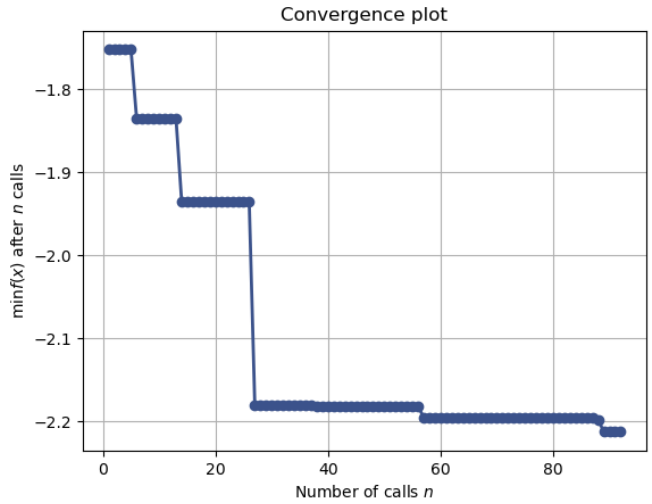
We first compute a distance metric called dist_overlap that is the mean of the distance between the center of mass of the predicted volume and the partner and the overlap between the highest scoring voxels of the predicted volume and the partner. We then pick the predicted volume that yields the best dist_overlap value and compute two additional metrics to encourage giving it a high rank. The first one is simply the exponential of its rank over its surface (to normalize the rank by the expected number of volumes). The second one, deemed as rank_overdist is an expectation-like formula as follows : $\sum_{\text{volume}\in\text{volumes}}(1-\frac{\text{rank(volume)}}{|\text{volumes}|})^2\text{dist\_overlap(volume)}$. We add a DVO score for the shape complementary. The DVO metric is only computed on successful prediction at 6 Å and consists in the volume of the overlap over the volume of the union. Finally, we add a penalty for having too many predicted volumes by subtracting to the metric value the number of predicted volumes divided by the surface.

All of these contributions are then summed and we empirically checked on the validation set that the satisfactory looking predictions had a high metric value while failed ones had a low value. We can then proceed to the automatic optimization of the hyper-parameters with regards to this scalar metric.

E.3.2 Hyper-optimization

We have tweaked the type of model (U-Net vs forward), the type of blocks used by the U-Net, the amount of neurons at the beginning at the network and the depth of the network. Moreover, we have also tuned some post-processing parameters, namely the maximum euclidean distance between two watershed basins and the threshold of the merging value between two neighboring basins.

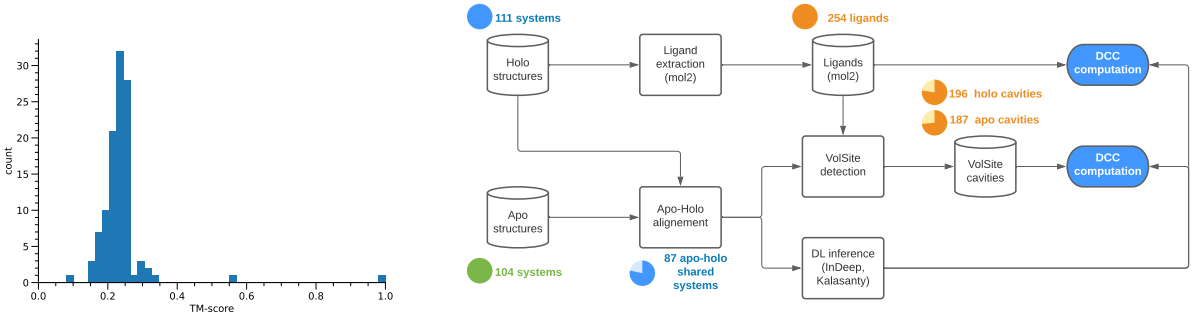
The convergence of this procedure is plotted in Supplementary Figure E.2. We see that the optimization procedure starts by decreasing in the first half of the optimization and then plateaus hinting it has converged.



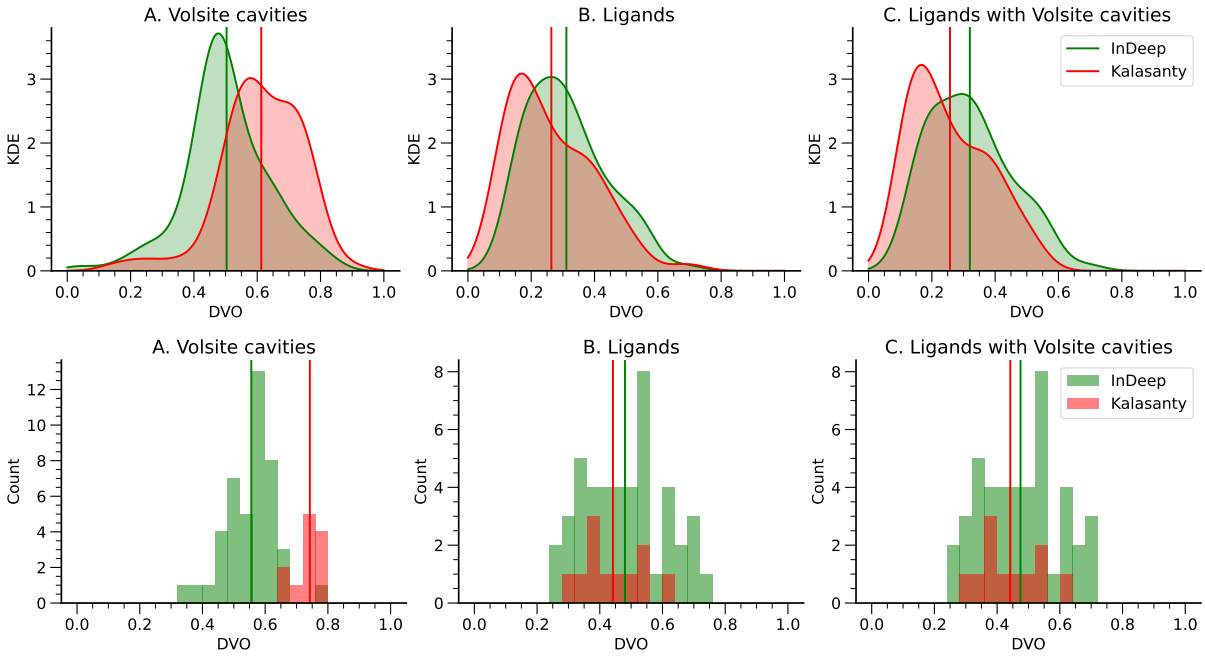
Supplementary Figure E.2 – Successive values of the HPO metric as a function of the HPO epochs

E.4 Supplementary Results

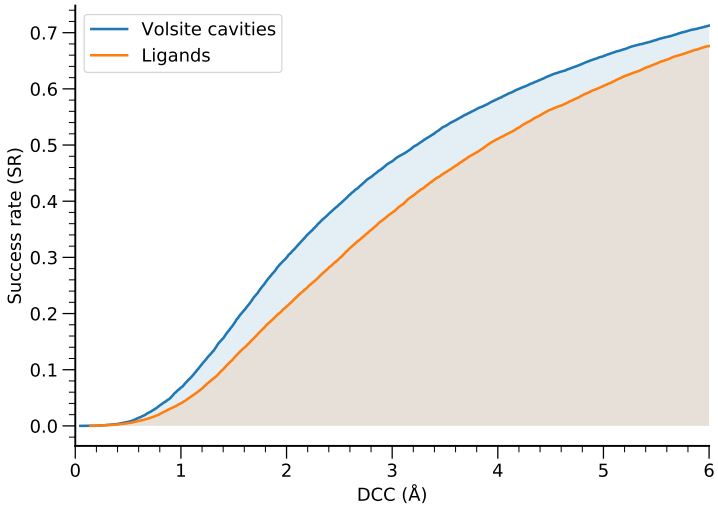
Additional metrics were computed to compare InDeep to Kalasanty on the Chen benchmark (Supplementary Figure E.3), and on the PL datasets (Supplementary Figure E.4). The Success rate metric on the sc-PDB dataset was evaluated (Supplementary Figure E.5) as well as the channels' validation (Supplementary Figure E.6).



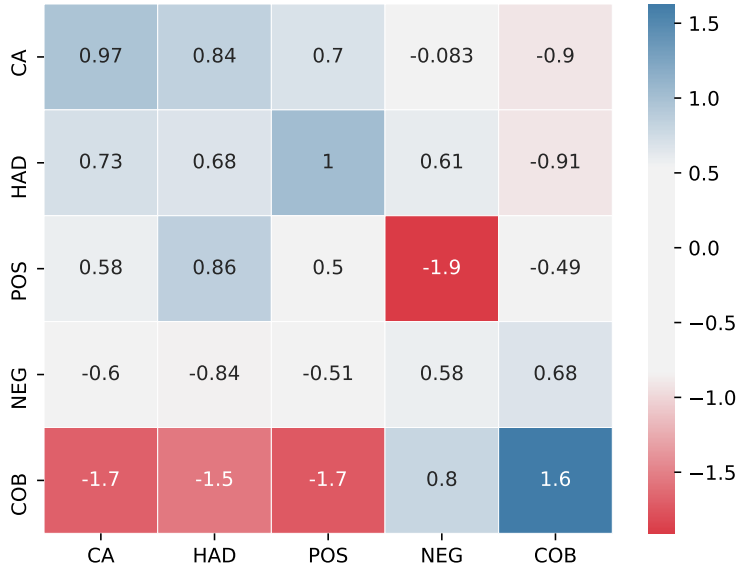
Supplementary Figure E.3 – *Left :* TM-score between the Chen benchmark and the `InDeep` training set. As can be seen, there is only little overlap, showing that most systems are still relevant for our comparisons. Systems with a TM-score higher than 0.5 were discarded for this comparison. *Right :* Overall process of the metric calculations for `PL` to evaluate the performance of our model, in the cav, lig and cavlig settings. We rely on the aligned holo and apo structures to place the ligands for both structures and compute cavities using Volsite around this ligand for each structure. The numbers presented in this diagram pertain to the Chen dataset.



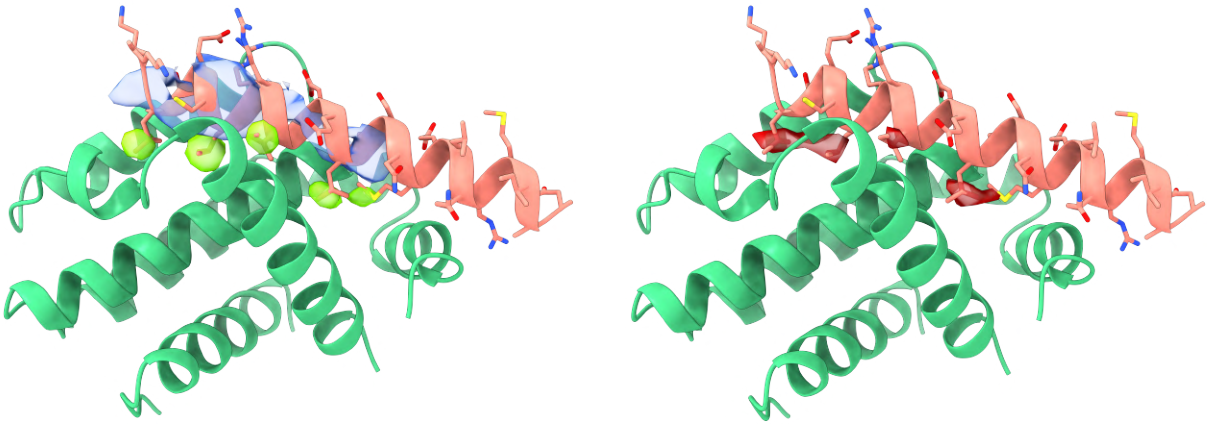
Supplementary Figure E.4 – Distributions of the DVO values for `InDeep` and Kalasanty on the Chen benchmark (*Top*) and on the test set filtered by TM-score (*Bottom*). We only plot the values for the systems for which the method has a successful DCC of less than 6 Å. The metrics are computed with the VolSite cavities associated with the ligand position given by the PDB (**A.**), the ligand position itself (**B.**) and the ligand position having a cavity detected by VolSite (**C.**).



Supplementary Figure E.5 – Success rate as defined in the main text for different DCC thresholds on the scPDB dataset



Supplementary Figure E.6 – Z-scores of the predicted distributions of the probabilities for each channel (one distribution per line). Each line corresponds to a distribution around atoms of the partner that have a certain channel annotation.



Supplementary Figure E.7 – *Top:* The hydrophobic-atom typed channel (green surface), predicted on Bcl-2, (green cartoon) correctly matches with the known hydrophobic hotspots of the protein partner Bax (orange cartoon). Likewise, the Cα-atom typed channel (blue surface) correctly predicts the α-helix shape of the Bax backbone (depicted as a shape close to a cylinder). *Bottom:* `InDeep` ligandability prediction on Bcl-2 (pdb 2xa0). The red surface patches of `InDeep` ligandability are localized around the known hot spots of the Bcl-2/Bax interaction.

E.5 Molecular Dynamics setup

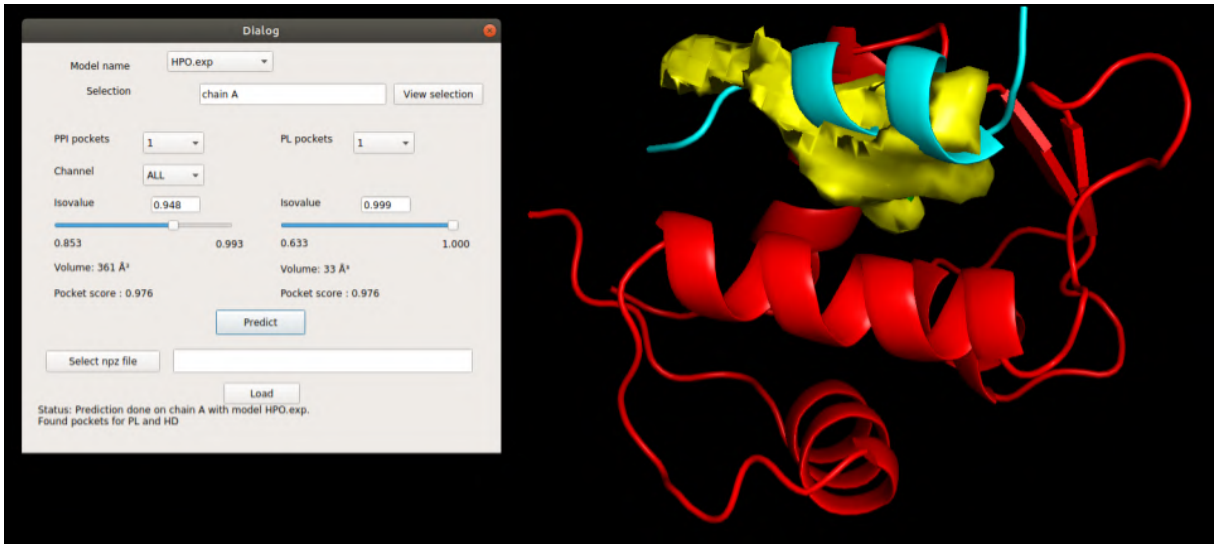
The MD input files were prepared from the unbound form of Bcl-2 (pdb 1gjh) with CHARMM-GUI solution builder server [Lee et al., 2015b] and CHARMM36m force field [Huang et al., 2016]. The protein was placed in a cubic unit cell with a minimum distance of 10 Å to the box edge. The system was solvated with an explicit water model (TIP3P) and KCl ions were added at 0.15 M concentration. Steepest descent minimization was performed in 5000 steps. Subsequently, a 125 ps equilibration in the NVT ensemble was performed at 303.15 K. Finally, production was run for 1µs in the NPT ensemble. GROMACS [Spoel et al., 2005] was used for the simulations. MD frames were saved every 50 picoseconds resulting in 20 000 frames used as `InDeep` input structures.

E.6 Binding site definition for RMSD calculations and InDeep predictions during MD

In order to obtain the list of residues defining the Bcl-2 orthosteric binding site we have retrieved from the dataset 16 PL structures of Bcl-2: 1ysw, 2o21, 2o22, 2o2f, 2w3l, 4aq3, 4ieh, 4lvt, 4lxd, 4man, 6gl8, 6o0k, 6o0l, 6o0m, 6o0o and 6o0p. Then, for each structure the solvent accessible area by residue was measured with naccess [Hubbard et al., 1993] in presence and absence of the orthosteric ligand. Residues having lower solvent accessible area when the ligand is present are considered as part of the binding site. As a result the following residues were used to define the binding site: 96, 99, 100, 103, 104, 107, 108, 111, 112, 115, 133, 136, 137, 144, 145, 146, 148, 149, 152, 153, 156, 198, 202 and 143. Heavy atoms of these residues have been selected to measure the RMSD along the MD compared to the 16 PL structures and to the HD structure (pdb 2xa0). The resulting RMSD curves are shown in Figure 6.5 in the main text.

E.7 Pymol plugin

The tool can be added as a PyMol plugin for an interactive visualisation of the results. A screenshot is shown in Supplementary Figure E.8.



Supplementary Figure E.8 – The results can be visualized through a PyMol plugin, that lets the user see the different pockets, scores and their corresponding volumes at different probability levels.

Appendix F

OptiMol - Appendix

Contents

F.1 Model Architecture and Training

The Variational Autoencoders (VAE), first introduced in [Kingma and Welling, 2013], couples a generator network with an encoder network that performs approximate inference on the data. It relies on the hidden variable hypothesis that postulates that the data distribution can be generated from simpler, common hidden variables that live in a space called the latent space. A prior $p_{model}(\mathbf{z})$, usually chosen as a normal distribution, governs the prior distribution of the latent variables and ensures the smoothness of the latent space.

Given an input $\mathbf{x}$, the encoder computes its encoding $\mathbf{z} = \phi(\mathbf{x})$ as the parameters of a distribution over latent variables $q_\phi(\mathbf{z}|\mathbf{x})$. The generator then draws a sample $\tilde{\mathbf{z}}$ from $q_\phi(\mathbf{z}|\mathbf{x})$ and decodes it through the decoder network to get $\hat{\mathbf{x}} = \theta(\tilde{\mathbf{z}})$. The decoder network represents the function $p_\theta(\mathbf{x}|\mathbf{z})$.

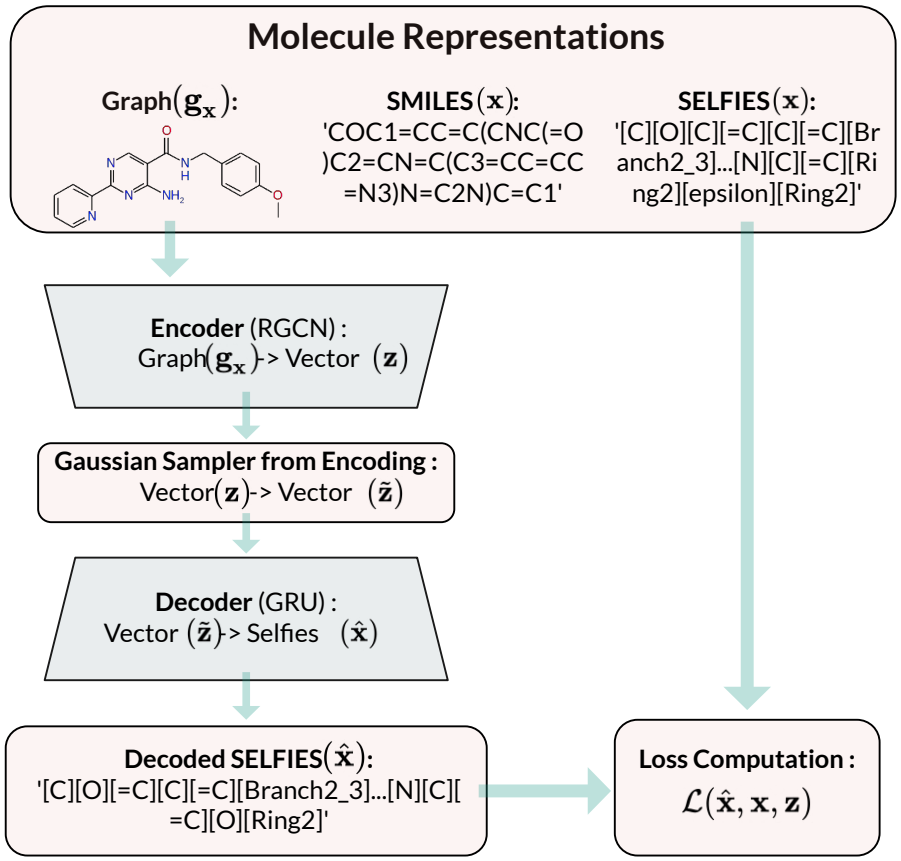
VAEs can be trained via Variational Inference: The encoder network outputs the parameters of the distribution $q_\phi(\mathbf{z}|\mathbf{x})$. Usually, this distribution is chosen as gaussian, hence the encoder outputs a mean and standard deviation that parametrize q. Gradient descent is then used to minimize the objective function (F.1) with respect to the encoder and decoder parameters:

$$L_{\text{VAE}}(\mathbf{x}) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\ p_\theta(\mathbf{x}|\mathbf{z})\right] + D_{KL}(\ p_{model}(\mathbf{z})\ ||\ q_\phi(\mathbf{z}|\mathbf{x})\ ) \tag{F.1}$$

Where the KL-divergence D_{KL} of two distributions is defined as

$$D_{KL}(p||q) = E[\log p(\mathbf{x}) - \log q(\mathbf{x})] \tag{F.2}$$

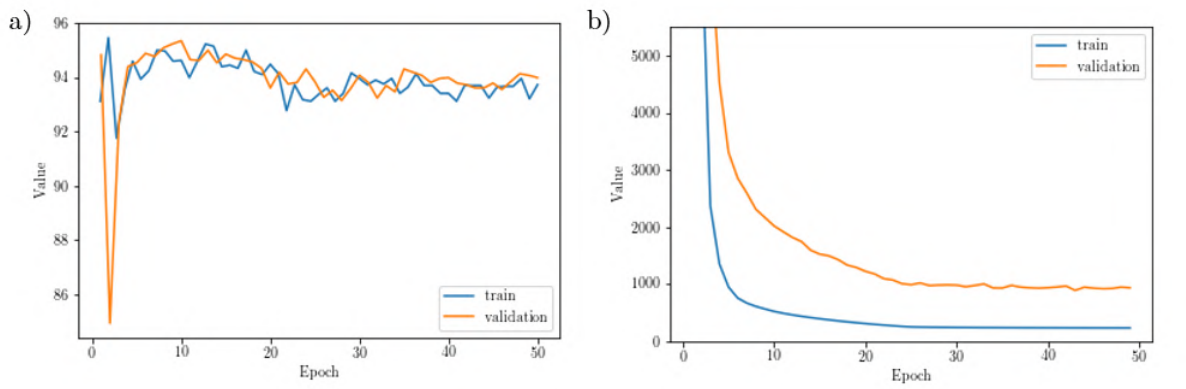
The expectation can be estimated by Monte Carlo sampling, and the KL-divergence between two gaussian distributions has a closed form solution. Hence the VAE approach is easy to implement. In our case, the encoder is an RGCN model that takes in a molecular graph, encodes it in the latent space and decodes it as a one-hot representation of a SELFIES. This pipeline is illustrated in Supplementary Figure F.1.



Supplementary Figure F.1 – Training of the VAE : The molecule gets encoded based on its molecular graph into a vector through an Relational Graph Convolution Network. Then this vector gets decoded into a selfie using a Gated Recurrent Unit Network. Finally a loss gets computed between the decoded SELFIES and the canonical one representing the input molecule. The error is back-propagated through the network.

F.2 Model Implementation and Results

The encoder consists in 3 Relational-GCN layers of hidden size 32, with skip connections, resulting in 96-dimensional embeddings. Two dense layers map to the mean and log standard deviation of the latent embeddings, of dimension 56. The decoder is a 3-layer GRU with hidden states of dimension 450. The model was implemented in PyTorch [Paszke et al., 2019] and DGL [Wang et al., 2019]. The model was trained for 50 epochs using Adam optimizer, a learning rate of 10^{-3} and an exponential decay with rate 0.9 every 40k steps. Batch size was set to 64. For the first 40k steps, we train the model only on reconstruction loss, and then progressively increase the weight of the KL term by 0.02 every 2k steps, until it reaches 0.5



Supplementary Figure F.2 – Fraction of correctly reconstructed characters (a) and KL divergence term (b) during model training, for training and validation set

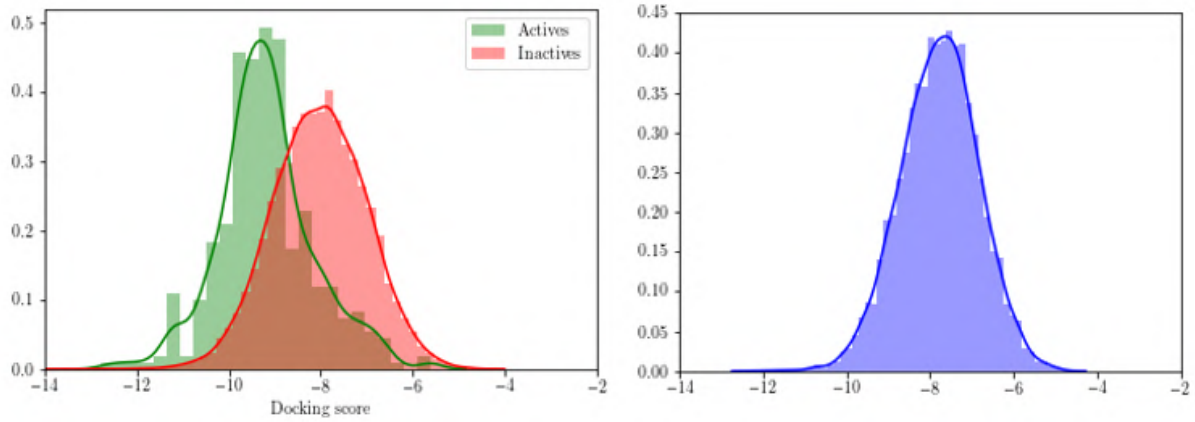
Model	Valid	Unique 1k	Unique 10k	IntDiv	IntDiv2	Filters	Novelty
HMM	0.0760	0.6230	0.5671	0.8466	0.8104	0.9024	**0.9994**
NGram	0.2376	0.9740	0.9217	0.8738	0.8644	0.9582	0.9694
Combinatorial	0.9979	0.9983	0.9948	**0.8812**	**0.8741**	0.7912	0.9913
CharRNN	0.9748	**1.0000**	**1.0000**	0.8562	0.8503	0.9943	0.8419
AAE	0.9368	**1.0000**	0.9973	0.8557	0.8499	0.9960	0.7931
VAE	0.9767	**1.0000**	0.9984	0.8558	0.8498	**0.9970**	0.6949
JTN-VAE	**1.0000**	**1.0000**	0.9992	0.8512	0.8453	0.9778	0.9153
LatentGAN	0.8970	**1.0000**	0.9970	0.8565	0.8504	0.9727	0.9488
graph2selfies	**1.0000**	**1.0000**	0.9998	0.8560	0.8496	0.9557	0.9097

Table F.1 – Moses metrics for all models benchmarked in Moses [Polykovskiy et al., 2018] and graph2selfies

F.3 Docking Scores Distributions

We set the exhaustiveness of the conformation space search to 16, as a reasonable compromise between running time and enrichment factor. We see that the distribution of actives compound

scores is well shifted from the inactives one with a mean around -9.5 kcal/mol.



Supplementary Figure F.3 – Distribution of docking scores of active molecules vs ExCAPE inactives *(left)* and random compounds from the ZINC training set used to train the prior *(right)*

F.4 Query-efficient Optimization

F.4.1 Bayesian Optimization

Bayesian optimization uses Gaussian processes as a surrogate model for black box functions that are costly to evaluate. It enables optimizing the objective in a query-efficient way, by sampling points that maximize expected improvement. However, it has inherent limitations for the lead generation problem we attempt to solve. To take samples, some kind of rigid (not learnt nor adaptive) sampling is generally used, meaning that the expected improvement under a Gaussian process model is computed over each point of a grid of a certain resolution.

This computation of the expected improvement over the samples space does not scale well to a high number of dimensions for a large batch size as the grid evaluation becomes intractable (This amounts to finding an estimate on all molecules and only picking the most promising candidates for docking). In addition, BO was shown to perform better when the latent space is shaped by the objective function [Gómez-Bombarelli et al., 2018; Griffiths and Hernández-Lobato, 2017]. This is not the case for binding affinities, since the chemical space is likely to exhibit activity cliffs and scattered activity peaks. The final goal is to be able to take samples from the model, and this method uses rejection sampling to generate favorable samples. This choice limits scalability when sampling tens of thousands of compounds.

Bayesian optimization was implemented using BoTorch [Balandat et al., 2019]. A Gaussian process was trained to predict the objective on 500 initial samples selected to be maximally diverse in the Moses training set. The Gaussian process was then trained for 20 steps by sampling a batch of 50 compounds using Expected Improvement as the acquisition function. For the optimization of QSAR activity scores, the 500 initial samples were selected as maximally diverse from the union of Moses and the QSAR train actives.

F.4.2 CbAS

To address the limitations of Bayesian Optimization, we turn to a recently published method : Conditioning by Adaptive Sampling (CbAS) [Brookes et al., 2019]. This method trains a generative model that also seeks to maximize an objective function. This method uses a prior generative model and shifts its distribution to maximize an expectation. Queries are used in an efficient way thanks to an importance sampling scheme coupled with reachable objectives for the model. The alternating phases of tuning and sampling also enable a more efficient implementation.

CbAS starts with a prior generative model with parameters $\boldsymbol{\theta}^{(0)}$, $p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$ and the optimization is formulated by conditioning this probability on the random variable $\boldsymbol{S}, p(\mathbf{x}|\boldsymbol{S}, \boldsymbol{\theta}^{(0)})$. This random variable represents the values for a probabilistic (or noisy) oracle : $\boldsymbol{S} = (\mathbf{f}(\mathbf{x}) > \gamma)|\mathbf{x}$. We now see that this conditioned probability model is a distribution that maximizes the function $\mathbf{f}$ when γ goes to the maximum value of the function. However this conditional probability is intractable and the authors propose to use variational inference to approximate it. The parametric family is a generative model with parameters $\boldsymbol{\phi}$, $q(\mathbf{x}|\boldsymbol{\phi})$ that solves :

$$\begin{aligned}\boldsymbol{\phi}^* &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\, D_{KL}\left(p(\mathbf{x}|\boldsymbol{S}, \boldsymbol{\theta}^{(0)})||q(\mathbf{x}|\boldsymbol{\phi})\right) \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}\left[p(\boldsymbol{S}|\mathbf{x})\log(q(\mathbf{x}|\boldsymbol{\phi})\right]\end{aligned}$$

The authors then use importance sampling instead of always sampling from the prior to estimate this expectation. The proposal distributions are the successive generative models obtained at each iteration. The last key idea is to use a fixed quantile of the successive generative models distributions as a value for $\gamma^{(t)}$, to set reachable objectives for the model : $\boldsymbol{S}^{(t)} = \mathbf{f}(\mathbf{x}) > \gamma^{(t)}$. A detailed derivation can be found in the original paper and result in solving for $\boldsymbol{\phi}^{(t)}$ for each t in the following equation :

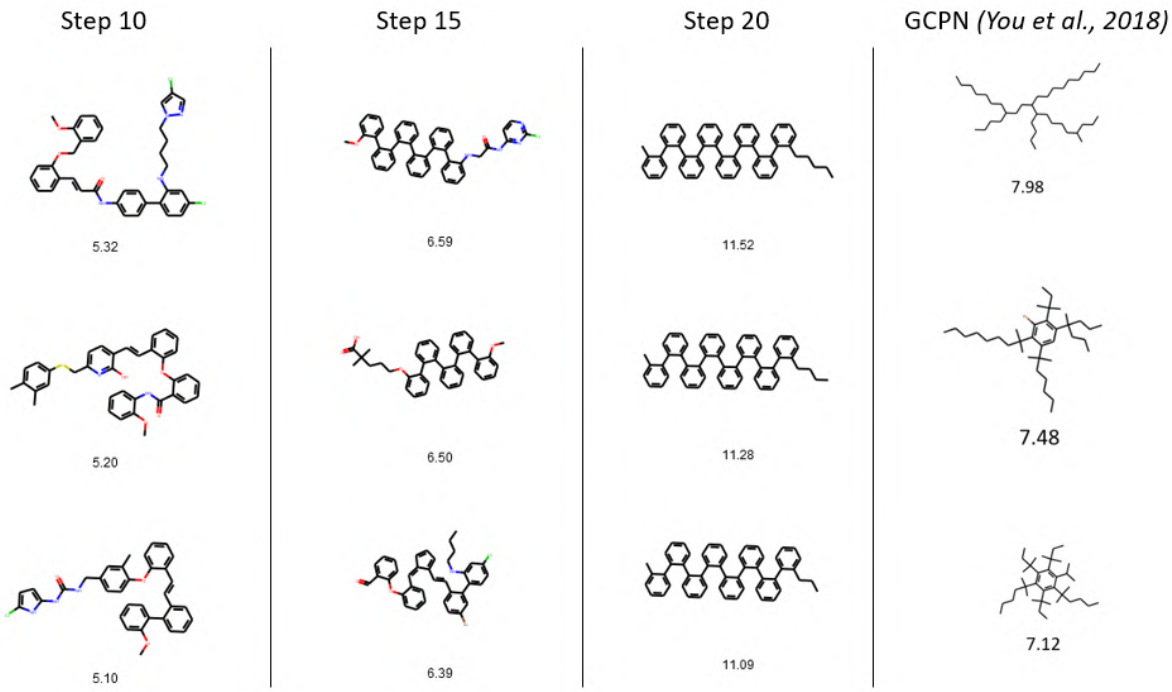
$$\boldsymbol{\phi}^{(t+1)} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, \mathbb{E}_{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})}\left[\frac{p(\mathbf{x}|\theta^{(0)})}{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})} p(\boldsymbol{S}^{(t)}|\mathbf{x})\log(q(\mathbf{x}|\boldsymbol{\phi})\right]$$

As a prior model $p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$, one can either use a model trained on a broad chemical space, or leverage previously discovered actives to narrow-down the chemical space by fine tuning the prior on the actives. Then, we take samples from the search model (initialized with the prior), use an oracle with a noise model to get $p(\boldsymbol{S}^{(t)}|\mathbf{x})$ and fine tune the search model with the adequately re-weighted samples.

F.5 Additional OptiMol Results

F.5.1 clogP

The top 3 molecules found by OptiMol after 12, 16 and 20 steps are shown in Figure F.4. Like in [You et al., 2018], we note that the best molecules look significantly different from the ZINC clean leads of the 250k dataset. This is due to the nature of the composite logP objective, and more reasonable molecules, though lower-scoring, can be obtained using early-stopping OptiMol.



Supplementary Figure F.4 – *(left)* Top 3 molecules found by OptiMol after 12, 15 and 20 steps *(right)* Top 3 molecules reported by [You et al., 2018]

F.5.2 QSAR validation

QSAR model details

The Excape database [Sun et al., 2017a] contains 3482 active molecules for human Dopamine Receptor 3, and more than 300k inactives. We follow the procedure in [Olivecrona et al., 2017] to split the actives into 5 folds: First, we cluster the actives using rdkit's Morgan fingeprints ($r = 3$) and the Butina algorithm with a similarity cutoff of 0.4. We then sort the clusters by size and iteratively assign them to each of the 5 folds. We randomly sample 100k inactives and assign them in the same proportions, to obtain 5 folds of equal size.

We then use the same model architecture as Skalic et al. [2019b]: LightGBM gradient boosting decision tree algorithm [Ke et al., 2017] (1000 trees, 0.8 colsample-bytree and keeping default values of other parameters) was applied to train a QSAR classification model for DRD3. The model was evaluated on 5-fold cross-validation, and obtained an average f1-score of 0.89 (sd = 0.014).

F.5.3 QSAR results

We note that DRD3 is part of the training set of Skalic et al. [2019b], hence there is an inherent bias towards the generation of compounds that share a similar shape with known DRD3 binders. The performance obtained in this experiment is therefore an upper bound of their performance, but still significantly outperforms `OptiMol` in the first decile. The upper deciles of the ranked list contain significantly more samples from `OptiMol` than decoys, showing they ability of OptiMol to design compounds that score well under QSAR, without ever leveraging known actives.

Model Optimization

We have tried tuning several parameters and found the training dynamics quite subtle, as too much training increased the scores but crashed the diversity and too little did not optimize the objective functions. We have tuned the number of samples at each epochs, the number of epochs, the usage of teacher forcing, different noise model for the oracle, the number of epochs, the optimizer used (Adam or SGD), the scheduler used, the initial learning rate, the impact of the quantile picked as well as using a non-decreasing γ. We found that that the best combination was reached using 30 epochs of 1000 samples, teacher forcing, gaussian noise of variance of the same magnitude as the variance of the data, Adam optimizer with no scheduler and default learning rate, 6th quantile and a non-decreasing γ.

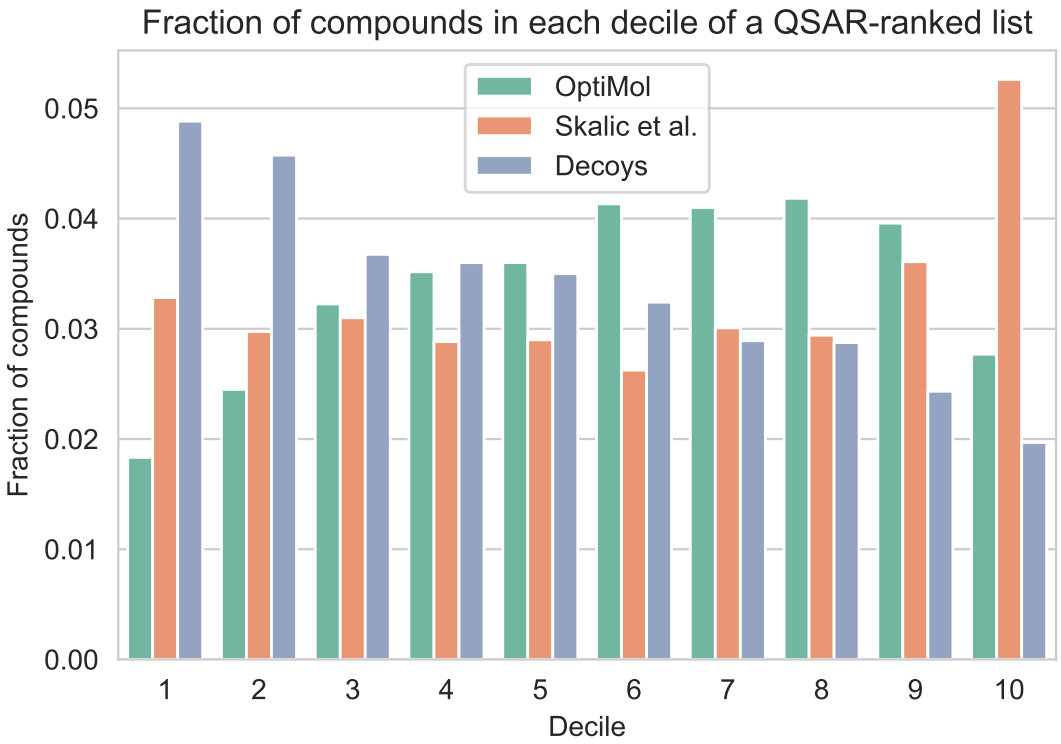
Model robustness

Multi-objective training

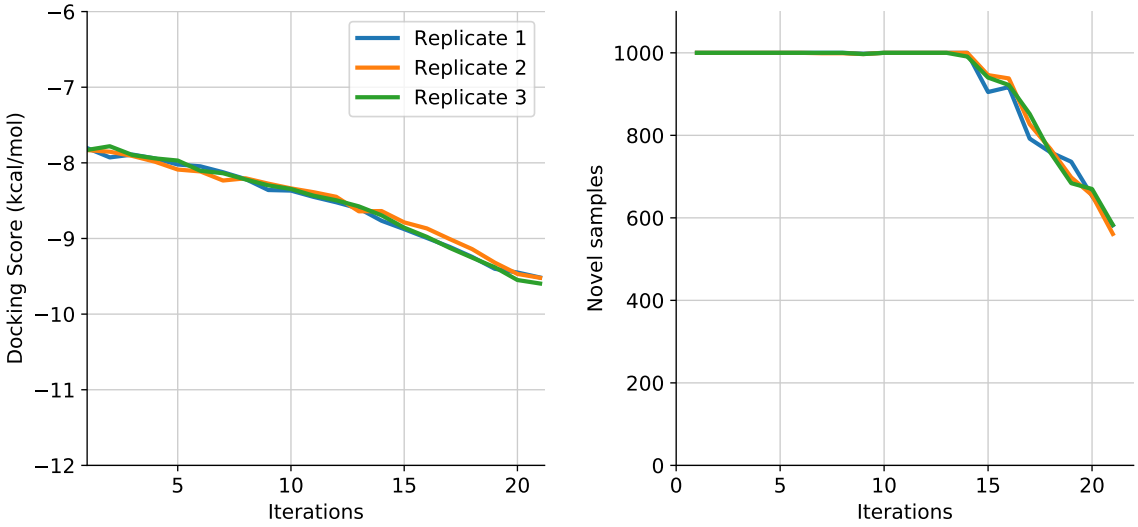
As in the main text, the dynamics of learning for the multi-objective model are in Figure F.7.

Sampling quality assessment

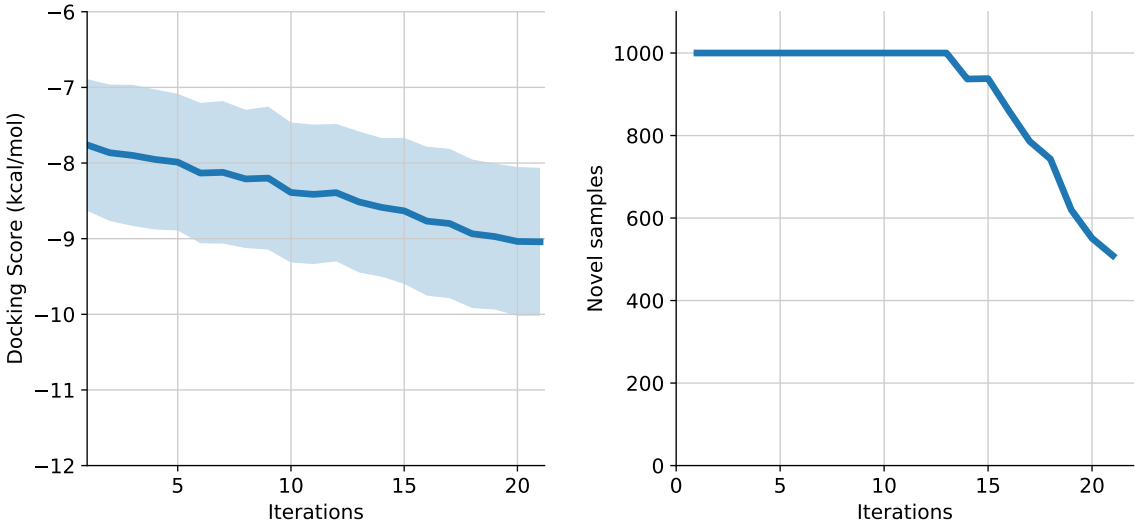
Figure F.8 shows the distributions of docking scores for the first new samples from the generative model and the last ones, when taking 100k random samples. As stated in the main text, the difference in distributions is statistically significant (p-value of 10^{-14}) but small compared to the



Supplementary Figure F.5 – Distribution of samples from Skalic et al., `OptiMol` and decoys when ranked by QSAR score. The results are split by scores deciles (higher is better) and in each decile, we report the fraction of compounds originating from each population. The list contains 12k compounds, equally split between samples from Skalic et al., `OptiMol` samples and decoys from ZINC (filtered as in Skalic et al. [2019b]).

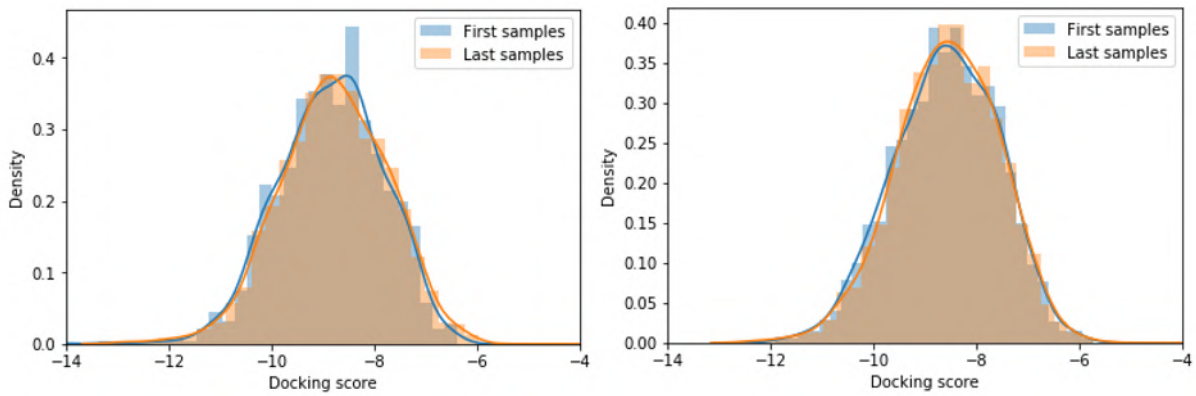


Supplementary Figure F.6 – Trajectories of the mean result obtained on three independent runs of `OptiMol`.

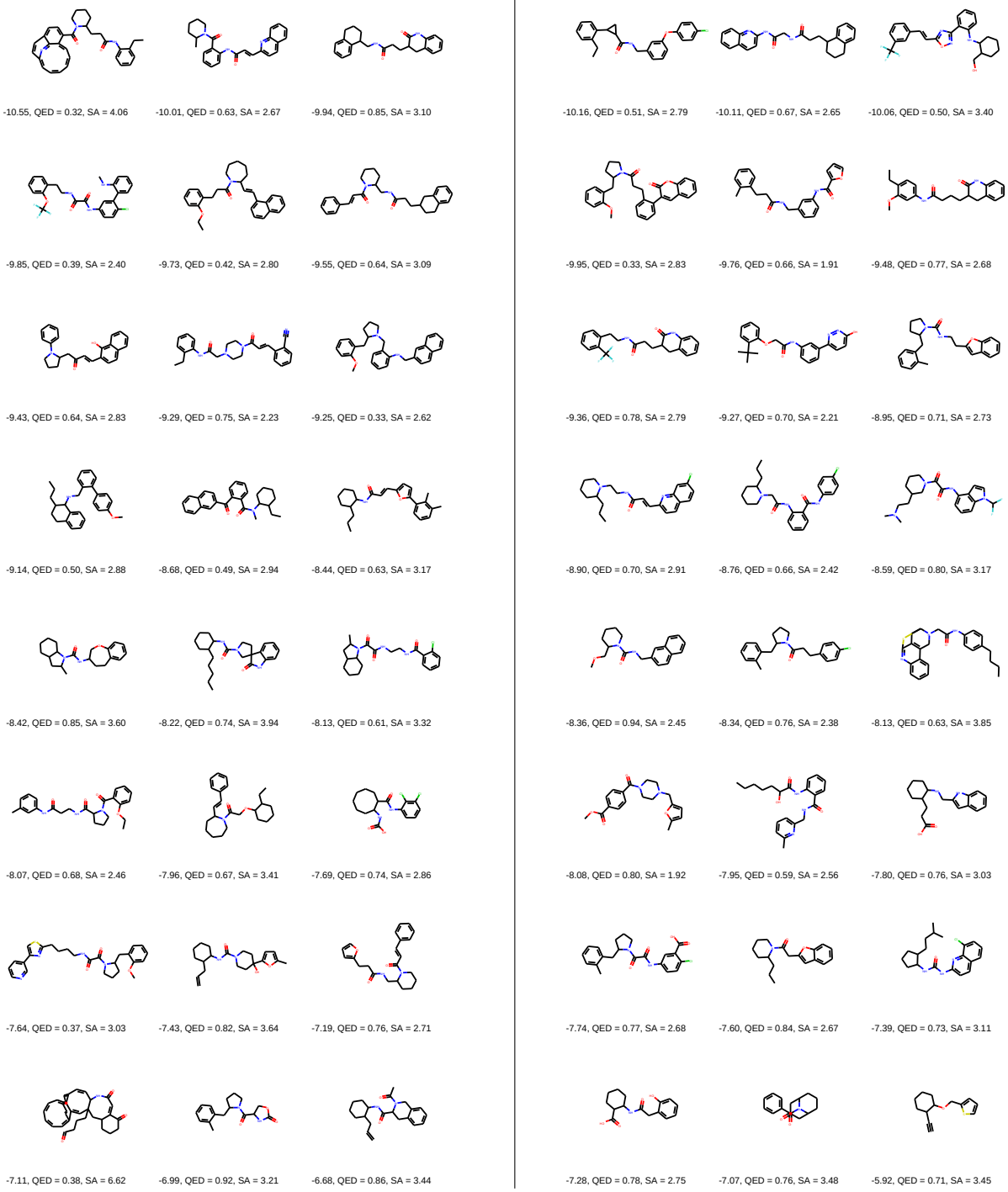


Supplementary Figure F.7 – *Left*: The average and standard deviation of the docking scores obtained at each iteration. *Right*: The number of compounds never sampled before at each iteration.

difference with uniform sampling over ZINC. This shows the OptiMol generative models have learned to sample uniformly a large chemical space with enhanced docking scores.



Supplementary Figure F.8 – Distribution of docking scores of the 2500 first molecules sampled by the generative model and the 2500 last molecules sampled, for OptiMol *(left)* and OptiMol-multiobjective *(right)*



Supplementary Figure F.9 – 24 random samples from Optimol *(left)* and OptiMol-multiobjective *(right)*, sorted by docking scores (lower is best).