Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

De Novo Design of Protein-Protein Interactions Modulators

Autor: Eduardo González García
Tutores: Antonio Jiménez Martín
Nuria E. Campillo Martín

Madrid, 7-2024

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: De Novo Design of Protein-Protein Interactions Modulators

7-2024

Autor(a): Eduardo González García
Tutores: Antonio Jiménez Martín
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Nuria E. Campillo Martín
Centro de Investigaciones Biológicas Margarita Salas

Acknowledgments

Me gustaría agradecer a mi tutora, Nuria E. Campillo, por darme la oportunidad de trabajar en este interesante proyecto. Agradezco también al doctor Roi Naveiro por su ayuda en la corrección de la memoria y a la doctora Eugenia Ulzurrun por su labor en los cálculos de docking empleados en este trabajo.

Muchas gracias a mi familia por el apoyo incondicional en todo lo que hago.

Por último, quiero expresar mi agradecimiento a mis compañeros matemáticos, con quienes he vivido intensos días de biblioteca mientras ellos sufrían sus exámenes finales. Gracias por vuestra compañía durante todo este tiempo.

Resumen

El proceso de desarrollo de fármacos es intrínsicamente largo y costoso, a menudo dura más de una década y requiere una sustancial inversión económica. Los modelos de aprendizaje profundo ofrecen una solución prometedora al mejorar tanto la exploración del espacio químico con modelos generativos como la identificación de potenciales fármacos con predictores de propiedad. Este trabajo se centra en la generación de moduladores de interacciones proteína-proteína (PPIs), que son importantes dianas terapéuticas debido a su papel central en numerosos procesos biológicos y su potencial en el tratamiento de diversas enfermedades.

Presentamos un marco integral para la generación *de novo* de moduladores de PPIs, estructurado en torno a tres pilares principales. En primer lugar, empleamos un conjunto diverso de generadores moleculares del estado del arte para explorar eficazmente el espacio químico, garantizando una amplia gama de diversidad molecular. En segundo lugar, adaptamos la investigación existente para desarrollar un nuevo regresor de la bioactividad de PPIs, diseñado para predecir la afinidad de las moléculas con PPIs especificados por un usuario. Este regresor guía el proceso generativo hacia compuestos con alto potencial terapéutico. En tercer lugar, incorporamos varios filtros de propiedades auxiliares, centrados en la predicción de la toxicidad, la capacidad de penetración en la barrera hematoencefálica, el parecido a ser un fármaco, la accesibilidad sintética, la similitud molecular a compuestos conocidos y la afinidad de unión a bolsillos proteicos específicos. Estos filtros son esenciales para eliminar compuestos de baja calidad y refinar la selección de candidatos viables.

Este marco se aplica para generar moduladores de dos PPIs específicas, NCS-1/Ric-8A y NCS-1/D2R. Los compuestos resultantes se compilan en dos bibliotecas químicas virtuales, que actualmente se están sometiendo a más pruebas para identificar moduladores nuevos y eficaces. Este trabajo destaca el potencial del aprendizaje profundo en el descubrimiento de fármacos y propone una metodología robusta para desarrollar moduladores dirigidos a PPIs.

Abstract

The process of drug development is inherently lengthy and costly, often taking over a decade and requiring substantial financial investment. Deep learning models offer a promising solution by enhancing both the exploration of the chemical space with generative models and the identification of potential drug candidates with property predictors. This master final project (MFP) focuses on generating modulators of protein-protein interactions (PPIs), which are important therapeutic targets due to their central role in numerous biological processes and their potential in treating various diseases.

We present a comprehensive framework for the *de novo* generation of PPI modulators, structured around three main pillars. First, we employ a diverse set of state-of-the-art molecular generators to effectively explore the chemical space, ensuring a wide range of molecular diversity. Second, we adapt existing research to develop a novel PPI bioactivity regressor, designed to predict the affinity of molecules to specific PPIs. This regressor guides the generative process toward compounds with high therapeutic potential. Third, we incorporate various auxiliary property filters, focusing on predicting toxicity, blood-brain barrier penetration capability, drug-likeness, synthetic accessibility, molecular similarity to known compounds, and the binding affinity to specific protein pockets. These filters are essential for eliminating low-quality compounds and refining the selection of viable candidates.

This framework is applied to generate modulators for two specific PPIs, NCS-1/Ric-8A and NCS-1/D2R. The resulting compounds are compiled into two virtual chemical libraries, which are currently undergoing furhter testing to identify new and effective modulators. This MFP highlights the potential of deep learning in drug discovery and proposes a robust methodology for developing targeted PPI modulators.

Contents

Chapter 1

Introduction

1.1 Motivation

Drug development is an inherently lengthy and costly process, often requiring over a decade and millions of dollars to bring a single new drug to market [29]. This complex process involves multiple stages, from initial discovery through preclinical and clinical trials, each with significant financial and temporal investments. One of the most challenging aspects of drug development is identifying and developing promising molecular candidates. The chemical space of feasible compounds is vast, encompassing an estimated 10^{33} potential molecules [112]. This immense diversity makes it exceptionally difficult to pinpoint molecules with the desired properties and therapeutic potential.

Artificial Intelligence *de novo* drug design can significantly accelerate and streamline the drug development process in two key ways:

1. **Property prediction:** AI models can predict the properties of molecules with high accuracy, enabling the identification of candidates that possess desired biological and chemical characteristics. These property predictors can rapidly evaluate vast libraries of compounds, filtering out those with suboptimal properties and highlighting promising candidates for further development.
2. **Compound generation:** Deep learning generative models have the capability to create novel molecular structures, exploring unknown regions of the chemical space. These models can generate new molecular candidates that may not be immediately apparent through traditional methods, thus expanding the scope of drug compounds with therapeutic potential.

Specifically, we are interested in designing small molecules capable of modulating protein-protein interactions (PPIs). PPIs play a crucial role in a wide array of biological processes, governing cellular functions such as signal transduction, immune responses, and metabolic regulation. Given their central role in numerous biological systems, PPIs represent very promising therapeutic targets [147]. Targeting PPIs can potentially modulate or disrupt pathological processes, offering novel approaches for treating diseases ranging from cancer to infectious diseases and neurological disorders [93].

Despite their therapeutic potential, designing small molecules capable of effectively

modulating PPIs has proven to be a significant challenge. Unlike traditional drug targets, PPIs often involve large, flat, and featureless interaction surfaces with limited well-defined pockets, making it difficult for small molecules to bind with high specificity and affinity [132]. These structural characteristics pose a considerable problem in the identification and optimization of PPI modulators, as conventional drug design strategies are typically less effective in this context. Hence why, recent advancements in AI *de novo* molecular design offer new tools to overcome these challenges.

1.2 Objectives

The goal of this master final project (MFP) will be to build a general PPI modulator generation framework, leveraging both PPI and compound data to design drugs for any specified PPI in a low data regime. To achieve this goal, we have identified the following subobjectives:

- Review the current state-of-the-art in molecular property prediction and generation.
- Adapt existing work from the literature to develop a general PPI modulator bioactivity predictor.
- Define auxiliary screening filters for our properties of interest.
- Integrate molecular generators with our bioactivity predictor and screening filters.

Additionally, this MFP will be applied in a case study focusing on generating modulators of two PPIs, NCS-1/Ric-8A and NCS-1/D2R.

1.3 Structure

This MFP is structured as follows:

- Chapter 2 briefly reviews how compounds are computationally represented and outlines the general methodology employed in this MFP for *de novo* drug design.
- Chapter 3 provides an in-depth description of the state-of-the-art in predicting our properties of interest, as well as how models from the literature are adapted to build our screening filters for *de novo* design of PPI-modulators.
- Chapter 4 describes a comprehensive classification of the state-of-the-art molecular generative models, along with how models from the literature are combined with our filters to create new PPI-modulators.
- Chapter 5 presents a case study of the application of this MFP for the generation of modulators of NCS-1/Ric-8A and NCS-1/D2R.
- Chapter 6 concludes with a summary of the achieved results, the limitations of this MFP, and a brief review of future research directions.

Chapter 2

Design methodology

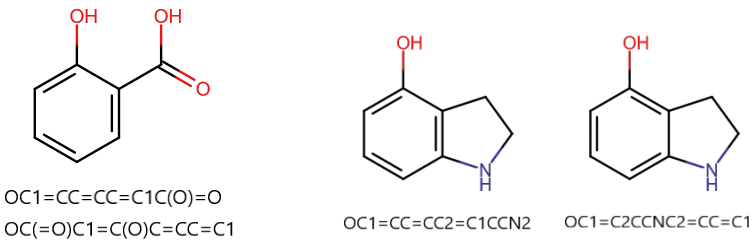
In this chapter, we will cover the overarching methodology used for generating PPI modulators. We will start by discussing how our data will be represented.

2.1 Molecular and Protein Representations

Molecules can be represented in various ways. In this thesis, we will use the three main types of representations: text-based, graph-based, and fingerprint-based.

2.1.1 Text-based Molecular Representations

The predominant molecular representation through text is the SMILES (Simplified Molecular-Input Line-Entry System) encoding [153], which describes molecules using short ASCII strings. SMILES is a compact data format capable of representing complex molecules with a few characters, making it the most commonly used encoding for storing molecular data in large databases. However, SMILES has three substantial limitations. First, as shown in Figure 1a, the same molecule can have different SMILES representations, which may hinder a model's learning capabilities. Second, the SMILES encoding does not properly characterize similarities, as two similar molecules may have very different SMILES representations (as seen in Figure 1b). Finally, a large fraction of SMILES strings do not correspond to valid molecules, which is a significant limitation when using deep generative models employing SMILES encoding.



(a) (b)

Figure 1: (a) A molecule with two different SMILES representations. (b) Two similar molecules with very different SMILES representations.

To address the issue of invalid molecules, other text-based representations have been developed. Such as SELFIES [73], a sequence-based representation of semantically constrained graphs in which every string corresponds to a valid molecule.

2.1.2 Graph-based Molecular Representations

A molecular graph G is defined as a tuple of tensors $G = (V, E)$, where nodes $v_i \in V$ represent atoms (excluding hydrogen, which can be inferred by chemical rules) and edges $(v_i, v_j) \in E$ represent bonds between atoms. Nodes and edges are labeled based on the type of atom and bond, respectively. Additionally, feature vectors can be included for both atoms and bonds to increase the information density of the representation. Representing molecules as graphs has the advantage of directly reflecting the chemical structure in the data format.

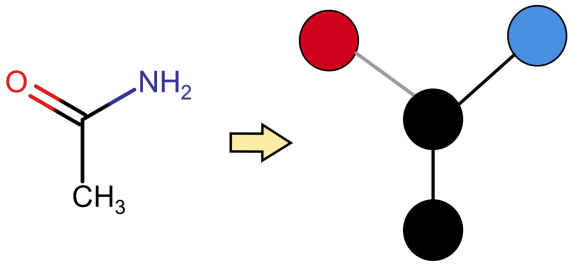


Figure 2: Example of a molecular graph with colors indicating different node and edge features.

2.1.3 Fingerprint-based Molecular Representations

Fingerprint-based representations describe the presence (or absence) of substructures within molecules. This encoding can be derived by matching substructures against an expert-defined set or through the algorithmic enumeration and hashing of substructures in a molecule. One of the most commonly used fingerprint representations are Morgan fingerprints [118], which encode the atom groups of a molecule into a binary vector. This encoding is parameterized by the length of the vector and a radius that defines the size of the atom groups.

2.1.4 Protein Representations

Proteins are large and complex macromolecules formed by chains of amino acid residues. There are 20 different amino acids that combine in different ways to create a protein. A common method to uniquely describe a protein is through a string sequence, where each character represents an amino acid. Additionally, information on how the amino acid sequence folds into a 3D structure is important, which is why proteins are also represented in a 3D space with coordinates for each amino acid residue. This data is stored in the Protein Data Bank (PDB) [9]. The structural information can be implemented in different file types, with the pdb data format being one of the most commonly used.

2.2 Properties of Interest

In the development of drugs, different biological, toxicological and physicochemical properties must be taken into account. Some of them are key to getting the drug to the clinic. In this MFP, we have taken into account the following properties:

1. **Biological activity:** the capacity of a molecule to modulate a specific biological target. Different metrics are used to quantify this activity, like the half-maximal inhibitory concentration (IC_{50}), dissociation constant (*Kd*), inhibition constant (*Ki*)...
2. **Toxicity:** the harmfulness of a molecule towards an organism. Evaluating toxicity involves various assays and studies, including in vitro cell-based tests or in vivo animal studies.
3. **Blood-brain barrier penetration:** the capacity of a molecule to penetrate through the blood-brain barrier and reach its target destination.
4. **Drug-likeness:** a qualitative property of a compound that determines its potential to become an effective drug.
5. **Synthetic accessibility:** how difficult is to synthesize a particular compound.
6. **Molecular similarity:** how similar are generated molecules to known compounds and to each other.
7. **Binding affinity:** the capacity of a molecule to bind to specific protein pockets.

2.3 Pipeline for *de Novo* Design

The general pipeline for the *de novo* design of PPI modulators used in this MFP follows a funnel shape. As described in Figure 3, the pipeline comprises five steps: first, tens of thousands of candidates are created using generative deep learning models; second, these molecules are filtered based on predictions of their biological activity, toxicity, drug-likeness, synthetic accessibility, and similarity to known modulators; third, software for automatic docking is employed to calculate binding affinities and rank the best candidates; fourth, a classical molecular docking study is performed on the most promising molecules; and finally, the best molecules are handpicked and sent for synthesis in the laboratory.

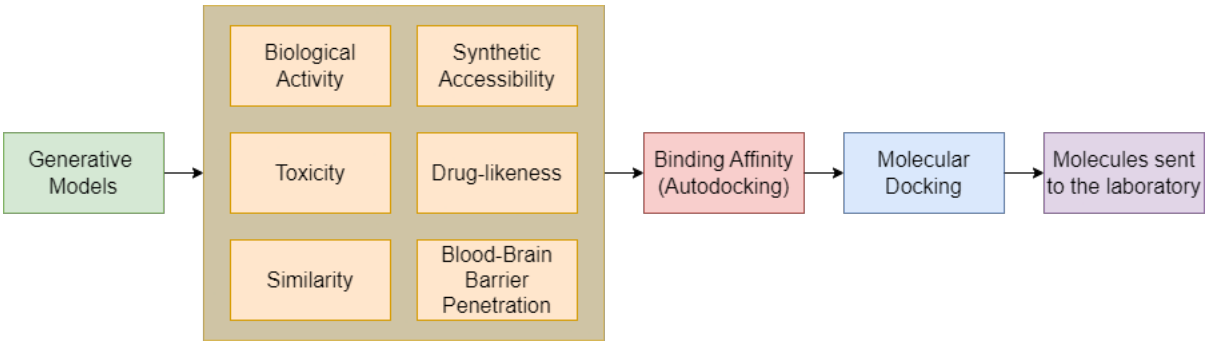


Figure 3: Pipeline for *de novo* design.

Chapter 3

Screening Filters

In this chapter, we will begin covering how our properties of interest are calculated. For bioactivity, toxicity and blood-brain barrier penetration (BBBP), predictions will be made using machine learning models, while the remaining properties will be directly inferred from predefined functions. Then, we will define the filters imposed on each property for newly generated molecules.

3.1 Previous Works in Bioactivity Prediction

In the literature, the bioactivity prediction task is typically framed either as a classification problem, where compounds are categorized as active or non-active, or as a regression one, where experimental measurements (such as IC_{50}, *Kd*, *Ki*...) are predicted, usually on a logarithmic scale.

Currently, bioactivity prediction of protein-protein interactions (PPI) modulators revolves around the use of molecular fingerprint representations combined with molecular descriptors to train machine learning models. These models exclusively take molecular data as input and thus, are limited to predicting bioactivity either for specific PPI families or for general PPI modulators (i.e. independent of PPI family) without protein information. For example, SMMPPI [46] adopts this methodology to build several random forest classifiers capable of predicting the bioactivity of both general PPI modulators and modulators within 11 specific PPI families. Similarly, pdCSM-PPI [117] utilizes graph-based signatures to create feature vectors and train ExtraTrees models for regression and classification. Meanwhile, SELPPI [33] explores the use of a stacking strategy with a genetic algorithm to identify the optimal combination of machine learning models and molecular descriptor pairs for the regression and classification of bioactivity within specific PPI families.

These models share a common limitation: they typically require a dataset of hundreds of molecules for a useful prediction within a given PPI family. However, bioactivity data is often scarce, and many PPI families lack datasets of this size. While predicting the bioactivity of general PPI modulators can partially address this issue, the absence of protein data in the input significantly compromises the quality and specificity of predictions for novel PPI targets. To address this limitation, Sun et al. developed MultiPPIMI [137], a deep learning framework that classifies bioactivity between any two proteins, forming a PPI; and a modulator. This approach offers a key advantage:

it can extrapolate knowledge gained from interactions of modulators with one PPI family to another. Consequently, the abundance of data from one PPI family may help alleviate the scarcity of data for others.

3.1.1 MultiPPIMI

The MultiPPIMI framework [137] comprises four modules trained end-to-end with cross-entropy loss: a molecule feature extractor, which processes a molecular graph and generates an embedding; a PPI feature extractor, which accepts two chains of amino acids (representing each protein) as input and produces an embedding; an interaction encoder, which combines the molecular and PPI embeddings to generate a representation of the PPI-modulator complex; and finally, a multilayer perceptron (MLP) classifier consisting of fully connected layers with ReLU activations. This classifier takes the PPI-modulator representation and predicts the probability of the modulator being active through a sigmoid function. The architecture is illustrated in Figure 4.

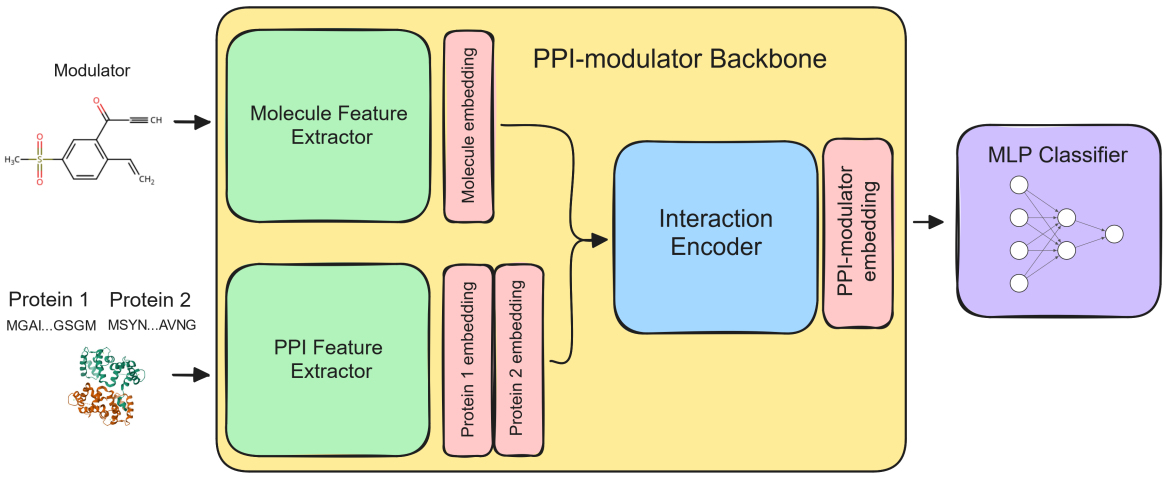


Figure 4: MultiPPIMI architecture adapted from [137].

The main contribution of MultiPPIMI is the development of a PPI-modulator backbone, which generates valuable representations of a PPI-modulator complex. We will now delve into the inner workings of the modules within the PPI-modulator backbone.

3.1.1.1 Molecule Feature Extractor

The molecule feature extractor module is based on the Graph Multi-View Pre-training (GraphMVP) framework [89]. GraphMVP is formed by a 2D graph neural network (GNN) model that is pre-trained using an auxiliary 3D GNN through self-supervised learning on the GEOM database [5]. The idea behind GraphMVP is to develop a 2D graph encoder capable of generating embeddings that integrate 3D geometrical information.

GraphMVP employs two self-supervised tasks for pre-training. The first task revolves around contrastive learning, where positive labels are assigned to pairs of 2D and 3D

representations of the same molecule, while pairs of 2D and 3D representations of different molecules receive negative labels. Throughout training, the embeddings of positive instances are optimized to become similar, while those of negative instances are pushed to be different via a loss function L_C. The second task focuses on generative learning, requiring the 2D and 3D GNN models to learn molecular representations that can accurately reconstruct their 3D and 2D counterparts. This training method adopts a VAE-like approach, incorporating a loss function L_G that combines a reconstruction term with a Kullback-Leibler divergence [75] term. Both tasks are integrated into a single loss function with two weighting coefficients α_1 and α_2:

$$L_{GraphMVP} = \alpha_1 L_C + \alpha_2 L_G \,. \tag{3.1}$$

To enhance the learned representations of GraphMVP, a variant called GraphMVP-C [89] is introduced, incorporating a third self-supervised contrastive learning task focused solely in 2D graphs. This additional task is integrated into the model's loss function via a third weighting coefficient α_3:

$$L_{GraphMVP-C} = L_{GraphMVP} + \alpha_3 L_{Contrastive2D} \,. \tag{3.2}$$

Finally, once the pre-trained GraphMVP-C model calculates an encoding, 10 complementary physicochemical properties of the molecule (such as number of rings, number of Nitrogens and Oxygens...) are concatenated to the feature vector becoming the N-dimensional output of the molecule feature extractor module.

3.1.1.2 PPI Feature Extractor

Proteins enter the PPI feature extractor as text representations of their amino acid sequence. This format does not explicitly encode their structural information, however, it can be indirectly inferred with a complex enough model.

The PPI feature extractor is based on the pre-trained protein language model ESM-2 [86]. ESM-2 adopts a BERT-like [25] transformer architecture trained on 25 million protein sequences sourced from the UniRef database [138]. It is trained with a masked language modeling objective, where 15% of the amino acids of a sequence are hidden and ESM-2 tries to predict these missing positions.

ESM-2 generates an embedding for each amino acid in the sequence. Therefore, to obtain an encoding for both proteins, a mean pooling operation is applied to all resulting embeddings. Then, after calculating both encodings, 19 physicochemical properties of each protein (such as composition of acidic residues, composition of basic residues...) are concatenated to the feature vectors becoming the M-dimensional outputs of the PPI feature extractor module.

3.1.1.3 Interaction Encoder

The interaction encoder is based on an adapted version of DrugBAN [7], a deep bilinear attention network (BAN) framework designed to learn representations of pairwise local interactions between molecules and proteins. The BAN module comprises two layers: a bilinear interaction map that captures pairwise attention weights, and a

bilinear pooling layer over the interaction map to extract a unified PPI-modulator representation.

Given the concatenated protein representations $\vec{H}_p = \left(h_p^1, \ldots, h_p^{2M}\right)$ and the modulator representation $\vec{H}_m = \left(h_m^1, \ldots, h_m^N\right)$. The bilinear interaction map $I \in \mathbb{R}^{N \times 2M}$ is defined as:

$$I = \left(\left(\vec{1} \cdot \vec{q}^\top\right) \circ \sigma\left(\vec{H}_m^\top U\right)\right) \cdot \sigma\left(V^\top \vec{H}_p\right) \tag{3.3}$$

where $\vec{1} \in \mathbb{R}^N$ is a fixed vector of ones, $\vec{q} \in \mathbb{R}^K$ is a learnable weight vector, $U \in \mathbb{R}^{D_m \times K}$ and $V \in \mathbb{R}^{D_p \times K}$ are learnable weight matrices for modulator and protein representations, $\sigma(\cdot)$ is an activation function, and $\circ$ denotes the element-wise product. Rewriting equation (3.3) to get the value of each element I_{ij}:

$$I_{ij} = \vec{q}^\top \left(\sigma\left(U^\top h_m^i\right) \circ \sigma\left(V^\top h_p^j\right)\right), \tag{3.4}$$

one can observe how matrices U and V take the modulator and protein representations to a common feature space, and then, the weight vector $\vec{q}$ regulates the strength of the interaction between each pair.

Once the bilinear interaction map I is computed, a bilinear pooling layer is applied to obtain the joint representation $f' \in \mathbb{R}^K$. The k-th element of f' is calculated as:

$$f'_k = \left(\sigma\left(\vec{H}_m^\top U\right)_k\right)^\top \cdot I \cdot \sigma\left(\vec{H}_p^\top V\right)_k, \tag{3.5}$$

where the sub-index k denotes taking the k-th column of a matrix. Note that the learnable parameters of the bilinear interaction map and bilinear pooling layer are shared to increase efficiency and reduce over-fitting. Additionally, a one-dimensional and non-overlapped sum pooling operation with stride s is applied to the joint representation f' to get a compacted representation $f \in \mathbb{R}^{K/s}$.

Finally, the interaction encoder includes multiple attention-heads through the summation of each of the single heads computed feature vectors. These attention heads share their U and V matrices, and only differ on their $\vec{q}$ weight vector so as to maintain parameter efficiency.

3.2 Bioactivity Filter

Our bioactivity filter is built atop MultiPPIMI's PPI-modulator backbone [137]. Rather than constructing a classification model, our focus lies in conducting bioactivity assessment through regression to rank and order candidate compounds. This deep learning-based filter will be implemented using PyTorch [107] and PyTorch Geometric [30]. Furthermore, due to the computational demands associated with inference using the ESM-2 model, the weights of the PPI feature extractor will remain fixed, as protein embeddings will be computed only once, using the Hugging Face library [157].

3.2.1 Data Processing

We will utilize PPI data sourced from the ChEMBL database [35], comprising 6900 compounds and spanning 123 distinct PPIs concerning the bioactivity of PPI modulators. Each entry in our dataset consists of three components: the SMILES code of a molecule, used for computing its molecular graph; two UniProt IDs [23] which

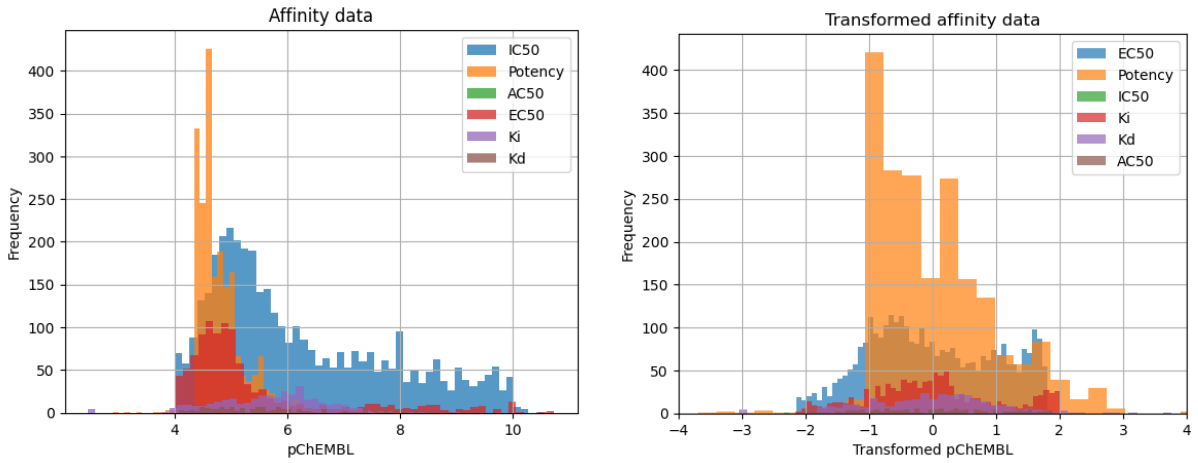
identify both proteins and are linked to their corresponding amino acid sequences; and a label known as the pChEMBL value. This quantity is calculated from one of six different types of bioactivity measurements as:

$$\text{pChEMBL} = -\log_{10}\left(\{\text{IC}_{50}, \text{Potency}, \text{AC}_{50}, \text{EC}_{50}, \text{Ki}, \text{Kd}\}\right). \tag{3.6}$$

Some molecules have several pChEMBL labels measured in different experiments. In such cases, we calculate the median value of the labels (avoiding using the mean to prevent skewing due to outliers). Although pChEMBL is commonly utilized as a unified value for various experimental measurements, we decide to apply transformations to the labels, segmenting them based on their measurement type. This segmentation is important because, despite the correlation between different types of measurements, they often assess distinct aspects of what constitutes bioactivity. Two transformations are applied to the labels: a Box-Cox transformation [15] and normalization. The Box-Cox transformation is first calculated using the SciPy library [148], as:

$$Y_i^{(\lambda)} = \begin{cases} \frac{Y_i^{\lambda}-1}{\lambda} & \lambda \neq 0 \\ \log(Y_i) & \lambda = 0 \end{cases}, \tag{3.7}$$

where Y_i represents a data point from the distribution Y, and $\lambda \in [-5, 5]$ is the optimal value which results in the best approximation of $Y^{(\lambda)}$ to a normal distribution. The Box-Cox transformation is employed to convert the distribution of regression labels into a normal distribution. This allows positive predictions from the bioactivity model to be interpreted as being above the average of the dataset distribution once the values are normalized. Normalizing the labels after the Box-Cox transformation aims to make the pChEMBL values more comparable across different types of measurements. Once all labels are transformed, they will be treated as representing the same magnitude. However, this separate transformations will need to be accounted for at the time of measuring a model's performance. The transformations applied can be observed in Figure 5.



(a) (b)

Figure 5: (a) Histogram of pChEMBL. (b) Histogram of transformed pChEMBL.

3.2.2 Modified MultiPPIMI

To design our filter, we will adapt MultiPPIMI's classification module into a regression module. We consider four architectures for our regression module: an MLP regressor trained using our pChEMBL dataset, a neural network trained using multi-task learning (MTL-NN) for both classification and regression, and two versions of the aforementioned architectures implemented as Bayesian neural networks (BNNs), utilizing Bayesian layers in place of the typical fully connected ones.

3.2.2.1 Multi Task Learning Neural Network

The MTL-NN is trained using both our regression dataset and a curated classification dataset comprising 16,265 molecules from MultiPPIMI's paper [137]. The MTL-NN ends up in a classification node connected to a final regression node, as illustrated in Figure 6. Throughout each epoch of training, the MTL-NN initially predicts and back-propagates through the classification node, followed by the same process through the regression node. It is worth noting that both tasks cannot be backpropagated simultaneously due to the scarcity of shared PPI-modulator pairs between the classification and regression datasets. The MTL-NN introduces two key enhancements over the naive MLP implementation: firstly, the exposure to a larger number of molecules during training enhances the learned PPI-modulator representation; secondly, the classification node could provide valuable insights to the regression node, thereby improving its predictions.

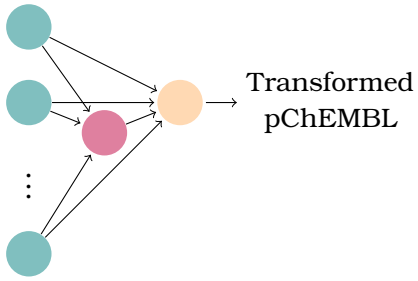


Figure 6: Output layers of a MTL-NN with a classifyng node in purple and a regression node in orange

3.2.2.2 Bayesian Neural Networks

BNNs change the usual NN paradigm by having variable outputs for the same input. This characteristic enables them to offer a measure of uncertainty for our model's predictions on bioactivity. Such uncertainty measures can be applied to filter out molecules with too uncertain estimations, thus improving the efficacy of our filter.

One way of introducing stochasticity to a NN output is by sampling its weights W from a distribution q_θ parameterized by θ. Therefore, the optimization process aims to minimize the expected loss L of the network given a dataset D:

$$\mathbb{E}_{(x,y)\sim D, W\sim q_\theta}\left[L\left(f\left(x, W\right), y\right)\right]. \tag{3.8}$$

Variational BNNs [37] modify this concept with a Bayesian approach. Given a predefined prior distribution $p(W)$, an approximate posterior $q_\theta(W) \approx p(W|D) \propto p(W)p(D|W)$

is attempted to be fit, maximizing the evidence lower bound (ELBO) with respect to θ:

$$F(\theta) = \mathbb{E}_{W\sim q_\theta}\left[\log p\left(D|W\right)\right] - D_{KL}\left(q_\theta \| p\right), \tag{3.9}$$

where the first term corresponds to the data likelihood, measuring how well the data is fitted, and the second term is the Kullback-Leibler divergence between the approximate and the true posterior, which is minimized when maximizing ELBO.

The Bayesian layers tested follow the variational BNN paradigm. They are implemented using the default parameters of the Bayesian-Torch library [74], employing the flipout training method [155] to decorrelate the gradients of the network within a mini-batch reducing variance.

3.2.3 Bioactivity Filter Results

In this section, we will present the results that guide our decision-making process regarding which model to employ as our bioactivity filter. We will start by obtaining regression metrics for each model. Then, we will move on to performing a customized evaluation of the various filters that can be constructed from each regression module.

3.2.3.1 Regression Metrics

The regression metrics are computed through 5-fold cross-validation on the regression dataset, although the MTL-NN regression modules employ all the classification data for training. Each model undergoes training for 100 epochs, and the weights from the epoch with the lowest Mean Squared Error (MSE) on the validation dataset is saved. Predictions from the BNN models are taken as the mean of 10 inference runs. During the calculation of the test results, each model's prediction is reverted to its typical pChEMBL value using the saved set of preprocessing parameters from the training dataset, depending on the measurement type. Furthermore, an untrained MLP regression module is established as a baseline. The regression metrics are detailed in the following table:

Table 1: Average 5-fold CV metrics for each regression module.

Model	$\mathbf{R^2}\uparrow$	**MAE** $\downarrow$	**RMSE** $\downarrow$	**MAPE** $\downarrow$
Untrained Baseline	0.1606	0.8760	1.2941	0.1403
MLP	0.8190	0.4182	0.6006	0.0740
MTL-NN	**0.8310**	**0.4086**	**0.5797**	**0.0723**
Bayesian MLP	0.7850	0.4514	0.6540	0.0809
Bayesian MTL-NN	0.7991	0.4301	0.6313	0.0749

As seen in Table 1, the basic MTL-NN module marginally outperforms the others across all metrics; showing a R^2 score that indicates a good knowledge on PPI bioactivity. However, the differences between models are so small, that there is no clear winner. Furthermore, while all models clearly outperform the baseline performance (as expected), the R^2 results for the baseline suggest a slight bias in the procedure for inverting the transformed pChEMBL predictions. It appears that employing distinct inverse transformations based on the measurement type of the test molecule leaks some information regarding its value.

It is worth noting that the dataset partition used considers the entire modulator-PPI complex. Thus, during testing, a model will not encounter a previously observed molecule-PPI data point. However, it will likely be tested on compounds for which it has seen their PPI combined with a different molecule (although there will also be unseen PPIs in the test dataset). Due to this fact, the achieved results are only extrapolatable to the largest PPIs families within our regression data, not necessarily to unseen PPIs. PPI modulator data outside the most tested families is very limited, thus it is not correct to directly assume that these results are extendable to our context. An alternate testing approach is required to thoroughly evaluate these models.

3.2.3.2 Few-shot filter test

The few-shot filter test is designed to evaluate a bioactivity filter's performance in a low data regime, where we have only a few known modulators with measured bioactivity values for a given PPI. The test is structured as follows:

1. The set M of all molecules associated with a specific PPI is removed from the regression and classification training data.
2. A 80-20 split is performed for training with the remaining data.
3. 10 randomly selected molecules are removed from M and added back into the training dataset (for both the regression and classification data).
4. The trained model is used to make predictions and pass a filter through all remaining molecules in M.
5. The quality of the chosen molecules is compared to that of the compounds in M, specifically, we define the metric τ as the difference between the mean experimental pChEMBL of the chosen molecules μ_{mols} and the mean experimental pChEMBL of M μ_M, scaled by the standard deviation of the population σ_M:

$$\tau = \frac{\mu_{mols} - \mu_M}{\sigma_M} \tag{3.10}$$

This test will be performed with the four most populated PPIs in the ChEMBL database. The UniRef IDs of each protein in the PPI complex, and the number of associated molecules is shown in the following table:

Table 2: Most populated PPIs in ChEMBL.

PPI	n° molecules
O00255|Q03164	1985
P04637|Q00987	1352
Q16236|Q14145	532
P37231|Q15596	338

During the few-shot filter testing, filters built from both the deterministic and Bayesian versions of the MLP and MTL-NN regression modules were evaluated. However, the BNNs exhibited a significant limitation in their uncertainty estimations. For all BNNs used, a prediction was computed as the mean of 10 inference runs, with the corresponding uncertainty represented by the standard deviation. Despite this, the BNN

models tended to assign nearly identical uncertainty values to all predictions, with the standard deviation of the uncertainties being close to zero. This indicates that the uncertainty values were not informative, as they remained nearly constant. The Bayesian-based filters were found to be inferior to the deterministic ones due to having equivalent performance but with higher computational costs for both inference and training. Therefore, their results are not included, as they are redundant.

We will consider a total of four different filters using the deterministic MLP and MTLNN regression modules. Given a molecule m, these filters will be parametrized by a θ value:

$$\begin{array}{llll} (1) & f_{MLP}(m) > \theta & (2) & f_{MTLNN}(m) > \theta \\ (3) & f_{class}(m) > \theta & (4) & f_{MTLNN}(m) > \theta \wedge f_{class}(m) > 0.5 \end{array}$$

where f_{MLP} and f_{MTLNN} denote the prediction of the transformed pChEMBL value, and f_{class} represents the predicted modulation probabilities of the MTLNN classification node. For each filter, we will calculate the metric τ (3.10) using θ values ranging from 0 up to the maximum possible θ, which is reached when no molecules pass the filter. We will then plot curves with coordinates (θ, τ). In Figure 7, we train 10 models per filter per PPI and graph each filter's performance as the mean and standard deviation of its (θ, τ) curves.

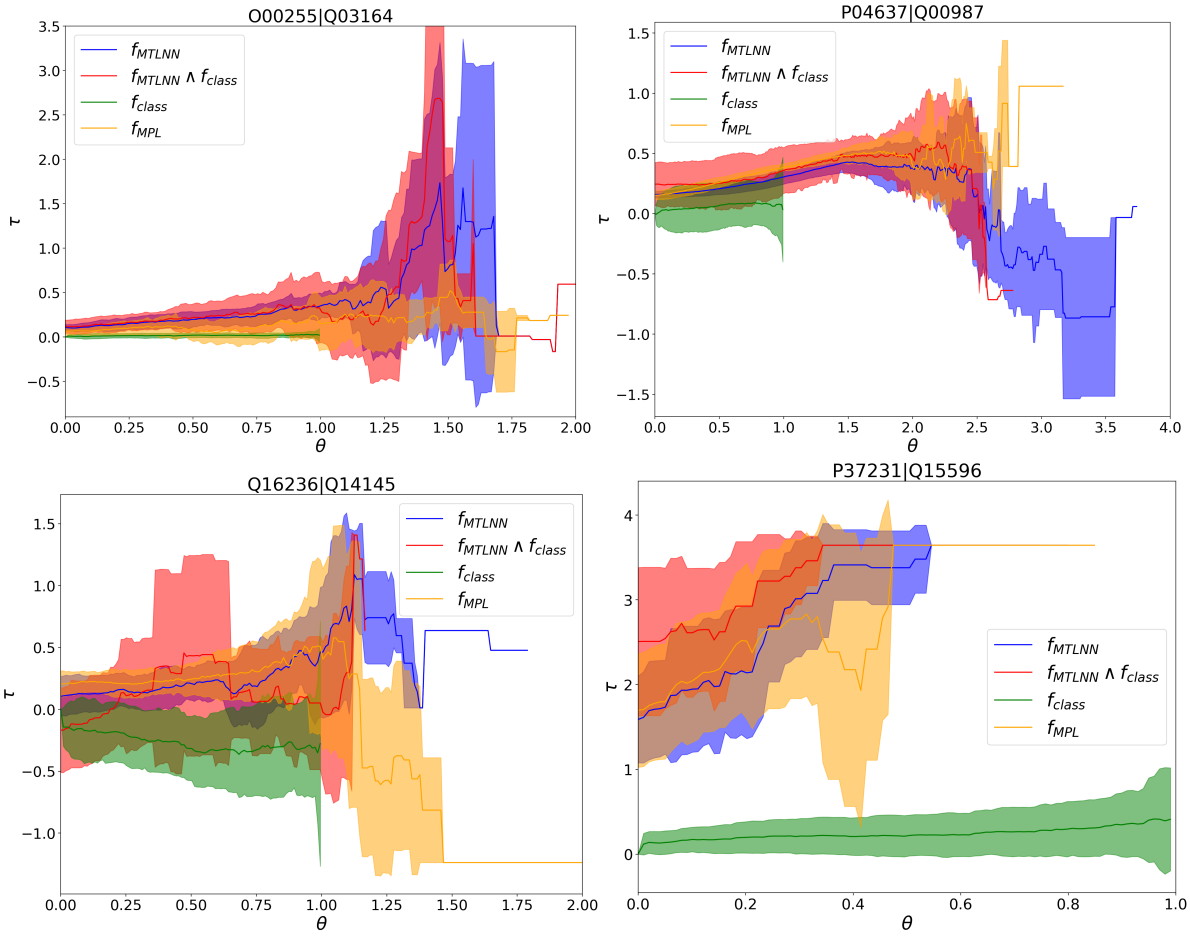


Figure 7: (θ, τ) curves for each deterministic filter

Looking at Figure 7, there are four key points that can be taken. First, the f_{class}

filter does not perform better than random selection, as its τ value consistently stays around 0. Therefore, it will be taken out of consideration as a bioactivity filter. Second, at high θ values, too many molecules are filtered out, causing the curves to become straight lines and increasing variability as τ loses statistical relevance. Third, the other filters oscillate around positive τ values at low and mid-range θ values, indicating that their chosen molecules are better on average than the rest of the population. This suggests that these filters can identify good compounds with limited training data. Moreover, they show upward τ trends (when θ is not too high), indicating that higher predicted pChEMBL values have some positive correlation to higher experimental labels. Fourth, it is difficult to compare these filters' performance solely based on graphical data, therefore, an objective evaluation method needs to be defined.

To quantify the quality of a filter's curve, we introduce a metric called τ_{filter} defined as the average of τ values weighted by their θ coordinates:

$$\tau_{filter} = \frac{\sum_i (\theta_i + 1)\tau_i}{\sum_i (\theta_i + 1)}. \tag{3.11}$$

We choose to use a weighted average instead of the usual mean since we recon that having high τ values at high θ values is relevant to establish a filter's quality. This is important because, even if these filters function as hard cutoffs, assigning the highest predicted values to the best molecules is more desirable, as the top-ranking compounds will be chosen first for further evaluation. Additionally, note that the weighted average in (3.11) is calculated at $\theta_i + 1$ to give some weight to τ when θ is close to 0.

Calculating τ_{filter}, we proceed to compare the different filters. Table 3 shows that the MTLNN-based filters score a higher mean τ_{filter} than f_{MLP} in 3 out of 4 PPIs. This advantage may be due to the MTLNN models' ability to learn a better representation of the PPI-modulator complex because they incorporate extra classification data during training. Comparing f_{MTLNN} and $f_{MTLNN} \wedge f_{class}$, each outperforms the other in 2 out of 4 PPIs. However, given that f_{class} does not show good predictive values, it is preferable to define the bioactivity filter solely based on the regression predictions of the MTLNN module.

One limitation of these filters is that, even when trained on the same data, they exhibit high run-to-run variance. To mitigate this, we propose using an ensemble of 10 trained models and combining their predictions using the median, which helps reduce the impact of outliers. Furthermore, having multiple predictions per molecule allows for the estimation of uncertainty using the standard deviation. Unlike the BNNs' uncertainty estimation, this method does not suffer from near-constant values. Considering this, we propose two additional filters based on an ensemble model: one that uses the median of predictions, and another that incorporates an extra term penalizing uncertainty.

$$(5)\ med\left(f^i_{MTLNN}\right) > \theta \quad (6)\ med\left(f^i_{MTLNN}\right) - \sigma\left(f^i_{MTLNN}\right) > \theta.$$

We proceed to graph the (θ, τ) curves of the new filters compared to f_{MTLNN} and f_{MLP}:

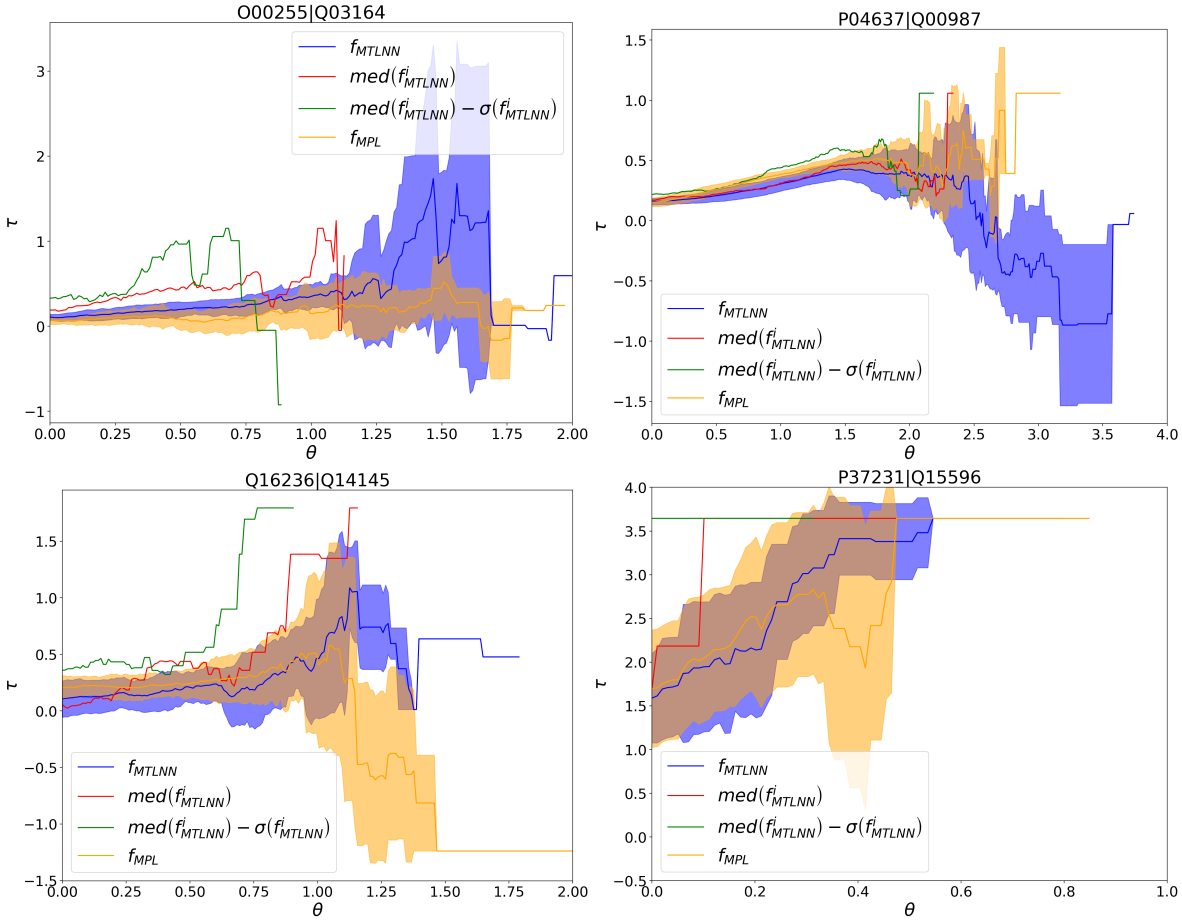


Figure 8: (θ, τ) curves for each ensemble filter, f_{MTLNN} and f_{MLP}

Looking at Figure 8, the ensemble method appears to improve the τ values of the filters. However, the new filters' curves are shorter due to the penalization term and the application of the median over several predictions, which makes it harder for a molecule to pass the filter. This does not pose a significant problem as long as it is accounted for when choosing a θ value as a hard cutoff filter. In Table 3, we calculate the τ_{filter} values for the new filters. Note that these calculations do not have an associated standard deviation since there is only one instance of the entire ensemble model.

Table 3: τ_{filter} results of bioactivity filters.

	PPI τ_{filter}			
	O00255|Q03164		**P04637|Q00987**	
Filter	**Mean ↑**	**Std**	**Mean ↑**	**Std**
f_{MLP}	0.160	0.102	0.438	0.077
f_{MTLNN}	0.439	0.212	0.249	0.230
f_{class}	0.018	0.026	0.064	0.182
$f_{MTLNN} \wedge f_{class}$	0.365	0.372	0.394	0.227
$med\left(f^i_{MTLNN}\right)$	0.472	-	0.368	-
$med\left(f^i_{MTLNN}\right) - \sigma\left(f^i_{MTLNN}\right)$	**0.515**	-	**0.469**	-

	PPI τ_{filter}			
	Q16236|Q14145		**P37231|Q15596**	
Filter	**Mean ↑**	**Std**	**Mean ↑**	**Std**
f_{MLP}	0.165	0.405	2.474	0.875
f_{MTLNN}	0.345	0.255	2.693	0.622
f_{class}	-0.272	0.290	0.251	0.260
$f_{MTLNN} \wedge f_{class}$	0.218	0.505	3.118	0.492
$med\left(f^i_{MTLNN}\right)$	0.682	-	3.419	-
$med\left(f^i_{MTLNN}\right) - \sigma\left(f^i_{MTLNN}\right)$	**0.882**	-	**3.645**	-

As shown in Table 3, the ensemble model with uncertainty penalization outperforms all other filters across all four PPIs. Moreover, this filter consistently exhibits a positive τ_{filter} value, indicating its effectiveness in identifying promising modulators even with limited training data. Therefore, this model is selected as our final bioactivity filter. Regarding the θ value, we will implement a hard cutoff of $\theta = 0$. However, it's important to note that molecules with the highest bioactivity estimates will be chosen first in case there's a need to limit the number of outputted candidates. In summary, we will train 10 f_{MTLNN} predictors on a 10-fold split of our dataset, and, for a given molecule m, the bioactivity filter used for screening will be:

$$f_{bio}(m) \equiv med\left(f^i_{MTLNN}(m)\right) - \sigma\left(f^i_{MTLNN}(m)\right) \tag{3.12}$$

$$f_{bio}(m) > 0. \tag{3.13}$$

3.3 Previous Works in Toxicity and Blood-Brain Barrier Penetration prediction

The toxicity and BBBP prediction tasks are commonly formulated as multi-label classification and binary classification problems respectively. Notable classification datasets for toxicity prediction include ClinTox [159], Tox21 [115], and ToxCast [116], while the most important BBBP dataset was compiled by Martins et al. [96].

Currently, state-of-the-art toxicity prediction is mainly centered around employing graph convolutional networks (GCNs) to learn useful representations, which are then classified using feed-forward neural networks (FFNs). Some noteworthy models utilizing this approach include TrimNet [80], comprising message-passing networks with attention weights between nodes and their connecting edges; ToxicBlend [167], an ensemble of models (GCNs, FFNs, and gradient boosting) applied to different molecule representations (graphs, molecular descriptors, fingerprints, etc.); and BioAct-Het [108], a siamese neural network that compares the learned molecule representation from a GCN with the representation of a toxicity class built from molecular fingerprints of active compounds in the class. Apart from GCNs, transformers are also emerging as the state-of-the-art in toxicity prediction. An example is elEmBERT-V1 [133], a BERT-like bidirectional architecture pre-trained with masked language modeling, which encodes tokenized SMILES representations into embeddings for classification.

State-of-the-art BBBP prediction follows similar principles to toxicity prediction. Some of the models mentioned earlier have been utilized for BBBP prediction, such as

elEmBERT-V1 and TrimNet. Additionally, other notable models in the literature include GLAM [81], a GCN method that automates the selection of various hyperparameters; and OT-GNN [18], a GNN incorporating an aggregation function designed to preserve molecular structural information more effectively than traditional sum aggregation methods.

3.4 Toxicity and Blood-Brain Barrier Penetration Filter

We will classify toxicity and BBBP using the Chemprop package for chemical property prediction [49]. The Chemprop package utilizes directed message passing neural networks (D-MPNNs) and FFNs to learn molecule embeddings for prediction across various regression or classification tasks.

D-MPNNs work with molecular directed graph representations, where each bond is represented by two directed edges in both directions. The process of calculating a molecular embedding using D-MPNNs, as described in [49], unfolds as follows: initially, directed edge features e_{vw}^d between nodes v and w are computed as the concatenation of the atom features x_v and the undirected bond features e_{vw},

$$e_{vw}^d = \text{cat}\left[x_v\,, e_{vw}\right], \tag{3.14}$$

then, the hidden directed edge features h_{vw}^0 of size h are calculated using a feed forward layer with learnable weights $W_i \in \mathbb{R}^{h\times h_i}$ and an activation function σ,

$$h_{vw}^0 = \sigma\left(W_i e_{vw}^d\right), \tag{3.15}$$

next, the hidden features are iteratively updated by the message passing process T times based on the neighbouring nodes $N(v)$ and learnable weights $W_h \in \mathbb{R}^{h\times h}$,

$$h_{vw}^t = \sigma\left(h_{vw}^0 + W_h \sum_{j\in N(v)\setminus w} h_{jv}^{t-1}\right), \tag{3.16}$$

where node w is excluded from the neighbourhood to improve numerical stability; afterwards, the sum of all incoming hidden features is combined into atomic embeddings h_v through learnable weights $W_o \in \mathbb{R}^{h\times h_o}$ as

$$h_v = \sigma\left(W_o\,\text{cat}\left[x_v\,, \sum_{w\in N(v)} h_{wv}^T\right]\right), \tag{3.17}$$

finally, a molecular embedding h_m is computed as the concatenation of molecular descriptors x_m and the sum of atomic features h_v,

$$h_m = \text{cat}\left[\sum_{v\in V} h_v\,, x_m\right]. \tag{3.18}$$

Using the Chemprop package, we will train D-MPNNs on the Tox21 [115] and BBBP [96] datasets. Tox21 comprises 11,764 molecules with binary labels representing their belonging in 12 different toxicity classes, while BBBP comprises 2,053 molecules with binary labels. For training, we will use the default hyperparameters for the

classification task. In the context of toxicity, as most data samples are not labeled in all 12 classes, Chemprop will used masked backpropagation, wherein only the known labels update the weights. This technique allows the model to be trained on the entire dataset without discarding incomplete samples and eliminates the need to train 12 different models for each class. To validate the model, we will perform 5-fold cross-validation and calculate the ROC-AUC score for both prediction tasks. It is worth noting that ROC-AUC is the established metric for both tasks due to the highly skewed distribution of toxicity data in Tox21 towards negative samples and in BBBP towards positive samples. The model's performance will be compared with other state-of-the-art methods in the following table:

Table 4: ROC-AUC comparison of state-of-the-art models in Tox21 and BBBP.

Tox21	
Model	**Mean ROC-AUC ↑**
Chemprop [49]	0.885
BioAct-Het [108]	0.898
ToxicBlend [167]	0.862
TrimNet [80]	0.860
elEmBERT-V1 [133]	**0.958**
BBBP	
Model	**Mean ROC-AUC ↑**
Chemprop [49]	0.896
GLAM [81]	**0.932**
OT-GNN [18]	0.920
TrimNet [80]	0.850
elEmBERT-V1 [133]	0.905

As shown in Table 4, the Chemprop models offer comparable results to other state-of-the-art techniques. Utilizing these models, the toxicity filter will discard all compounds with a predicted positive label in any of the 12 toxicity classes, while the BBBP filter will remove all negatively predicted molecules. Given a molecule m, the toxicity and BBBP filters used for screening will be:

$$\sum_{i=1}^{12} f_{tox}^{(i)}(m) = 0 \tag{3.19}$$

$$f_{BBBP}(m) = 1\,. \tag{3.20}$$

3.5 Descriptor and Fingerprint Based Filters

Three properties of interest can be directly calculated from molecular descriptors or fingerprints using the RDKit library [1]: drug-likeness, synthetic accessibility, and molecular similarity.

3.5.1 Drug-likeness

Drug-likeness is typically estimated using an established metric called QED (quantitative estimate of drug-likeness) [10]. QED quantifies drug-likeness with a value between 0 and 1, with 1 representing the highest score. It considers eight widely used

molecular descriptors, such as molecular polar surface area, the number of aromatic rings, etc. For each calculated molecular descriptor x, a desirability function $d(x)$ is defined as an asymmetric double sigmoidal function:

$$d(x) = a + \frac{b}{\left(1 + exp\left(-\frac{x-c+\frac{d}{2}}{e}\right)\right)}\left(1 - \frac{1}{\left(1 + exp\left(-\frac{x-c-\frac{d}{2}}{f}\right)\right)}\right), \tag{3.21}$$

where a, b, c, d, e and f are different parameters for each descriptor. Once all desirability functions are computed, QED is calculated as their weighted geometric mean:

$$\text{QED} = exp\left(\frac{\sum_{i=1}^{8} w_i \ln(d_i)}{\sum_{i=1}^{8} w_i}\right). \tag{3.22}$$

QED provides a useful tool to identify molecules unsuitable for drug development. However, rather than relying on QED for estimating drug-likeness, we will use the QEPPI index [71] in this MFP. QEPPI is an extension of QED with different weights, making it better suited as a screening tool in the context of PPIs. To eliminate potentially suboptimal PPI modulators, molecules with low QEPPI scores will be discarded. Therefore, for a given molecule m, the QEPPI filter used for screening will be:

$$f_{QEPPI}(m) > 0.5\,. \tag{3.23}$$

3.5.2 Synthetic accessibility

Synthetic accessibility is estimated using a established metric known as the SAS (Synthetic Accessibility Score) [27]. It quantifies the ease of synthesis with a score ranging from 1 to 10, with 1 representing the most accessible compounds. SAS comprises two primary components: a $fragment\ score$ and a $complexity\ score$. The $fragment\ score$ is calculated by comparing the substructures that form the Morgan fingerprint of a molecule with a predefined set of fragments with assigned scores. Meanwhile, the $complexity\ score$ incorporates a penalizing term derived from chemical descriptors, such as molecular size. Additionally, the RDKit implementation of SAS introduces a $density\ score$, rewarding compounds with dense fingerprint representations to positively account for highly symmetrical molecules. SAS is defined as:

$$\text{SAS} = \left[-\left(fragment\ score + complexity\ score + density\ score\right)\right]_{1-10}, \tag{3.24}$$

where the sub-index $1-10$ denotes scaling the value inside the closed interval $[1, 10]$.

SAS serves as a valuable metric for filtering out molecules that may prove costly to synthesize during laboratory stages of development. Considering typical SAS values for drugs, for a given molecule m, the SAS filter employed for screening is:

$$f_{SAS}(m) < 4\,. \tag{3.25}$$

3.5.3 Molecular similarity

Molecular similarity can be calculated as a distance metric between vectors of molecular fingerprint representations. There are many distance metrics that can be used, but, for this MFP we will calculate the Tanimoto index [141] of Morgan fingerprints.

This metric is widely adopted in the field and has been demonstrated to be a suitable measure for assessing similarity [8].

Given two sets of Morgan fingerprint bits m_1, m_2, the Tanimoto coefficient is calculated as:

$$T(m_1, m_2) = \frac{|m_1 \cap m_2|}{|m_1| + |m_2| - |m_1 \cap m_2|}, \tag{3.26}$$

where $|M_i|$ denotes the number of bits equal to 1. This coefficient ranges from 0 to 1, where 0 indicates no common bits and 1 represents identical fingerprints. Equation (3.26) only compares pairs of molecules. To extend this definition for calculating similarity between a molecular fingerprint m and a set of compounds M, we take the maximum of each Tanimoto coefficient.

$$T(m, M) = \max_{m' \in M} T(m, m'). \tag{3.27}$$

Molecular similarity serves as a valuable metric to filter out candidates that are too related to a set of known molecules. Furthermore, calculating molecular similarity between candidate compounds helps eliminate those that are overly similar to each other, thereby enhancing the diversity of the proposed molecules. Considering typical similarity values for molecules, given a molecule m and a set of compounds M, the similarity filter used for screening will be:

$$f_{similarity}(m, M) < 0.5. \tag{3.28}$$

3.6 Binding Affinity Filter

Binding affinity, is estimated through molecular docking calculations. Molecular docking is a computational technique used to predict how a small molecule ligand will bind to a larger biomolecule, such as a protein. To apply this technique to medium-scale compound databases, AutoDocking tools are used to automate molecular docking calculations, handling processes from preparing the molecules to analyzing the most probable binding modes. For our MFP, we will utilize the dockstring Python library [34], a Python wrapper of the established docking package AutoDock Vina [143].

AutoDock Vina is made up of two parts: a scoring function which predicts the free energy of binding of a ligand for a given 3D conformation, and an optimization algorithm trying to find the global minimum of the scoring function. The scoring function is defined as a weighted sum of different interactions between atoms. Both the choice of interaction functions and weights is done by empirical reasoning. The optimization algorithm, an Iterated Local Search (ILS) global optimizer [92], operates through multiple steps involving mutation and local optimization in the space of molecular conformations. The local optimization used for ILS is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [2], which follows a gradient descent-like formula for optimization, with its main difference being that the direction of a given optimization step is calculated using not only the gradient, but also an approximation of the Hessian matrix.

The calculation of molecular dockings will be performed on known specific modulating pockets of a PPI. In case of not having this information this filter is omitted. We

will follow the recommended steps outlined in dockstring [34] for protein preparation. This involves transforming the PDB file of a protein into a PDBQT file, adding extra charge information to each amino acid residue. Different protein pockets may be selected by adjusting the binding coordinates. Due to the computational complexity of these calculations, they will only be conducted on a medium-sized list (less than 1000 compounds) of already filtered molecules. The docking score filter applied will feature a variable threshold determined by the calculated docking score of the reference ligand m_i for each protein pocket p_i. Therefore, given a molecule m, the filter will take the following shape:

$$f_{docking}(m, p_i) < f_{docking}(m_i, p_i). \tag{3.29}$$

Note that, as docking scores model free binding energy, we will want to minimize said value.

3.7 Summary of Filters

Here, we present a summary of all the filters introduced in this chapter, along with their respective names, property descriptions, and references to their defining formulas:

Table 5: Overview of introduced filters

Name	**Description**	**Filter**
Bioactivity	Capacity to modulate a specific biological target	(3.13)
Toxicity	Harmfulness towards an organism	(3.19)
BBBP	Capability to penetrate through the blood-brain barrier	(3.20)
QEPPI	Potential to become a PPI modulating drug	(3.23)
SAS	Difficulty to synthesize	(3.25)
Similarity	Similarity to known and sampled compounds	(3.28)
Binding affinity	Capacity to bind to a specific protein pocket	(3.29)

Chapter 4

Generative models

Generative models tackle the task of exploring the vast chemical space of possible compounds in order to find useful molecules. In this chapter, we will first cover the main classifications of models that arise in the state-of-the-art. Then, we will provide an in-depth explanation of the generative models employed in this work and how they have been implemented to design PPI modulators.

4.1 Generative Models in the Literature

Deep learning models have flourished in the literature due to their success in learning useful representations of highly intricate data. Several reviews exist on this context [11, 20, 106], where they generally group generative models into three classifications based on how molecules are represented at their input, how they generate compounds, and which deep learning architecture or technique they employ.

First, generative models can be classified based on the molecular representation they utilize: text-based and graph-based models. Text-based models generally work with SMILES strings, such as CVAE [82] or the work by Olivecrona et al. [103]. The SMILES encoding has a weak syntax where it is easy to generate invalid strings, making the percentage of valid molecules a metric for ranking generative models. However, two key improvements have practically fixed this issue: the introduction of new architectures capable of handling complex grammar rules, such as transformers like MolGPT [6], and the development of more robust encoding formats like SELFIES [73]. Graph-based models, on the other hand, directly encode structural information, but they come with the challenge of being harder to work with. Examples of graph-based models include CGVAE [88] and MoFlow [166].

Second, generative models can be classified based on how they grow the compounds they form: atom by atom, motif by motif, or one-shot. Atom by atom models build compounds by adding one atom (or symbol in text representations) at a time, such as MolDQN [169]. This method allows exploration of the entire chemical space but requires many inference steps to reconstruct larger molecules, often resulting in chemically infeasible compounds. Motif by motif models build compounds by connecting substructures from a predefined motif dictionary, as seen in GCPN [165] and JT-VAE [65]. These dictionaries are usually constructed with expert knowledge or heuristic rules. Using motifs can improve the generation process's effectiveness by

utilizing known chemically relevant structures as building blocks. However, the quality and diversity of a model's output depend on the quality of the chosen dictionary. Meanwhile, one-shot models generate all molecular data simultaneously, a method often used by diffusion models, such as EDM [51].

Finally, generative models can be classified based on the architecture or technique they use. We have identified eight (not necessarily mutually exclusive) classes: RNN-based models, Transformer-based models, VAE-based models, GAN-based models, Flow-based models, Diffusion-based models, RL-based models, and deep guided heuristic search models.

4.1.1 RNN-based Models

NLP methods for next token prediction can be easily adapted to the SMILES and SELFIES encoding domains. This is one of the main applications of Recurrent Neural Networks (RNNs) for generating molecules, as they are designed for processing sequential data. The generation process of an RNN is illustrated in Figure 9.

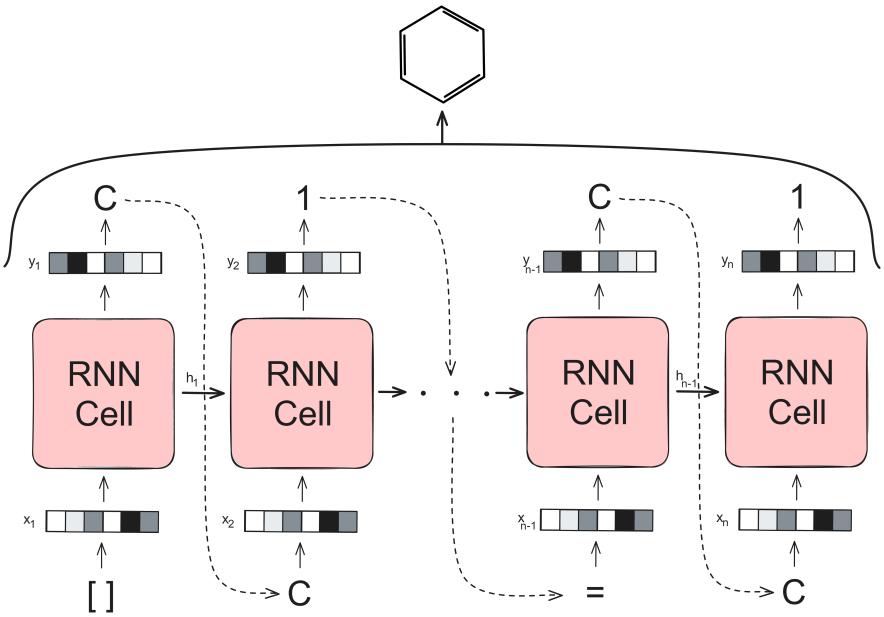


Figure 9: RNN architecture for SMILES generation.

Most RNN-based generative models employ LSTM cells [44, 100, 125] or GRU cells [168] to better model long-term dependencies. To tackle the inverse problem of generating a compound with certain properties, Kotsias et al. [72] developed a conditional RNN that takes molecular descriptors and bioactivity labels and transforms them into the initial LSTM states for generation. These models focus on unidirectional SMILES generation; however, since there is no defined start or end of a molecule, Grisoni et al. propose a bidirectional method, BIMODAL [40], which uses two RNNs to process a sequence in both directions and combine their predictions.

4.1.2 Transformer-based Models

The introduction of Transformers [145] as an architecture developed for NLP has also influenced the field of molecular design. Transformers are ideal for text-based compound generation because the self-attention mechanism allows them to consider all previous tokens during generation, thus addressing the memory problems that affect RNNs. They can learn representations that better capture molecular structures [120] and reduce the number of chemically invalid compounds sampled. Transformer architectures in the literature are built either with decoder blocks, following an autoregressive scheme, or with both a decoder and an encoder that takes initial data (such as a scaffold to build from, a protein to inhibit, etc.) and embeds it into the decoder for conditional generation, as illustrated in Figure 10.

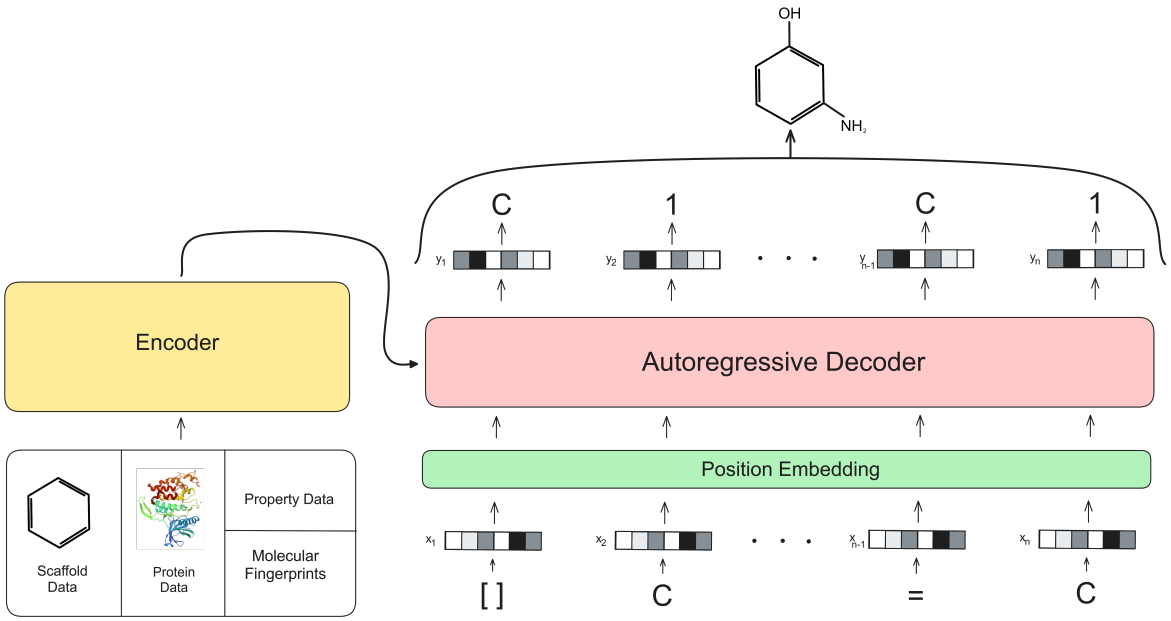


Figure 10: Transformer architecture for SMILES generation.

MolGPT [6] is an example of a GPT-like [114] model that follows the diagram in Figure 10. MolGPT encodes scaffold and property data to guide the generation task. Other researchers, like Dablain et al. [24] and Wang et al. [150], modify this idea by using molecular fingerprint data to steer the design process. Grechishnikova [38] proposes treating the generation task for protein inhibitors as a machine translation problem, with amino acid sequences as input and compounds as output. Meanwhile, cMolGPT [152] achieves conditional generation for a specific protein target by varying the different keys and values of the multi-head attention. Another approach is from Born et al., who propose the Regression Transformer [14], a multitask method that combines property prediction and molecule generation to design compounds with desired property constraints.

Besides text, Transformers can also be trained with graph data. This is the case with DrugEx v3 [90], which employs a novel positional encoding for the graph representation of input scaffolds, and GraphGPT [94], which integrates graph representations through a Graph Attention Mechanism.

4.1.3 VAE-Based Models

Variational autoencoders (VAEs) [69] consist of an encoder that maps input data to a smooth latent space and a decoder that reconstructs compounds from latent samples, as illustrated in Figure 11. Searching for ideal molecules through a VAE's latent space, as opposed to using the original chemical space, simplifies the property optimization task, making it a state-of-the-art technique for *de novo* generation.

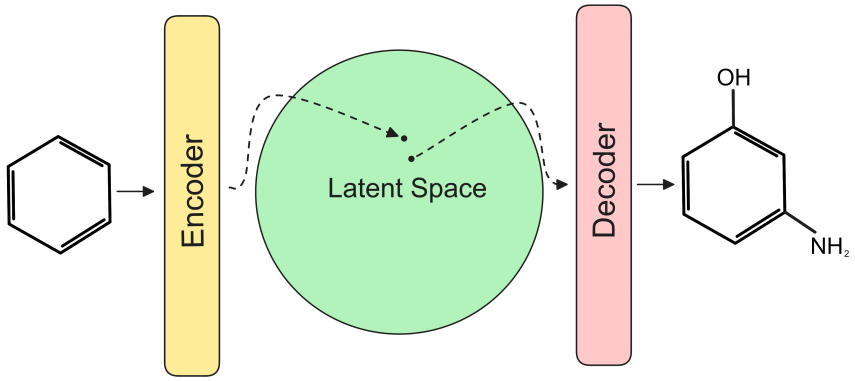


Figure 11: VAE architecture for molecular generation.

The use of VAEs in the literature varies depending on the molecular representation, the architectures employed for the encoder and decoder, and the method of traversing the latent space during the generation process. One of the most influential works in molecular generation with VAEs was done by Gómez-Bombarelli et al. [47]. In this work, the generation process uses SMILES representations, with a encoder composed of 1D CNNs and a decoder consisting of GRU layers. Additionally, a feed-forward network trained on property prediction is connected to the latent space to help shape it, and a Gaussian process guides the search for desirable compounds in the latent space. Lim et al. [83] extend this idea by including molecular properties in both the encoder and decoder to enhance conditional generation. CogMol [21] employs this method, using target data in their property prediction scheme to improve the binding affinity of generated compounds. Transformers have also been explored as an architectural advance for the encoding and decoding modules of a VAE, as seen in TransVAE [26]. However, these models still suffer from limitations, such as generating invalid molecules due to the use of SMILES encoding.

Bayesian property optimization in the latent space is often used to search for molecules. However, Griffiths et al. [39] demonstrate that this method tends to produce invalid structures when the optimization samples embeddings far from the data on which the VAE was trained. To address this, they propose a constrained Bayesian optimization scheme. Other search schemes involve heuristics, such as in Dragonet [162], where the latent coordinates of known desirable molecules are calculated, and a search is performed using one of those coordinates as a starting point, moving towards the other latent representations.

The use of graph data has also been deeply studied in the context of VAEs. GraphVAE [135] employs GCNs for encoding and MLPs to decode the latent data into an adjacency matrix, edge attribute tensor, and node attribute matrix that define a molecular graph. Calculating the reconstruction loss requires costly graph match-

ing operations during training, which is why MoVAE [87] works with nodes and edges individually to simplify this operation. To address the chemical validity problems affecting generative models, Jin et al. developed JT-VAE [65], a VAE that generates molecules through a tree-like structure with nodes representing simple substructures. This method constructs molecules motif by motif, ensuring that it always generates valid molecules. Additionally, some works focus on generation tasks in 3D space, such as NeVAE [124], which estimates 3D coordinates for the atoms in the constructed graph.

4.1.4 GAN-Based Models

Generative adversarial networks (GANs) [36] consist of two modules: a generator that creates new data from random noise, attempting to trick a discriminator; and a discriminator that tries to distinguish between real and generated samples, as illustrated in Figure 12. These models are trained through a zero-sum game, where their opposing goals drive mutual improvement. GANs have been widely successful in many generative tasks, leading to their broad adoption in molecular generation. However, they face significant challenges: high training instability and susceptibility to mode collapse, which reduces the diversity of generated compounds. To address these limitations, WGAN [3] was proposed as a GAN variant with modifications to its loss function, making it the predominant architecture for molecular generation using GAN-based models.

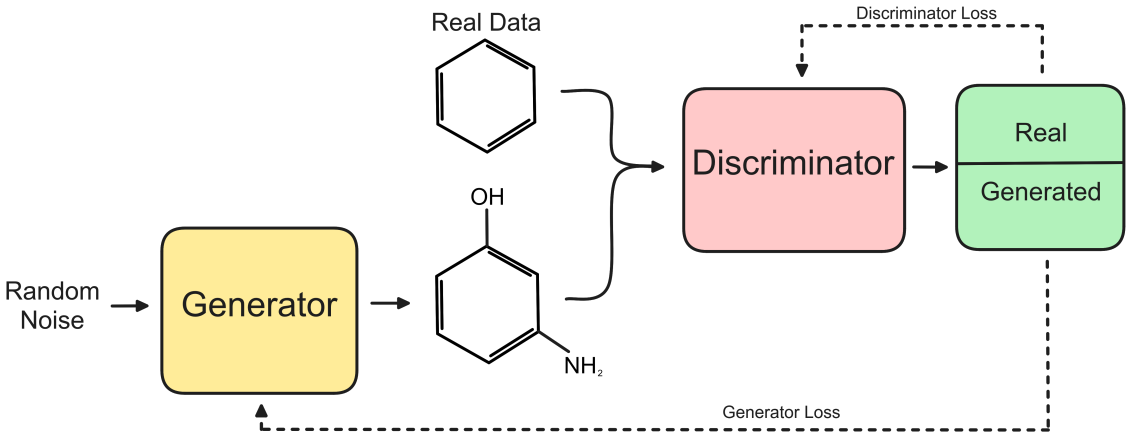


Figure 12: GAN architecture for molecular generation.

DrugGAN [67] is one of the first models to employ the GAN architecture using molecular fingerprint representations. However, SMILES representations are more commonly used in the literature, such as in ORGAN [42], which utilizes a text discriminator made up of 1D CNNs and an LSTM-based generator that optimizes a desired property through reinforcement learning. Due to the inherent difficulties of training GANs, a common technique to simplify this process is proposed in LatentGAN [113], where training is performed inside the latent space of an autoencoder, freeing the GAN model from having to learn the syntax behind SMILES strings. Several generation strategies emerge from the use of GANs, such as FeedbackGAN [45], which uses a predictor to score generated samples and gradually replaces real data points with the highest-scoring samples to guide the generative process toward a desired subset of the chemical space. Meanwhile, DeepTarget [19] concatenates random noise

with a protein embedding from a transformer as the input to the generator, guiding the design process toward generating molecules capable of binding to specific protein pockets. Other examples of using transformers in GAN architectures include SpotGAN [78] and TenGAN [79], which employ a transformer encoder as the discriminator and a transformer decoder as the generator, combined with a reward function to improve the design process with reinforcement learning.

With respect to graph-based molecular generation, GANs have lower adoption compared to other architectures. Some examples include MolGAN [17], which aims at generating the adjacency tensor and annotation matrix that define a graph, and MolCycleGAN [98], which, given a compound, tries to generate a similar structure with improved properties.

Finally, one last example worth mentioning is iPPIGAN [149], a GAN-based framework for PPI inhibitors generation. This model's generation process comprises two steps: first, a GAN using 3D CNNs to design 3D molecular shapes, and an auxiliary network of CNNs and LSTMs trained to parse SMILES strings from 3D molecular representations. To adapt iPPIGAN for PPI inhibitor generation, the model is trained on data samples filtered by their QEPPI value. Additionally, this model is used to design inhibitors for a specific PPI (MDM2/p53) by selecting generated compounds with a light gradient boosting machine regression model trained on MDM2/p53 bioactivity data.

4.1.5 Flow-Based Models

Normalizing flow models [140] aim to transform the distribution of data into a Gaussian distribution through a series of learnable invertible transformations. A sample from the latent space can then be converted back into real-world data by inverting the flow. A key strength of flow-based models compared to VAEs is that their invertible transformations enable the calculation of the exact data likelihood. Flow models in the field of molecular generation focus on graph data, transforming the graph's node and adjacency tensors with two separate parallel flows, where the adjacency tensor aids the node flow, as illustrated in Figure 13.

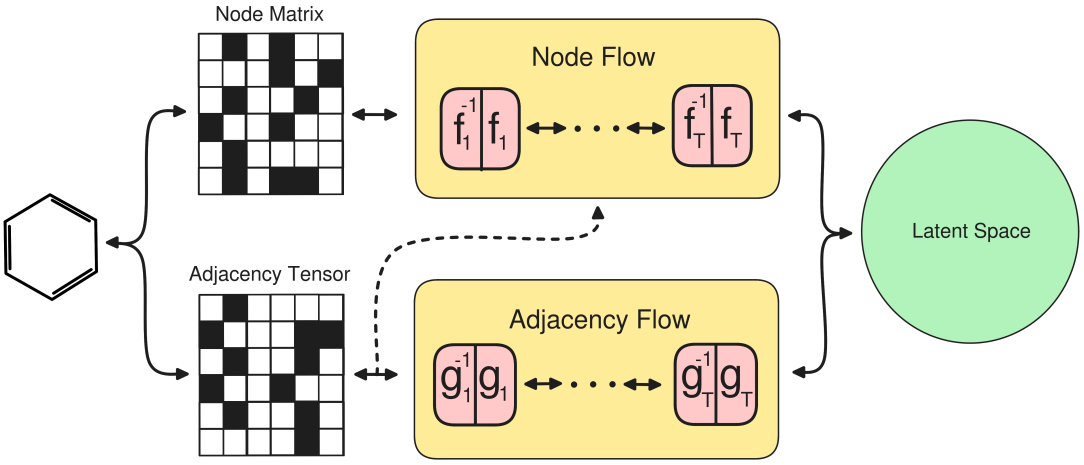


Figure 13: Flow architecture for graph molecular generation.

GraphNVP [68] is the first example of a flow-based model applied to molecular graph generation. Later, Honda et al. [50] proposed changing the transformations of the flow model with residual layers, achieving similar performance to GraphNVP while being more parameter efficient. However, these models suffer from generating invalid molecules. To mitigate this problem, GrapAF [134], an autoregressive model that sequentially adds new bonds and atoms to a given graph structure, was developed. GrapAF includes chemical checks between steps to ensure the molecule remains valid. Another example of a flow model aimed at generating valid molecules is MoFlow [166], which builds its node flow as a graph conditional flow to improve chemical validity. These models map the tensors that form a molecular graph to a continuous latent space. However, since these matrices are sparse and discrete, a continuous prior distribution is suboptimal for modeling them. Thus, GraphDF [95] is proposed as a flow model with discrete latent variables (using invertible modulo shift transformations), better suited for the molecular graph context. Additionally, another application of flow models in molecular design is the optimization of compounds by developing connections between given fragments or adding new radicals, as seen in FFLOM [62].

Finally, flow models are also being researched for molecular generation aided by target data. Current works of this nature include PocketFlow [61], an autoregressive model that considers protein information when choosing the next node and edge to grow, and the work by Rozenberg et al. [122], which encodes both the protein and molecular 3D graphs into the normalizing flow, imposing semi-equivariance conditions to ensure equivariance against rigid body transformations acting jointly on the protein and molecular graph.

4.1.6 Diffusion-Based Models

Diffusion models [136] aim to learn a mapping from random noise to real-world data. These models operate through a series of sequential steps where noise is gradually introduced to a sample, and then learn to reverse this process through gradual denoising steps, as illustrated in Figure 14. The use of diffusion models is relatively recent in the field of molecular generation, with research focusing significantly on 3D molecular generation, where molecules are represented as point clouds with associated features.

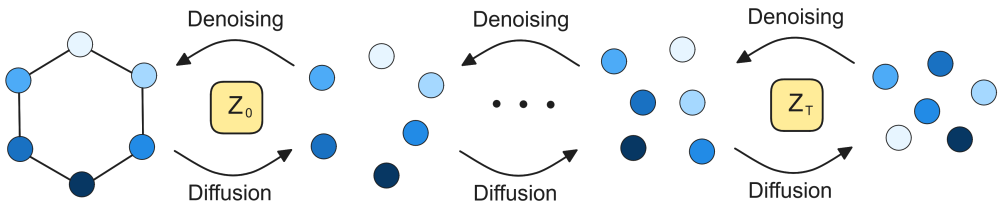


Figure 14: Diffusion architecture for 3D molecular generation.

Hoogeboom et al. [51] were the first to introduce molecular diffusion models, learning the denoising process with neural networks modified to be equivariant against translations, rotations, and reflections of atoms. To simplify the generation of complex structures, Xu et al. [161] later proposed the use of autoencoders (AEs) to work in a latent space more suitable for the diffusion process. Meanwhile, Huang et al. [53]

and Gong et al. [158] tried to improve the diffusion process by incorporating physical information, encoding interatomic forces. One limitation of diffusion models is that bonds are determined from predefined rules applied to generated point clouds, which can result in unrealistic molecules. Some models designed to mitigate this issue include MiDi [146], which generates both a graph and its corresponding 3D conformation, and MolDiff [110], which jointly samples atoms and bonds.

To generate molecules with desired properties, the diffusion process can be guided towards valuable regions of the chemical space by the gradients of property predictors. This approach is used by models such as GaUDI [154] and MOOD [77]. Additionally, diffusion models can be conditioned with target data to generate compounds based on protein structures. Examples include TargetDiff [41] and DiffSBDD [126], which consider the physical interaction between proteins and molecules during diffusion and denoising, DiffBP [84], which generates compounds in a non-autoregressive way, and the work by Lin et al. [85], which splits molecules into two types of components: substructures as functional groups and atoms connecting functional groups as linkers.

Other types of molecular generative tasks besides *de novo* design can be addressed with diffusion models, such as molecular linker design (generating connections between two given fragments) with models like DiffLinker [54], and R-group decoration (adding new fragments to a given molecule) with models like DiffDec [160].

4.1.7 RL-Based Models

Reinforcement learning (RL) is a widely used technique to build generators which optimize certain properties. There are two approaches towards the use of RL in molecular generation: using RL to fine-tune a pre-trained generator, or using RL to train an agent from scratch. The first approach is generally more adopted.

REINVENT [12, 104] is one of the most commonly used family of methods for molecular generation with RL. REINVENT fine-tunes a pre-trained RNN by maximizing the expected return of the model's policy given by the probabilities of the next chosen token. Some examples of papers improving this base idea include the work of Blaschke et al. [13], with an added memory unit that records already generated molecules to force a higher diversity output; or the work by Guo et al. [43], which introduces curriculum learning, a process that divides learning into stages that the model must pass by fulfilling increasingly complex objectives. REINVENT 4 [91] is one of the latest frameworks of the REINVENT family, incorporating some of the aforementioned improvements, and adding transformers as pre-trained generators. Additionally, some works like Atance et al.'s [4], apply the REINVENT RL method to graph-based generative models.

Other fine-tuning based methods outside from the REINVENT family include: RationaleRL [64], which defines a vocabulary of substructures, and tries to maximize the expected return of a VAE trained to complete graphs given a set of chosen substructures; or MolRL-MGPT [52], which applies multi agent RL to a group of GPT agents searching the chemical space.

Fine-tuning pre-trained models has the disadvantage when trying to explore the chemical space of being bounded by a pre-training distribution. Hence why, training agents from scratch also has practical applications. GCPN [165] is an example of

this approach, with a graph convolutional policy network that adds atoms and bonds sequentially. GCPN is trained for property optimization with an included adversarial loss given by a trained discriminator. Although, GCPN is not completly free from a predefined dataset, as the discriminator used (trained on said dataset) guides and limits the generation process. Another important example is MolDQN [169], which employs value function learning as opposed to policy optimization. This model's actions include adding atoms, bonds or deleting bonds. Some adaptations from MolDQN appear in the literature, such as MORLD [60], a MolDQN model using docking calculations as rewards.

Finally, it's worth noting the work by Ohue et al. [102], where they develop iPPI-REINVENT, a REINVENT based model designed for the generation of PPI inhibitors. iPPI-REINVENT optimizes the QEPPI of proposed molecules trying to look for general PPI inhibitors, however, their compounds are not designed to target any specific PPI.

4.1.8 Deep Guided Heuristic Search Models

Deep models, both generators and property predictors, can be combined with different heuristics to perform a guided search through the chemical space. The two most relevant heuristics applied to *de novo* design are Genetic Algorithms (GA) and Monte Carlo Tree Search (MCTS).

GAs are an extensive family of algorithms with a similar structure. First, the algorithm starts with a set P of subjects. Then, a subset $S \subseteq P$ is sampled from the population P. Next, S is used to generate a new set of subjects N through random modifications or combinations of subjects, called mutations and crossovers, respectively. Finally, a new population P' is selected from $P \cup N$, choosing the best-ranked subjects, and the process is repeated. In the context of molecular generation, the subjects are molecules, the mutations are operations such as adding or removing atoms and bonds, and molecules may be ranked with deep property predictors. Examples include the work by Kwon et al. [76], which optimizes Morgan fingerprints of molecules and employs an RNN to decode the fingerprints into SMILES; the work by Fu et al. [32], which uses an RL approach to select the mutations and crossovers performed by a policy network considering ligand and protein information; and GARel [151], an algorithm that involves sampling from a generative model, optimizing the samples with a GA, and reintroducing the optimized molecules into the generative model's training set. Software tools utilizing the GA methodology include DENOPTIM [31] and AlvaBuilder [97].

Meanwhile, the MCTS method aims to efficiently explore a tree of actions by iteratively repeating four steps: selection, where child nodes are selected, based on the number of times they have been visited and a reward, until reaching a leaf node; expansion, where child nodes are created from the selected leaf node and one is chosen; rollout, where actions from the chosen child node are selected (randomly or following a strategy) until a terminal state is reached; and backpropagation, where the reward of the terminal state is calculated and backpropagated to the set of visited nodes. In the context of molecular generation, nodes employed by MCTS represent either characters (in a text-based representation), atoms, or entire substructures. Examples of MCTS-based models include ChemTS [55, 164], which uses an RNN to perform the rollouts of a SMILES string; GB-GM-MCTS [59], which applies rollouts through random GA-based mutations of graphs; and VGAE-MCTS [56], which expands molecules

based on the feature map output by a graph VAE.

4.1.9 Generative Models Selection

Considering the aforementioned state-of-the-art, we will select several generative models spanning various architecture types based on the following criteria: relevance in the literature while being current, availability of the model's weights and code, and ease of implementation.

Seven models where tested as generators for our case study in chapter 5: MolGen [28], a pre-trained molecular language model; Taiga [99], a Transformer optimized with RL; HierVAE [63], a graph-based VAE; MOOD [77], a graph based diffusion model; FREED [163], a graph policy network trained with RL; SyntheMol [139], a MCTS-based generative model; and Pocket2Mol [111], an equivariant network for sampling in 3D space with protein data. However, only four models were capable of generating molecules that went through all filters. The reason behind this is that the bioactivity filter is, by design, hard to pass, hence why, only models aimed at directly optimizing the bioactivity property were capable of surpassing the filter.

In the following sections we will cover in depth the inner workings of the four successful models, HierVAE, FREED, SyntheMol and Taiga, and explain how they are combined with our trained filters for PPI modulator generation. A summary of the generative models classification presented can be consulted in appendix A.

4.2 HierVAE

HierVAE [63] is a graph VAE designed to reconstruct molecules by concatenating structural motifs. This approach reduces the number of decoding steps required to generate large molecules while maintaining molecular complexity and improving validity.

HierVAE employs hierarchical graphs H_G, a data structure that represents molecules through a hierarchical stacking of three layers of graphs. Each layer represents the molecular graph G at different resolutions, as illustrated in Figure 15.

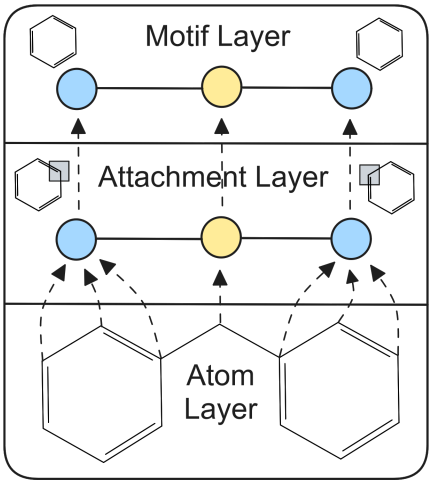


Figure 15: Example of a hierarchical graph adapted from [63].

The motif layer H_G^s describes the motifs that make up a molecule. It follows a tree structure with nodes S_i representing motifs and edges (S_i, S_j) between intersecting motifs. The attachment layer H_G^a describes how motifs are connected; each node $A_i = (S_i, \{v_j\})$ represents the attachment configuration of a motif S_i and its bonded atoms $\{v_j\}$. The atom layer H_G^g represents the usual molecular graph. These three layers are connected by directed edges, with atoms from the atom layer linked to their corresponding motifs from the attachment layer, and nodes from the attachment layer linked to the motif layer. The motifs and attachment configurations used are predefined in vocabularies V_S ($|V_S| = 5623$) and $V_A(S_i)$, respectively. These vocabularies are derived from large databases using simple heuristics to identify common structural motifs.

4.2.1 Hierarchical Graph Encoder

The hierarchical graph encoder utilizes message passing networks (MPNs) to compute embeddings for the nodes in each layer. The embedding h_v of a node v is calculated as the output of a MLP given as inputs the concatenation of the node features x_v and the sum of messages $\nu_{uv}^{(T)}$ at iteration T from all neighboring nodes $u \in N(v)$:

$$h_v = MLP\left(\text{cat}\left[x_v, \sum_{u \in N(v)} \nu_{uv}^{(T)}\right]\right). \tag{4.1}$$

A message $\nu_{uv}^{(t)}$ at iteration t is calculated as the hidden state of a modified LSTM cell with the node and edge features x_u, x_{uv} as input features, the messages from neighbouring nodes $\{\nu_{wu}^{(t-1)}\}_{w \in N(u) \setminus v}$ (excluding v) as the last hidden state, and a cell state $\{c_{wu}^{(t-1)}\}_{w \in N(u) \setminus v}$ also given by the cell states of the neighbouring nodes:

$$\nu_{uv}^{(t)} = LSTM\left(\text{cat}\left[x_u, x_{uv}\right], \{\nu_{wu}^{(t-1)}, c_{wu}^{(t-1)}\}_{w \in N(u) \setminus v}\right). \tag{4.2}$$

Using this MPN block, the graph encoder computes the embedding of the hierarchical graph H_G^g sequentially by layer. First, the set of atom layer H_G^g nodes $\{v\}$ is encoded as $\{h_v\}$:

$$\{h_v\} = MPN_{\psi_1}\left(H_G^g, \{x_u\}, \{x_{uv}\}\right). \tag{4.3}$$

Then, the set of attachment layer H_G^a nodes $\{A_i\}$ is encoded as $\{h_{A_i}\}$:

$$\{h_{A_i}\} = MPN_{\psi_2}\left(H_G^a, \{f_{A_i}\}, \{e(d_{ij})\}\right), \tag{4.4}$$

where $e(d_{ij})$ is the embedding vector of d_{ij}, a measure of relative ordering between A_i and A_j, and f_{A_i} is an attachment feature vector calculated from the features x_{A_i} and the embeddings $\{h_v\}$ of its corresponding atoms:

$$f_{A_i} = MLP\left(\text{cat}\left[x_{A_i}, \sum_{v \in S_i} h_v\right]\right). \tag{4.5}$$

Next, the set of motif layer H_G^s nodes $\{A_i\}$ is similarly encoded as $\{h_{S_i}\}$:

$$\{h_{S_i}\} = MPN_{\psi_3}\left(H_G^s, \{f_{S_i}\}, \{e(d_{ij})\}\right), \tag{4.6}$$

with $\{f_{S_i}\}$ being given by:

$$f_{S_i} = MLP\left(\text{cat}\left[x_{S_i}, h_{A_i}\right]\right). \tag{4.7}$$

Finally, the latent vector z_G encoding the hierarchical graph H_G is sampled from the embedding of the root motif h_{S_1} using the reparametrization trick applied to the output of two MLPs representing the mean $\mu\left(h_{S_1}\right)$ and log variance $\Sigma\left(h_{S_1}\right)$:

$$z_G = \mu\left(h_{S_1}\right) + \exp\left(\Sigma\left(h_{S_1}\right)\right) \cdot \epsilon, \quad \epsilon \sim N(0,1). \tag{4.8}$$

4.2.2 Hierarchical Graph Decoder

The decoder expands a hierarchical graph $H_G^{(t)}$ at step t by connecting a new motif S_t to a motif $S_k \in F$ from a maintained list of frontier motifs. This process continues until there are no more frontier nodes or a maximum number of steps is reached. Besides the latent representation z_G, an encoding of the partial hierarchical graph $H_G^{(t)}$ is required for decoding. Decoding proceeds inversely to encoding. First, a new motif S_t to attach to S_k is predicted as a classification task over the motif vocabulary V_S:

$$p_{S_t} = \text{Softmax}\left(MLP\left(\text{cat}\left[h_{S_k}, z_G\right]\right)\right). \tag{4.9}$$

Then, the attachment configuration A_t of motif S_t is predicted as a classification task over the attachment vocabulary $V_A(S_t)$:

$$p_{A_t} = \text{Softmax}\left(MLP\left(\text{cat}\left[h_{S_k}, z_G\right]\right)\right). \tag{4.10}$$

Finally, the model decides how S_t is attached to S_k. Given the atom pairs $M_{tk} = \{(u_j, v_j) | u_j \in A_k, v_j \in A_t\}$, where atom u_j and v_j would be attached together, the probability p_M of a candidate attachment M is calculated as:

$$p_M = \text{Softmax}\left(h_M \cdot z_G\right) \tag{4.11}$$

$$h_M = \sum_j MLP\left(\text{cat}\left[h_{u_j}, h_{v_j}\right]\right). \tag{4.12}$$

Combining the hierarchical graph encoder and decoder, they can be trained following the usual VAE method of minimizing the negative ELBO:

$$-\mathbb{E}_{z\sim Q}\left[\log P(G|z)\right] + \lambda_{KL}\mathcal{D}_{KL}\left[Q(z|G)\|P(z)\right], \tag{4.13}$$

with the first factor representing a reconstruction loss and the second a divergence of the latent space from a predefined distribution $P(z) \sim N(0, I)$.

4.2.3 Implementation

Given a HierVAE model pre-trained on the ChEMBL database [35], we proceed to finetune the model towards the generation of modulators of a specified PPI through the following steps:

1. Finetune the model on a training set D of known modulators of the given PPI.
2. Sample 20,000 molecules from the model.
3. Apply the bioactivity filter (3.13) and similarity filter (3.28) to the sampled molecules.

4. Save and append the remaining molecules to the training set D, and return to step 1 until 10 epochs have concluded.

This procedure ensures that the model is trained in an expanding subregion of interest within the chemical space, while avoiding overfitting by preventing repeated or highly similar molecules to be added to the finetuning dataset. After this generative process, the remaining molecules are passed through the rest of the filters and saved as potential PPI modulators.

4.3 FREED

FREED [142, 163] i RL model that employs a GCN-based policy to grow compounds by concatenating molecular fragments. This methodology has similar advantages to HierVAE (e.g. increased validity generation and a decreased number of decoding steps per sampled molecule), while not having its exploration capabilities bounded by a pre-training dataset.

4.3.1 Reinforcement Learning Setup

FREED works with extended molecular graphs as states, which include special nodes called attachment points representing locations where a bond can be made between fragments. Each action $a = (a_1, a_2, a_3)$ consists of three sequential subactions: first, one of the molecule's attachment points is selected for expansion; then, a fragment is chosen from a predefined motif vocabulary (for this MFP, we use a vocabulary of 91 fragments from the original paper, plus 5 fragments containing bromine to include more element diversity); finally, one of the fragment's attachment points is selected to connect to the compound. The episodes played by the model have a fixed length of 4 steps, meaning that molecules will be sampled as combinations of four fragments starting with benzene as the initial state, and all remaining attachment points at $t = 4$ will be bonded to hydrogen atoms. The reward at s_t is calculated as:

$$r_t(s_t, a_t, s_{t+1}) = \begin{cases} 0 & t < T = 4 \\ \max(0, R(s_{t+1})) & t = T = 4 \end{cases}, \tag{4.14}$$

where R is the docking score to a particular protein. In this MFP, R is replaced by our bioactivity model f_{bio} (3.12) scaled as $2f_{bio} + 5$ to move the predicted rewards into a range better suited for a RL setup. Additionally, the replay buffer is sampled with prioritized experience replay, where a (s_t, a_t, s_{t+1}, r_t) tuple is sampled with a higher probability if the reward r_t differs significantly from a prediction $f_r(s_t, a_t)$ given by a reward predictor network f_r.

4.3.2 Action Sampling and Policy/Critic Architecture

Before describing how the model's policy is defined, we first explain how actions are sampled from a categorical distribution $\pi = (\pi_1, \cdots, \pi_n)$. Due to the non-differentiability of sampling from a categorical distribution, the Gumbel-Softmax reparameterization trick [58] is applied to allow gradient backpropagation during training. Given $\vec{g} = (g_1, \cdots, g_n)$ i.i.d samples drawn from a Gumbel(0,1) distribution, a sample vector

$\vec{y}$ is generated as:

$$\vec{y} = \text{Gumbel-Softmax}(\pi) = \text{Softmax}\left(\frac{\log(\pi) + \vec{g}}{\tau}\right), \tag{4.15}$$

where $\tau > 0$ is a temperature parameter. Then, a sample $a \sim \pi$ is drawn as:

$$a = \text{one\_hot}\left(\text{argmax}(\vec{y})\right), \tag{4.16}$$

and using the straight-through Gumbel-Softmax estimator [58] the gradient is estimated as $\nabla_\theta a \approx \nabla_\theta \vec{y}$, since $\vec{y}$ is a smooth aproximation of a such that $\vec{y} \xrightarrow{\tau \to 0} a$.

We now discuss how the model's policy π_θ is autoregressively computed. Subaction a_1 decides which attachment point is used to expand the molecule. First, a GCN G_η is applied over the state graph s to obtain the node embeddings $\widetilde{V} \in \mathbb{R}^{|V| \times d}$, the graph embedding $\widetilde{S} \in \mathbb{R}^d$, and a submatrix of attachment node embeddings $\widetilde{V}^{att} \in \mathbb{R}^{|V^{att}| \times d}$:

$$\widetilde{V} = G_\eta(s), \quad \widetilde{S} = \sum_{\vec{v} \in \widetilde{V}} \vec{v}, \quad \widetilde{V}^{att} \subset \widetilde{V}. \tag{4.17}$$

Then, the graph $\widetilde{S}$ and attachment nodes embeddings $\widetilde{V}^{att}$ are combined by a multiplicative interaction layer $f_{\phi_1}^{MI}$ into a matrix encoding $\widetilde{A}_1 \in \mathbb{R}^{|V^{att}| \times d}$:

$$\widetilde{A}_1 = f_{\phi_1}^{MI}\left(\widetilde{S}, \widetilde{V}^{att}\right) = \widetilde{V}^{att^T} W_1 \widetilde{S} + U_1 \widetilde{V}^{att} + V_1 \widetilde{S} + b_1, \tag{4.18}$$

where W_1 is a learnable 3D tensor, U_1 and V_1 are learnable matrices, and b_1 is a learnable bias term. Next, an MLP $f_{\psi_1} : \mathbb{R}^{|V^{att}| \times d} \to \mathbb{R}^{|V^{att}|}$ is composed with a Gumbel-Softmax operator σ (4.15) to sample subaction $a_1 \sim \pi_1(\cdot|s)$:

$$\pi_1(\cdot|s) = \sigma\left(f_{\psi_1}\left(\widetilde{A}_1\right)\right). \tag{4.19}$$

Subaction a_2 decides which fragment to concatenate. First, the a_1-st row $\widetilde{a}_1$ of $\widetilde{A}_1$, and the matrix of Morgan fingerprint (MFP) embeddings of fragments $\widetilde{F}^{MFP}$ are combined into a matrix encoding $\widetilde{A}_2 \in \mathbb{R}^{|F| \times d}$:

$$\widetilde{A}_2 = f_{\phi_2}^{MI}\left(\widetilde{a}_1, \widetilde{F}^{MFP}\right). \tag{4.20}$$

Then, a MLP $f_{\psi_2} : \mathbb{R}^{|F| \times d} \to \mathbb{R}^{|F|}$ is used to sample subaction $a_2 \sim \pi_2(\cdot|s, a_1)$:

$$\pi_2(\cdot|s, a_1) = \sigma\left(f_{\psi_2}\left(\widetilde{A}_2\right)\right). \tag{4.21}$$

Subaction a_3 decides which of the fragment's attachment points is bonded to the molecule. First, $\widetilde{a}_2$ is selected the same way as $\widetilde{a}_1$. Then, the attachment points embeddings $\widetilde{F}_{a_2}^{att} \in \mathbb{R}^{|f_{a_2}^{att}| \times d}$ of the selected fragment f_{a_2} with attachment points $f_{a_2}^{att}$ are calculated using the same GCN:

$$\widetilde{F}_{a_2}^{att} \subset \widetilde{F}_{a_2} = G_\eta(f_{a_2}). \tag{4.22}$$

Next, $\widetilde{a}_2$ and $\widetilde{F}_{a_2}^{att}$ are combined into $\widetilde{A}_3 \in \mathbb{R}^{|f_{a_2}^{att}| \times d}$:

$$\widetilde{A}_3 = f_{\phi_3}^{MI}\left(\widetilde{a}_2, \widetilde{F}_{a_2}^{att}\right), \tag{4.23}$$

and, finally, a MLP $f_{\psi_3} : \mathbb{R}^{|f_{a_2}^{att}| \times d} \to \mathbb{R}^{|f_{a_2}^{att}|}$ is used to sample the last subaction $a_3 \sim \pi_3(\cdot|s, a_1, a_2)$:

$$\pi_3(\cdot|s, a_1, a_2) = \sigma\left(f_{\psi_3}\left(\widetilde{A}_3\right)\right). \tag{4.24}$$

Therefore, the model's policy π_θ is:

$$\pi_\theta((a_1, a_2, a_3)|s) = \pi_1(a_1|s)\pi_2(a_2|s, a_1)\pi_3(a_3|s, a_1, a_2). \tag{4.25}$$

Meanwhile, the critic architecture is a MLP $f_\omega : \mathbb{R}^{d \times |V^{att}| \times |F| \times |f_{a_2}^{att}|} \to \mathbb{R}$, taking as input the concatenation of: graph embedding $\widetilde{S}$, probabilities of the first subaction $p_1 = \text{Softmax}\left(f_{\psi_1}\left(\widetilde{A}_1\right)\right)$, one-hot encoding of selected fragment f_{a_2}, and probabilities of the third subaction $p_3 = \text{Softmax}\left(f_{\psi_3}\left(\widetilde{A}_3\right)\right)$:

$$Q_\omega(s, a) = f_\omega\left(\text{cat}\left[\widetilde{S}, p_1, f_{a_2}, p_3\right]\right). \tag{4.26}$$

4.3.3 Soft Actor Critic Training

Soft actor critic (SAC) [48] is an off-policy algorithm used to optimize the stochastic policie defined in FREED. The main idea is to maximize the trade off between expected return and entropy, a measure of randomness in the policy. An optimal policy π^* should try to exploit rewards as best as possible, while being random enough to output diverse molecules:

$$\pi^* = \arg\max_\pi \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{T=4} \gamma^t \left(r(s_t, a_t, s_{t+1}) - \alpha \log \pi(a_t|s_t)\right)\right], \tag{4.27}$$

where α is a variable trade-off coefficient. Given equation (4.27), it can be derived [48] that the Bellman equation for the associated Q-value function Q_ω^π is:

$$Q_\omega^\pi(s_t, a_t) = \mathop{\mathbb{E}}_{\substack{s_{t+1}\sim P \\ \widetilde{a}\sim\pi(\cdot|s_{t+1})}} \left[r(s_t, a_t, s_{t+1}) + \gamma\left(Q_\omega^\pi(s_{t+1}, \widetilde{a}) - \alpha \log \pi(\widetilde{a}|s_{t+1})\right)\right]. \tag{4.28}$$

SAC learns simultaneously the policy π_θ and two equivalent Q-functions $Q_{\omega_1}, Q_{\omega_2}$ (used to smoot the Q-values predictions). Both Q-functions are trained with a MSE loss given by a target value $\hat{Q}_\omega^\pi(s_t, a_t, d)$ obtained with the Bellman equation (4.28):

$$\mathcal{L}(\omega_i, \mathbb{D}) = \mathop{\mathbb{E}}_{(s_t, a_t, s_{t+1}, d)\sim\mathbb{D}}\left[\left(Q_{\omega_i}^\pi(s_t, a_t) - \hat{Q}_\omega^\pi(s_t, a_t, d)\right)^2\right] \tag{4.29}$$

$$\hat{Q}_\omega^\pi(s_t, a_t, d) = r(s_t, a_t, s_{t+1}) + \gamma(1-d)\left(\min_{i=1,2} Q_{\substack{targ \\ \omega_i}}^\pi(s_{t+1}, \widetilde{a}) - \alpha \log \pi(\widetilde{a}|s_{t+1})\right), \widetilde{a} \sim \pi_\theta(\cdot|s_{t+1}), \tag{4.30}$$

where $d = 1$ if s_{t+1} is a terminal state, otherwise $d = 0$.

The policy π_θ is learned by optimizing expected future return plus expected future entropy. Thus, let $\widetilde{a}_\theta(s) \sim \pi_\theta(\cdot|s)$ indicate drawing a sample with the Gumbel-Softmax reparametrization trick, then the SAC algorithm optimizes:

$$\max_\theta \mathop{\mathbb{E}}_{s\sim\mathbb{D}}\left[\min_{i=1,2} Q_{\omega_i}(s, \widetilde{a}_\theta(s)) - \alpha \log \pi_\theta(\widetilde{a}_\theta(s)|s)\right]. \tag{4.31}$$

4.3.4 Issues

As stated in the work by Telepov et al. [142], there are four critical issues in the original implementation of FREED. First, the critic does not receive information about the selected attachment points of the molecule and fragment, as it only gets a vector of probabilities. Therefore, learning an optimal policy for subactions a_1 and a_3 is impossible. To address this, the critic architecture should receive the node embeddings of the chosen attachment points as input:

$$Q_\omega(s,a) = f_\omega\left(\text{cat}\left[\widetilde{S}, \widetilde{V}_{a_1}^{att}, f_{a_2}, \left(\widetilde{F}_{a_2}^{att}\right)_{a_3}\right]\right). \tag{4.32}$$

Furthermore, instead of utilizing the one-hot encoding f_{a_2}, it would be more adequate to use the fragment embedding $\widetilde{F}_{a_2}$.

Second, in the original implementation of FREED, the Q-functions are not updated taking into account the entropy term, which is a critical aspect of the SAC algorithm.

Third, in the original code implementation, the target networks Q_{targ}^π are set to equal the Q^pi networks at every step, leading to the moving target problem, which destabilizes training. Instead, it is preferred to smoothly transition from Q_{targ}^π to Q^π as follows:

$$Q_{targ}^\pi = (1-\tau)Q_{targ}^\pi + \tau Q^\pi, \quad \tau \in (0.001, 0.01). \tag{4.33}$$

Fourth, in the original implementation, the Gumbel-Softmax function is incorrectly evaluated at the probabilities instead of the logits. Moreover, a parameter ν is included without being necessary.

Taking into account these critical issues, along some additional minor problems, the authors in [142] propose a fixed version of FREED called FFREED, which we reimplement inside the original FREED's code.

4.3.5 FREED++

Apart from FFREED, Telepov et al. [142] also propose a model with improved efficiency called FREED++. This model introduces three changes to simplify the FREED framework while maintaining performance. First, the fusing functions $f_{\phi_i}^{MI}$ are replaced by concatenation layers $f_{\phi_i}^{CAT}$ to reduce the number of parameters:

$$f_{\phi_i}^{CAT}(x,z) = Q_i \text{cat}\,[x,z] + b_i, \tag{4.34}$$

where Q_i and b_i are a learnable matrix and bias respectively.

Second, the fragment selection procedure is modified. Subaction a_2 is sampled from a distribution calculated directly from the $\widetilde{a}_1$ representation:

$$\pi_2(\cdot|s,a_1) = \sigma\left(f_{\psi_2}(\widetilde{a}_1)\right), \tag{4.35}$$

and the embedding $\widetilde{a}_2$ is later computed using a concatenation layer over $\widetilde{a}_1$ and the fragment embedding $\widetilde{F}_{a_2}$ obtained from the GCN G_η (rather than using Morgan fingerprints):

$$\widetilde{a}_2 = f_{\phi_2}^{CAT}(\widetilde{a}_1, \widetilde{F}_{a_2}). \tag{4.36}$$

Third, the prioritized experience replay approach used for sampling the replay buffer is removed as it is deemed problematic and more computationally inefficient. Instead, the replay buffer is sampled uniformly.

FREED, FFREED, and FREED++ were compared in the context of PPI modulator generation for the case study in Chapter 5. FREED++ was selected over the others due to its improved performance. The results can be found in the appendix B.

4.4 Taiga

Taiga [99] is a transformer-based model that generates molecules by autoregressively sampling SMILES characters. The attention mechanisms built inside transformers make them well-suited for generating valid SMILES sequences. Taiga is trained in two stages: first, it undergoes pre-training for next-token prediction using a large molecular database; then, it is finetuned using reinforcement learning to optimize the properties of the sampled molecules.

4.4.1 Language Model

Taiga [99] uses a GPT-like [114] decoder architecture trained for next token prediction similar to MolGPT [6]. The model consists of stacked decoder blocks applying the masked self-attention mechanism [145]:

$$Attention(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4.37}$$

where d_k is the dimension of the key vector and Q, K and V represent queries, keys and values respectively. For a single attention head, the tuple (Q, K, V) is calculated from the token embedding matrix X (or from the output of the previous block) as:

$$head_i(X) = Attention\left(XW_i^Q, XW_i^K, XW_i^V,\right), \tag{4.38}$$

where W_i^Q, W_i^K and W_i^V are learnable matrices. Multihead attention (MHA) is achieved by concatenating the outputs of several heads with a learnable matrix W^o:

$$MHA(X) = \text{cat}\left[head_1(X), \cdots, head_n(X)\right]W^o. \tag{4.39}$$

Finally, a transformer decoder block is defined as:

$$Z_l = X_{l-1} + MHA\left(LayerNorm(X_{l-1})\right) \tag{4.40}$$

$$X_l = Z_l + MLP\left(LayerNorm(Z_l)\right), \tag{4.41}$$

where X_{l-1} denotes the output from the previous block. After the last decoder block, an MLP layer with a softmax activation is applied to ouput the next token probabilities.

4.4.2 Reinforcement Learning Setup

Once Taiga is pre-trained on molecular generation, reinforcement learning is applied to optimize the compounds sampled by the model. States are defined by sequences of tokens representing the SMILES code of a molecule, with actions corresponding

to deciding which next token to add. An episode continues until an end of sequence [EOS] token is generated. Rewards are calculated at terminal nodes the same way as FREED (4.14) (utilizing our bioactivity model f_{bio} (3.12)), but with an added -1 penalty for non-terminal states and molecules shorter than 35 tokens, to encourage the sampling of middle sized compounds. Given a state s, the model's policy π_θ is defined as the predicted probability distribution for the next token.

In Taiga [99], the policy is finetuned using the REINFORCE algorithm [156]. The objective is to maximize the expected return J of a trajectory τ under policy π_θ with a discount factor $\gamma \approx 1$:

$$J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\gamma^t r_{t+1}\middle|\pi_\theta\right] = \sum_{\tau}\sum_{t=0}^{T-1}\pi_\theta(a_0,\cdots a_t|\tau)\gamma^t r_{t+1}. \tag{4.42}$$

To do so, the gradient $\nabla_\theta J(\theta)$ needs to be calculated:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{\tau}\sum_{t=0}^{T-1}\nabla_\theta\pi_\theta(a_0,\cdots a_t|\tau)\gamma^t r_{t+1} && (4.43)\\
&= \sum_{\tau}\sum_{t=0}^{T-1}\pi_\theta(a_0,\cdots a_t|\tau)\frac{\nabla_\theta\pi_\theta(a_0,\cdots a_t|\tau)}{\pi_\theta(a_0,\cdots a_t|\tau)}\gamma^t r_{t+1} && (4.44)\\
&= \sum_{\tau}\sum_{t=0}^{T-1}\pi_\theta(a_0,\cdots a_t|\tau)\nabla_\theta\log\pi_\theta(a_0,\cdots a_t|\tau)\gamma^t r_{t+1} && (4.45)\\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_0,\cdots a_t|\tau)\gamma^t r_{t+1}\right]. && (4.46)
\end{aligned}$$

During learning, the expectation (4.46) is approximated by sampling episodes. Since the policy π_θ is independent of time t, the gradient can be simplified to:

$$\begin{aligned}
\nabla_\theta J(\theta) &\sim \sum_{t=0}^{T-1}\gamma^t r_{t+1}\left(\sum_{t'=0}^{t}\nabla_\theta\log\pi_\theta(a_{t'}|s_{t'})\right) && (4.47)\\
&= \sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t|s_t)\left(\sum_{t'=t+1}^{T}\gamma^{t'-t-1}r_{t'}\right). && (4.48)
\end{aligned}$$

After estimating $\nabla_\theta J(\theta)$, Taiga is finetuned following the negative of the gradient with a gradient descent-based optimizer.

Finally, it is worth noting that the RL training of Taiga does not require an entropy term like in FREED. This is the case because the pre-training stage of Taiga ensures that the molecules sampled have enough diversity.

4.5 SyntheMol

SyntheMol [139] is a MCTS-based model that generates synthesizable molecules from a chemical space constructed from a set B of 132,479 real molecular building blocks combined through a set R of 13 different chemical reactions. The model applies MCTS to explore the combinatorial space guided by a property predictor, which in this MFP will be our bioactivity model f_{bio} (3.12).

4.5.1 Monte Carlo Tree Search Implementation

In SyntheMol [139], the MCTS is performed within a tree T that represents all potential synthetic routes formed using building blocks from B and reactions from R. Each node N contains a set of molecules from B or from $R(B)$ (the set of compounds that can be produced with blocks from B and reactions $r \in R$): $N \subset B \cup R(B)$. Moreover, every molecule $m \in N$ must be compatible inside a reaction with the rest of compounds in the node. The tree T is initialized with an empty root node $\emptyset$.

Given a node N, it is expanded into child nodes N_{child} through two mechanisms. First, all reactions $r \in R$ that can be applied to molecules of N are performed, and, for every new compound produced by each reaction r, a child node N_{child} is created. Second, a child node N_{child} is created as $N_{child} = N \cup b$ for each molecular block $b \in B$ that is compatible with the rest of compounds in N.

During generation, SyntheMol runs $n_{rollouts} = 20.000$ rollouts through the chemical tree T until reaching a terminal node N_{ter}, defined as nodes containing a final molecule m produced by $n_{reaction} \in \{1, 2, 3\}$ reactions. The tree T is traversed by selecting at each layer nodes N that maximize the scoring function $S(N)$, composed of four factors:

$$S(N) = \frac{Q(N) + P(N)U(N)}{D(N)}. \tag{4.49}$$

The exploitation factor $Q(N)$ measures the quality of the future terminal nodes of a given node N:

$$Q(N) = \frac{\sum_{m \in \bigcup N_{ter}^{i}} f_{bio}\,(m)}{N_{visit}}, \tag{4.50}$$

where $\bigcup N_{ter}^{i}$ represents the union of all final molecules reachable from the node N, and N_{visit} counts the number of times N has been visited during rollouts. The property prediction factor $P(N)$ measures the mean quality of the molecular blocks that make up node N:

$$P(N) = \frac{\sum_{m \in N} f_{bio}(m)}{|N|}. \tag{4.51}$$

The exploration factor $U(N)$ balances exploration and exploitation by favoring nodes with fewer visits compared to their siblings:

$$U(N) = c \cdot \frac{\sqrt{1 + N_{visit} + \sum_{N' \in N_{siblings}} N'_{visit}}}{1 + N_{visit}}, \tag{4.52}$$

where $c = 10$ is a hyperparameter controlling the trade-off, and $N_{siblings}$ represents the set of nodes with the same parent node as N. The building block diversity factor $D(N)$ discourages selecting nodes made up of compounds that have been produced with overused molecular blocks:

$$D(N) = \exp\left(\frac{\max_{b \in N_B} n_b - 1}{100}\right), \tag{4.53}$$

where N_B represents the set of building blocks that have been used in any molecule in N, and n_b is a counter for the number of times that a block b has been used across all nodes searched.

After exhausting $n_{rollouts}$ rollouts, SyntheMol returns a list of final molecules from searched terminal nodes, which later are passed through our bioactivity and auxiliary filters.

Chapter 5

Case Study: Modulation of NCS-1/Ric-8A and NCS-1/D2R

Neuronal Calcium Sensor-1 (NCS-1) is a highly conserved calcium binding protein which contributes to the maintenance of intracellular calcium homeostasis and regulation of calcium-dependent signaling pathways. NCS-1 is the most abundant protein of the family of Neuronal Calcium Sensors and controls a wide range of important processes in the Central Nervous System due to its ability to recognize and regulate different target proteins, such as G-protein coupled receptors (GPCRs). Interestingly, NCS proteins have key roles in neuronal function and pathologies such as neurodevelopmental disorders, neurodegeneration or cancer. NCS-1 binds to a big number of proteins, such as guanine exchange factor Ric-8A and dopamine 2 receptor (D2R) [101], and dysregulation of these interactions has been linked to neurological disorders such as Parkinson's disease [66], making them attractive targets for therapeutic intervention.

The NCS-1/Ric-8A complex regulates synapse number and neurotransmitter release [119]. Its inhibition reduces synapse number and improves learning in animal models of Fragile X syndrome[1] [22]. By contrast, its stabilization prevents synapse loss and the following movement impairment in a Drosophila model of Alzheimers disease [16]. Meanwhile, in mice, the regulation of D2R surface expression by NCS-1 in the hippocampus is involved in exploration, synaptic plasticity, and memory formation [123].

Thus, we have tested our PPI modulator generation framework by designing molecules for these specific complexes: NCS-1/Ric-8A and NCS-1/D2R. This case study is performed in an extremely low data regime, with only 6 molecules having measured biological activities for the NCS-1/Ric-8A complex and 12 molecules for NCS-1/D2R. Therefore, it is necessary to use a bioactivity model trained on general PPI data, as there is insufficient PPI-specific data to train an affinity predictor.

5.1 Bioactivity Model Experimental Validation

Before delving into the generation of modulators, we first tested the capabilities of our bioactivity model through the task of drug repurposing, which involves searching for

[1] a genetic disorder characterized by intellectual disability

modulators of our target PPIs in a dataset of known safe drugs. We conducted a virtual screening of a chemical library of 2,582 FDA-approved compounds curated from the DrugBank database [70]. We calculated the bioactivity score of each molecule for NCS-1/Ric-8A and NCS-1/D2R. The top 20 ranking molecules for each PPI were used to construct two lists of potential modulators.

Next, we subjected these compounds to docking studies against NCS-1, using the protein structure 5AAN from the Protein Data Bank [9], to verify which compounds presented an acceptable binding. These studies included both AutoDocking, following the procedure described in section 3.6, and traditional docking with Schrödinger's Glide software [128], performed by Dr. Eugenia Ulzurrun at CIB-CSIC[2] through the procedure described in appendix C. Finally, four compounds were selected based on their docking scores, which vary depending on the docking method, and were sent for experimental evaluation.

Experimental assays were performed by Dr. Maria José Sánchez-Barrena's group at IQF-CSIC[3] to determine the affinity of the proposed compounds towards the protein NCS-1. These experiments involved the use fluorescence emission techniques to estimate the dissociation constant (*Kd*) of each molecule at a 5μM protein concentration. One of the selected candidates showed clear affinity towards NCS-1, suggesting that the bioactivity model could effectively identify modulators for our PPIs of interest. However, further experimental testing is required. The results are presented in the following table:

Table 6: Results of the selected FDA compounds

		Bioactivity score ↑		
DrugBank ID	**Name**	**NCS-1/Ric-8A**	**NCS-1/D2R**	**Autodocking Score ↓**
DB00480	Lenalidomide	0.5651	0.2782	−7.3
DB08910	Pomalidomide	0.7078	0.0816	−7.1
DB01041	Thalidomide	0.3593	−0.0193	−7.3
DB00975	Dipyridamole	0.1785	−0.6794	−5.6
DrugBank ID	**Name**	**Glide Docking Score ↓**	***Kd* ↓**	**NCS-1 Binding**
DB00480	Lenalidomide	−5.902	154.42	No
DB08910	Pomalidomide	−5.708	X	Inconclusive
DB01041	Thalidomide	−5.323	X	Not determined yet
DB00975	Dipyridamole	−5.032	39.434	Yes

5.2 PPI Modulators Generation Setup

We employ the PyTorch [107] implementations of the four generative models described in Chapter 4. We use the pre-training weights provided by the authors of each model and finetune them towards *de novo* design of NCS-1/Ric-8A and NCS-1/D2R modulators following the pipeline described in Chapter 2 (Figure 3). All generated molecules during training and sampling are saved and filtered as described in Chapter 3 (refer to table 5 for an overview of the filters used). However, in this case study the binding affinity filter (3.29) is not used to discard compounds due to the manageable number

[2]Centro de Investigaciones Biológicas Margarita Salas
[3]Instituto de Física Química Blas Cabrera

of molecules that pass the other filters; instead, it is applied only as an informative metric.

All generative models use their default hyperparameters except for the target entropy α in FREED++. We chose a value of $\alpha = 5$ (instead of $\alpha = 3$), as the lower entropy resulted in an extremely low diversity of sampled compounds.

5.3 Generation Results and Discussion

Given the aforementioned generation setup, we proceed to sample modulators of NCS-1/Ric-8A and NCS-1/D2R with our generation framework. The number of molecules that are pushed through our filtering pipeline is presented in the following figure:

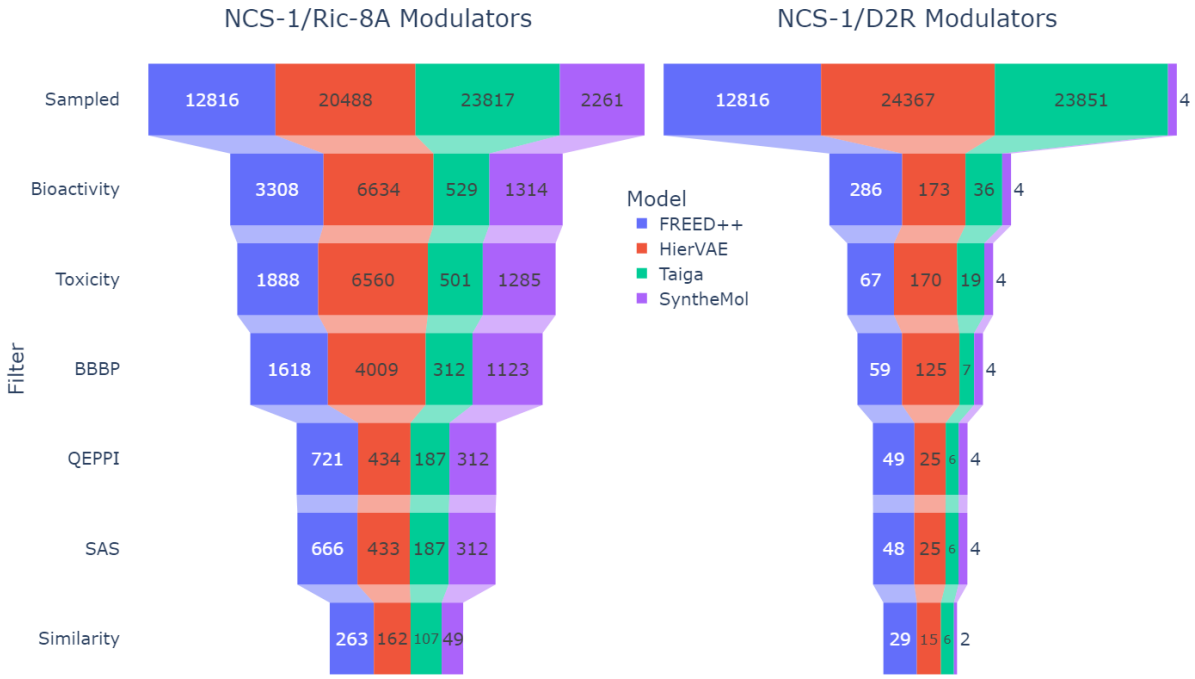


Figure 16: Funnel plots of sampled and remaining modulators after each subsequent filter.

As shown in Figure 16, FREED++ outperforms the rest of models with respect the number of molecules that pass all filters. Meanwhile, SyntheMol presents a low quantity of generated NCS-1/D2R modulators due to its inability to reach non repeated terminal nodes. Moreover, it is worth noting that the bioactivity filter for NCS-1/D2R is much more strict than for NCS-1/Ric-8A. One hypothesis to explain this fact is that NCS-1/D2R training data has lower transformed pChEMBL values on average and with a higher variance than NCS-1/Ric-8A data. However, further work with explainable AI techniques is required to comprehend this phenomenon.

We now analyze the chemical space explored by our generative models during sampling. To do so, we calculate embeddings for all saved molecules using MolFormer [121], a transformer-based large-scale chemical language model developed for molecular representation learning. The 768-dimensional embeddings are then projected

into a 2-dimensional space using the t-SNE [144] dimensionality reduction technique from the scikit-learn [109] Python package.

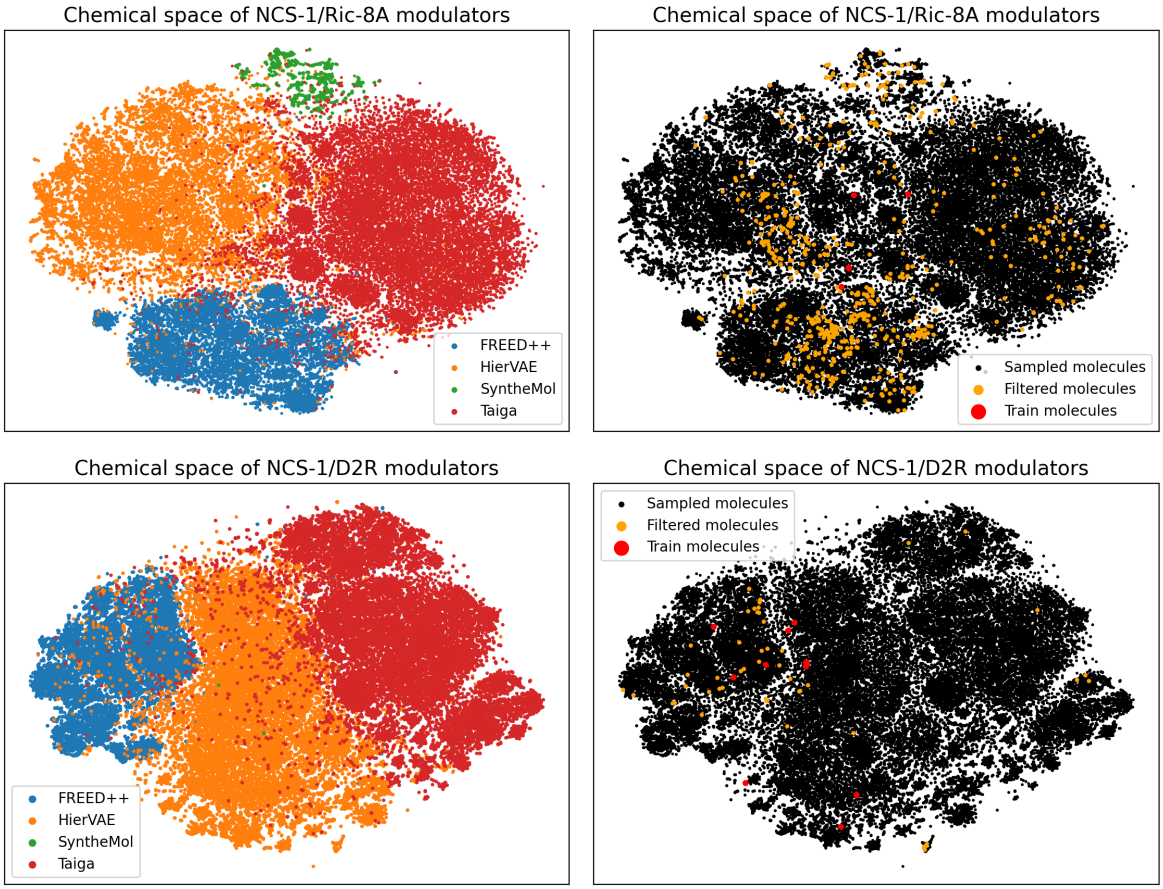


Figure 17: Scatter plots of sampled PPI modulators. Left side plots are divided based on the generative model used. Right side plots are divided based on the compounds that pass all filters and includes the molecules used for training.

Figure 17 shows the distribution of molecules in the chemical space. The left-side plots reveal four distinct clusters (three in the case of NCS-1/D2R modulators due to the low number of samples from SyntheMol). This indicates that each generative model explores different subregions of the chemical space due to their inherent biases, such as the motif vocabularies used in FREED++, HierVAE, and SyntheMol, or the pre-training dataset used in Taiga. The right-side plots show that compounds passing the filters are distributed across all clusters, without a clear preference for any specific region.

We now compare the performance of the generative models with respect to the predicted bioactivity scores of their sampled molecules. We draw violin plots for the bioactivity distribution of the sampled unfiltered molecules. For HierVAE, these plots are based on the sampled molecules from the last epoch of finetuning, while for FREED++ and Taiga, they are based on the last 20% of samples obtained during training.

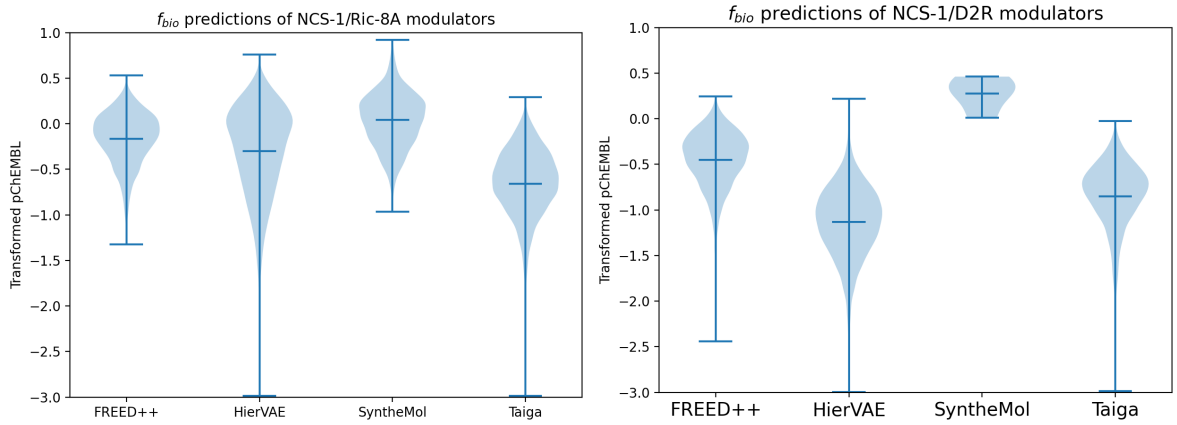


Figure 18: Violin plots of predicted bioactivity distribution of sampled molecules.

Figure 18 shows that SyntheMol outperforms the other models on average. This may happen because SyntheMol directly optimizes bioactivity predictions, which might not completely represent the real-life performance of the sampled molecules. Additionally, the low quantity of sampled molecules by SyntheMol in the NCS-1/D2R context prevents obtaining any statistically significant insights. Among the other models, all show somewhat comparable distributions, with Taiga slightly falling behind the rest.

Given the *de novo* molecules obtained in Figure 16, we aim to construct two virtual chemical libraries of modulators. We combine the filtered molecules and apply the similarity filter once more to remove similar compounds created by different generative models. This results in two sets of 579 and 52 potential modulators for NCS-1/Ric-8A and NCS-1/D2R, respectively.

These libraries are then subjected to docking studies against NCS-1, using the pdb protein structures 5AAN and 6QI4, in a simliar way as in section 5.1 and following the procedure outlined in appendix C. During the ligand preparation stage, several variations of each molecule are generated to account for possible ionization states at physiological pH and, most importantly, undetermined cis-trans isomerisms (i.e., different arrangements of atoms in 3D space that map to the same molecular graph). These variations are again passed through all our filters (except the similarity filter), as they have different chemical properties, leading to the creation of two final virtual chemical libraries with the following sizes:

- **NCS-1/Ric-8A:** 1129 molecules.
- **NCS-1/D2R:** 87 molecules.

The results of the docking studies are shown in Figure 19 and Table 7. A significant percentage of molecules exhibit lower docking scores than the reference ligand of each protein structure in both libraries. The generation framework successfully produces novel molecules with clear *in silico* affinity towards NCS-1, which suggests that there will be successful modulators of NCS-1/Ric-8A and NCS-1/D2R inside the proposed virtual chemical libraries. However, further experimental testing will be carried out to evaluate the properties of some of the new generated molecules.

Table 7: Molecules outperforming the reference ligand in docking score.

Library	5AAN	6QI4
NCS-1/Ric-8A	126 (11.16%)	723 (64.04%)
NCS-1/D2R	16 (18.39%)	64 (73.56%)

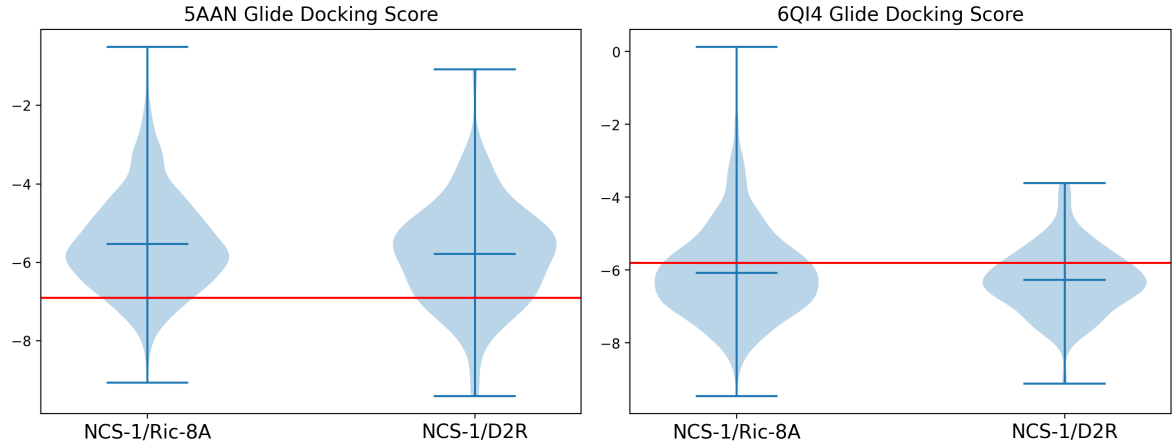


Figure 19: Violin plots of docking scores of the virtual chemical libraries. Red lines represent the docking score of the reference ligand of each protein structure.

Sample figures of the top-performing molecules in the bioactivity predictions for each model in both virtual libraries can be found in appendix D.

Chapter 6

Conclusion

In this MFP, we have developed a comprehensive framework for the *de novo* design of small molecules capable of modulating PPIs. This effort is motivated by the urgent need to accelerate the drug development process, which is traditionally slow and costly, and to address the challenges inherent in targeting PPIs, which are vital for numerous biological functions and represent promising therapeutic targets.

Our framework integrates state-of-the-art deep learning techniques for molecular property prediction and compound generation. One central aspect of this project has been the adaptation of work from the literature towards the development of a novel general bioactivity regression model for any specified PPI. By combining this predictor with a diverse set of molecular generation models and auxiliary property filters, we have demonstrated a powerful approach to exploring the vast chemical space and identifying potential PPI modulators, even in low data regimes.

In the case study, we applied our framework to generate modulators for two PPIs: NCS-1/Ric-8A and NCS-1/D2R. These interactions are linked to neurological disorders, including Parkinson's disease and Alzheimer's disease, highlighting their therapeutic relevance. The results from our case study indicate that our bioactivity prediction model is capable of identifying potential modulators of a specified PPI, as demonstrated by the experiment on affinity towards NCS-1 using a set of FDA-approved molecules screened by our model. Furthermore, the applied generative models effectively explore diverse regions of the chemical space and produce molecules with promising bioactivity profiles and binding affinities estimated through *in silico* docking studies.

While our framework shows promise, there are some limitations that must be acknowledged:

- **Limited experimental validation:** The generated molecules need further experimental validation to confirm their predicted properties and therapeutic potential.
- **Scalability and generalization:** Further testing is required to assess the scalability of our framework across different PPIs.
- **Bias in biological activity data:** Experimental data on bioactivity profiles of PPI modulators can be biased towards the testing and publishing of the most successful modulating molecules.

Moreover, several future directions may be explored to enhance the proposed framework:

- **Integration of protein structural data:** Incorporating explicit protein structural data, rather than relying just on amino acid sequences, could significantly improve the accuracy and robustness of our bioactivity predictions.
- **Integration of molecular conformation data:** As well as with protein data, including the 3D structure of molecular data, as opposed to 2D chemical graphs, could enhance the generation capabilities of the framework.
- **Integration of Explainable Artificial Intelligence (XAI) techniques:** Utilizing XAI techniques to understand the reasoning behind the predictions of our bioactivity, toxicity, and blood-brain barrier penetration models could provide valuable insights. This could lead to improvements in the models and a better understanding of which molecular features are critical for enhancing these properties.
- **Discrimination between inhibiting and stabilizing modulators:** Further curation of training data to develop bioactivity models capable of differentiating between modulators that inhibit or stabilize a given PPI.

6.1 Data Availability

All training data, virtual chemical libraries, and the weights of the finetuned models are available on the following github repository: `https://github.com/Darmator/De-Novo-Design-of-Protein-Protein-Interactions-Modulators`.

Bibliography

[1] "Rdkit: Open-source cheminformatics," https://www.rdkit.org, [Online; accessed 17-May-2024].

[2] *Quasi-Newton Methods*. New York, NY: Springer New York, 2006, pp. 135–163. [Online]. Available: https://doi.org/10.1007/978-0-387-40065-5_6

[3] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 214–223. [Online]. Available: https://proceedings.mlr.press/v70/arjovsky17a.html

[4] S. R. Atance, J. V. Diez, O. Engkvist, S. Olsson, and R. Mercado, "De novo drug design using reinforcement learning with graph-based deep generative models," *Journal of Chemical Information and Modeling*, vol. 62, no. 20, pp. 4863–4872, 2022. [Online]. Available: https://doi.org/10.1021/acs.jcim.2c00838

[5] S. Axelrod and R. Gómez-Bombarelli, "Geom, energy-annotated molecular conformations for property prediction and molecular generation," *Scientific Data*, vol. 9, no. 1, p. 185, 2022. [Online]. Available: https://doi.org/10.1038/s41597-022-01288-4

[6] V. Bagal, R. Aggarwal, P. K. Vinod, and U. D. Priyakumar, "MolGPT: Molecular generation using a transformer-decoder model," *Journal of Chemical Information and Modeling*, vol. 62, no. 9, pp. 2064–2076, 2022.

[7] P. Bai, F. Miljković, B. John, and H. Lu, "Interpretable bilinear attention network with domain adaptation improves drug–target prediction," *Nature Machine Intelligence*, vol. 5, no. 2, pp. 126–136, 2023. [Online]. Available: https://doi.org/10.1038/s42256-022-00605-1

[8] D. Bajusz, A. Rácz, and K. Héberger, "Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations?" *Journal of Cheminformatics*, vol. 7, no. 1, p. 20, 2015. [Online]. Available: https://doi.org/10.1186/s13321-015-0069-3

[9] 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*, vol. 28, no. 1, pp. 235–242, 2000. [Online]. Available: https://doi.org/10.1093/nar/28.1.235

[10] G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan, and A. L. Hopkins, "Quantifying the chemical beauty of drugs," *Nature Chemistry*, vol. 4, no. 2, pp. 90–98, 2012. [Online]. Available: https://doi.org/10.1038/nchem.1243

[11] C. Bilodeau, W. Jin, T. Jaakkola, R. Barzilay, and K. F. Jensen, "Generative models for molecular discovery: Recent advances and challenges," *WIREs Computational Molecular Science*, vol. 12, no. 5, p. e1608, 2022. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1608

[12] T. Blaschke, J. Arús-Pous, H. Chen, C. Margreitter, C. Tyrchan, O. Engkvist, K. Papadopoulos, and A. Patronov, "REINVENT 2.0: An AI tool for de novo drug design," *Journal of Chemical Information and Modeling*, vol. 60, no. 12, pp. 5918–5922, 2020. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c00915

[13] T. Blaschke, O. Engkvist, J. Bajorath, and H. Chen, "Memory-assisted reinforcement learning for diverse molecular de novo design," *Journal of Cheminformatics*, vol. 12, no. 1, p. 68, 2020. [Online]. Available: https://doi.org/10.1186/s13321-020-00473-0

[14] J. Born and M. Manica, "Regression transformer enables concurrent sequence regression and generation for molecular language modelling," *Nature Machine Intelligence*, vol. 5, no. 4, pp. 432–444, 2023. [Online]. Available: https://doi.org/10.1038/s42256-023-00639-z

[15] G. E. Box and D. R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 26, no. 2, pp. 211–243, 1964.

[16] A. Canal-Martín, J. Sastre, M. J. Sánchez-Barrena, A. Canales, S. Baldominos, N. Pascual, L. Martínez-González, D. Molero, M. E. Fernández-Valle, E. Sáez, P. Blanco-Gabella, E. Gómez-Rubio, S. Martín-Santamaría, A. Sáiz, A. Mansilla, F. J. Cañada, J. Jiménez-Barbero, A. Martínez, and R. Pérez-Fernández, "Insights into real-time chemical processes in a calcium sensor protein-directed dynamic library," *Nature Communications*, vol. 10, no. 1, p. 2798, 2019. [Online]. Available: https://doi.org/10.1038/s41467-019-10627-w

[17] N. D. Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," *arXiv preprint arXiv:1805.11973*, 2022.

[18] B. Chen, G. Bécigneul, O.-E. Ganea, R. Barzilay, and T. Jaakkola, "Optimal transport graph neural networks," *arXiv preprint arXiv:2006.04804*, 2020.

[19] Y. Chen, Z. Wang, L. Wang, J. Wang, P. Li, D. Cao, X. Zeng, X. Ye, and T. Sakurai, "Deep generative model for drug design from protein target sequence," *Journal of Cheminformatics*, vol. 15, no. 1, p. 38, 2023. [Online]. Available: https://doi.org/10.1186/s13321-023-00702-2

[20] Y. Cheng, Y. Gong, Y. Liu, B. Song, and Q. Zou, "Molecular design in drug discovery: a comprehensive review of deep generative models," *Briefings in Bioinformatics*, vol. 22, no. 6, p. bbab344, 2021. [Online]. Available: https://doi.org/10.1093/bib/bbab344

[21] V. Chenthamarakshan, P. Das, S. Hoffman, H. Strobelt, I. Padhi, K. W. Lim, B. Hoover, M. Manica, J. Born, T. Laino, and A. Mojsilovic, "CogMol: Target-specific and selective drug design for covid-19 using deep generative models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 4320–4332. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/2d16ad1968844a4300e9a490588ff9f8-Paper.pdf

[22] P. Cogram, L. C. Fernández-Beltrán, M. J. Casarejos, S. Sánchez-Yepes, E. Rodríguez-Martín, A. García-Rubia, M. J. Sánchez-Barrena, C. Gil, A. Martínez, and A. Mansilla, "The inhibition of NCS-1 binding to Ric8a rescues fragile X syndrome mice model phenotypes," *Frontiers in Neuroscience*, vol. 16, 2022. [Online]. Available: http://dx.doi.org/10.3389/fnins.2022.1007531

[23] T. U. Consortium, "UniProt: the Universal Protein Knowledgebase in 2023," *Nucleic Acids Research*, vol. 51, no. D1, pp. D523–D531, 2022. [Online]. Available: https://doi.org/10.1093/nar/gkac1052

[24] D. Dablain, G. Siwo, and N. Chawla, "Generative ai design and exploration of nucleoside analogs," 2021.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:52967399

[26] O. Dollar, N. Joshi, D. A. C. Beck, and J. Pfaendtner, "Attention-based generative models for de novo molecular design," *Chem. Sci.*, vol. 12, pp. 8362–8372, 2021. [Online]. Available: http://dx.doi.org/10.1039/D1SC01050F

[27] P. Ertl and A. Schuffenhauer, "Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions," *Journal of Cheminformatics*, vol. 1, no. 1, p. 8, 2009. [Online]. Available: https://doi.org/10.1186/1758-2946-1-8

[28] Y. Fang, N. Zhang, Z. Chen, X. Fan, and H. Chen, "Domain-agnostic molecular generation with chemical feedback," in *ICLR*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/pdf?id=9rPyHyjfwP

[29] S. S. Farid, M. Baron, C. Stamatis, W. Nie, and J. Coffman, "Benchmarking biopharmaceutical process development and manufacturing cost contributions to R&D," *mAbs*, vol. 12, no. 1, p. 1754999, 2020.

[30] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[31] M. Foscato, V. Venkatraman, and V. R. Jensen, "DENOPTIM: Software for computational de novo design of organic and inorganic molecules," *Journal of Chemical Information and Modeling*, vol. 59, no. 10, pp. 4077–4082, 2019. [Online]. Available: https://doi.org/10.1021/acs.jcim.9b00516

[32] T. Fu, W. Gao, C. Coley, and J. Sun, "Reinforced genetic algorithm for structure-based drug design," in *Advances in Neural Information Processing*

Systems, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 12 325–12 338. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/4fe1859112230a032c7143a9adc3be78-Paper-Conference.pdf

[33] M. Gao, L. Zhao, Z. Zhang, J. Wang, and C. Wang, "Using a stacked ensemble learning framework to predict modulators of protein-protein interactions," *Computers in Biology and Medicine*, vol. 161, p. 107032, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482523004973

[34] M. García-Ortegón, G. N. C. Simm, A. J. Tripp, J. M. Hernández-Lobato, A. Bender, and S. Bacallado, "Dockstring: Easy molecular docking yields better benchmarks for ligand design," *Journal of Chemical Information and Modeling*, vol. 62, no. 15, pp. 3486–3502, 2022. [Online]. Available: https://doi.org/10.1021/acs.jcim.1c01334

[35] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, and J. P. Overington, "ChEMBL: a large-scale bioactivity database for drug discovery," *Nucleic Acids Research*, vol. 40, no. D1, pp. D1100–D1107, 2011. [Online]. Available: https://doi.org/10.1093/nar/gkr777

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, p. 139144, 2020. [Online]. Available: https://doi.org/10.1145/3422622

[37] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf

[38] D. Grechishnikova, "Transformer neural network for protein-specific de novo drug generation as a machine translation problem," *Scientific Reports*, vol. 11, no. 1, p. 321, 2021. [Online]. Available: https://doi.org/10.1038/s41598-020-79682-4

[39] R.-R. Griffiths and J. M. Hernández-Lobato, "Constrained bayesian optimization for automatic chemical design using variational autoencoders," *Chem. Sci.*, vol. 11, pp. 577–586, 2020. [Online]. Available: http://dx.doi.org/10.1039/C9SC04026A

[40] F. Grisoni, M. Moret, R. Lingwood, and G. Schneider, "Bidirectional molecule generation with recurrent neural networks," *Journal of Chemical Information and Modeling*, vol. 60, no. 3, pp. 1175–1183, 2020. [Online]. Available: https://doi.org/10.1021/acs.jcim.9b00943

[41] J. Guan, W. W. Qian, X. Peng, Y. Su, J. Peng, and J. Ma, "3d equivariant diffusion for target-aware molecule generation and affinity prediction," in *International Conference on Learning Representations*, 2023.

[42] 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," 2018.

[43] J. Guo, V. Fialková, J. D. Arango, C. Margreitter, J. P. Janet, K. Papadopoulos, O. Engkvist, and A. Patronov, "Improving de novo molecular design with curriculum learning," *Nature Machine Intelligence*, vol. 4, no. 6, pp. 555–563, 2022. [Online]. Available: https://doi.org/10.1038/s42256-022-00494-4

[44] A. Gupta, A. T. Müller, B. J. H. Huisman, J. A. Fuchs, P. Schneider, and G. Schneider, "Generative recurrent networks for de novo drug design," *Molecular Informatics*, vol. 37, no. 12, 2017. [Online]. Available: http://dx.doi.org/10.1002/minf.201700111

[45] A. Gupta and J. Zou, "Feedback GAN for DNA optimizes protein functions," *Nature Machine Intelligence*, vol. 1, no. 2, pp. 105–111, 2019. [Online]. Available: https://doi.org/10.1038/s42256-019-0017-4

[46] P. Gupta and D. Mohanty, "SMMPPI: a machine learning-based approach for prediction of modulators of protein-protein interactions and its application for identification of novel inhibitors for RBD:hACE2 interactions in SARS-CoV-2," *Briefings in Bioinformatics*, vol. 22, no. 5, p. bbab111, 2021. [Online]. Available: https://doi.org/10.1093/bib/bbab111

[47] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS Central Science*, vol. 4, no. 2, pp. 268–276, 2018, pMID: 29532027. [Online]. Available: https://doi.org/10.1021/acscentsci.7b00572

[48] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[49] E. Heid, K. P. Greenman, Y. Chung, S.-C. Li, D. E. Graff, F. H. Vermeire, H. Wu, W. H. Green, and C. J. McGill, "Chemprop: A machine learning package for chemical property prediction," *Journal of Chemical Information and Modeling*, vol. 64, no. 1, pp. 9–17, 2024. [Online]. Available: https://doi.org/10.1021/acs.jcim.3c01250

[50] S. Honda, H. Akita, K. Ishiguro, T. Nakanishi, and K. Oono, "Graph residual flow for molecular graph generation," *arXiv preprint arXiv:1909.13521*, 2019.

[51] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling, "Equivariant diffusion for molecule generation in 3D," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 8867–8887. [Online]. Available: https://proceedings.mlr.press/v162/hoogeboom22a.html

[52] X. Hu, G. Liu, Y. Zhao, and H. Zhang, "De novo drug design using reinforcement learning with multiple GPT agents," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[53] L. Huang, H. Zhang, T. Xu, and K.-C. Wong, "MDM: Molecular diffusion model for 3d molecule generation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, pp. 5105–5112, 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/25639

[54] I. Igashov, H. Stärk, C. Vignac, A. Schneuing, V. G. Satorras, P. Frossard, M. Welling, M. Bronstein, and B. Correia, "Equivariant 3d-conditional diffusion model for molecular linker design," *Nature Machine Intelligence*, vol. 6, no. 4, pp. 417–427, 2024. [Online]. Available: https://doi.org/10.1038/s42256-024-00815-9

[55] S. Ishida, T. Aasawat, M. Sumita, M. Katouda, T. Yoshizawa, K. Yoshizoe, K. Tsuda, and K. Terayama, "ChemTSv2: Functional molecular design using de novo molecule generator," *WIREs Computational Molecular Science*, vol. 13, no. 6, p. e1680, 2023. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1680

[56] H. Iwata, T. Nakai, T. Koyama, S. Matsumoto, R. Kojima, and Y. Okuno, "VGAE-MCTS: A new molecular generative model combining the variational graph auto-encoder and monte carlo tree search," *Journal of Chemical Information and Modeling*, vol. 63, no. 23, pp. 7392–7400, 2023. [Online]. Available: https://doi.org/10.1021/acs.jcim.3c01220

[57] M. P. Jacobson, D. L. Pincus, C. S. Rapp, T. J. Day, B. Honig, D. E. Shaw, and R. A. Friesner, "A hierarchical approach to all-atom protein loop prediction," *Proteins: Structure, Function, and Bioinformatics*, vol. 55, no. 2, pp. 351–367, 2004.

[58] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[59] J. H. Jensen, "A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space," *Chem. Sci.*, vol. 10, pp. 3567–3572, 2019. [Online]. Available: http://dx.doi.org/10.1039/C8SC05372C

[60] W. Jeon and D. Kim, "Autonomous molecule generation using reinforcement learning and docking to develop potential novel inhibitors," *Scientific Reports*, vol. 10, no. 1, p. 22104, 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-78537-2

[61] Y. Jiang, G. Zhang, J. You, H. Zhang, R. Yao, H. Xie, L. Zhang, Z. Xia, M. Dai, Y. Wu, L. Li, and S. Yang, "PocketFlow is a data-and-knowledge-driven structure-based molecular generative model," *Nature Machine Intelligence*, vol. 6, no. 3, pp. 326–337, 2024. [Online]. Available: https://doi.org/10.1038/s42256-024-00808-8

[62] J. Jin, D. Wang, G. Shi, J. Bao, J. Wang, H. Zhang, P. Pan, D. Li, X. Yao, H. Liu, T. Hou, and Y. Kang, "FFLOM: A flow-based autoregressive model for fragment-to-lead optimization," *Journal of Medicinal Chemistry*, vol. 66, no. 15, pp. 10 808–10 823, 2023. [Online]. Available: https://doi.org/10.1021/acs.jmedchem.3c01009

[63] W. Jin, D. Barzilay, and T. Jaakkola, "Hierarchical generation of molecular graphs using structural motifs," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 2020, pp. 4839–4848. [Online]. Available: https://proceedings.mlr.press/v119/jin20a.html

[64] ——, "Multi-objective molecule generation using interpretable substructures," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4849–4859. [Online]. Available: https://proceedings.mlr.press/v119/jin20b.html

[65] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 2323–2332. [Online]. Available: https://proceedings.mlr.press/v80/jin18a.html

[66] J. Jones-Tabah, "Targeting G protein-coupled receptors in the treatment of parkinsons disease," *Journal of Molecular Biology*, vol. 435, no. 12, p. 167927, 2023, molecular Mechanisms of Neurodegeneration in Parkinsons Disease. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022283622005540X

[67] A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, and A. Zhavoronkov, "druGAN: An advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico," *Molecular Pharmaceutics*, vol. 14, no. 9, pp. 3098–3104, 2017. [Online]. Available: https://doi.org/10.1021/acs.molpharmaceut.7b00346

[68] M. Kaushalya, I. Katushiko, N. Kosuke, and A. Motoki, "GraphNVP: An invertible flow model for generating molecular graphs," *arXiv preprint arXiv:1905.11600*, 2019.

[69] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[70] C. Knox, M. Wilson, C. M. Klinger, M. Franklin, E. Oler, A. Wilson, A. Pon, J. Cox, N. E. L. Chin, S. A. Strawbridge, M. Garcia-Patino, R. Kruger, A. Sivakumaran, S. Sanford, R. Doshi, N. Khetarpal, O. Fatokun, D. Doucet, A. Zubkowski, D. Y. Rayat, H. Jackson, K. Harford, A. Anjum, M. Zakir, F. Wang, S. Tian, B. Lee, J. Liigand, H. Peters, R. Q. R. Wang, T. Nguyen, D. So, M. Sharp, R. da Silva, C. Gabriel, J. Scantlebury, M. Jasinski, D. Ackerman, T. Jewison, T. Sajed, V. Gautam, and D. S. Wishart, "DrugBank 6.0: the DrugBank Knowledgebase for 2024," *Nucleic Acids Research*, vol. 52, no. D1, pp. D1265–D1275, 2023. [Online]. Available: https://doi.org/10.1093/nar/gkad976

[71] T. Kosugi and M. Ohue, "Quantitative estimate index for early-stage screening of compounds targeting protein-protein interactions," *International Journal of Molecular Sciences*, vol. 22, no. 20, 2021. [Online]. Available: https://www.mdpi.com/1422-0067/22/20/10925

[72] P.-C. Kotsias, J. Arús-Pous, H. Chen, O. Engkvist, C. Tyrchan, and E. J. Bjerrum, "Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks," *Nature Machine Intelligence*, vol. 2, no. 5, pp. 254–265, 2020. [Online]. Available: https://doi.org/10.1038/s42256-020-0174-5

[73] M. Krenn, F. Häse, A. Nigam, P. Friederich, and A. Aspuru-Guzik, "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation," *Machine Learning: Science and Technology*, vol. 1, no. 4, p. 045024, 2020. [Online]. Available: https://dx.doi.org/10.1088/2632-2153/aba947

[74] R. Krishnan, P. Esposito, and M. Subedar, "Bayesian-Torch: Bayesian neural network layers for uncertainty estimation," https://github.com/IntelLabs/bayesian-torch, 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5908307

[75] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79 – 86, 1951. [Online]. Available: https://doi.org/10.1214/aoms/1177729694

[76] Y. Kwon, S. Kang, Y.-S. Choi, and I. Kim, "Evolutionary design of molecules based on deep learning and a genetic algorithm," *Scientific Reports*, vol. 11, no. 1, p. 17304, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-96812-8

[77] S. Lee, J. Jo, and S. J. Hwang, "Exploring chemical space with score-based out-of-distribution generation," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 2023, pp. 18872–18892. [Online]. Available: https://proceedings.mlr.press/v202/lee23f.html

[78] C. Li and Y. Yamanishi, "Spotgan: A reverse-transformer GAN generates scaffold-constrained molecules with property optimization," in *Machine Learning and Knowledge Discovery in Databases: Research Track*, D. Koutra, C. Plant, M. Gomez Rodriguez, E. Baralis, and F. Bonchi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 323–338.

[79] ——, "TenGAN: Pure transformer encoders make an efficient discrete GAN for de novo molecular generation," in *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Dasgupta, S. Mandt, and Y. Li, Eds., vol. 238. PMLR, 2024, pp. 361–369. [Online]. Available: https://proceedings.mlr.press/v238/li24d.html

[80] P. Li, Y. Li, C.-Y. Hsieh, S. Zhang, X. Liu, H. Liu, S. Song, and X. Yao, "TrimNet: learning molecular representation from triplet messages for biomedicine," *Briefings in Bioinformatics*, vol. 22, no. 4, p. bbaa266, 2020. [Online]. Available: https://doi.org/10.1093/bib/bbaa266

[81] Y. Li, C.-Y. Hsieh, R. Lu, X. Gong, X. Wang, P. Li, S. Liu, Y. Tian, D. Jiang, J. Yan, Q. Bai, H. Liu, S. Zhang, and X. Yao, "An adaptive graph learning method for automated molecular interactions and properties predictions,"

Nature Machine Intelligence, vol. 4, no. 7, pp. 645–651, 2022. [Online]. Available: https://doi.org/10.1038/s42256-022-00501-8

[82] J. Lim, S. Ryu, J. W. Kim, and W. Y. Kim, "Molecular generative model based on conditional variational autoencoder for de novo molecular design," *Journal of Cheminformatics*, vol. 10, no. 1, p. 31, 2018. [Online]. Available: https://doi.org/10.1186/s13321-018-0286-7

[83] ——, "Molecular generative model based on conditional variational autoencoder for de novo molecular design," *Journal of Cheminformatics*, vol. 10, no. 1, p. 31, 2018. [Online]. Available: https://doi.org/10.1186/s13321-018-0286-7

[84] H. Lin, Y. Huang, M. Liu, X. Li, S. Ji, and S. Z. Li, "Diffbp: Generative diffusion of 3d molecules for target protein binding," *arXiv preprint arXiv:2211.11214*, 2022.

[85] H. Lin, Y. Huang, O. Zhang, Y. Liu, L. Wu, S. Li, Z. Chen, and S. Z. Li, "Functional-group-based diffusion for pocket-specific molecule generation and elaboration," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 34 603–34 626. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/6cdd4ce9330025967dd1ed0bed3010f5-Paper-Conference.pdf

[86] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, A. dos Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, and A. Rives, "Evolutionary-scale prediction of atomic-level protein structure with a language model," *Science*, vol. 379, no. 6637, pp. 1123–1130, 2023. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.ade2574

[87] Z. Lin, Y. Zhang, L. Duan, L. Ou-Yang, and P. Zhao, "Movae: a variational autoencoder for molecular graph generation," in *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM, 2023, pp. 514–522.

[88] Q. Liu, M. Allamanis, M. Brockschmidt, and A. L. Gaunt, "Constrained graph variational autoencoders for molecule design," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 78067815.

[89] S. Liu, H. Wang, W. Liu, J. Lasenby, H. Guo, and J. Tang, "Pre-training molecular graph representation with 3d geometry," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=xQUe1pOKPam

[90] X. Liu, K. Ye, H. W. T. van Vlijmen, A. P. IJzerman, and G. J. P. van Westen, "Drugex v3: scaffold-constrained drug design with graph transformer-based reinforcement learning," *Journal of Cheminformatics*, vol. 15, no. 1, p. 24, 2023. [Online]. Available: https://doi.org/10.1186/s13321-023-00694-z

[91] H. H. Loeffler, J. He, A. Tibo, J. P. Janet, A. Voronov, L. H. Mervin, and O. Engkvist, "Reinvent 4: Modern AI–driven generative molecule design," *Journal of Cheminformatics*, vol. 16, no. 1, p. 20, 2024. [Online]. Available: https://doi.org/10.1186/s13321-024-00812-5

[92] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated Local Search: Framework and Applications*. Boston, MA: Springer US, 2010, pp. 363–397. [Online]. Available: https://doi.org/10.1007/978-1-4419-1665-5_12

[93] 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*, vol. 5, no. 1, p. 213, 2020.

[94] H. Lu, Z. Wei, X. Wang, K. Zhang, and H. Liu, "GraphGPT: A graph enhanced generative pretrained transformer for conditioned molecular generation," *International Journal of Molecular Sciences*, vol. 24, no. 23, 2023. [Online]. Available: https://www.mdpi.com/1422-0067/24/23/16761

[95] Y. Luo, K. Yan, and S. Ji, "GraphDF: A discrete flow model for molecular graph generation," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 7192–7203. [Online]. Available: https://proceedings.mlr.press/v139/luo21a.html

[96] I. F. Martins, A. L. Teixeira, L. Pinheiro, and A. O. Falcao, "A bayesian approach to in silico blood-brain barrier penetration modeling," *Journal of Chemical Information and Modeling*, vol. 52, no. 6, pp. 1686–1697, Jun 2012. [Online]. Available: https://doi.org/10.1021/ci300124c

[97] A. Mauri and M. Bertola, "AlvaBuilder: A software for de novo molecular design," *Journal of Chemical Information and Modeling*, vol. 64, no. 7, pp. 2136–2142, 2024. [Online]. Available: https://doi.org/10.1021/acs.jcim.3c00610

[98] Ł. Maziarka, A. Pocha, J. Kaczmarczyk, K. Rataj, T. Danel, and M. Warchoł, "Mol-CycleGAN: a generative model for molecular optimization," *Journal of Cheminformatics*, vol. 12, no. 1, p. 2, 2020. [Online]. Available: https://doi.org/10.1186/s13321-019-0404-1

[99] E. Mazuz, G. Shtar, B. Shapira, and L. Rokach, "Molecule generation using transformers and policy gradient reinforcement learning," *Scientific Reports*, vol. 13, no. 1, p. 8799, 2023.

[100] M. Moret, I. Pachon Angona, L. Cotos, S. Yan, K. Atz, C. Brunner, M. Baumgartner, F. Grisoni, and G. Schneider, "Leveraging molecular structure and bioactivity with chemical language models for de novo drug design," *Nature Communications*, vol. 14, no. 1, p. 114, 2023. [Online]. Available: https://doi.org/10.1038/s41467-022-35692-6

[101] T. Y. Nakamura, S. Nakao, and S. Wakabayashi, "Emerging roles of neuronal ca2+ sensor-1 in cardiac and neuronal tissues: A mini review," *Front. Mol. Neurosci.*, vol. 12, p. 56, 2019.

[102] M. Ohue, Y. Kojima, and T. Kosugi, "Generating potential protein-protein interaction inhibitor molecules based on physicochemical properties," *Molecules*, vol. 28, no. 15, 2023. [Online]. Available: https://www.mdpi.com/1420-3049/28/15/5652

[103] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *Journal of Cheminformatics*,

vol. 9, no. 1, p. 48, 2017. [Online]. Available: https://doi.org/10.1186/s13321-017-0235-x

[104] ——, "Molecular de-novo design through deep reinforcement learning," *Journal of Cheminformatics*, vol. 9, no. 1, p. 48, 2017. [Online]. Available: https://doi.org/10.1186/s13321-017-0235-x

[105] M. H. Olsson, C. R. Søndergaard, M. Rostkowski, and J. H. Jensen, "PROPKA3: consistent treatment of internal and surface residues in empirical pka predictions," *Journal of Chemical Theory and Computation*, vol. 7, no. 2, pp. 525–537, 2011.

[106] C. Pang, J. Qiao, X. Zeng, Q. Zou, and L. Wei, "Deep generative models in de novo drug molecule generation," *Journal of Chemical Information and Modeling*, vol. 64, no. 7, pp. 2174–2194, 2024. [Online]. Available: https://doi.org/10.1021/acs.jcim.3c01496

[107] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[108] M. Paykan Heyrati, Z. Ghorbanali, M. Akbari, G. Pishgahi, and F. Zare-Mirakabad, "BioAct-Het: A heterogeneous siamese neural network for bioactivity prediction using novel bioactivity representation," *ACS Omega*, vol. 8, no. 47, pp. 44 757–44 772, 2023. [Online]. Available: https://doi.org/10.1021/acsomega.3c05778

[109] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[110] X. Peng, J. Guan, Q. Liu, and J. Ma, "MolDiff: Addressing the atom-bond inconsistency problem in 3D molecule diffusion generation," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 2023, pp. 27 611–27 629. [Online]. Available: https://proceedings.mlr.press/v202/peng23b.html

[111] X. Peng, S. Luo, J. Guan, Q. Xie, J. Peng, and J. Ma, "Pocket2mol: Efficient molecular sampling based on 3d protein pockets," in *International Conference on Machine Learning*. PMLR, 2022, pp. 17 644–17 655.

[112] P. G. Polishchuk, T. I. Madzhidov, and A. Varnek, "Estimation of the size of drug-like chemical space based on GDB-17 data," *Journal of Computer-Aided Molecular Design*, vol. 27, no. 8, pp. 675–679, 2013. [Online]. Available: https://doi.org/10.1007/s10822-013-9672-4

[113] 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*, vol. 11, no. 1, p. 74, 2019. [Online]. Available: https://doi.org/10.1186/s13321-019-0397-9

[114] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[115] A. M. Richard, R. Huang, S. Waidyanatha, P. Shinn, B. J. Collins, I. Thillainadarajah, C. M. Grulke, A. J. Williams, R. R. Lougee, R. S. Judson, K. A. Houck, M. Shobair, C. Yang, J. F. Rathman, A. Yasgar, S. C. Fitzpatrick, A. Simeonov, R. S. Thomas, K. M. Crofton, R. S. Paules, J. R. Bucher, C. P. Austin, R. J. Kavlock, and R. R. Tice, "The Tox21 10k compound library: Collaborative chemistry advancing toxicology," *Chemical Research in Toxicology*, vol. 34, no. 2, pp. 189–216, 2021. [Online]. Available: https://doi.org/10.1021/acs.chemrestox.0c00264

[116] A. M. Richard, R. S. Judson, K. A. Houck, C. M. Grulke, P. Volarath, I. Thillainadarajah, C. Yang, J. Rathman, M. T. Martin, J. F. Wambaugh, T. B. Knudsen, J. Kancherla, K. Mansouri, G. Patlewicz, A. J. Williams, S. B. Little, K. M. Crofton, and R. S. Thomas, "ToxCast chemical landscape: Paving the road to 21st century toxicology," *Chemical Research in Toxicology*, vol. 29, no. 8, pp. 1225–1251, 2016. [Online]. Available: https://doi.org/10.1021/acs.chemrestox.6b00135

[117] C. H. M. Rodrigues, D. E. V. Pires, and D. B. Ascher, "pdCSM-PPI: Using graph-based signatures to identify protein-protein interaction inhibitors," *Journal of Chemical Information and Modeling*, vol. 61, no. 11, pp. 5438–5445, 2021.

[118] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[119] J. Romero-Pozuelo, J. S. Dason, A. Mansilla, S. Baños-Mateos, J. L. Sardina, A. Chaves-Sanjuán, J. Jurado-Gómez, E. Santana, H. L. Atwood, . Hernández-Hernández, M.-J. Sánchez-Barrena, and A. Ferrús, "The guanine-exchange factor Ric8a binds the calcium sensor NCS-1 to regulate synapse number and probability of release," *Journal of Cell Science*, 2014. [Online]. Available: http://dx.doi.org/10.1242/jcs.152603

[120] J. Ross, B. Belgodere, V. Chenthamarakshan, I. Padhi, Y. Mroueh, and P. Das, "Large-scale chemical language representations capture molecular structure and properties," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1256–1264, 2022. [Online]. Available: https://doi.org/10.1038/s42256-022-00580-7

[121] ——, "Large-scale chemical language representations capture molecular structure and properties," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1256–1264, 2022. [Online]. Available: https://doi.org/10.1038/s42256-022-00580-7

[122] E. Rozenberg and D. Freedman, "Semi-equivariant conditional normalizing flows, with applications to target-aware molecule generation," *Machine Learning: Science and Technology*, vol. 4, no. 3, p. 035037, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1088/2632-2153/ace58c

[123] B. J. Saab, J. Georgiou, A. Nath, F. J. Lee, M. Wang, A. Michalon, F. Liu, I. M. Mansuy, and J. C. Roder, "NCS-1 in the dentate gyrus promotes exploration, synaptic plasticity, and rapid acquisition of spatial memory," *Neuron*, vol. 63, no. 5, pp. 643–656, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0896627309006266

[124] B. Samanta, A. De, G. Jana, V. Gómez, P. Chattaraj, N. Ganguly, and M. Gomez-Rodriguez, "NEVAE: A deep generative model for molecular graphs," *Journal of Machine Learning Research*, vol. 21, no. 114, pp. 1–33, 2020. [Online]. Available: http://jmlr.org/papers/v21/19-671.html

[125] M. V. S. Santana and F. P. Silva-Jr, "De novo design and bioactivity prediction of sars-cov-2 main protease inhibitors using recurrent neural network-based transfer learning," *BMC Chemistry*, vol. 15, no. 1, p. 8, 2021. [Online]. Available: https://doi.org/10.1186/s13065-021-00737-2

[126] A. Schneuing, Y. Du, C. Harris, A. Jamasb, I. Igashov, W. Du, T. Blundell, P. Lió, C. Gomes, M. Welling, M. Bronstein, and B. Correia, "Structure-based drug design with equivariant diffusion models," *arXiv preprint arXiv:2210.13695*, 2022.

[127] Schrödinger, LLC, *Schrödinger Release 2023-2: Epik*, 2023, new York, NY.

[128] ——, *Schrödinger Release 2023-2: Glide*, 2023, new York, NY.

[129] ——, *Schrödinger Release 2023-2: LigPrep*, 2023, new York, NY.

[130] ——, *Schrödinger Release 2023-2: Maestro*, 2023, new York, NY.

[131] *Schrödinger Release 2023-2: Protein Preparation Wizard; Epik, Schrödinger, LLC, New York, NY, 2023; Impact, Schrödinger, LLC, New York, NY; Prime, Schrödinger, LLC, New York, NY, 2023*, Schrödinger, LLC, New York, NY, 2023.

[132] D. E. Scott, A. R. Bayly, C. Abell, and J. Skidmore, "Small molecules, big targets: drug discovery faces the protein–protein interaction challenge," *Nature Reviews Drug Discovery*, vol. 15, no. 8, pp. 533–550, 2016. [Online]. Available: https://doi.org/10.1038/nrd.2016.29

[133] S. Shermukhamedov, D. Mamurjonova, and M. Probst, "Structure to property: Chemical element embeddings and a deep learning approach for accurate prediction of chemical properties," *arXiv preprint arXiv:2309.09355*, 2023.

[134] C. Shi*, M. Xu*, Z. Zhu, W. Zhang, M. Zhang, and J. Tang, "GraphAF: a flow-based autoregressive model for molecular graph generation," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=S1esMkHYPr

[135] M. Simonovsky and N. Komodakis, "GraphVAE: Towards generation of small graphs using variational autoencoders," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham: Springer International Publishing, 2018, pp. 412–422.

[136] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37.

Lille, France: PMLR, 07–09 Jul 2015, pp. 2256–2265. [Online]. Available: https://proceedings.mlr.press/v37/sohl-dickstein15.html

[137] H. Sun, J. Wang, H. Wu, S. Lin, J. Chen, J. Wei, S. Lv, Y. Xiong, and D.-Q. Wei, "A multimodal deep learning framework for predicting ppi-modulator interactions," *Journal of chemical information and modeling*, vol. 63, no. 23, pp. 7363–7372, 2023.

[138] B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu, and the UniProt Consortium, "UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches," *Bioinformatics*, vol. 31, no. 6, pp. 926–932, 2014. [Online]. Available: https://doi.org/10.1093/bioinformatics/btu739

[139] K. Swanson, G. Liu, D. B. Catacutan, A. Arnold, J. Zou, and J. M. Stokes, "Generative AI for designing and validating easily synthesizable and structurally novel antibiotics," *Nature Machine Intelligence*, vol. 6, no. 3, pp. 338–353, 2024. [Online]. Available: https://doi.org/10.1038/s42256-024-00809-7

[140] E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.21423

[141] T. Tanimoto, *An Elementary Mathematical Theory of Classification and Prediction*. International Business Machines Corporation, 1958. [Online]. Available: https://books.google.es/books?id=yp34HAAACAAJ

[142] A. Telepov, A. Tsypin, K. Khrabrov, S. Yakukhnov, P. Strashnov, P. Zhilyaev, E. Rumiantsev, D. Ezhov, M. Avetisian, O. Popova, and A. Kadurin, "FREED++: Improving RL agents for fragment-based molecule generation by thorough reproduction," *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=YVPb6tyRJu

[143] 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*, vol. 31, no. 2, pp. 455–461, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21334

[144] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[145] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[146] C. Vignac, N. Osman, L. Toni, and P. Frossard, "MiDi: Mixed graph and 3d denoising diffusion for molecule generation," in *Machine Learning and Knowledge*

Discovery in Databases: Research Track, D. Koutra, C. Plant, M. Gomez Rodriguez, E. Baralis, and F. Bonchi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 560–576.

[147] B. O. Villoutreix, M. A. Kuenemann, J. Poyet, H. Bruzzoni-Giovanelli, C. Labbé, D. Lagorce, O. Sperandio, and M. A. Miteva, "Druglike protein-protein interaction modulators: Challenges and opportunities for drug discovery and chemical biology," *Molecular Informatics*, vol. 33, no. 67, p. 414437, 2014. [Online]. Available: http://dx.doi.org/10.1002/minf.201400040

[148] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[149] J. Wang, Y. Chu, J. Mao, H.-N. Jeon, H. Jin, A. Zeb, Y. Jang, K.-H. Cho, T. Song, and K. T. No, "De novo molecular design with deep molecular generative models for PPI inhibitors," *Briefings in Bioinformatics*, vol. 23, no. 4, p. bbac285, 2022. [Online]. Available: https://doi.org/10.1093/bib/bbac285

[150] J. Wang, J. Mao, M. Wang, X. Le, and Y. Wang, "Explore drug-like space with deep generative models," *Methods*, vol. 210, pp. 52–59, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1046202323000129

[151] M. Wang, Z. Wu, J. Wang, G. Weng, Y. Kang, P. Pan, D. Li, Y. Deng, X. Yao, Z. Bing, C.-Y. Hsieh, and T. Hou, "Genetic algorithm-based receptor ligand: A genetic algorithm-guided generative model to boost the novelty and drug-likeness of molecules in a sampling chemical space," *Journal of Chemical Information and Modeling*, vol. 64, no. 4, pp. 1213–1228, Feb 2024. [Online]. Available: https://doi.org/10.1021/acs.jcim.3c01964

[152] Y. Wang, H. Zhao, S. Sciabola, and W. Wang, "cMolGPT: A conditional generative pre-trained transformer for target-specific de novo molecular generation," *Molecules*, vol. 28, no. 11, 2023. [Online]. Available: https://www.mdpi.com/1420-3049/28/11/4430

[153] D. Weininger, "SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, 1988. [Online]. Available: https://doi.org/10.1021/ci00057a005

[154] T. Weiss, E. Mayo Yanes, S. Chakraborty, L. Cosmo, A. M. Bronstein, and R. Gershoni-Poranne, "Guided diffusion for inverse molecular design," *Nature Computational Science*, vol. 3, no. 10, pp. 873–882, 2023. [Online]. Available: https://doi.org/10.1038/s43588-023-00532-0

[155] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," in *International*

Conference on Learning Representations, 2018. [Online]. Available: https://openreview.net/forum?id=rJNpifWAb

[156] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992. [Online]. Available: https://doi.org/10.1007/BF00992696

[157] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Q. Liu and D. Schlangen, Eds. Online: Association for Computational Linguistics, 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[158] L. Wu, C. Gong, X. Liu, M. Ye, and Q. Liu, "Diffusion-based molecule generation with informative prior bridges," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 36 533–36 545. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/eccc6e11878857e87ec7dd109eaa9eeb-Paper-Conference.pdf

[159] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "MoleculeNet: a benchmark for molecular machine learning," *Chemical Science*, vol. 9, no. 2, p. 513530, 2018. [Online]. Available: http://dx.doi.org/10.1039/c7sc02664a

[160] J. Xie, S. Chen, J. Lei, and Y. Yang, "Diffdec: Structure-aware scaffold decoration with an end-to-end diffusion model," *Journal of Chemical Information and Modeling*, vol. 64, no. 7, pp. 2554–2564, 2024. [Online]. Available: https://doi.org/10.1021/acs.jcim.3c01466

[161] M. Xu, A. Powers, R. Dror, S. Ermon, and J. Leskovec, "Geometric latent diffusion models for 3D molecule generation," in *International Conference on Machine Learning*. PMLR, 2023.

[162] C. Yamanaka, S. Uki, K. Kaitoh, M. Iwata, and Y. Yamanishi, "De novo drug design based on patient gene expression profiles via deep learning," *Molecular Informatics*, vol. 42, no. 8-9, p. 2300064, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/minf.202300064

[163] S. Yang, D. Hwang, S. Lee, S. Ryu, and S. J. Hwang, "Hit and lead discovery with explorative RL and fragment-based molecule generation," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 7924–7936. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/41da609c519d77b29be442f8c1105647-Paper.pdf

[164] X. Yang, J. Zhang, K. Yoshizoe, K. Terayama, and K. Tsuda, "ChemTS: an efficient python library for *de novo* molecular generation," *Science and technology of advanced materials*, vol. 18, no. 1, p. 972976, 2017. [Online]. Available: https://europepmc.org/articles/PMC5801530

[165] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 64126422.

[166] C. Zang and F. Wang, "MoFlow: An invertible flow model for generating molecular graphs," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 617626. [Online]. Available: https://doi.org/10.1145/3394486.3403104

[167] M. Zaslavskiy, S. Jégou, E. W. Tramel, and G. Wainrib, "ToxicBlend: Virtual screening of toxic compounds with ensemble predictors," *Computational Toxicology*, vol. 10, pp. 81–88, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468111318300744

[168] S. Zheng, X. Yan, Q. Gu, Y. Yang, Y. Du, Y. Lu, and J. Xu, "QBMG: quasi-biogenic molecule generator with deep recurrent neural network," *Journal of Cheminformatics*, vol. 11, no. 1, p. 5, 2019. [Online]. Available: https://doi.org/10.1186/s13321-019-0328-9

[169] Z. Zhou, S. Kearnes, L. Li, R. N. Zare, and P. Riley, "Optimization of molecules via deep reinforcement learning," *Scientific Reports*, vol. 9, no. 1, p. 10752, 2019. [Online]. Available: https://doi.org/10.1038/s41598-019-47148-x

Appendix

A Classification of Generative Models

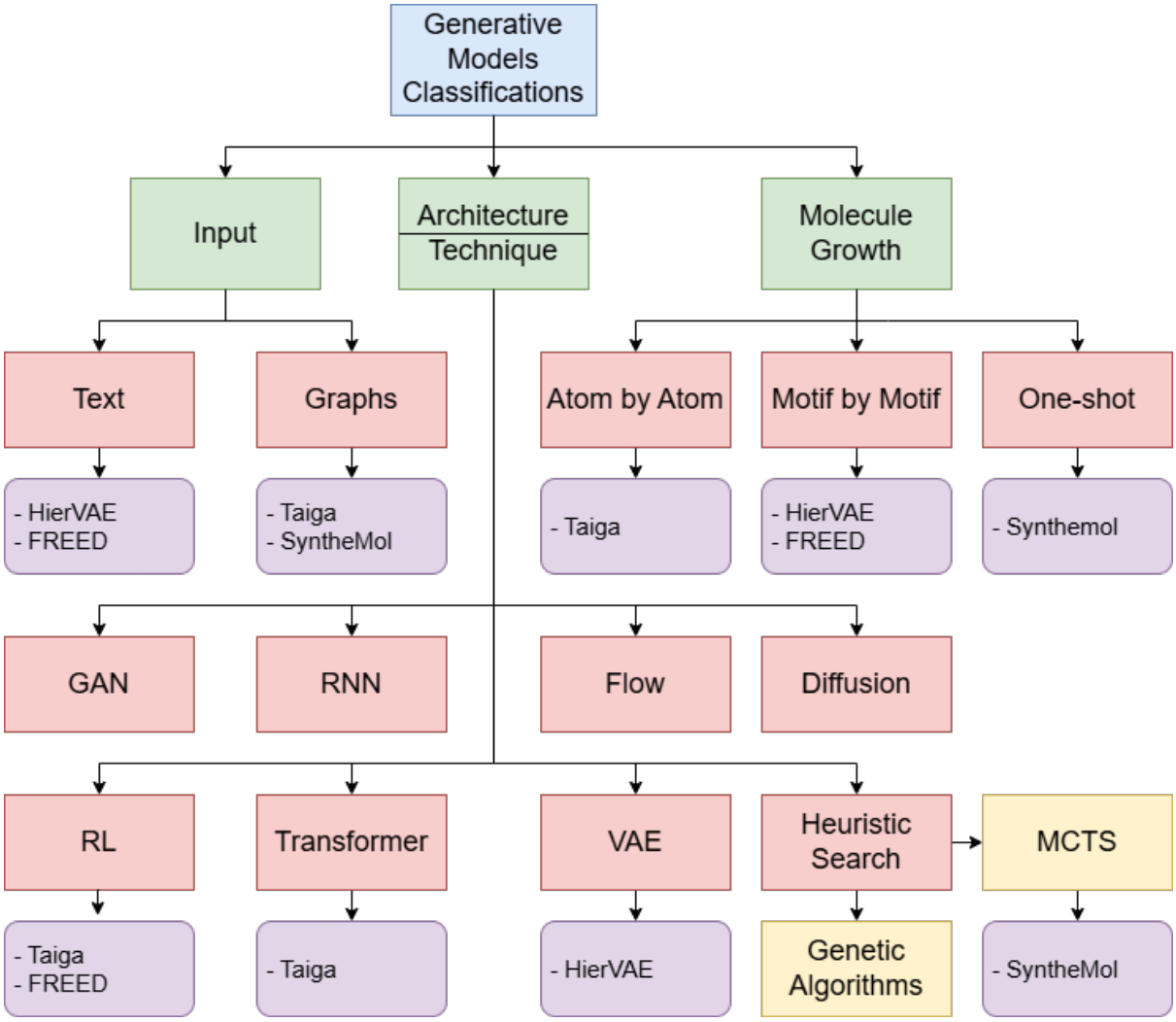


Figure 20: Classification of generative models in the literature.

B FREED Models Comparison

We compared the rewards obtained during the generation of NCS-1/Ric-8A and NCS-1/D2R modulators using FREED with default hyperparameters, a custom reimplementation of FFREED, and FREED++. During each step of RL training, the rewards for the generated molecules were recorded, and graph curves were calculated as the rolling mean of 100 samples. As shown in Figure 21, FREED++ consistently outperforms the other models across all graphs. It is also worth noting that models with a higher target entropy α tend to obtain lower rewards, as expected. However, this trade-off is necessary to ensure the generation of diverse enough molecules.

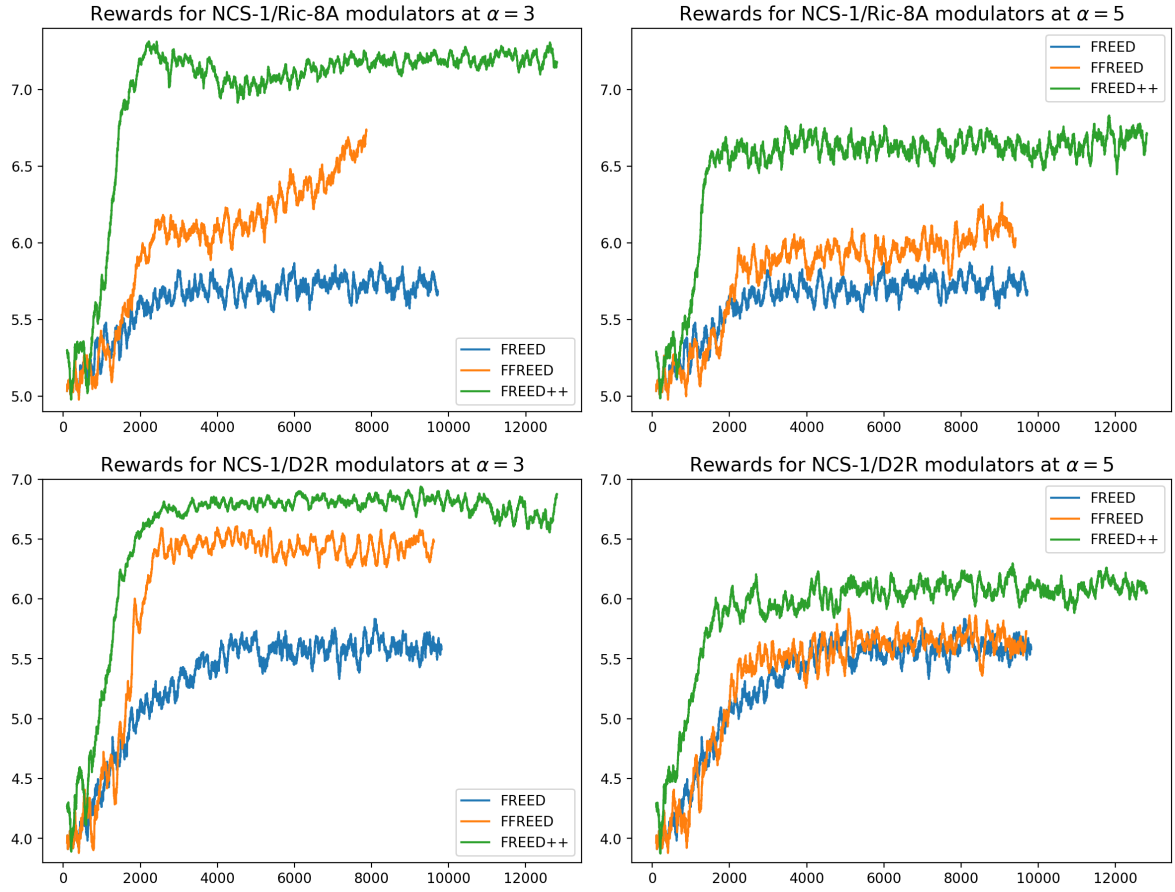


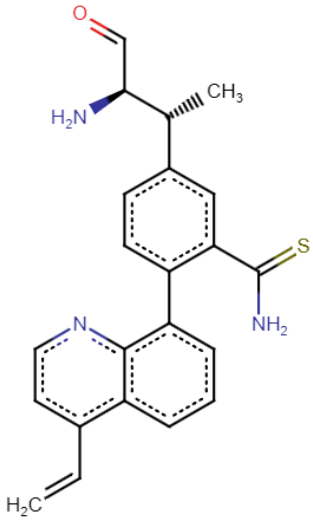
Figure 21: Comparison of rewards obtained during training with different FREED-based models

C Docking Procedure

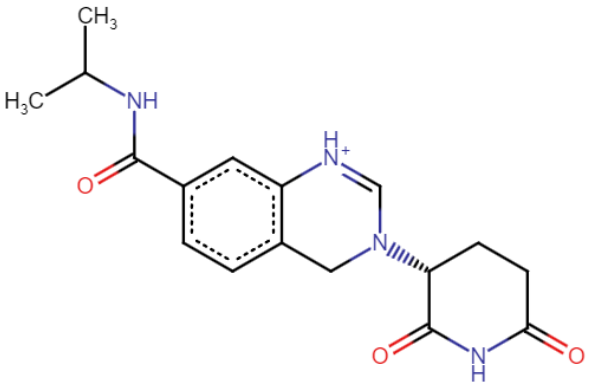
Protein-ligand docking using the Schrödinger interface Maestro [130] involves the computational prediction of the binding orientation and affinity of a small molecule (ligand) to a target protein. This process begins with the preparation of the NCS-1 protein structure utilizing the Protein Preparation Wizard [131]. The protocol initiates with the preprocessing of the protein structure, where bond orders are assigned, hydrogens are added, and ligand protonation states at pH 7 ± 2 are generated using Epik [127]. To facilitate conformational adjustments in receptor structures, Prime is employed [57]. Subsequently, the hydrogen-bonding network is optimized, and the protonation states of residues are calculated at pH 7 using PROPKA [105]. Water molecules, beyond a 5 Å radius from non-protein residues, are excluded, followed by a final restrained minimization with the OPLS4 force field. For receptor grid generation, the centroid of the crystallized ligand served as the grid center. During grid generation, a van der Waals radius scaling factor of 1.0 and a partial charge cutoff of 0.25 are applied.

The ligand preparation and optimization are conducted using LigPrep [129], generating possible ionization states at physiological pH, potential tautomers, and desalting the structures. Final energy minimizations are performed using the OPLS4 force field, with default parameters set for stereoisomers. The docking process is then executed using Schrödinger's Glide software [128] in extra precision (XP) mode, without applying any constraints. Default parameters are employed for ligand settings, including flexible ligand sampling and the incorporation of Epik state penalties into the docking score. The procedure concludes with post-docking minimization using default settings, providing a detailed prediction of the ligand's binding pose and affinity to the target protein.

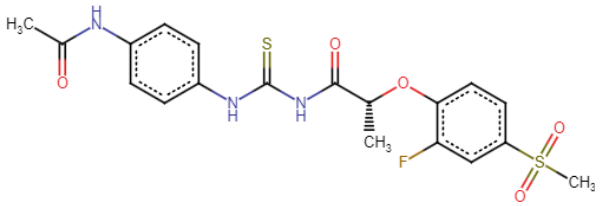
D Molecules from the Virtual Chemical Library



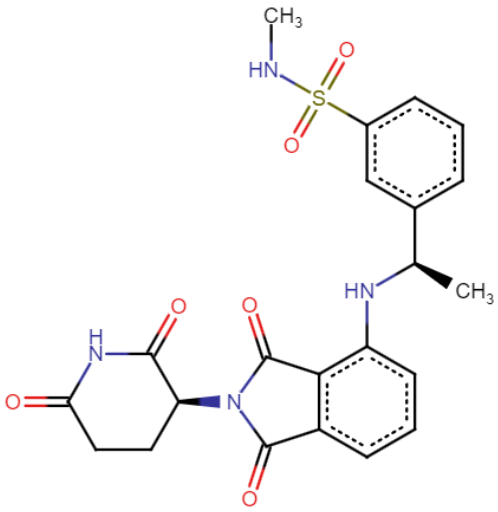
(a) Freed++



(b) HierVAE

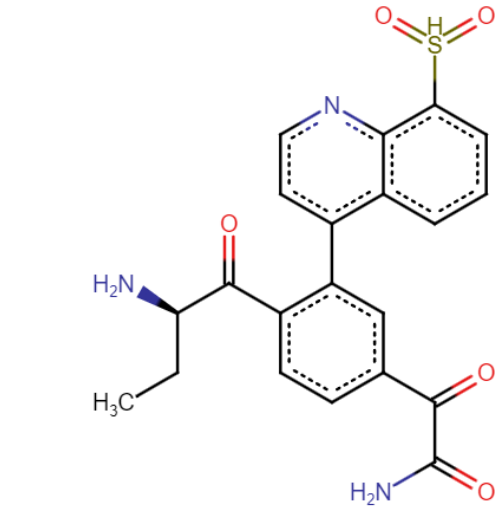


(c) Taiga

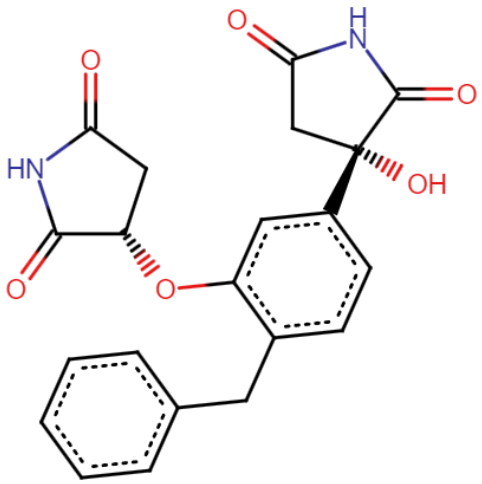


(d) Synthemol

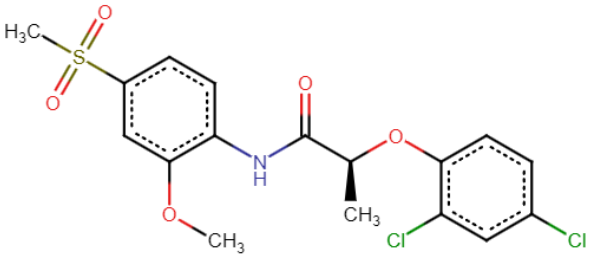
Figure 22: Top performing molecules of each model in NCS-1/Ric8 bioactivity prediction



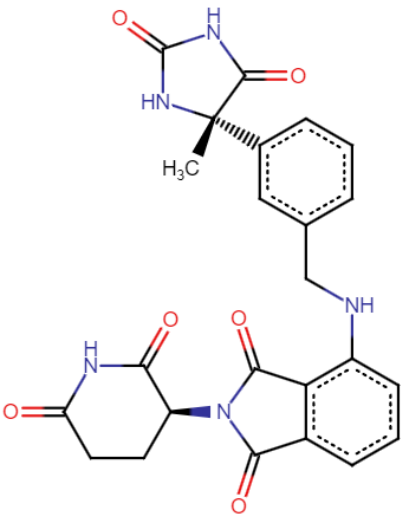
(a) Freed++



(b) HierVAE



(c) Taiga



(d) Synthemol

Figure 23: Top performing molecules of each model in NCS-1/D2R bioactivity prediction