EXPLORATION ON DEEP DRUG DISCOVERY: REPRESENTATION AND LEARNING

by

Shengchao Liu

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science

(Computer Science)

at the

UNIVERSITY OF WISCONSIN-MADISON

September 2018

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science.

(Anthony Gitter) Principal Adviser

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science.

(Yingyu Liang)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science.

(Spencer S. Ericksen)

Approved for the University of Wisconsim-Madison Committee on Graduate Studies

Acknowledgments

I would first like to thank my thesis advisor, Professor Anthony Gitter. The door to Prof. Gitter office was always open when I ran into problems on my research projects or paper writing. He can guide me in the right direction for my research problems.

I would also thank Prof. Yingyu Liang and Dr. Spencer Ericksen for helping with my thesis project and being in my committee. Prof. Liang helped guide me to solve about problem from very special aspect, like apply n-gram idea on the graph structural data. And Dr. Ericksen helped offering rich domain knowledge for my questions on the drug discovery projects.

In addition, I would like to thank all my co-authors and labmates. Special thanks to Moayad Alnammi, Scott A. Wildman, Thevaa Chandereng, Small Molecule Screening Facility group at University of Wisconsin-Madison, Chengpeng Wang for helpful discussion.

Finally, I am grateful for computing resources from NVIDIA and the University of Wisconsin-Madison Center for High Throughput Computing and support provided by the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funcing from the Wisconsin Alumni Research Foundation.

Contents

List of Tables

List of Figures

Chapter 1

Starting from A Standard Pipeline

We start from a benchmark project, so as to build up the whole framework for applying machine learning on drug discovery. Co-authors of [54] contributed to the contents of this chapter.

1.1 Introduction

Drug discovery is time consuming and expensive. After a specific protein or mechanistic pathway is identified to play an essential role in a disease process, the search begins for a chemical or biological ligand that can perturb the action or abundance of the disease target in order to mitigate the disease phenotype. A standard approach to discover a chemical ligand is to screen thousands to millions of candidate compounds against the target in biochemical or cell-based assays via a process called high-throughput screening (HTS), which produces vast sets of valuable pharmacological data. Even though HTS assays are highly automated, screens of thousands of compounds sample only a small fraction of the millions of commercially-available, drug-like compounds. Cost and time preclude academic laboratories and even pharmaceutical companies from blindly testing the full set of millions of drug-like compounds in HTS assays. Thus, there is a crucial need for an effective virtual screening (VS) process as a preliminary step in prioritizing compounds for HTS assays.

Virtual screening comprises two categories: structure-based [16, 53] and ligand-based methods [42, 94]. Structure-based methods require that the target protein's molecular structure be known so that the 3D interactions between the target and each chemical compound (binding poses) may be predicted *in silico*. These interactions are given numerical scores, which are then used to rank compounds for potential binding to the target. These methods do not require or typically make use of historical screening data in compound scoring. In contrast, ligand-based methods require no structural information about the target but use data generated from testing molecules in biochemical or functional assays of the target to fit empirical models that relate compound attributes to assay outcomes.

For targets with abundant assay data or where a druggable binding site is not well-defined, such as the targets considered here, ligand-based methods are generally superior to structure-based methods [33, 43, 96]. Confronted with the variety of ligand-based model building methods (e.g., regression models, random forests, support vector machines, etc.) [65], compound input representations, and performance metrics, how should one proceed? The Merck Molecular Activity Challenge [64] incited the development of many ligand-based deep learning VS methods [18, 58, 61, 95, 77], as recently reviewed [12]. These methods are often assessed with cross-validation on existing HTS data, but there is presently little experimental evidence on the best option for prioritizing new compounds given a fixed screening budget.

We critically evaluated a collection of VS algorithms that include both structure-based and ligand-based methods. We present a VS workflow that first uses available HTS training data to systematically prune the specific versions of the algorithms and calculate their cross-validation performance on a variety of evaluation metrics. Based on the cross-validation results and analysis of the various evaluation metrics, we selected a single virtual screening algorithm. The selected method, a random forest model, was the best option for prioritizing a small number of compounds from a new library, as verified by experimental screening. These model selection and evaluation strategies can guide VS practitioners to select the best model for their target even as the landscape of available VS algorithms continues to evolve.

1.2 Background

1.2.1 Dataset

Our case studies were on new and recently generated datasets [98, 97] for the targets PriA-SSB and RMI-FANCM. The PriA-SSB interaction is important in bacterial DNA replication and is a potential target for antibiotics [74]. The RMI-FANCM interaction is involved in DNA repair that is induced in human cancer cells to confer chemoresistance to cytotoxic DNA-cross-linking agents, making it an attractive drug target [59]. We previously screened these targets with a library of compounds obtained from Life Chemicals Inc. (LC) in different assay formats. In addition, we screened new LC compounds on the PriA-SSB target to evaluate our VS models. The four datasets derived from these screens are described below and summarized in Table 1.1.

PriA-SSB AlphaScreen: We refer to these continuous values as "PriA-SSB % inhibition". Those compounds that tested above a certain activity threshold ($\geq$ 35% inhibition) and passed PAINS chemical structural filters [8, 47] were retested in the same AS assay. Compounds that were confirmed in the AS retest screen ($\geq$ 35% inhibition) were marked as actives, creating the binary dataset **PriA-SSB AS**.

PriA-SSB fluorescence polarization: Compounds that had PriA-SSB % inhibition $\geq$ 35% and passed the PAINS filters were also tested in a fluorescence polarization (FP) assay as a secondary

screen. Those compounds with FP inhibition ≥30% were labeled as actives, creating the binary dataset **PriA-SSB FP**, with all other compounds in the screening set labeled inactive.

RMI-FANCM fluorescence polarization: The RMI-FANCM interaction was initially screened with a subset of 49,796 compounds from the same LC library as PriA-SSB [98]. We refer to these continuous values as "RMI-FANCM % inhibition". Those compounds that demonstrated activity ≥ 2 standard deviations (SD) above the assay mean and passed PAINS filters were marked as actives in the binary dataset **RMI-FANCM FP**.

PriA-SSB prospective: In subsequent screens used for prospective testing, we tested an additional 22,434 compounds, purchased from LC after the initial screening was complete. There was no overlap between these compounds and the 72,423 LC compounds described above that were used to train VS models. Actives were defined with the same criteria used for the binary dataset **PriA-SSB AS**. Compounds with at least 35% inhibition that passed the PAINS filters were retested with the AS assay. Those with at least 35% inhibition in the AS retest were labeled as actives, creating the binary dataset **PriA-SSB prospective**.

Because secondary screens and structural filters were used to define the active compounds, there was no single primary screen % inhibition threshold that separated the actives from the inactives. Some compounds exhibiting high % inhibition values were labeled as inactive because they did not satisfy the structural requirements or were not active in the secondary screen.

Table 1.1: Summary statistics for the four binary datasets.

Stage	Dataset	% inhibition threshold	# actives	# inactives
Cross-validation	**PriA-SSB AS**	≥ 35%	79	72,344
	PriA-SSB FP	≥ 30%	24	72,399
	RMI-FANCM FP	≥ 2 SD	230	49,566
Prospective	**PriA-SSB prospective**	≥ 35%	54	22,380

To help learn a better chemical representation with multi-task neural networks, we considered other screening contexts from which to transfer useful knowledge. We used a specific subset of 128 assays (AIDs) from the **PubChem BioAssay (PCBA)** [100] repository. This dataset was used in previous work on multi-task neural networks [77]. This subset contained assays that probe a specific protein target and provide dose-response measurements (assay query filters, see Appendix A.1). Potency and curve quality are factored into a PubChem Activity Score. Regardless of assay, compounds receiving a PubChem Score of 40 or greater (0-100) were assigned a PCBA Bioactivity outcome (label) of "Active". Compounds receiving PC Activity Scores 1-39 were labeled "Inconclusive," and those receiving 0 are labeled "Inactive" (Appendix A.2 and Appendix A.3).

1.2.2 Compound Features

Ligand-based virtual screening methods require each chemical compound to be represented in a particular format as input to the model. We adopted two common representations. All of the ligand-based algorithms except the Long Short-Term Memory (LSTM) neural network use Morgan fingerprints (1024 bits) [80]. For LSTM networks, we used the Simplified Molecular Input Line Entry System (SMILES) representation [101], where the characters were treated as sequential features. Both Morgan fingerprints and canonical SMILES were generated in RDKit [1].

Besides, people have found there exists a bottleneck when using Morgan fingerprints and SMILES. Recently more and more graph-based neural network models have been proposed, and in Chapter 2 we will have a further discussion on it. This chapter is focusing on building up a general framework so as to give a intuition on how to apply machine learning on drug discovery.

Further exploration of molecule representation will be discussed in Section 2.3.

Morgan Fingerprints

Morgan fingerprints [80] are produced by a widely accepted featurization mechanism to convert molecules to fixed-length bit strings. It is an iterative algorithm that encodes the circular substructures centered on each atom (node) of the molecule's 2D graph as identifiers at increasing levels with each iteration. In each iteration, hashing is applied to generate new identifiers, and thus, there is a chance that two substructures are represented by the same identifier. In the end, a list of identifiers encoding the substructures are folded to bit positions of a fixed-length bit string. A 1-bit at a particular position indicates the presence of a substructure (or multiple substructures) and a 0-bit indicates its absence. The number of iterations, also called the radius, d and length of the bit string l is set by the user. We used the common setting of $d = 4$ and $l = 1024$. Section 1.2.2 illustrates the concept with a small fixed-length bit string.

Morgan fingerprints are commonly used as features for molecules in predicting drug activity. [62] report that a Deep Neural Network trained on Morgan fingerprints had similar performance to one trained on molecular descriptors. We use them in a supervised learning setting where the input features are the Morgan fingerprints and the target activities are the output labels. The goal is to train a supervised learning model that is able to generalize to fingerprint instances outside of the training set.

Simplified Molecular Input Line Entry System

The second option for feature representation is [101] Simplified Molecular Input Line Entry System (SMILES). Each molecule can be represented via a SMILES sequence, which consists of around 35 different characters. For example, c1cc(oc1C(=O)Nc2nc(cs2)C(=O)OCC)Br is a canonical SMILES for the molecule in Figure 1.2. All atoms, except for Br, Cl, and Si, are represented by single

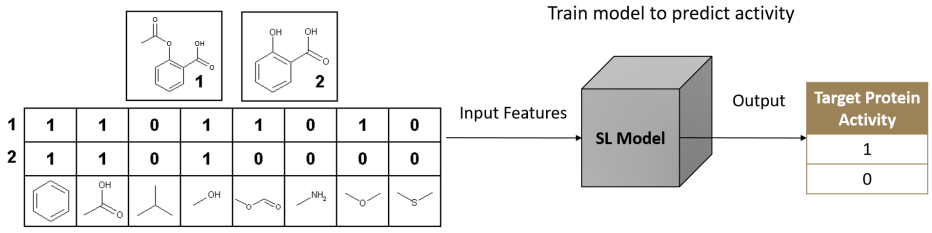


Figure 1.1: Morgan fingerprints used in a supervised learning (SL) setting for learning drug activity. Credit goes to S. Lusher and G. Schaftenaar. 2-D searching Tutorial (http://www.cmbi.ru.nl/edu/bioinf4/2D-Prac/2d.shtml).

alphabetic characters. Atoms adjacent to each other or linked by '=' or '#' sign means they are bonded, where any sequence between numbered elemtns is in a ring closed by those numbered elements.

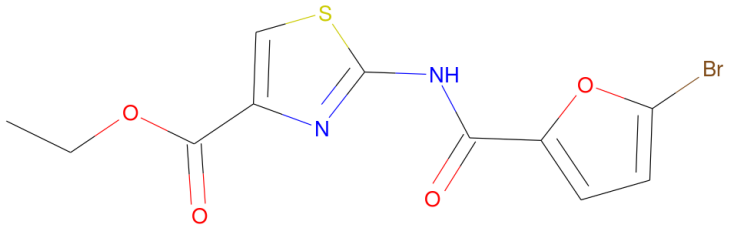


Figure 1.2: Three variants of representations for a molecule example. The graph for one molecule example is displayed as above. The canonical SMILES is c1cc(oc1C(=O)Nc2nc(cs2)C(=O)OCC)Br, and Morgan fingerprints is [000000...00100100100...000000].

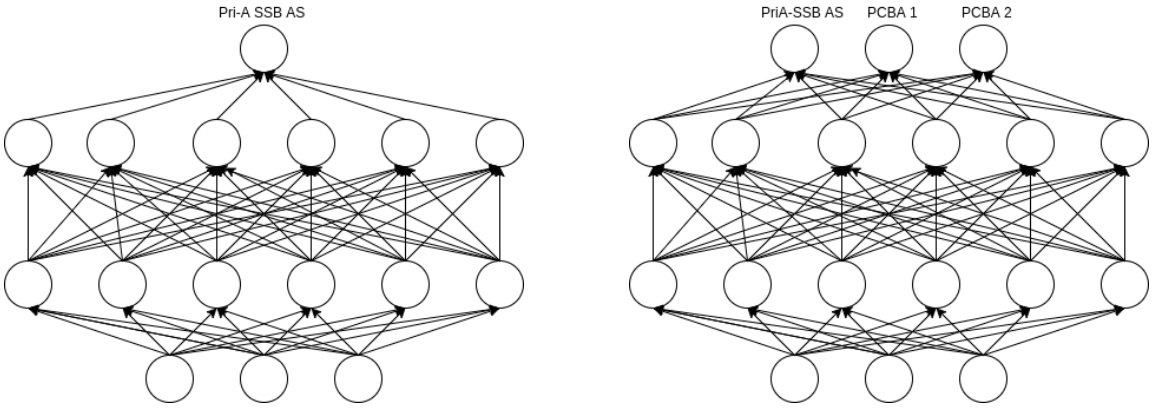
1.3 Virtual Screening Models

We selected a variety of existing virtual screening approaches for our benchmarks and prospective predictions. These included ligand-based supervised learning approaches, structure-based docking, and a chemical similarity baseline. Table 1.2 summarizes the types of training data used by each algorithm. We discuss hyperparameter tuning for these models in Section 1.3.6.

1.3.1 Ligand-Based Neural Networks

Deep learning is a machine learning approach that encompasses neural network models with multiple hidden layer architectures and the techniques for training these models. It represents the state of the art for many predictive tasks, which has generated extensive interest in deep learning for biomedical

research, including virtual screening [12]. We evaluated multiple types of established neural network architectures for virtual screening.



(a) Single-task Neural Network (b) Multi-task Neural Network

Figure 1.3: Neural network structures. The neural networks map the input features (e.g. fingerprints) in the input (bottom) layer to intermediate chemical representations in the hidden (middle) layers and finally to the output (top) layer, which makes either continuous or binary predictions. Figure 1.3(a) has only one unit in the output layer. Figure 1.3(b) has multiple units in the output layer representing different targets, one for our new target of interest and the others for PCBA targets.

Single-task Neural Network (STNN): A single-task neural network (Figure 1.3(a)) makes a single prediction for a single target (also referred to as a task). We trained a separate model for each of the **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** datasets, taking a compound's Morgan fingerprints elements (1024 bits) as the input features. We trained the neural networks using Keras [13] with the Theano backend [90]. The single-task neural networks were trained on each task to predict either the binary activity label in the classification setting (STNN-C) or the continuous % inhibition in the regression setting (STNN-R).

Multi-task Neural Network (MTNN): Multi-task neural networks make different predictions for multiple targets or tasks but share knowledge by training the first few hidden layers together. Each of our multi-task neural networks included one target task (**PriA-SSB AS**, **PriA-SSB FP**, or **RMI-FANCM FP**) and 128 tasks from PCBA. We only trained multi-task neural networks in the classification setting (MTNN-C).

Single-task Atom-level LSTM (LSTM): The LSTM is one of most prevalent recurrent neural network models [34], which has been applied previously in virtual screening [37]. An LSTM assumes there exists a sequential pattern in the input string. We used a one-hot encoding of the SMILES strings as input for the LSTM model. In a one-hot encoding, each character in a SMILES string

is replaced by a binary vector. The binary vector has one bit for each possible unique character in all SMILES strings. At each position in a SMILES string, the bit corresponding to the character at that position is set to 1, and all other bits are set to 0. We trained the LSTM model to predict the binary activity labels.

Influence Relevance Voter (IRV): IRV [88, 57] is a hybrid between k-nearest neighbors and neural networks. Each compound's output value is a non-linear combination of the similarity scores from its most closely related compounds in the training dataset. We used Morgan fingerprints as the input and trained separate IRV models for each dataset.

1.3.2 Other Ligand-Based Models

Random Forest (RF): Random forests are ensembles of decision trees that are often used as a baseline in virtual screening benchmarks [78, 102]. We used scikit-learn [75] to train a random forest classifier for each binary label with Morgan fingerprints as features.

1.3.3 Protein-Ligand Docking

Target Preparation: Our structure-based VS approach involved the docking-based ranking of the LC library to the holo-form of PriA, using the crystal structure (PDB: 4NL8) [9] in which it is bound to a C-terminal segment of an SSB protein. A missing loop in this structure was added from the the apo-form (PBD: 4NL4), though this is not near the SSB binding site. The docking search space was limited to 8Å from the coordinates of the co-crystallized SSB C-terminal tripeptide.

Compound Preparation: LC library compounds were assigned 3D coordinates and Merck Molecular Force Field partial charges using OpenEye OMEGA and Molcharge [32], respectively. Compounds in the LC library with ambiguous stereochemistry were enumerated to test all possible configurations, and the best resulting docking score was retained for each.

Docking (Dock) and Consensus Docking (CD): We ran eight different docking programs and generated nine docking scores. The docking programs and names we use for their scores are AutoDock v4.2.6 [68] (Dock_ad4), Dock v6.7 [3] (Dock_dock6), FRED v3.0.1 [63] (Dock_fred), HYBRID v3.0.1 [63] (Dock_hybrid), PLANTS v1.2 [41] (Dock_plants), rDock v2013.1 [83] (Dock_rdocktot and Dock_rdockint), Smina v1.1.2 [40] (Dock_smina), and Surflex-Dock v3.040 [14] (Dock_surflex). In addition, we calculated consensus docking scores using three traditional approaches (CD_mean, CD_median, and CD_max) and two versions of the Boosting Consensus Score (CD_efr1_opt and CD_rocauc_opt) [25]. The Boosting Consensus Score method was trained using 21 of the DUD-E benchmark targets [70] without any PriA-SSB or RMI-FANCM assay data. Compounds with missing scores due to failures in preparation or docking were not considered during evaluation.

1.3.4 Chemical Similarity Baseline

In the prospective screening stage, we introduced a simple baseline compound ranking method against which to compare more sophisticated VS methods. Each of the 22,434 compounds in the prospective screening set was compared to the actives in **PriA-SSB AS** using Tanimoto similarity on the Morgan fingerprints. Compounds in the prospective library were ranked by their similarity, and those with Tanimoto similarity ≥ 0.45 to any active compound in the initial screen were predicted to be active.

In addition, all compounds were clustered by two separate approaches to describe chemical series. Chemical similarity based hierarchical clusters on Morgan fingerprints using Ward's clustering are described as SIM. Maximum common substructure clusters, used to group molecules with similar scaffolds, are described as MCS. JKlustor was used for both types of clustering (JChem v17.26.0, ChemAxon).

Model	Continuous % inhibition	Binary label	PCBA binary labels
Dock			
CD			
STNN-C		✓	
STNN-R	✓		
MTNN-C		✓	✓
LSTM		✓	
RF		✓	
IRV		✓	
Similarity baseline		✓	

Table 1.2: A summary of the virtual screening methods and labels used by each model. The docking and consensus docking models do not train on the PriA-SSB or RMI-FANCM datasets.

1.3.5 Evaluation Metrics

Given our goal of developing VS methods that enable very small, cost-effective, productive screens, we considered how metrics weight early active retrieval. All of the VS algorithms produce a ranked list of compounds, where compounds are ordered by the probability of being active, the continuous predicted % inhibition, the docking score, or a comparable output value. For a ranked list of compounds, we can threshold the ranked list and consider all compounds above the threshold as positive (active) predictions and those below the threshold as negative (inactive). By comparing those predictions to the experimentally-observed activity, we can compute true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions for that ranked list at that

threshold. We explored several options for summarizing how well each algorithm ranks the known active compounds.

The area under the receiver operating characteristic curve (AUC[ROC]) has been recommended for virtual screening because it is robust, interpretable, and does not depend on user-defined parameters [72]. The ROC curve plots the relationship between true positive rate (TPR, also known as sensitivity or recall) and false positive rate (FPR, equivalent to 1 - specificity), which are defined in Section 1.3.5. As the FPR goes to 100%, all ROC curves will converge, whereas early active retrieval (a more meaningful characteristic of VS performance) can be assessed in the low FPR region of the ROC curve, which exhibits greater variability across VS methods. Thus, we also considered the Boltzmann-enhanced discrimination of receiver operating characteristic (BEDROC) [93]. It emphasizes the early part of the ROC curve through a scaling function α, which we set to 10 for our purposes of early enrichment up to 20%. We used the BEDROC implementation from the CROC Python package [87].

$$TPR = \frac{TP}{TP + FN}, \qquad FPR = \frac{FP}{FP + TN} \tag{1.1}$$

$$Recall = \frac{TP}{TP + FN}, \quad Precision = \frac{TP}{TP + FP} \tag{1.2}$$

Area under the precision-recall curve (AUC[PR]) is another common metric (Section 1.3.5). AUC[PR] has an advantage over AUC[ROC] for summarizing classifier performance when the class labels are highly-skewed, as in virtual screening where there are few active compounds in a typical library. AUC[PR] evaluates a classifier on its ability to retrieve actives (recall) and which of the predicted actives correctly classified (precision) as we vary the prediction threshold. We used the PRROC R package [29] to compute AUC[PR] because it correctly interpolates between points in the PR curve when there are ties in the compound rankings [19].

Another VS metric is enrichment factor (EF), which is the ratio between the number of actives found in some prioritized subset of compounds versus the expected number of actives in a random subset of same size. In other words, it assesses how much better the VS method performs over random compound selection. Let $R \in [0\%, 100\%]$ be a pre-defined fraction of the compounds from the total library of compounds screened.

$$EF_R = \frac{\#\text{ actives in top } R \text{ ranked compounds}}{\#\text{ actives in entire library} \times R} \tag{1.3}$$

$$EF_{max,R} = \frac{min\{\#\text{ actives, total }\#\text{ compounds} \times R\}}{\#\text{ actives in entire library} \times R} \tag{1.4}$$

$EF_{max,R}$ represents the maximum enrichment factor possible at R. Difficulty arises when interpreting EF scores because they vary with the dataset and threshold R. We defined the normalized enrichment factor (NEF) as:

$$NEF_R = \frac{EF_R}{EF_{max,R}} \tag{1.5}$$

Because $NEF_R \in [0, 1]$, it is easier to compare performance across datasets and thresholds. 1.0 is the perfect normalized enrichment factor. Furthermore, we can create an NEF curve as NEF_R versus $R \in [0\%, 100\%]$ and compute the area under that curve to obtain AUC[NEF] $\in [0, 1]$. However, most models tend to exhibit similar late enrichment behavior. We are typically interested in early enrichment behavior so we computed AUC[NEF] using $R \in [0\%, 20\%]$.

Finally, we introduce the metric n_{hits} which is simply the number of actives found in a selected number of tested compounds (e.g. how many hits or actives were found in 250 tested compounds). This metric represents the typical desired utility of a screening process: retrieve as many actives as possible in the selected number of tests (denoted as n_{tests}). We compare n_{hits} at various n_{tests} to the different evaluation metrics to identify which metrics best mimic the n_{hits} utility.

1.3.6 Pipeline

Our virtual screening workflow contains three stages:

1. Tune hyperparameters in order to prune the model search space.
2. Train, evaluate, and compare models with k-fold cross-validation results.
3. Assess the best models' ability to prospectively identify active compounds in a new set.

In contrast to most other virtual screening studies, the experimental screen was not conducted until after all models were trained and evaluated in the cross-validation stage. For the first two stages, we first split the **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** datasets into five stratified folds as described in Appendix A.2.

Hyperparameter Sweeping Stage

Hyperparameters are model configurations or settings that can be set by an expert as opposed to the weights or parameters that are learned or fit during model training. For example, in neural networks the hyperparameters include the number of hidden layers, the number of hidden units in each layer, types of activation functions, drop out ratios, learning rates, etc. In random forests, hyperparameters include the number and depth of trees, the size of the subsamples, etc.

For most of the ligand-based machine learning models, the hyperparameter space was too large for exhaustive searches using the full dataset. Therefore, we applied a grid search on a pre-defined

set of hyperparameters in a smaller dataset and pruned those that performed poorly. We performed a single iteration of training on the first four folds of **PriA-SSB AS** to avoid over-fitting. The hyperparameters considered are listed in Appendix A.4.

Cross-Validation Stage

To identify which VS algorithms are likely to have the best performance in a prospective screen, we applied a traditional cross-validation training strategy on datasets **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** after reducing the hyperparameter combinations to consider. Selecting the best model is non-trivial. Ideally, the best model would have dominant performance on all evaluation metrics, but this is rarely observed with existing models. Each evaluation metric prioritizes different performance characteristics. Our cross-validation results illustrate which models consistently perform well over different metrics, the correspondence of metrics relative to a desired utility (n_{hits}), and how to choose models and evaluation metrics in order to successfully identify active compounds in a prospective screen.

Cross-validation is commonly used to avoid over-fitting when there are few training samples. We split the training data into 5 folds: 4 folds for training and 1 for testing. Models like RF and IRV that do not require a hold-out dataset for early stopping used 4 folds for training. Other models like the neural networks perform early-stopping based on a hold-out set, so we iteratively selected 1 of the 4 training data folds for this purpose. This led to a nested cross-validation with $5 \times 4 = 20$ trained neural networks.

Prospective Screening Stage

Our prospective screen used a library of 22,434 new LC compounds that were not present in the training set. We used each VS model to prioritize 250 of these compounds that are most likely to be active. This emulates virtual screening on much larger compound libraries, in which only a small fraction of all computationally scored compounds can be tested experimentally. After finalizing the models' predictions, we screened all 22,434 compounds in the wet lab and assigned actives based on a 35 % inhibition threshold and structural filters (**PriA-SSB prospective**). Finally, we evaluated how many of the wet lab actives each VS method identified in its top 250 predictions, the number of distinct chemical clusters recovered, and the number of active compounds that were not in the top 250 predictions from any of the VS algorithms. The prospective screen allowed us to assess how well the cross-validation results generalized to new compounds and further verified our conclusions from the retrospective cross-validation tests.

1.3.7 Data and Software Availability

Code implementing our ligand-based virtual screening algorithms is available at `https://github.com/gitter-lab/pria_lifechem`. This GitHub repository also contains additional Jupyter notebooks to reproduce the visualizations and analyses. Version 0.1.0 of our software is archived on Zenodo (doi:10.5281/zenodo.1257674). Our PriA-SSB and RMI-FANCM HTS data will be made available on PubChem. A formatted version of this dataset for training virtual screening algorithms is available on Zenodo (doi:10.5281/zenodo.1257462).

1.4 Cross-Validation Results

In the cross-validation stage we assessed 35 models: 8 neural networks (STNN-C, STNN-R, MTNN-C, and LSTM), 8 random forests (RF), 5 influence relevance voter (IRV), and 14 from docking (Dock) or consensus docking (CD). When there are multiple versions of a model that use different hyperparameters, we distinguish them with alphabetic suffixes such as "_a" and "_b". [54] describes the hyperparameters associated with these suffixes. We highlight the **PriA-SSB AS** dataset as a representative example, but the VS workflow is applicable for all tasks.

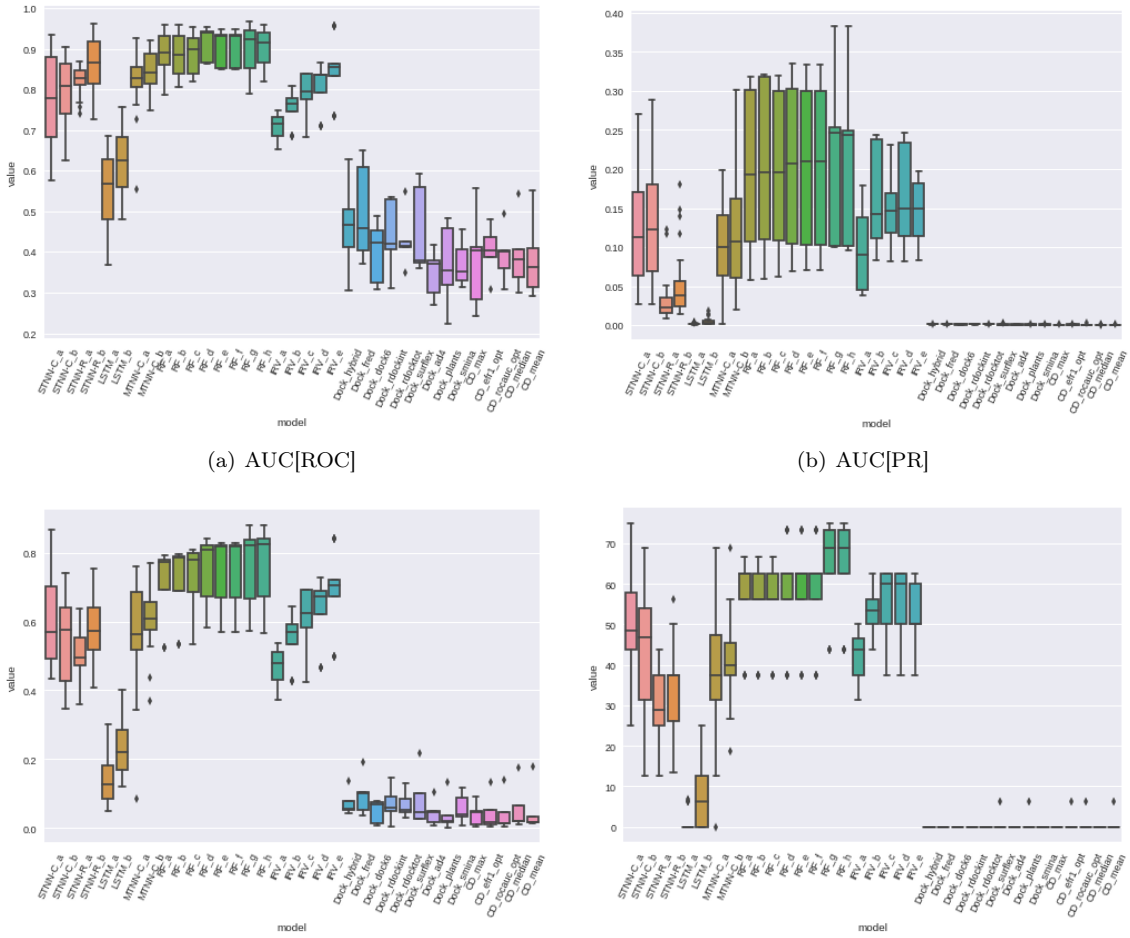
Comparing Virtual Screening Algorithms

We tested all 35 models on three datasets, and the results for four evaluation metrics on the **PriA-SSB AS** dataset are shown in Figure 1.4. Appendix A.5 contains the results for **PriA-SSB FP** and **RMI-FANCM FP**. The **PriA-SSB AS** performance using AUC[ROC] was comparable for most models, where nearly all models except LSTM and the docking programs were above 0.8 AUC[ROC]. Random forest was the best model, reaching AUC[ROC] of approximately 0.9.

Random forest was again the best method for the **RMI-FANCM FP** dataset for most of the evaluation metrics (Appendix A.5). On the **PriA-SSB FP** dataset, STNN-R achieved the highest scores over the majority of the metrics (Appendix A.5). The other types of VS models were effectively tied for most metrics (details in [54]).

Evaluation Metrics

Given a fixed evaluation metric, we could compare two models with a t-test to assess if one statistically outperforms the other. However, we needed to make such comparisons repeatedly between each pair of models and required a statistical test that accounts for multiple hypothesis testing. Due to unequal variances and sample sizes (Figure 1.4), we used Dunnett's modified Tukey-Kramer test (DTK) [48, 21] for pairwise comparison to assess whether the mean metric scores of two models were significantly different.



(a) AUC[ROC]

(b) AUC[PR]

(c) AUC[BEDROC]

(d) EF@1%

Figure 1.4: The evaluation metric distributions on **PriA-SSB AS** on all models over the cross-validation folds.

Using DTK results for *each metric*, we scored each model based on how many times it attained a statistically significantly better result than other models (details in [54]). For most metric-target pairs, many models have the same rank because DTK does not report a significant difference. Based on DTK alone, RF models consistently place in the top five ranks for each metric-target pair.

In a prospective screen, our goal is to maximize the number of active compounds identified by a VS algorithm given a fixed budget (number of predictions). We wanted to determine which of the evaluation metrics commonly used for VS best aligns with n_{hits}; i.e. maximizing the metric leads to maximizing n_{hits}. We assume that n_{hits} is ultimately the desired utility to maximize. Thus, we compared the model ranking induced by each metric with the model ranking induced by n_{hits} at varying number of tests.

To score the evaluation metrics, we used Spearman's rank correlation coefficient based on the model rankings induced by the metric of concern versus n_{hits} at a specific n_{tests}. We then ranked the metrics based on their correlation with n_{hits}. The correlation coefficients and metric rankings can be found in [54]. The metric ranking varies depending on n_{tests} and the target. Some metrics overtake one another as we increase or decrease n_{tests}. For **PriA-SSB AS**, NEF_R consistently placed in the top ranking correlations when R coincided with n_{tests}. This is evident when we focus on a single metric and see the top ranking metrics for $n_{tests} \in [100, 250, 500, 1000, 2500]$. Only for a large enough n_{tests} do metrics like AUC[ROC] that evaluate the complete ranked list become comparable. This suggests that if we know *a priori* how many new compounds we can afford to screen, then NEF_R at a suitable R is a viable metric for choosing a VS algorithm during cross-validation in the hopes of maximizing n_{hits}.

Selecting the Best Model

Based on these results, we selected the VS screening models that are most likely to generalize to new compounds and identify actives in our experimental screen of 22,434 new compounds. We focus on PriA-SSB for the prospective screen using models trained on **PriA-SSB AS** because the assay was more readily available for us to generate data for the new compounds.

Table 1.3 compares model selection based on evaluation metric means alone versus the DTK+Mean approach for multiple evaluation metrics on the three tasks. The complete model rankings for means only and DTK+Means can be found in [54]. DTK+Mean ranks models by statistical significance and uses the mean value only for tie-breaking. Both strategies selected the same models for a fixed evaluation metric, except for AUC[PR] on all three tasks and $\text{NEF}_{1\%}$ for **RMI-FANCM FP**. This is mainly due to DTK not detecting statistically significant differences among the models' evaluation scores, so tie-breaking by means selected the same models as ranking by means. Recall that **PriA-SSB FP** has fewer actives than **PriA-SSB AS** and **RMI-FANCM FP** (Table 1.1). Similar models from random forest and STNN-C were selected for **PriA-SSB AS** and **RMI-FANCM FP**. However, **PriA-SSB FP** identified STNN-R models exclusively.

Table 1.3: Top-ranked models by means versus DTK+Mean on the three tasks. Evaluation metric means were computed over all cross-validation folds. The prospective screening was only performed on PriA-SSB. Model names are mapped to their hyperparameter values in [54].

Metric	Best by Mean Model			Best by DTK+Mean Model		
	PriA-SSB AS	**PriA-SSB FP**	**RMI-FANCM FP**	**PriA-SSB AS**	**PriA-SSB FP**	**RMI-FANCM FP**
AUC[ROC]	RF_d	STNN-R_a	RF_h	RF_d	STNN-R_a	RF_h
AUC[BEDROC]	RF_h	STNN-R_b	RF_h	RF_h	STNN-R_b	RF_h
AUC[PR]	RF_g	STNN-R_a	RF_h	STNN-C_b	STNN-R_b	STNN-C_b
AUC[NEF]	RF_h	STNN-R_b	RF_h	RF_h	STNN-R_b	RF_h
$\text{NEF}_{1\%}$	RF_h	STNN-R_b	RF_g	RF_h	STNN-R_b	RF_h

In our prospective screen, each model prioritizes 250 top-ranked compounds, approximately 1%

of the new LC library. In this setting where each model has a fixed budget for the predicted compounds, NEF_R is a suitable metric. Therefore, we used $NEF_{1\%}$ with DTK+Means to choose the best models from each class. The best-in-class models were RandomForest_h, SingleClassification_a, SingleRegression_b, MultiClassification_b, LSTM_b, IRV_d, and ConsensusDocking_efr1_opt, with RandomForest_h being the strongest model overall (details in [54]).

1.5 Prospective Screening Results

After selecting the best model from each class based on cross-validation and the $NEF_{1\%}$ metric, we retrained the models on all 72,423 LC compounds to predict PriA-SSB inhibition using the same types of data shown in Table 1.2. This provided a single version of each model instead of one for each cross-validation fold. All models then ranked 22,434 new LC compounds that were provided blind, without activity labels. We selected the top 250 ranked new compounds from each model. Then, we experimentally screened all 22,434 new compounds to assess PriA-SSB % inhibition and defined actives based on a 35% inhibition and PAINS filters. The new binary dataset **PriA-SSB prospective** contained 54 actives.

Table 1.4 presents how many of the 54 actives were identified by each of the best-in-class virtual screening methods, along with the chemical structure similarity baseline. For comparison, randomly selecting 250 compounds from the **PriA-SSB prospective** dataset is expected to identify less than one active.

Table 1.4: The number of active compounds in the top 250 predictions from the seven selected models and the chemical similarity baseline compared to the number of experimentally-identified actives. These selected models are the best in each algorithm category from cross-validation. The last two columns correspond to the number of distinct chemical clusters from similarity or maximum common substructure clustering that are represented among the 54 actives. Complete hits on all models are in Appendix A.7.

Model	Actives	Actives not in baseline	SIM clusters	MCS clusters
Experimental	54	–	27	35
Similarity baseline	31	–	12	9
CD_efr1_opt	0	0	0	0
STNN-C_a	29	4	11	8
STNN-R_b	28	7	10	8
LSTM_b	1	1	1	1
MTNN-C_b	27	3	13	9
RF_h	37	7	13	9
IRV_d	29	4	13	9

Table 1.4 also lists the number of distinct chemical clusters identified by each method, with the goal of identifying as many diverse compounds as possible. The 54 experimental hits represent 27 SIM and 35 MCS clusters or chemical series. Commonly, virtual screening is followed by a medicinal

chemistry effort that would be expected to identify other members of these clusters.

In general, the number of distinct chemical clusters captured in the top 250 predictions is correlated with the number of actives (Table 1.4), meaning that the methods selected structurally diverse hits. The best ligand-based approaches predicted at least one compound from half of the SIM clusters but a smaller fraction of the MCS clusters. RF found the most actives and the most distinct chemical clusters.

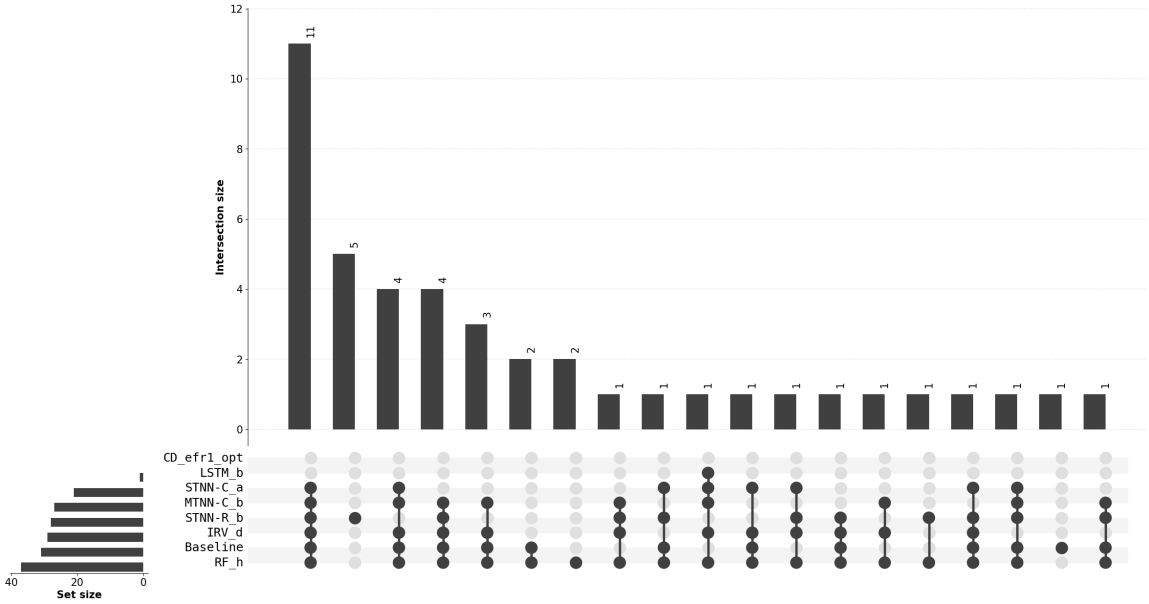


Figure 1.5: An UpSet plot showing the overlap between the top 250 predictions from the selected VS models and the chemical similarity baseline on **PriA-SSB prospective**. The plot generalizes a Venn diagram by indicating the overlapping sets with dots on the bottom and the size of the overlaps with the bar graph [51]. Altogether, the combined predictions from the best-in-class VS methods found 43 of the 54 actives.

On the **PriA-SSB prospective** dataset, only the RF model recovered more active compounds in its top 250 predictions than the chemical similarity baseline. Cross-validation with $NEF_{1\%}$ as the metric successfully identified the best PriA-SSB model before the prospective screen. The similarity baseline included one active compound that was excluded from the top 250 compounds from RF (Figure 1.5), but RF recovered a different member from this active compound's SIM and MCS clusters (Appendix A.8). The STNN-R model identified fewer total active compounds than the similarity baseline but more unique chemical clusters.

The ligand-based methods recovered many of the same actives as the chemical similarity baseline, but they also found actives that were missed by the baseline (Figure 1.5). There was a group of four active compounds that were identified by most ligand-based methods, including the baseline model. The best model, RF, identified two unique actives that were not detected by any other methods.

STNN-R was not the top performer overall but found five unique actives.

Trained Models are Target-Specific

As a control, we assessed the performance of models trained on **RMI-FANCM FP** instead of **PriA-SSB AS** for making **PriA-SSB prospective** predictions on the new 22,434 compounds. As expected, the **RMI-FANCM FP** models perform poorly on **PriA-SSB prospective** (Appendix A.6), indicating that the best **PriA-SSB AS** models have learned compound properties that are specific to PriA-SSB.

1.6 Summary

We followed a VS pipeline with the goal of maximizing the number of active compounds identified in a prospective screen with a limited number of predictions. From an initial pool of structure-based and ligand-based models, we pruned models in a hyperparameter search stage and conducted cross-validation with multiple evaluation metrics. We used DTK+Means with the $NEF_{1\%}$ metric to select the best models based on the cross-validation results and evaluated their top 250 prospective predictions from a new library of 22,434 compounds. The single best model from our pool, which was RandomForest_h for **PriA-SSB AS**, was also the top performing model on **PriA-SSB prospective**. Therefore, our overall pipeline successfully identified the best prospective model.

Metrics like AUC[ROC] can compare models in general, regardless of cost or other additional constraints [72]. However, for virtual screening in practice, one typically only experimentally tests a small fraction of all compounds in the test library. In this setting, metrics like EF that capture early enrichment are preferable. In our prospective screen, STNN-R had higher AUC[ROC] than the random forest (Appendix A.6), but the random forest found 11 more active compounds in its top 250 predictions (Table 1.4). Our study suggests that EF_R, or its normalized version NEF_R, are the preferred metrics for identifying the best target-specific virtual screening method that maximizes n_{hits} when there is a budget for experimental testing. Other metrics like AUC[ROC] or AUC[PR], which is more appropriate for problems where the inactive compounds far outnumber the actives [19], may still be reasonable for benchmarking virtual screening methods on large existing datasets where the entire ranked list of compounds is evaluated [102].

Some recent studies [50, 78, 42] reported that deep learning models substantially outperform traditional supervised learning approaches, including random forests. Our finding that a random forest model was the most accurate in both cross-validation and our prospective screen does not refute those results. Rather, it reinforces that the ideal virtual screening method can depend on the training data available, target attributes, and other factors. Therefore, careful target-specific cross-validation is important to optimize prospective performance. One cannot assume that deep learning models will be dominant for all targets and all virtual screening scenarios. We also recommend

hyperparameter exploration for all models, including traditional supervised learning methods. For example, our best random forest model contained 8000 estimators, whereas a previous benchmark considered at most 50 estimators [42].

Ramsundar et al. [77] showed that performance improved in multi-task neural networks as they added more training compounds and tasks. Furthermore, the degree of improvement varied across the datasets and was moderately correlated with number of shared active compounds among the targets within a single dataset. Task-relatedness also affects the success of multi-task learning but is difficult to quantify [38, 4]. We observed that **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** have no shared actives with any of the PCBA tasks, and multi-task neural networks were not substantially better than single-task neural networks in **PriA-SSB AS** cross-validation (Figure 1.4). The MTNN-C model outperformed the STNN-C model in the prospective evaluation (Table 1.4), possibly because multi-task learning can help prevent overfitting [81], but was still considerably worse than the random forest.

We focused on docking and well-established machine learning models instead of more recent deep learning models, such as graph-based neural networks [23, 39, 15, 60, 102]. This is because our main goal was to investigate the virtual screening principles for choosing the best model for a specific task (**PriA-SSB AS**) in a practical setting instead of broadly benchmarking virtual screening algorithms. In addition, a recent benchmark showed that conventional methods outperformed graph-based methods on most biophysics datasets [102].

Consensus docking [25] failed to recover any actives in the **PriA-SSB prospective** dataset, even though some of the individual docking programs did. It should be noted that the individual docking and consensus methods do not make use of the initial HTS screening data like the machine learning methods do. A more direct comparison might be to re-train a custom consensus scoring function to include the initial HTS data, though this effort is out of scope for this study.

Specifically for the PriA-SSB protein-protein interaction, docking is also limited by the large, flat nature of the binding site. Many compounds that are inactive in the experimental screen have good scores and reasonable binding poses (per visual inspection) but fail to interrupt necessary specific interactions in the protein-protein interface. This will the limit overall performance by pushing true actives down the ranked list. Therefore, our docking results are not necessarily representative for other targets with other structural features, once again demonstrating the need for target-specific model selection during virtual screening.

The random forest model performed the best overall, but there were ten active compounds identified by the other methods that the random forest missed (Figure 1.5). The single-task regression neural network recovered nine of those ten and three unique active compound clusters (Appendix A.8). In addition, this regression model performed the best on **PriA-SSB FP** during cross-validation (Table 1.3), possibly because there are fewer binary actives in this dataset. In future work, we will explore whether ensembling classification and regression models, potentially in

combination with structure-based VS algorithms, can further improve accuracy.

We emphasize our prospective performance on the new LC library, which minimizes the biases that make evaluation with retrospective benchmarks challenging [99]. There are many sources of experimental error in HTS, and the active compounds in the prospective evaluation must still be interpreted conservatively. However, a VS algorithm that can prioritize compounds with high % inhibition in primary and retest screens is valuable for further compound optimization even if not all of the actives confirm experimentally. Our study provides guidelines for selecting a target-specific VS model and complements other practical recommendations for VS pertaining to hit identification, validation, and filtering [105] as well as avoiding common pitfalls [84]. Having established that our best virtual screening model successfully prioritized new active compounds in the LC library, another future direction will be to test prospective performance on much larger, more diverse chemical libraries.

Chapter 2

Structural Representation and Learning

If we treat the success of deep learning as a useful but black-box function for feature representation, then it can explain why we have not seen such advance in abstract topics like drug discovery.

Some argue that machine learning models cannot outperform human's performance, from easy tasks (like image recognition) to more abstract tasks. This conjecture is interesting to justify, but the first step is to make machine understand the tasks like expert, or how to make a good representation for machine remains unsolved. This chapter explores such problems from a graph's perspective, considering that molecules have rich properties and can be treated as **structural** data.

Co-authors of [55] contributed to the contents of this chapter.

2.1 Introduction

The advancement in deep learning has brought tremendous advancement in the area of image classification and speech recognition. The benefits of applying deep neural networks in the horizon of drug discovery (prediction on molecules) are yet to be fully realized. Recent publication demonstrates poor prediction when distortion (noise) was added to images [85]. Unlike images and speech, the most common input features in drug discovery provide highly abstract representations of the chemicals, thus creating a stumbling block in making a good prediction on molecules.

Chemical fingerprints, such as the Morgan fingerprints, are perhaps the most widely used feature representations of molecules, encoding each as a fixed length bit vector. In Morgan fingerprints, each bit vector corresponds to a set of substructures and collisions may happen after the hashing operation. Therefore, Morgan fingerprints are easy for the machine to utilize but complex for a human interpretation.

Simplified Molecular Input Line Entry System (SMILES) is a character sequence describing molecular structures. There are some inherent issues in SMILES, the biggest being that molecules cannot be simply represented as a linear sequence: drug-like organic molecules usually have ring structures and tree-like branching.

Graph-based representation, on the other hand, can include the most comprehensive information for a molecule, including the molecule skeleton, conformational information, and atom features. But common molecule input representations used in machine learning models, such as chemical fingerprints or SMILES, may not be optimal or even adequate in some tasks. A fully-connected neural network is a good fit for models using Morgan fingerprints, and both convolution and a recurrent neural network can map SMILES strings into meaningful latent space. [24] introduces neural fingerprints (NEF), and it first starts to apply a graph layer on deep neural networks. Beyond this, developments in molecule representations for machine learning models are few. One limitation for graph-based models is that most efforts focus on message passing between adjacent atoms as representation, and it may over represent the local structure and ignore the molecule general shape.

Current graph-based neural networks apply message passing for information delivery and are only designed for end-to-end deep neural networks. This paper introduces a novel graph-based representation called **N-gram graph**. It allows non-deep supervised machine learning methods to reach the most up-to-date performance.

The contributions of this chapter are: (1) An introduction to a novel representation on graph-like data, **N-gram graph**. It is much simpler than existing graph-based neural network, yet the performance can compete with the most up-to-date models. (2) The N-gram graph does not require an end-to-end training process, therefore multiple non-deep supervised machine learning methods can be trained on it. (3) The N-gram graph shows very promising generalization performance on deep neural networks. (4) These findings support a conclusion that molecule representation has become a bottleneck in virtual screening tasks, and significant gains might be achieved by new representation methods that can fully utilize the expressive capacity of deep neural networks.

2.2 Related Work

Deep learning methods captured the attention among scientists in the drug discovery domain in [64, 17] Merck Molecular Activity Challange, 2012. [18, 38, 58, 61, 77, 95] Efforts expanded to investigate the benefits of multi-task deep neural network, frequently showing outstanding performance when comparing with shallow models. All of these works used Morgan fingerprints as input representations.

Another option for molecule representation is the SMILES string. SMILES can be treated as a sequence of atoms and bonds, and each molecule has a unique canonical SMILES string among a frequently vast set of noncanonical, but completely valid, SMILES strings. Therefore attempts were made to make SMILES feed into more complicated neural networks. [37] makes model comparisons

based on input features, including recurrent neural network language model (RNN) and convolutional neural networks (CNN) with SMILES, and shows that CNN is best when evaluated on the log-loss.

SMILES as molecule representation is now common in molecule generation tasks. [28] first applies SMILES for automatic molecule design, and [46] proposes using a parser tree on SMILES so as to produce more grammatically-valid molecules, where the input is the one-hot encoded rules.

Recent works are also exploring the molecule graph, which can potentially represent each molecule without losing any information. [24] first utilized message passing on graph. At each step, this method passes the hidden message layer to the intermediate feature layer, the summed-up neural fingerprints then feed into neural network as feature. Following this, [4] made small adaptations by using the last message layer as feature inputs for neural network. In work along these lines, [39] proposed a new module called weave for delivering information among atoms and bonds.

It should also be noted that there are other research lines not focusing on message-passing for graph representation. Patchy-SAN(Select-Assemble-Normalize), introduced by [73], imposes an order on nodes to construct local structure as graph features, and [60] applies weave operation on a graph. Distributed graph representations are also of interest in other domains [2, 71].

2.3 Background

Generally, molecules can be represented in three formats for machine learning models. The ideal representation contains comprehensive information for each molecule, like molecule graph. SMILES represents each molecule as a string of characters, while Morgan fingerprints maps molecules to a bit-vector, where each bit represents the existence of one set of substructures. [91] Molecular descriptors is another option, but we are not going to include that in this chapter, because heuristically coming up with descriptors and dynamically adjusting it as tasks is not easy and requires a lot of domain knowledge; one of the goal here is to get a generalized feature representation.

2.3.1 Morgan Fingerprints and Simplified Molecular Input Line Entry System

Morgan fingerprints [80] has been the most widely used for featurization in virtual screening tasks. It is an iterative algorithm that encodes the circular substructures of the molecule as identifiers at increasing levels with each iteration. In each iteration, hashing is applied to generate new identifiers, and thus, there is a chance that two substructures are represented by the same identifier. In the end, a list of identifiers encoding the substructures is folded to bit positions of a fixed-length bit string. A 1-bit at a particular position indicates the presence of a substructure (or multiple substructures) and a 0-bit indicates the absence of corresponding substructures.

Simplified Molecular Input Line Entry System (SMILES) [101] is another representation option. For any given compound structure with a valid 2-D graph representation, a SMILES string can be

generated as a 1-D line notation for the chemical structure. By this transformation, SMILES can be treated as a sequential representation and fits the setting of a recurrent neural network.

An example of Morgan fingerprints and SMILES is illustrated in Figure 1.2.

2.3.2 Graph Representation

Nearly all drug-like molecules can be potentially represented as a graph, where each atom is a node and each bond is an edge. Suppose there are in total m atoms in the graph, each atom is a_i, where $i \in \{0, 1, ...m-1\}$.

Each atom entails useful information, like atom symbol, number of charges, etc. These atom features are encoded into node attribute matrix $\mathcal{N} \in \{0,1\}^{m \times d}$, where d is the dimension of atom feature. For atom a_i, its attribute vector $\mathcal{N}_{i,\cdot}$ is defined in Equation (2.1). Adjacency matrix $\mathcal{A} \in \{0,1\}^{m \times m}$ is able to depict the skeleton of a molecule. As in Equation (2.2), $\mathcal{A}_{i,j} = 1$ iff a_i and a_j are bonded. And distance matrix $\mathcal{D}_{i,j}$ is defined in Equation (2.3).

$$\mathcal{N}_{i,\cdot} = [\underbrace{\mathrm{C}, \mathrm{Cl}, \mathrm{I}, \mathrm{F}, \ldots}_{\text{atom symbol}}, \underbrace{0,1,2,3,4,5,6}_{\text{atom degree}}, \ldots, \underbrace{0,1}_{\text{is acceptor}}, \underbrace{0,1}_{\text{is donor}}] \tag{2.1}$$

$$\mathcal{A}_{i,j} = \begin{cases} 1, & \text{atom}_i \text{ and atom}_j \text{ are bonded} \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

$$\mathcal{D}_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{2.3}$$

Each N-gram path in a graph is represented by V, and $|V|$ is the length of that path. A path with length n is represented by V_n. The set of all N-gram paths with same length is called a N-gram path set.

2.3.3 Message Passing in Graph-based Neural Network

In recent works, message passing has been dominant in graph-based neural networks. Message passing has T iterations. At step t, each atom will pass its information only to neighbors. After continuing for T steps, each atom is able to pass its own information to atoms at most T-steps away. Therefore, message passing is capable of encoding molecule local structure within the radius of T around each atom.

Let the intermediate matrix at step t be $\mathcal{M}_t$, and operation $\mathcal{A} \cdot \mathcal{M}_t$ allows each atom to pass its own information to its neighbors, where $\cdot$ is the matrix multiplication. Message passing will then multiply it with hidden layer H_t followed by activation function σ. Repeat this process for T times,

and the output matrix $\mathcal{M}_T$ is assumed to capture the molecule information. This can be nicely written in Equation (2.4). $\mathcal{M}_0 = \mathcal{N}$ at initial step.

$$\mathcal{M}_{t+1} = \sigma(H_t[\mathcal{M}_t + \mathcal{A} \cdot \mathcal{M}_t]) \tag{2.4}$$

One characteristic of graph-based representation is that its restriction applies only to end-to-end deep neural networks, where all parameters are learned by back-propagation.

2.4 Structural Learning Method: Graph-based Representation

2.4.1 Motivation

This section helps explain the motivation to combine adjacency matrix $\mathcal{A}$ and distance matrix $\mathcal{D}$ for input representation.

The **adjacency matrix** can be used to represent the neighborhood structure for each atom. But it lacks the ability to capture overall molecular structure.

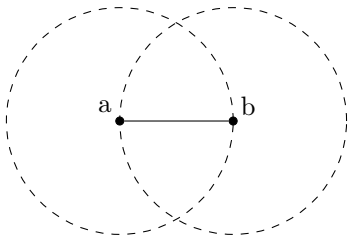
The **distance matrix** is defined as the distance between each pair of atoms in a molecule. In other words, the distance matrix is able to capture the global structure, but may miss some local information.

For example, both atom a and c are bonded with atom b, and they are close in the euclidean space (either 2-D or 3-D), but not directly bonded. In this case, the adjacency matrix can reveal the skeleton from a to b to c, while the distance matrix is able to distinguish that atom a, b, and c are close in the euclidean space. Combining such two matrices for molecule representation, each molecule can be potentially represented in a lossless way. Therefore it is possible to get better performance for drug discovery using this representation. In figure 2.1, it illustrates how adjacency matrix and distance matrix together specify the graph structure in a 2D euclidean space.

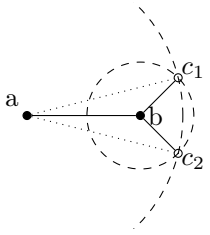
2.4.2 Relaxation and Problem Formulation

This chapter formulates the problem as follows. For each molecular graph, an attempt is made to find a shared substructure among all the actives. The substructure must satisfy some latent constraints, including the atom types, atom distance, and bond types. The predicted molecules will have positive labels if and only if they contain such substructure, otherwise the molecules should have inactive interactions against the target.

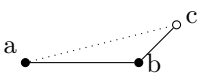
The substructure is represented by a candidate set $s = \{0, 1\}^{n \times 1}$, where each one bit in s means this atom is crucial for the target task. One relaxation strategy is, instead of requiring the atoms with exact positions and exact symbols, only adding exact constraints on the atom positions; and



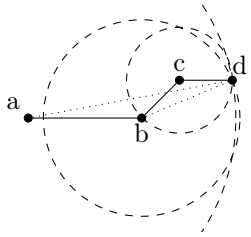
(a) stage 1, fix atom a and b locations. The distance between atoms is given by distance matrix.



(b) Stage 2, construct the molecule graph following the adjacency matrix. There are two possible locations for atom c.



(c) Stage 3, choose c_1 as atom C.



(d) Stage 4, all the remaining atoms will be uniquely defined in this 2D space after fixing first three atoms.

Figure 2.1: Illustrations on how the adjacency matrix and the distance matrix can be combined to recover a graph structure in 2-D euclidean space.

for atom symbols, the only constraints is that each positive molecules should contain the exact total number of key atoms. After relaxation, this problem can be interpreted in the following rigorous way. Suppose each molecule has n atoms, and the goal is to find if there exists one candidate set s in this molecule, such that

1. $\mathcal{N} \cdot s = c_1$
2. $\mathcal{A} \otimes (s \cdot s^T) \cong c_2$
3. $\mathcal{D} \otimes (s \cdot s^T) \cong c_3$

, where $\cdot$ is the matrix multiplication, $\otimes$ is the element-wise multiplication, and $A \cong B$ means A and B are two matrices with order-invariant property.

In the above constraints, $c_1 \in \mathbb{R}^d$ in Item 1 represents the number of each feature in the potential structure, like atom symbol and atom degree. The $c_2 \in \mathbb{R}^n$ and $c_3 \in \mathbb{R}^n$ are the corresponding skeleton and affinity structures.

The complete pipeline for structure-based neural network is shown in Figure 2.2. However, under this setting, the candidate set s is sensitive to the atom order, therefore **order invariant** becomes the biggest issue. Like shuffling the atom orders, same molecule may have totally different candidate set s.

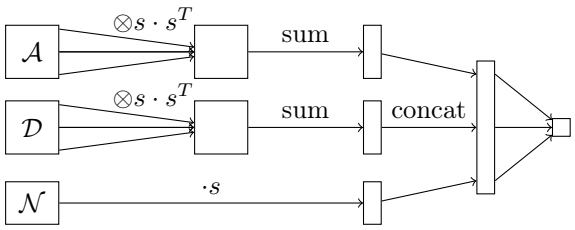


Figure 2.2: Pipeline for Structure Graph Neural Network.

In following sections, we will introduce a primary solution called **N-Gram Graph**. We will go step by step, from problem formulation to motivation of the novel representation in Section 2.4.3. Segmented random projection and N-gram graph will be explained in Section 2.4.4 and Section 2.4.5 respectively.

2.4.3 N-gram Graph: A Novel Molecule Representation

N-gram graph is an order-invariant representation for a graph. It splits the graph into different N-gram path sets. The high-level process for the N-gram Graph is described as follows:

1. Apply **atom**-level representation, segmented random projection.
2. **N-gram path** is formulated by the production of all the nodes (atoms) in that path.
3. For a fixed path length, sum up all the N-gram paths to denote **N-gram path set**.
4. Concatenate N-gram path sets with multiple N, and output is the **N-gram graph**.

Formulation on N-Gram Graph

For each molecular graph, the goal is to find a set of shared substructures among all the active molecules. The substructure should satisfy several constraints, like the atom features and relative positions. The predicted molecules will have positive labels if and only if they contain similar substructures.

The target substructures can be represented by a candidate set $C \in \{0,1\}^{m\times 1}$, where each bit set to 1 in C means this atom is crucial for the target task. Therefore drug discovery problems can be interpreted as follows.

$$\mathcal{N}^T \cdot C = c_1 \tag{2.5}$$

$$\mathcal{A} \otimes (C \cdot C^T) \cong c_2 \tag{2.6}$$

, where $\cdot$ is the matrix multiplication, $\otimes$ is the element-wise multiplication, and $A \cong B$ means A and B are isomorphic. For the above constraints in Equation (2.5), $c_1 \in \mathbb{R}^d$ represents the number

of each feature in the candidate substructures, like atom symbol and atom degree, and $c_2 \in \mathbb{R}^m$ is the skeleton among candidate substructures.

One issue for the candidate set C is that it is sensitive to the atom ordering. Once the indices of atoms are switched, it's still the same molecule but representation may change totally. And this is the motivation for introducing **N-gram graph**, an order-invariant graph representation.

2.4.4 Atom Level: Segmented Random Projection

Before introducing the novel representation, we need to apply **segmented random projection** as atom-level feature. Recall that $\mathcal{A}$ is the adjacency matrix, and $\mathcal{N}$ is the node attribute matrix. Atom features can be treated as the combination of S feature segments, where each segment is a one-hot vector. Similarly, the node attribute matrix can be divided into S segments, $\mathcal{N} = [\mathcal{N}^0, \mathcal{N}^1, \ldots, \mathcal{N}^{(S-1)}]$.

In natural language processing tasks, a word is first mapped to a one-hot vector. Recent work [7] has shown that random embedding for words can be simple yet effective, and we extend this idea to map each feature segment into a random space, from one-hot vector to one-hot matrix. The segmented random projection is the concatenation of S one-hot matrices.

Let $\mathcal{G} = [\mathcal{G}^0, \mathcal{G}^1, \ldots, \mathcal{G}^{(S-1)}] \in \mathbb{R}^{S \times r \times d}$ be the randomized Gaussian matrix, where d is the dimension of the atom feature and r is the dimension of the random space. It can be divided into S segments according to atom features. $\mathcal{N}_{i,\cdot}$ is the feature for atom a_i, and g_i is the corresponding randomized representation. The segmented randomized projection function $f : \mathcal{N}_{i,\cdot} \to g_i$ is defined in Equation (2.7).

$$\begin{aligned} g_i &= f(\mathcal{N}_{i,\cdot}) \\ &= f([\mathcal{N}^0_{i,\cdot}, \mathcal{N}^1_{i,\cdot}, \ldots, \mathcal{N}^{(S-1)}_{i,\cdot}]) \\ &= [\sum(\mathcal{G}^0 \cdot \mathcal{N}^0_{i,\cdot}), \sum(\mathcal{G}^1 \cdot \mathcal{N}^1_{i,\cdot}), \ldots, \sum(\mathcal{G}^{(S-1)} \cdot \mathcal{N}^{(S-1)}_{i,\cdot})] \end{aligned} \tag{2.7}$$

, where $\sum$ is the summation along the axis of feature dimension d_s. $\sum(\mathcal{G}^s \cdot \mathcal{N}^s_{i,\cdot}) \in \mathbb{R}^{r \times 1}$ is the random projection on s-th feature segment for a_i. Figure 2.3 describes the whole projection process.

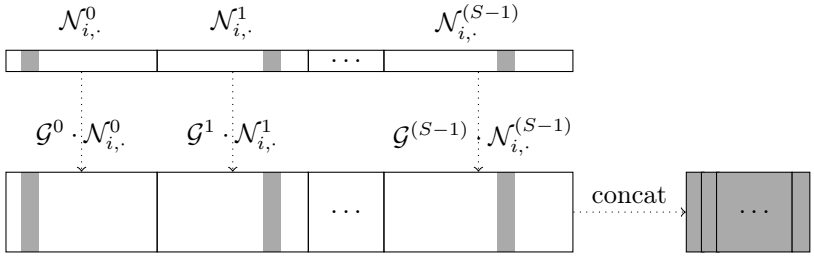


Figure 2.3: Segmented random projection on atom a_i. Each atom's features can be split into S segments. Each group of feature with dimension d_s corresponds to a one-hot vector $N_{i,\cdot}^s \in \{0,1\}^{1\times d_s}$ (marked in grey). This vector is then multiplied by the Gaussian random matrix $\mathcal{G}^s \in \mathbb{R}^{r\times d_s}$, yielding a projection into a random space. For each randomized atom feature g_i, the only non-zero column in output matrix $\mathcal{G}^s \cdot \mathcal{N}_{i,\cdot}^s$ in each segment will be extracted and concatenated.

2.4.5 Molecule Level: N-Gram Graph

Atom ordering becomes one of the biggest challenges under the previous problem formulation. Reordering atoms in one molecule will not change its physical or molecule properties, but the candidate set C is not capable of recognizing this difference. Adding an order-invariant representation seems to be a reasonable solution.

The N-gram approach is a classic technique used in natural language processing. It represents a sentence as counts of the contiguous sequence of N words in the sentence. Viewing words as atoms and sentences as linear molecules inspire us to come up with a N-gram method for graph representation. Based on segmented random projection on atoms, this paper proposes a novel molecular representation named **N-gram graph**.

Let $\mathcal{V}_n \in \mathbb{R}^{r\times S}$, $p \in \{1, 2, \ldots, N\}$ represent the **N-gram path set**. It is defined as the sum of all **N-gram paths** with length n. And each N-gram path is represented by the product of randomized atoms $f(a_i)$ in that path. See definition in Equation (2.8).

$$\mathcal{V}_n = \underbrace{\sum_{\forall V, \text{s.t. } |V|=n} \overbrace{\prod_{a_i \in V} \underbrace{f(a_i)}_{\text{segmented random projection}}}^{\text{n-gram path}}}_{\text{n-graph path set}} \tag{2.8}$$

N-gram graph for each molecule $\mathbb{G} = [\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n] \in \mathbb{R}^{N\times r\times S}$ is the concatenation of N-gram path sets with multiple length n. Notice that with N-gram graph representation, each column of $\mathbb{G}$ corresponds to the path representation with different length. In comparison to the message-passing in the end-to-end graph neural network or Morgan fingerprints generation, the N-gram graph representation can offer a finer-grained view of the molecules, in the sense that it separates

different local structures by path length. Moreover, when generating $\mathbb{G}$, only path information is used, whereas in a message passing graph, important information can get delivered back-and-force within each pair of adjacency atoms, therefore it may lead to a biased representation focusing more on paths with smaller length.

To build up models, N-gram graph is flattened into a 1-dimensional vector for compatibility as input features into non-deep models like random forest or XGBoost. Deep neural networks can directly apply fully-connected layers on the N-gram graph with flatten operation between some intermediate layers.

2.5 Experiments

Six models and three different feature representations were tested on 12 classification and 3 regression tasks. The corresponding feature representation and model combinations are listed in Table 2.1.

Table 2.1: Feature representation for each different machine learning model. NEF, GCNN, and Weave are end-to-end neural networks.

Model	Feature Representation
Neural Fingerprints (NEF) [24]	Message-Passing Graph
Graph CNN (GCNN) [4]	Message-Passing Graph
Weave NN (Weave) [39]	Message-Passing Graph
Random Forest (RF)	Morgan fingerprints / N-Gram Graph
XGBoost (XGB) [11]	Morgan fingerprints / N-Gram Graph
Fully-connected Deep Neural Network (DNN)	Morgan fingerprints / N-Gram Graph

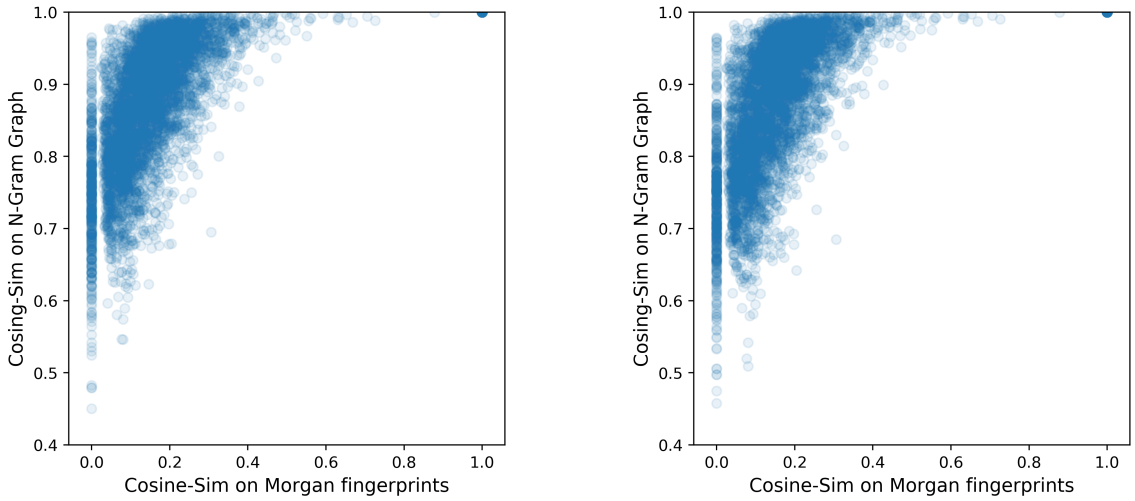
For N-gram graph, the specific $d = 42$ dimension features and $S = 8$ segments can be found in Appendix B.2. NEF, GCNN and Weave use hyperparameters in [24, 4, 39]. For other models, we run a comprehensive grid search for hyperparameter sweeping and explore two non-deep machine learning algorithms, RF and XGB. All data sets are split into five folds with one selected as hold-out test set. We run a comprehensive grid search for hyperparameter sweeping, and all the details and implementation are available on GitHub (https://github.com/chao1224/n_gram_graph).

2.5.1 Qualitative Analysis

To further explain how N-gram graph can help with representation, pairwise cosine similarities on different representations are compared. Here we compare representations from a randomly sampled subset of molecules from Delaney dataset [20].

As displayed in Figure 2.4, similarity based on Morgan fingerprints tends to concentrate around 0, while N-gram graph is inclined to make molecules concentrate on similar representation. This is not telling us which representation is better or worse, and it is just an observation. However,

some molecule pairs are observed to be overlapped on Morgan fingerprints while N-gram graph can distinguish among them.



(a) $N = 6, r = 50$ (b) $N = 6, r = 100$

Figure 2.4: Comparison of pairwise molecule similarity between Morgan fingerprints and N-gram graph. Larger random dimension ($r = 100$ on the right) shows slightly wider distribution than lower dimension ($r = 50$ on the left). The vertical band of points on the left side of the plots show compounds with no detectable similarity (cosine similarity=0) by Morgan fingerprints representation that do show various levels of similarity when comparing N-gram graph representations.

2.5.2 Classification Tasks

Tox21: [92] "Toxicology in the 21st Century" initiative created a public database measuring toxicity of compounds, which was used in the 2014 Tox21 Data Challenge. Twelve representation and model pairs are tested on twelve tasks, and Table 2.2 summarizes the AUC[ROC] on the test set.

Table 2.2: AUC[ROC] on test set on Tox21. Top three results are **bolded** and the best performance is **underlined**. Each row corresponds to a task, except that last row measures the general performance over all tasks.

Representation	Morgan			Message-Passing Graph			N-Gram Graph		
Method	RF	XGB	DNN	NEF	GCNN	Weave	RF	XGB	DNN
NR-AR	0.820	0.822	0.751	0.723	**0.843**	0.796	0.826	**0.839**	**0.830**
NR-AR-LBD	**0.882**	0.853	0.808	**0.859**	**0.869**	0.822	0.841	0.845	0.845
NR-AhR	**0.883**	**0.887**	0.850	0.846	0.881	0.869	0.870	**0.884**	0.859
NR-Aromatase	0.828	0.782	0.693	0.759	0.826	0.802	**0.849**	**0.833**	**0.832**
NR-ER	0.717	**0.742**	0.709	0.690	**0.739**	**0.742**	0.697	0.704	0.729
NR-ER-LBD	**0.837**	**0.827**	0.796	0.810	0.803	0.783	0.812	**0.829**	0.802
NR-PPAR-gamma	0.809	0.709	0.738	0.743	0.784	**0.829**	**0.826**	**0.817**	0.772
SR-ARE	**0.838**	**0.839**	0.781	0.759	0.804	0.813	**0.827**	0.826	0.794
SR-ATAD5	**0.865**	0.808	0.746	0.727	0.843	0.837	**0.858**	**0.844**	0.808
SR-HSE	**0.797**	0.745	0.723	0.747	0.770	**0.778**	0.773	**0.821**	0.764
SR-MMP	0.874	0.866	0.870	0.854	0.875	**0.878**	**0.893**	**0.905**	0.862
SR-p53	**0.875**	**0.869**	0.710	0.813	0.822	0.808	0.843	**0.872**	0.794
Average	**0.835**	0.812	0.765	0.778	0.822	0.813	**0.826**	**0.835**	0.807

After a thorough hyperparameter sweeping, RF on Morgan fingerprints in Table 2.2 shows promising results. Other methods, like GCNN and Weave on Message-Passing Graph, also exhibit competitive performance. Switching from Morgan fingerprints to N-gram graph representations appears to benefit XGB and DNN performance, where XGB model with n-gram graph representations show highest overal ROCAUC. Though all the test set performance are similar, as shown in Appendix B.3.2, N-gram graph on DNN possesses quite stable performance.

However, we are not able to conclude which representation and model pair can make the best prediction. As observed from Table 2.2, no model can be singled out as the top performer, and best algorithm may vary according to different tasks. In addition, performance is very sensitive to data splits and hyperparameters. In many domains, a comprehensive representation and a deep neural network can outperform other modeling frameworks. Yet we have not observed that in this domain, and the bottleneck seems likely to be in the compound representation.

2.5.3 Regression Tasks

Compound representations were also compared on three different regression tasks from the same datasets used in [24].

- **Delaney:** 1144 molecules were measured with respect to the aqueous solubility [20].

- **Malaria:** [27] measures the drug efficacy of 10,000 molecules against the parasite that causes malaria.
- **CEP:** A subset of 2,000 molecules from Havard Clean Energy Project (CEP) [31]. It aims at estimating organic photovoltaic efficiency.

Table 2.3: RMSE on three regression tasks (test set). Top three results are **bolded** and the best performance is **underlined**. Baseline results (*) are from [24, 39].

Representation	Method	Delaney	Malaria	CEP
Morgan	RF	1.251	**1.011**	1.667
	XGB	1.120	**0.998**	1.442
	DNN (*)	1.40	1.13	2.00
Message-Passing Graph	NEF (*)	**0.52**	1.15	1.43
	GCNN	0.98	1.02	**1.17**
	Weave (*)	**0.46**	1.07	**1.10**
N-Gram Graph	RF	0.802	**1.011**	1.367
	XGB	0.771	**1.003**	**1.296**
	DNN	**0.665**	1.085	1.359

As demonstrated in Table 2.3, performance on regression tasks varies a lot. But we can still get some common observations out of it. When comparing N-gram graph with Morgan fingerprints, all three models can obtain better RMSE in most cases (except RF and XGB on Malaria). Message-passing graph shows slightly better performance on Delaney and CEP, but other models based on n-gram graph are very comparative.

2.6 Summary

This chapter introduces a novel graph-based representation called **N-gram graph** for virtual screening task. The first step in tackling this task is the derivation of problem formulation which requires an order-invariant representation. Then, the idea of N-gram is taken and extended to graph data and this can be applied to most supervised machine learning methods. After a comprehensive hyperparameter sweeping, we show the potential benefits of N-gram graph by reaching state-of-the-art performance on many benchmarks using different classification and regression models. However, the limitation is that N-gram graph does not take the edge information or molecule shape into consideration, which might be important for making predictions.

The work here suggests an interesting way to handle graph-based representation. A future direction for this work would involve deeper exploration of molecule representation and molecule

generation based on the randomized latent space.

Chapter 3

Reinforced Multi-task Learning

3.1 Introduction

Multi-task classification as one of transfer learning methods, has shown remarkable advantages in natural language processing, computer vision, and [17, 77] drug discovery arenas. An assumption in multi-task learning application is that the tasks are potentially related, and comparing to the single-task setting, training multiple tasks together can lead to a better represented feature space, obtain higher performance, and alleviate the data insufficiency issue.

Traditional MTL methods are based on shallow models. But now, as the deep neural network becomes prevalent, more work has adopted deep multi-task learning–with examples arising in diverse applications such as the [64, 17] Merck Molecule Activity Challenge, [56] facial attributes detection, language translation, and [36] wellbeing detection. But experiments illustrate that multi-task learning is not always stable. We refer to this phenomenon as **negative transfer**, where unrelated tasks in joint-training show decreased performance relative to that from single-task training. There are two main reasons: either there exists one domain group of tasks taking over the training process towards its dominant representation space, or all the tasks are widely spread, and heading towards different latent space can result in a useless latent representation. On average MTL can outperform STL, but there's no guarantee when focusing on target tasks, the remaining ones can act as auxiliary tasks that support the main training.

So here, we first try to address this negative transfer issue in deep multi-task training by proposing a general reinforced multi-task learning (RMTL) framework. The goal is to train a distilled policy for the multi-task neural network. Originally, the reinforcement learning was to solve the decision-making problem in gaming and robotics [67]. The target agent learns how to make decisions based on partially observed environments and scalar rewards; by enough trials and errors, it can learn an optimal action given the environment state. To apply this idea in our RMTL framework, we trained an agent model, which given the state of multi-task network, automatically decides which group of

similar tasks to be updated in each epoch.

3.2 Related Work

3.2.1 Self-Paced Curriculum Learning

Self-paced curriculum learning borrows idea from education: it is reasonable for people to take curricula in progression from easy ones to complex ones. Curriculum learning suggests using easy samples to train the model first, then continue by adding more complicated samples. The easiness is hard to determine, and [45] introduces self-paced learning, where the easiness is identified by the learned model.

Then following works extend this from single-task model to multi-task case. [52] introduces a joint objective function by adding weights and regularizers on each task. And [69] uses a threshold on the residual of each task to guide which group of tasks are easy, so as to update gradients. Above methods need the assumption that all tasks are related, and [76] handles dissimilar tasks by applying multiple task learning sequences.

3.2.2 Clustering-based Multi-task Learning

Many examples have been proposed in literature that use clustered tasks, grouping tasks using different notions of grouping. Some methods assume that parameters of tasks that can be grouped in the same cluster are either close to each other in some distance metric or share a common probabilistic prior, whereas tasks assigned in different clusters didn't interact with each other [35, 10]. Similar to the method proposed in this chapter, many methods are trying to find a probabilistic model that attempts to extract a covariance matrix among task pairs for use in training predictors[104, 30].

Another assumption in clustering-based method is that task parameters lie in a low dimensional subspace, which captures model structure shared among all tasks. Some methods assume that some features (with all raw and transformed features contained) are inactive for all tasks. And thus all tasks parameters are pushed to a low dimensional subspace[6]. Other methods follow the low dimensional subspace assumption but allows the tasks in different task groups to overlap with each other in one or more bases [44].

3.2.3 Deep Multi-task Learning

Classification tasks have been increasingly addressed by deep neural network, which benefits from taking the raw data as input, extracting the latent features, so as to make better predictions.

However, only a few works [82] have focused on a better generalized deep multi-task framework. Ensembling is one traditionally adopted option, and [49] introduces a similar idea called TreeNets, an ensembled deep neural network. In TreeNets, the first part of layers are shared among tasks, and

following layers are trained independently. [56] proposes a framework that is able to dynamically construct a hierarchical network structure and selectively share information among closely-related tasks. Both methods require vast computation memory, especially when the hidden space is in high dimension.

3.2.4 Meta Learning and Reinforcement Learning

Meta learning on deep networks is a novel approach. Meta learning, or learning to learn, aims at making a model being able to learn the learning process, so it can generalize well on new tasks or new samples. When it comes to deep network framework, there are many interesting and potentially constructive points, like optimal learning rates, batch size, predicting loss, and gradients. There has been recent work on such idea, [5] uses a meta learner to learn the gradient, and [79] predicts the next step hidden layer parameter with recurrent neural network.

Reinforcement learning provides another option for meta learning. The intuition behind reinforcement learning is that once a model or agent observed the environment, with the feedback, it can update its policy space to make decisions towards higher expected rewards. [26] Therefore the policy agent can fit well as meta learner, considering that agent can apply meta learning to find a better base model, facilitating quick transfer to new unseen but related tasks under some task distribution, where only a few data points are available. This setting is also called few-shot learning, where insufficient data is very common on new tasks, and models are supposed to converge well within a few epochs on limited data. Distral [89] proposes a framework for simultaneously training multiple reinforcement tasks by learning a distilled policy, but this is not applicable towards the multi-task classification problems.

3.3 Problem Background

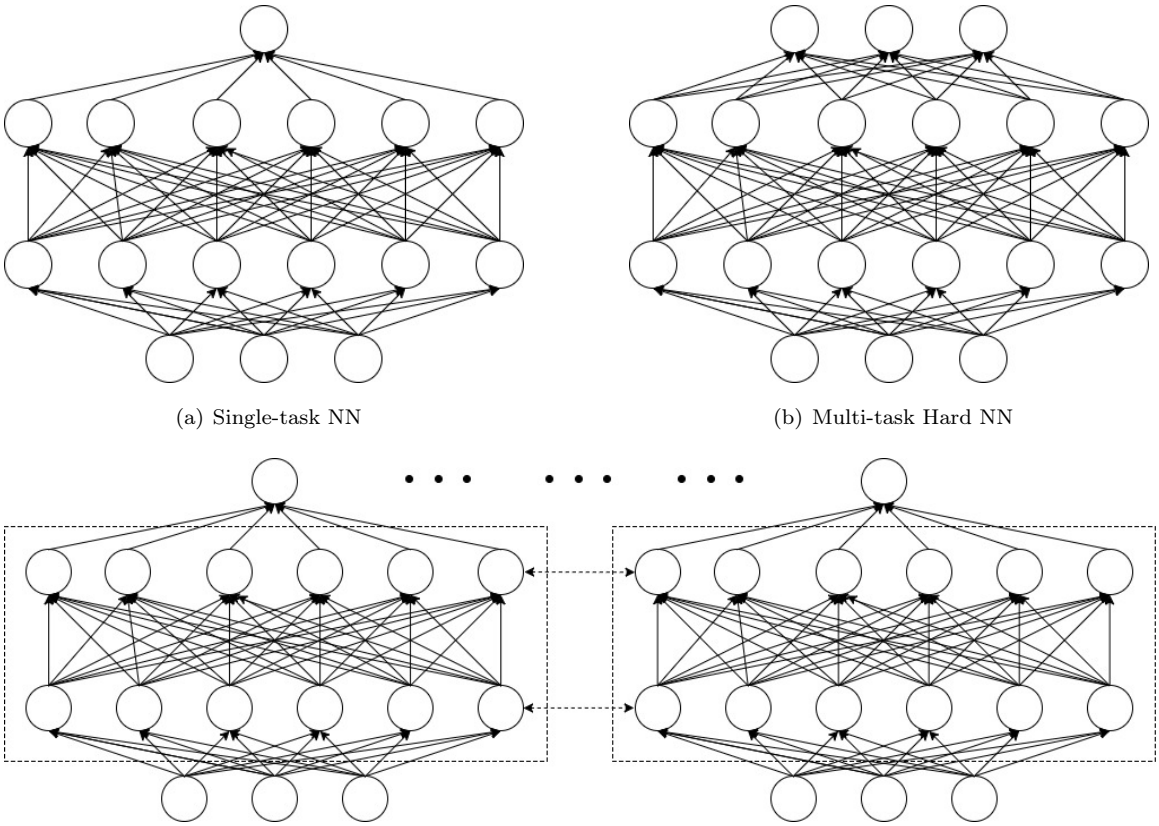
3.3.1 Annotation

Suppose we have T tasks, corresponding to T datasets, and each dataset $S_t = \{X_t, Y_t\}, t \in \{0, 1, \ldots, T-1\}$. All the instance-task labels can compose a complement matrix $C \in \mathbb{R}^{n \times T}$, where n is the number of instances in its union set and T is the number of tasks. All the missing values are filled with 'None' in the complement matrix C. In the context of stochastic gradient descent, one epoch, or one data pass, refers to using all samples for gradient updates once. W is the weight parameters for neural network.

In reinforcement learning, a problem is episodic if it has a repeated decision making pattern, where each round of decision making process is called episode. Policy π is an action to make best decision with respect to some predefined reward function, where the reward function returns a value after the task environment changes.

3.3.2 Deep Single-task and Multi-task Learning

Deep Single-task learning treats each column of complement matrix C independently, and trains T independent tasks. Multi-task learning uses either **hard parameter sharing** or **soft parameter sharing** to transfer knowledge, depending on network construction. In **hard parameter sharing** network, all the hidden layers are shared, except the output layer. While in **soft parameter sharing**, each model has its own layers and parameters. Constraints are commonly applied to force similarity among models of tasks, for example implementing ℓ_2 regularization [22] or trace norm [103] on distances among model parameters. Figure 3.1 presents more details.



(a) Single-task NN

(b) Multi-task Hard NN

(c) Multi-task Soft NN

Figure 3.1: Deep Neural Network Structures. A Single-task NN in Figure 3.1(a) has only one unit on output layer, and requires T independent neural networks for T tasks. A Multi-task hard NN in Figure 3.1(b) shares parameters except the output layer, and each one unit in the output layer corresponds to one task predictions. For Multi-task soft NN in Figure 3.1(c), each task has its own model, but parameters are constrained to force similarity. Therefore, this only needs one such network for T tasks.

The intuition behind MTL approaches is to transform knowledge among tasks, augment representation learning among tasks, and improve performance especially when they are highly correlated. Besides, when data are insufficient, tasks can benefit from learning representation from closely related ones. The structures of neural networks are illustrated in Figure 3.1.

3.4 Methodology

3.4.1 Multi-task Reinforcement Learning

One of the biggest differences between shallow and deep models is the training process. In a shallow model, all the features are provided, either by manual extraction or feature extractor model. While in a deep model, only raw data are required from which the model learns its own latent space. Therefore, we can divide the training process of deep neural networks into two stages, encoding and decoding. Encoding maps raw input data to meaningful latent features; decoding maps latent features to an output scalar (single-task) or vector (multi-task).

Traditional MTL works well under the assumption that most of tasks are correlated, and therefore knowledge can be transferred across the tasks. Negative transfer will happen when this assumption does not hold. And here we propose a generalized framework called reinforced multi-task learning to address this issue. The idea comes very naturally: suppose there exists a generalized representation space for all tasks, and each tasks will possess its own decoding process after mapping to the latent space. The two causes of negative transfer may easily occur during decoding. Therefore, we believe that negative transfer might be mitigated by decoding only similar tasks at one time.

To put this in the reinforcement learning context, when given a deep network state, the policy agent will guide the network to update towards its optimal solution. In the RMTL setting, the state is the parameter W of network and the action can be a subset of tasks to update. So the action $a \in A = \{0, 1\}^T$ is task selection for gradient updates. Each epoch will be treated as one episode in the reinforcement learning context, therefore this RMTL can be treated as an episodic learning process. The agent will get a return value from the environment, like evaluation value on a hold-out dataset, and it aims for higher return values. If we are using Q-learning as exact solution, the Q-function is a mapping $W \times A \to \mathbb{R}$.

Reinforcement learning has two main strategies, value-based and policy-based. We will introduce both of them, compare the pros and cons, and illustrate how to select the condition.

3.4.2 Value-based Method

The value-based method learns the policy by maximizing the expected state-function values. At time step t, we choose the best policy π and use this policy to update the multi-task network weights.

$$\pi_{t+1}(W_t) = \underset{a}{arg\,max}\, Q(W_t, a)$$

The $Q(W_t, a)$ is a value function that maps the state space and action space to the reward space. [86] Temporal difference (TD) and Monte-Carlo (MC) are two traditional methods to approximate Q-value. Algorithm 1 sketches this iterative approach.

Algorithm 1: Value-based Method

Initialize Neural Network W_0, initialize $\pi_0 = \{1\}^T$
repeat
  $\pi_{t+1}(W_t) = \underset{a \in \{0,1\}^T}{arg\,max}\, Q(W_t, a)$
  $\mathcal{L} = \sum_{i=1}^{T} \pi_{t+1}^{(i)} \cdot \mathcal{L}(\mathcal{Y}^{(i)}, f(\mathcal{X}^{(i)}, W_t^{(i)}))$
  update weight W_{t+1}
until Convergence

If the action space is discrete, we can enumerate all possible policies. But if the action space is too large or continuous, Monte-Carlo can guarantee convergence by law of large numbers.

The classical value-based methods have one big drawback: given discrete action space, the computation time will grow exponentially as the number of tasks increases. One solution is to use a parametric model, like neural network in deep Q-network (DQN) [66], to approximate the Q-value. To be more specific, in our case, we use a meta-RL agent to model the policy, π_θ, where the input is the model state (like hidden layer parameter), and the output is the probability of each task updated in current epoch.

3.4.3 Policy-based Methods

Once we have a parametric model to approximate the Q-value, instead of applying it in the bootstrap framework for policy selection, a natural approach would be to directly output the policy. A more current method is to use policy gradient to train a parameterized (θ) policy π_θ, where the goal is to maximize the expectation $\mathbb{E}_{a\sim\pi_\theta}[Q(W, a)]$. The gradient is as follows:

$$\bigtriangledown_\theta \mathbb{E}_{a\sim\pi_\theta}[Q(W, a)] = \mathbb{E}_{a\sim\pi_\theta}[Q(W, a)\nabla_\theta\, log\, \pi_\theta(a|W)]$$

Applying this for gradient updates, and at each time step t, we determine next action according to this newly updated distribution. Details are described in Algorithm 2.

Algorithm 2: Policy-based Method

Initialize Neural Network W_0, and π_θ
repeat
$\theta_{t+1} = \theta_t + \alpha \cdot \mathbb{E}[Q(W_t, a)\nabla_{\theta_t} log\, \pi_{\theta_t}(a|W_t)]$
$\mathcal{L} = \sum_{i=1}^{T} \pi^{(i)}_{\theta_{t+1}} \cdot \mathcal{L}(\mathcal{Y}^{(i)}, f(X^{(i)}, W^{(i)}_t))$
update weight W_{t+1}
until Convergence

3.4.4 Pipeline

In Figure 3.2 we describe the complete pipeline of our framework. The solid arrow stands for the policy improvements based on the state-function or value-function. The dashed arrow is for policy-gradient based method.

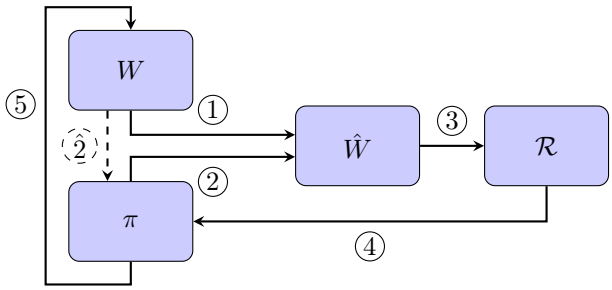


Figure 3.2: Pipeline for the multi-task reinforcement learning. At the start of each step t, $W = W_t$ is the deep neural network parameter. Run several episodes as trials, update $\hat{W}$ by taking ① current state W_t and ② the sampled policy π. Get ③ the return value and ④ use it as feedback to select best policy π. This optimal policy can then be used to ⑤ update the deep network parameter. In the case of policy-based method, policy is parameterized by θ. At the beginning of each epoch, ② will map from state W_t to an action, and use this selected group of tasks for trial. Correspondingly, ④ π_θ will be updated upon receiving return values.

In general, given the state or the weights of the multi-task network, and following some stochastic distribution, we can forecast the updated weight $\hat{W}$, which is the state of next epoch $t+1$. Since we have the updated weight, it is reasonable to obtain the return target, some pre-defined function with respect to W_t and $\hat{W}$. And according to Figure 3.2, this return target is $\mathcal{R} = \Delta[W_t, \hat{W}] \in \mathbb{R}^T$. Once we have this return, we can treat it as oracle for policy improvement at epoch t. We will repeat this forecasting step until reaching an acceptable policy distribution, then continue to the next epoch.

3.4.5 Reward Function on A Focused Task

There's a very common desire for domain experts to transfer domain knowledge towards one specific task, and we call this focused task. When the focused task has only a few data points, the problem is equivalent to few-show learning.

The reward function is the key component in reinforcement learning problems, given that the ultimate outputs can be very sensitive to it. It can be very sensitive to the ultimate outputs. This section will explore an applicable strategy for developing a reward function in the setting with one focused task.

Our motivation for the reward function is that task relatedness may not be easily represented by a latent task distribution. However, if we take a closer look at each data point separately, we can tell how tasks are relevant with respect to the data, and therefore useful information can be extracted. Here we apply cosine-similarity on task pairs based on gradient of each batch of data as reward scores. Details are explained in Algorithm 3.

Algorithm 3: Reward Function

```
Given a focused task.
Initialize neural network W, with depth d.
for batch of data B do
  for all candidate policy π. do
    Get gradient w.r.t. batch of data on the last shared layer ∇_{d-1}B.
    Compute cosine-similarity v w.r.t. the focused task based on ∇_{d-1}B.
    Fetch reward, r = max(v, 0).
  end for
  Select best π according to r.
  Multiply gradients ∇B by π · r.
  Update W by backpropagation.
end for
```

Algorithm 3 is the core function in RMTL. It reveals how we can get a finer-grained operation in the reinforcement learning structure. By assumption, how helpful each task in the policy π to the focused task is proportional to the corresponding cosine-similarity. In Algorithm 3, we get the reward by $r = \max(v, 0)$. This reports as to whether a candidate task in the policy has a negative effect to the focused one, in which case we will reduce its impact by multiplying 0.

3.5 Experiments and Preliminary Results

3.5.1 Experiment Setup

We test our RMTL algorithm on virtual chemical screening tasks, where the goal is to predict the biochemical activity or other properties of a chemical compound. In this domain, the number of

labeled training examples is typically limited, making multi-task learning appealing.

We are conducting experiments on one virtual screening dataset. In the PubChem BioAssay (PCBA) dataset [100], each task is a binary classification problem. The goal is to predict whether a chemical is active for any of 128 properties/tasks. For molecule representation, we use the Morgan fingerprints. It encodes each molecule structure into 1024 binary bits, where each bit represents one substructure. Previous studies using this dataset with multi-task neural networks exhibited negative transfer for some tasks [77, 78]. Thus this dataset is desirable for our efforts to eliminate negative transfer with RMTL.

3.5.2 Performance Comparison

For preliminary results, we picked 3 pairs of similar tasks, where there is one focused task and one policy task in each pair. Both single- and multi-task deep neural network (STDNN and MTDNN) are tested. For RMTL, each task is treated as the focused task once, and the other one in the corresponding task pair is the only task for policy.

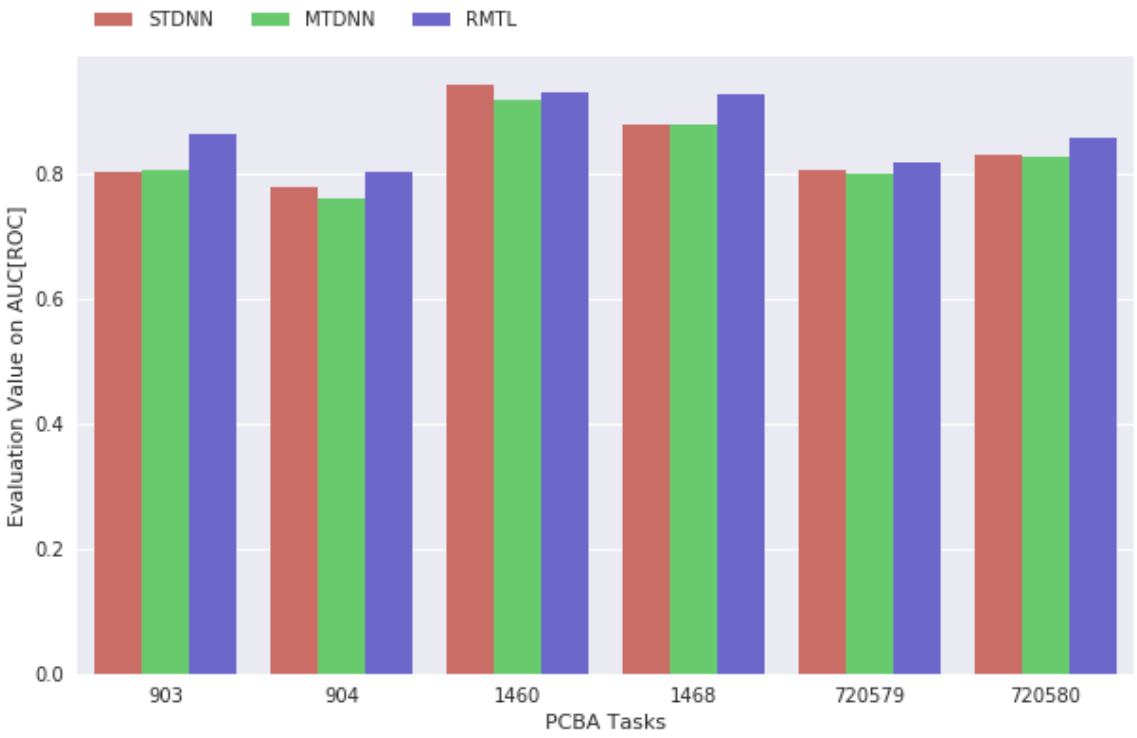


Figure 3.3: Test ROC performance. Three task pairs are 903-904, 1460-1468, 720579-720580. Four tasks have negative transfer issue, and RMTL is best in all those tasks. It also shows robust performance on the other two tasks. In general, RMTL performs better than both STDNN and MTDNN as expected.

The test results are summarized in Figure 3.3. We evaluated the models by using the area under the receiver operating characteristic curve (AUC[ROC]). RMTL shows improved performance over both STDNN and MTDNN, and successfully fixes the negative transfer issue.

3.6 Summary

In this chapter, we propose a general reinforced multi-task learning framework. It shows promising results in a subsample of the PCBA dataset. Future testing should be expanded to other benchmark datasets. Here we investigated the negative transfer issue only on one focused task cases. An ultimate solution would fit for multi-task learning with any number of focused tasks, and RMTL provides one possible path towards it.

Appendix A

Appendix: Starting from A Standard Pipeline

Co-authors of [54] contributed to this appendix chapter.

A.1 PCBA Query

We downloaded additional high-throughput screening datasets from the PubChem BioAssay database [100] using the following query: TotalSidCount from 10000, ActiveSidCount from 30, Chemical, Confirmatory, Dose-Response, Target: Single, NCGC. These correspond to the search query: (10000[TotalSidCount] : 1000000000[TotalSidCount]) AND (30[ActiveSidCount] : 1000000000[ActiveSidCount]) AND "small molecule"[filt] AND "doseresponse"[filt] AND 1[TargetCount] AND "NCGC"[SourceName]. This follows the query from Ramsundar et al. [77].

A.2 Data Preprocessing

A.2.1 Complex Matrix Composition

Each target dataset consists of compounds as rows. For each compound, it provides the biochemical features such as structure, SMILES, interaction scores, and the activity label (binary or continuous). For our purposes, we are only interested in the SMILES and activity label of the compounds. The first step was to extract these two properties for each target and construct the data matrix for training. We used RDKit [1] for navigating and extracting information from these datasets and for generating the fingerprints.

The second step was to merge the target matrices together into one consolidated matrix. We used an outer-join operation with the SMILES as the key. That is, given two matrices A and B with

two columns: SMILES and target-activity, an outer-join operation will merge rows of A and B that have the same SMILES value into a new matrix M. If there is a row in A with SMILES value s and no corresponding row in B with SMILES value s, then the merge would yield a row in M with an empty target-activity for B. In the resulting matrix, each row is a compound and the columns are: SMILES, Morgan fingerprints (1024 bits), and a column for the activity outcome of each target (5 columns for **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** and the associated % inhibition values and 128 columns for PCBA). As a result, we have two data matrices: **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** and **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** plus PCBA, on which we can train either single-task or multi-task learning methods. Note that merging all the targets introduces many empty cells for the activity outcome columns. For the distribution of active, inactive, and missing values for each target, refer to Appendix A.3. We observed severe data imbalance; the ratio of positive to negative labels is very small.

During model evaluation, we considered each task (column) separately and dropped the missing values.

A.2.2 Fold Splitting

The whole data set was split into 5 fixed folds for cross-validation. Label imbalance and the limited number of known active molecules is one of biggest challenges in virtual screening and must be accounted for during modeling. Stratified splitting is a way to divide data into folds while keeping the same active-to-inactive ratio for each label.

For a single-target task, stratified splits can be implemented by combining folds after sampling each class of labels. But this procedure becomes more complicated in the multi-task setting. With a total of 131 targets, each target has a set of molecules with activity outcomes that may or may not overlap across targets. After merging all molecules into one matrix, each row represents one molecule and each column represents one target. For each column (target), molecules can have missing, inactive, or active labels. Similarly, for each row (molecule), the molecule must have an active or inactive label for at least one of the 131 targets but can be missing for some of other targets. We construct this combined matrix of 131 targets using Algorithm 1 described below. We divide this matrix into 5 folds, while keeping the same data distribution.

A.2.3 Label Imbalance

PriA-SSB AS, **PriA-SSB FP**, and **RMI-FANCM FP** have only 79, 24, and 230 actives, respectively. To alleviate this class imbalance, one solution is to use a weighted schema. For single-target neural network models, we apply Eq (A.1).

$$weight_{(negative)} = 1, \quad weight_{(positive)} = \frac{n}{p} \tag{A.1}$$

Algorithm 4: Multi-task Data Splitting

Input: Initial pre-split molecule-target matrix M, number of desired folds k
Output: k folds F[1], F[2], ..., F[k] containing stratified splits of M

```
shuffle rows of M randomly
create k folds F[1], F[2], ..., F[k] which contain the row indexes only
indexList ← argsort columns of M from smallest active counts to largest
for i in indexList do
    currColumn ← M[:, i]
    split active indexes of currColumn into the k folds
    split inactive indexes of currColumn into the k folds
    split missing indexes of currColumn into the k folds
    take the unique compounds in each fold to remove duplicate row indexes
    greedily remove overlapping indexes from each fold (fold-by-fold manner)
take the unique compounds in each fold to remove duplicate row indexes
return F[1], F[2], ..., F[k]
```

where $weight_{positive}$ and $weight_{negative}$ are weight scalars for positive (active) and negative (inactive) compounds, respectively, and p and n represent the number of positive and negative samples on this target.

Similarly, we apply the weighted schema to multi-task models, defined as Eq (A.2).

$$weight_{(negative,\,i)} = t_i, \quad weight_{(positive,\,i)} = t_i \cdot \frac{n_i}{p_i} \tag{A.2}$$

where $weight_{(positive,\,i)}$ and $weight_{(negative,\,i)}$ are weight scalars for positive and negative labels for the i^{th} target, and p_i and n_i represent the number of positive and negative samples for i^{th} target. t_i is defined as Eq (A.3)

$$t_i = \begin{cases} \frac{\sum_i p_i}{p_i}, & i^{th} \text{ target is in PCBA} \\ \alpha \cdot \frac{\sum_i p_i}{p_i}, & i^{th} \text{ target is in } \textbf{PriA-SSB AS}, \textbf{PriA-SSB FP}, \text{ and } \textbf{RMI-FANCM FP} \end{cases} \tag{A.3}$$

In the multi-task setting, we give different weights to each target, focusing more on the **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** targets and the PCBA targets that have fewer positive samples. We emphasize **PriA-SSB AS**, **PriA-SSB FP**, and **RMI-FANCM FP** by setting $\alpha = 100$, and alleviate the data skewness among targets by the term $\frac{\sum_i p_i}{p_i}$.

A.2.4 Missing Label Imputation

1. For single-task machine learning models – random forest, single-task neural networks, and IRV – training is done on non-missing molecules (missing molecules are removed from the training

set).

2. In the case of multi-task neural networks, missing molecules were imputed as inactive. This was mainly due to Keras (at the time) not supporting sample-weighting for the multi-task case.

A.3 PCBA, PriA-SSB AS, PriA-SSB FP, and RMI-FANCM FP Data Distribution

Task name	Positive molecules	Negative molecules	Missing molecules	$ratio = \frac{\text{pos number}}{\text{neg number}}$
pcba-aid1030	15932	145369	335063	10.95970%
pcba-aid1379	561	196368	314806	0.28569%
pcba-aid1452	178	149367	362573	0.11917%
pcba-aid1454	513	115335	395935	0.44479%
pcba-aid1457	720	202110	308746	0.35624%
pcba-aid1458	5778	188852	311888	3.05954%
pcba-aid1460	5650	217010	283986	2.60357%
pcba-aid1461	2305	206016	301670	1.11885%
pcba-aid1468	1038	251148	259072	0.41330%
pcba-aid1469	170	272533	239423	0.06238%
pcba-aid1471	293	218258	293452	0.13424%
pcba-aid1479	793	269530	241180	0.29422%
pcba-aid1631	892	259030	251482	0.34436%
pcba-aid1634	154	261988	250000	0.05878%
pcba-aid1688	2375	201910	305636	1.17627%
pcba-aid1721	1087	289651	220471	0.37528%
pcba-aid2100	1157	291855	218127	0.39643%
pcba-aid2101	288	309907	201813	0.09293%
pcba-aid2147	3473	188764	316586	1.83986%
pcba-aid2242	715	183374	327492	0.38991%
pcba-aid2326	1065	259688	250478	0.41011%
pcba-aid2451	2005	271718	236568	0.73790%
pcba-aid2517	1138	332123	177897	0.34264%
pcba-aid2528	652	340938	170054	0.19124%
pcba-aid2546	10556	267886	223298	3.94048%
pcba-aid2549	1211	230450	279424	0.52549%
pcba-aid2551	16671	253653	225301	6.57236%
pcba-aid2662	110	285240	226836	0.03856%
pcba-aid2675	99	248789	263309	0.03979%
pcba-aid2676	1081	357341	152793	0.30251%
pcba-aid411	1563	69057	440113	2.26335%
pcba-aid463254	41	329171	183043	0.01246%
pcba-aid485281	253	314347	197443	0.08048%
pcba-aid485290	938	335859	174561	0.27928%
pcba-aid485294	148	309649	202351	0.04780%
pcba-aid485297	9128	301294	192746	3.02960%
pcba-aid485313	7569	304194	192964	2.48821%
pcba-aid485314	4493	312590	190720	1.43735%
pcba-aid485341	1729	325703	183135	0.53085%
pcba-aid485349	618	319466	191594	0.19345%
pcba-aid485353	603	322454	188636	0.18700%

Task name	Positive molecules	Negative molecules	Missing molecules	$ratio = \frac{\text{pos number}}{\text{neg number}}$
pcba-aid485360	1485	216997	292329	0.68434%
pcba-aid485364	10698	331470	159430	3.22744%
pcba-aid485367	557	325598	185584	0.17107%
pcba-aid492947	80	329301	182835	0.02429%
pcba-aid493208	342	41294	470318	0.82821%
pcba-aid504327	766	370995	139769	0.20647%
pcba-aid504332	30264	263754	188014	11.47433%
pcba-aid504333	15673	310114	170836	5.05395%
pcba-aid504339	16859	338757	139821	4.97672%
pcba-aid504444	7388	282993	214527	2.61067%
pcba-aid504466	4169	306751	197207	1.35908%
pcba-aid504467	7648	235607	261393	3.24608%
pcba-aid504706	201	302548	209346	0.06644%
pcba-aid504842	101	324570	187524	0.03112%
pcba-aid504845	100	372270	139826	0.02686%
pcba-aid504847	3509	376531	128747	0.93193%
pcba-aid504891	34	361224	151004	0.00941%
pcba-aid540276	4393	192748	310762	2.27914%
pcba-aid540317	2129	367917	140121	0.57866%
pcba-aid588342	25036	301746	160478	8.29704%
pcba-aid588453	3904	365862	138626	1.06707%
pcba-aid588456	51	384356	127838	0.01327%
pcba-aid588579	1980	384213	124123	0.51534%
pcba-aid588590	3931	352947	151487	1.11376%
pcba-aid588591	4700	367981	134915	1.27724%
pcba-aid588795	1307	376247	133435	0.34738%
pcba-aid588855	4897	347556	154946	1.40898%
pcba-aid602179	364	384856	126712	0.09458%
pcba-aid602233	165	379055	132911	0.04353%
pcba-aid602310	310	393819	117857	0.07872%
pcba-aid602313	762	372273	138499	0.20469%
pcba-aid602332	69	408322	103836	0.01690%
pcba-aid624170	838	397756	112864	0.21068%
pcba-aid624171	1239	394674	115144	0.31393%
pcba-aid624173	487	399643	111679	0.12186%
pcba-aid624202	3968	362543	141817	1.09449%
pcba-aid624246	101	364511	147583	0.02771%
pcba-aid624287	423	302226	209224	0.13996%
pcba-aid624288	1356	323051	186533	0.41975%
pcba-aid624291	222	331803	180049	0.06691%
pcba-aid624296	9840	282428	210188	3.48407%
pcba-aid624297	6213	301951	197919	2.05762%
pcba-aid624417	6389	319289	180229	2.00101%
pcba-aid651635	3784	343160	161568	1.10269%
pcba-aid651644	748	353982	156818	0.21131%
pcba-aid651768	1677	355992	152950	0.47108%
pcba-aid651965	6346	318038	181566	1.99536%
pcba-aid652025	238	364167	147653	0.06535%
pcba-aid652104	7126	368557	129487	1.93349%
pcba-aid652105	4072	318365	185787	1.27904%
pcba-aid652106	497	362334	148968	0.13717%
pcba-aid686970	5948	331060	169340	1.79665%
pcba-aid686978	62375	236628	150918	26.35994%
pcba-aid686979	48532	257279	157953	18.86357%
pcba-aid720504	10170	340357	151599	2.98804%

Task name	Positive molecules	Negative molecules	Missing molecules	$ratio = \frac{\text{pos number}}{\text{neg number}}$
pcba-aid720532	976	11815	498529	8.26069%
pcba-aid720542	733	356204	154626	0.20578%
pcba-aid720551	1265	341660	168106	0.37025%
pcba-aid720553	3259	336029	169749	0.96986%
pcba-aid720579	1908	280991	227489	0.67903%
pcba-aid720580	1508	304454	204826	0.49531%
pcba-aid720707	268	363257	148503	0.07378%
pcba-aid720708	661	356743	154231	0.18529%
pcba-aid720709	516	352850	158414	0.14624%
pcba-aid720711	290	363245	148471	0.07984%
pcba-aid743255	901	366915	143579	0.24556%
pcba-aid743266	306	398728	112956	0.07674%
pcba-aid875	34	73821	438407	0.04606%
pcba-aid881	590	103808	407308	0.56836%
pcba-aid883	1217	6647	503215	18.30901%
pcba-aid884	3396	6983	498521	48.63239%
pcba-aid885	160	12683	499293	1.26153%
pcba-aid887	1017	68423	441839	1.48634%
pcba-aid891	1564	6012	503156	26.01464%
pcba-aid899	1773	6141	502609	28.87152%
pcba-aid902	1865	117072	391494	1.59304%
pcba-aid903	338	52451	459169	0.64441%
pcba-aid904	528	50430	460810	1.04700%
pcba-aid912	453	56178	455212	0.80637%
pcba-aid914	221	7524	504330	2.93727%
pcba-aid915	421	7524	503930	5.59543%
pcba-aid924	1144	118813	391195	0.96286%
pcba-aid925	39	64140	448078	0.06080%
pcba-aid926	345	56230	455376	0.61355%
pcba-aid927	60	58565	453611	0.10245%
pcba-aid938	1781	60720	448014	2.93314%
pcba-aid995	699	65056	445842	1.07446%
PriA-SSB AS	79	72344	439794	0.10920%
PriA-SSB FP	24	72399	439849	0.03315%
RMI-FANCM FP	230	49566	462270	0.46403%

A.4 Hyperparameter Grid Search

Table A.2: Hyperparameter sweeping for classification neural networks (STNN-C and MTNN-C).

Hyperparameters	Candidate values
hidden layer sizes	[2000, 2000]
learning rate	0.00003, 0.0001, 0.003
optimizer	Adam
weighted schema	no_weight, weighted_sample
epoch patience	[epoch_size: 200, patience: 50], [epoch_size: 1000, patience: 200]
activations	[ReLU, Sigmoid, Sigmoid], [ReLU, ReLU, Sigmoid]
drop out	0.25

Table A.3: Hyperparameter sweeping for regression neural networks (STNN-R).

Hyperparameters	Candidate values
hidden layer sizes	[2000, 2000]
learning rate	0.00003, 0.0001, 0.003
optimizer	Adam
weighted schema	no_weight
epoch	200, 1000
activations	[ReLU, Sigmoid, Sigmoid], [ReLU, ReLU, Sigmoid]
drop out	0.25

Table A.4: Hyperparameter sweeping for LSTM neural networks.

Hyperparameters	Candidate values
hidden layer sizes	[50], [100], [100, 10], [100, 50], [50, 10]
embedding layer size	30, 50, 100
learning rate	0.00003, 0.0001, 0.003
optimizer	Adam
epoch patience	[epoch_size: 200, patience: 50]
drop out	0.2, 0.5

Table A.5: Hyperparameter sweeping for random forests (RF).

Hyperparameters	Candidate values
n_estimators	4000, 8000, 16000
max_features	None, sqrt, log2
min_samples_leaf	1, 10, 100, 1000
class_weight	None, balanced_subsample, balanced

Hyperparameters	Candidate values
number of neighbors	5, 10, 20, 40, 80
epoch patience	[epoch_size: 1000, patience: 20]
batch size	8192
learning rate	0.01
penalty	0.05

Table A.6: Hyperparameter sweeping for IRV.

During the hyperparameter sweeping stage, we trained models with all combinations of the hyperparameters in the tables above. For the neural networks, 80% of the 4 folds were used for training and 20% for validation. For random forest, the first 3 folds were used for training and the 4th fold for validation to prune 108 models down to 8 models. In both cases, the goal was to prune the model search space before the cross-validation stage. IRV has one primary hyperparameter for the number of neighbors so we did not need to prune the model search space before the cross-validation stage.

A.5 Cross-Validation Results on PriA-SSB FP and RMI-FANCM FP

A.5.1 Cross-Validation Performance on PriA-SSB FP

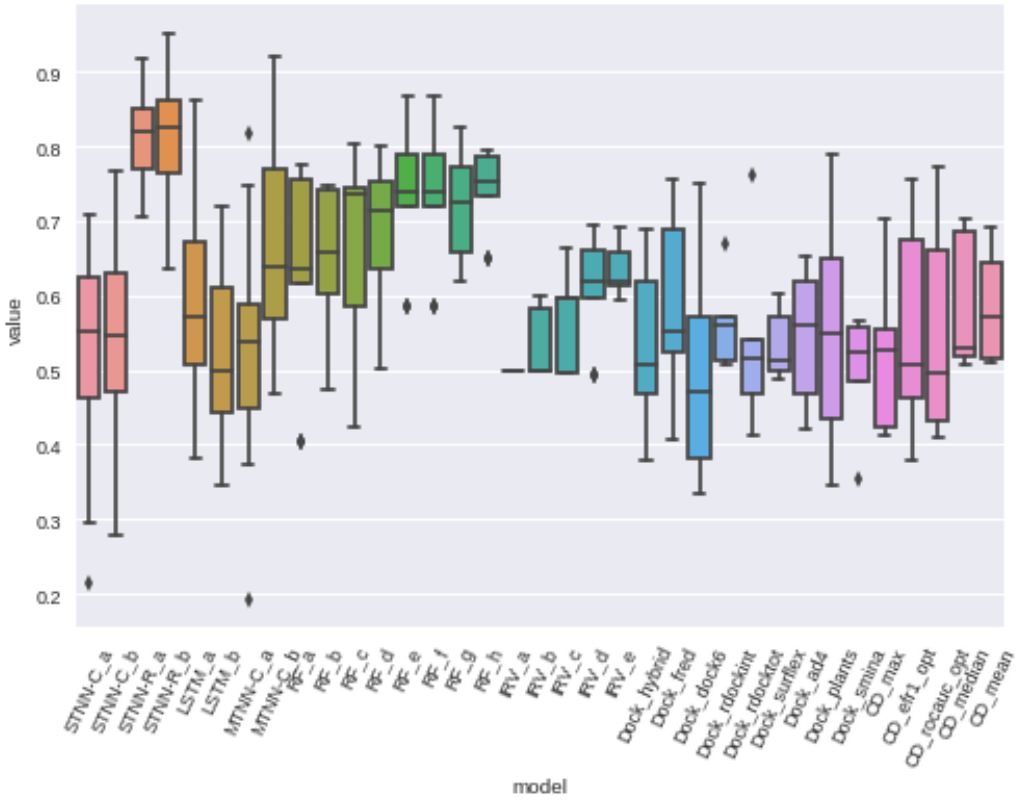


Figure A.1: Cross-validation performance with AUC[ROC]

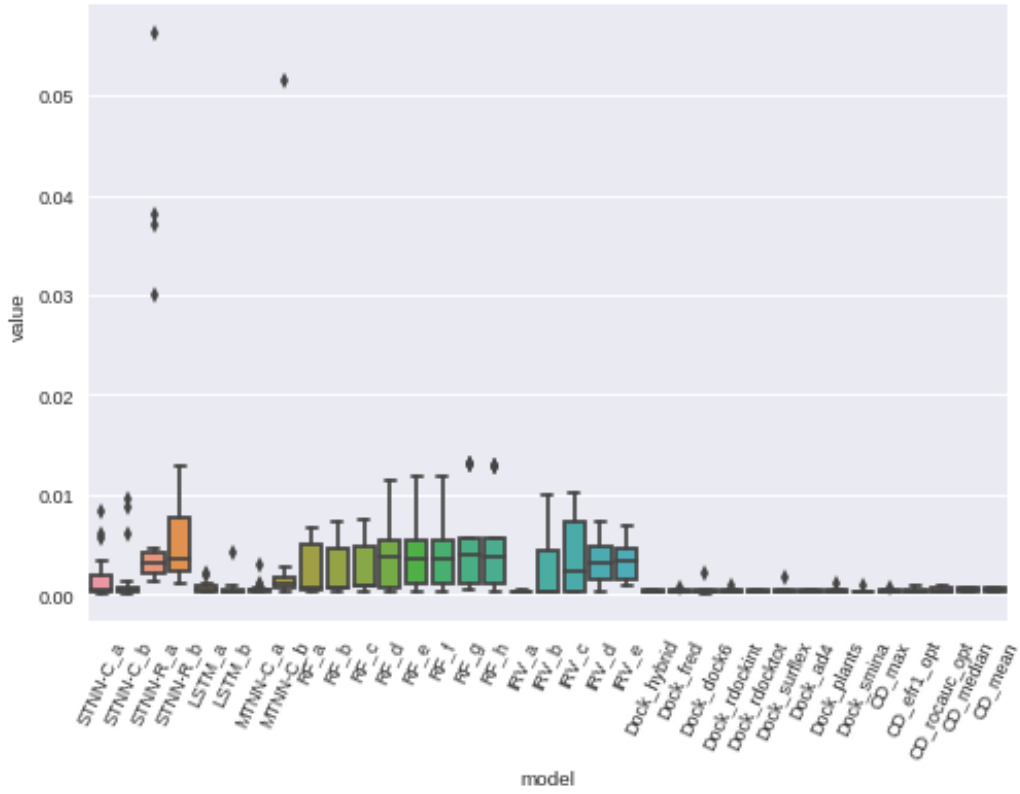


Figure A.2: Cross-validation performance with AUC[PR]

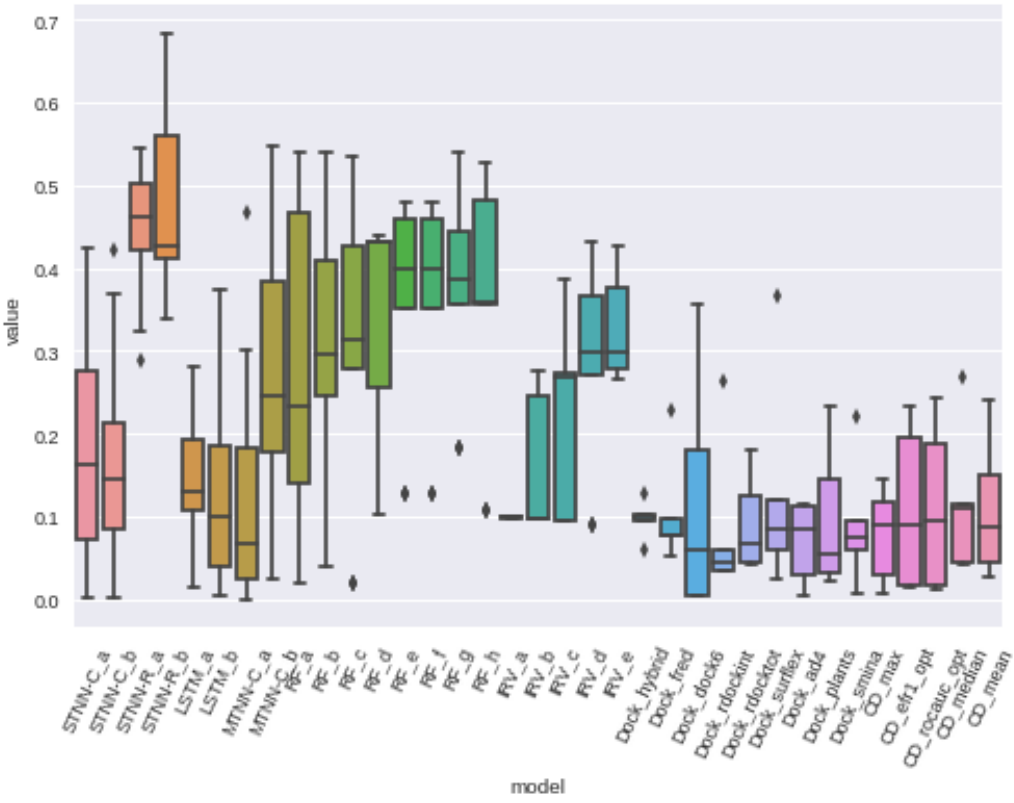


Figure A.3: Cross-validation performance with AUC[BEDROC]

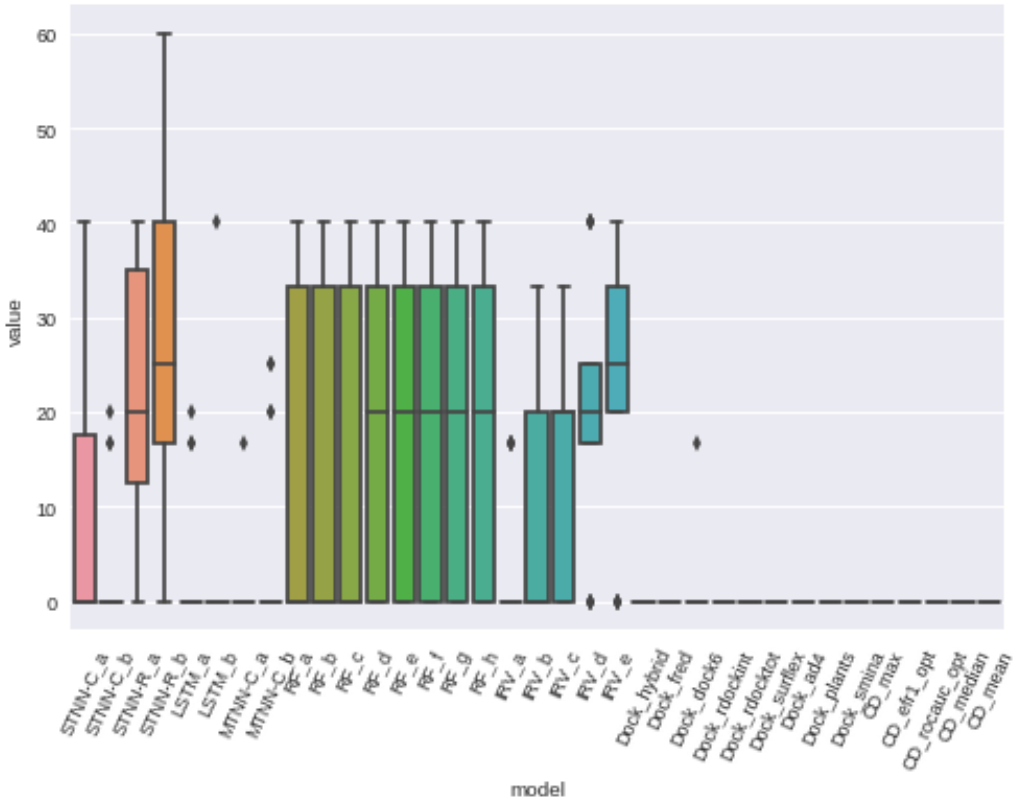


Figure A.4: Cross-validation performance with EF@1%

A.5.2 Cross-Validation Performance on RMI-FANCM FP

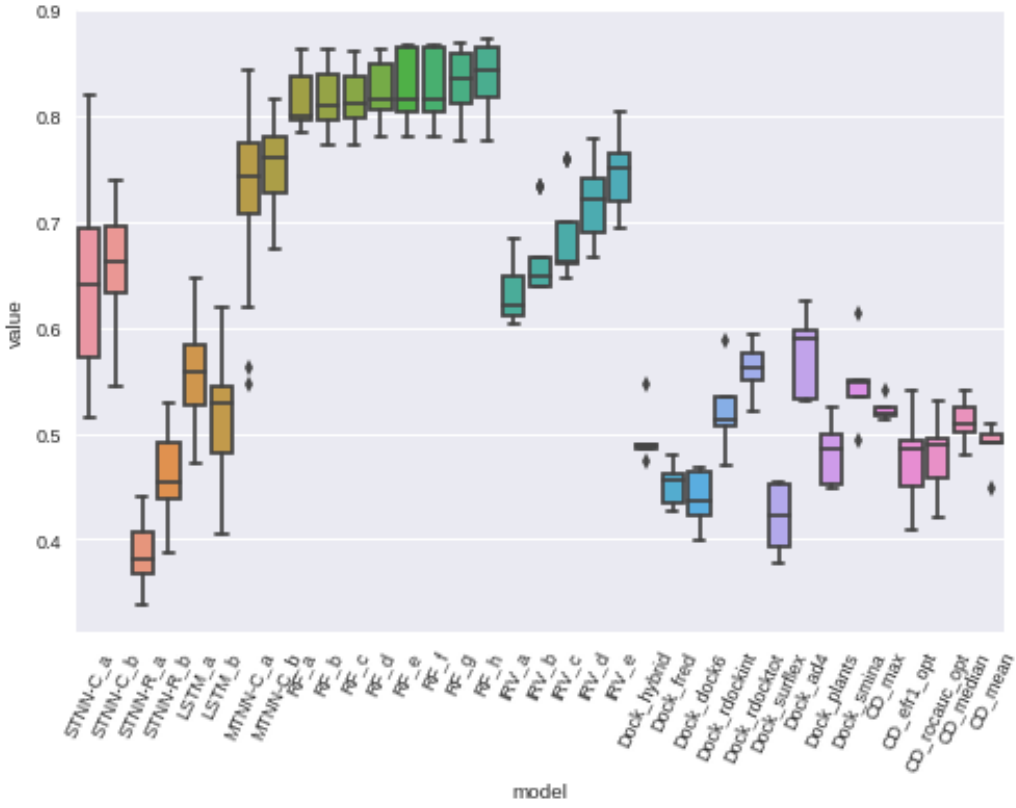


Figure A.5: Cross-validation performance with AUC[ROC]

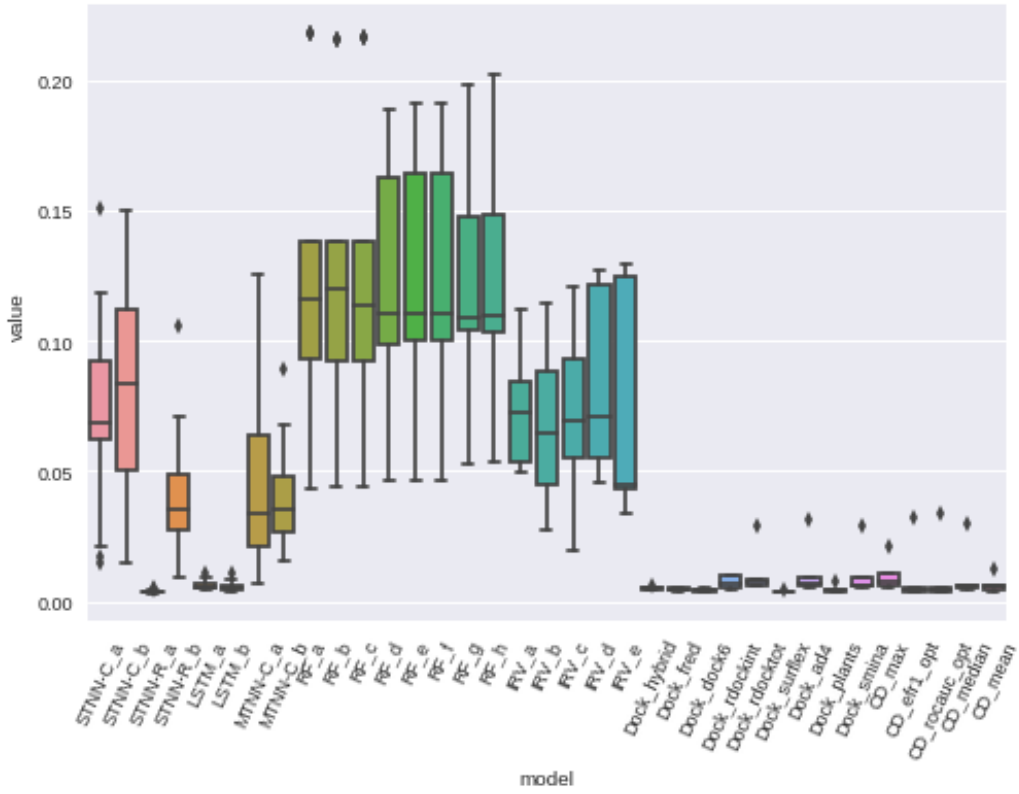


Figure A.6: Cross-validation performance with AUC[PR]

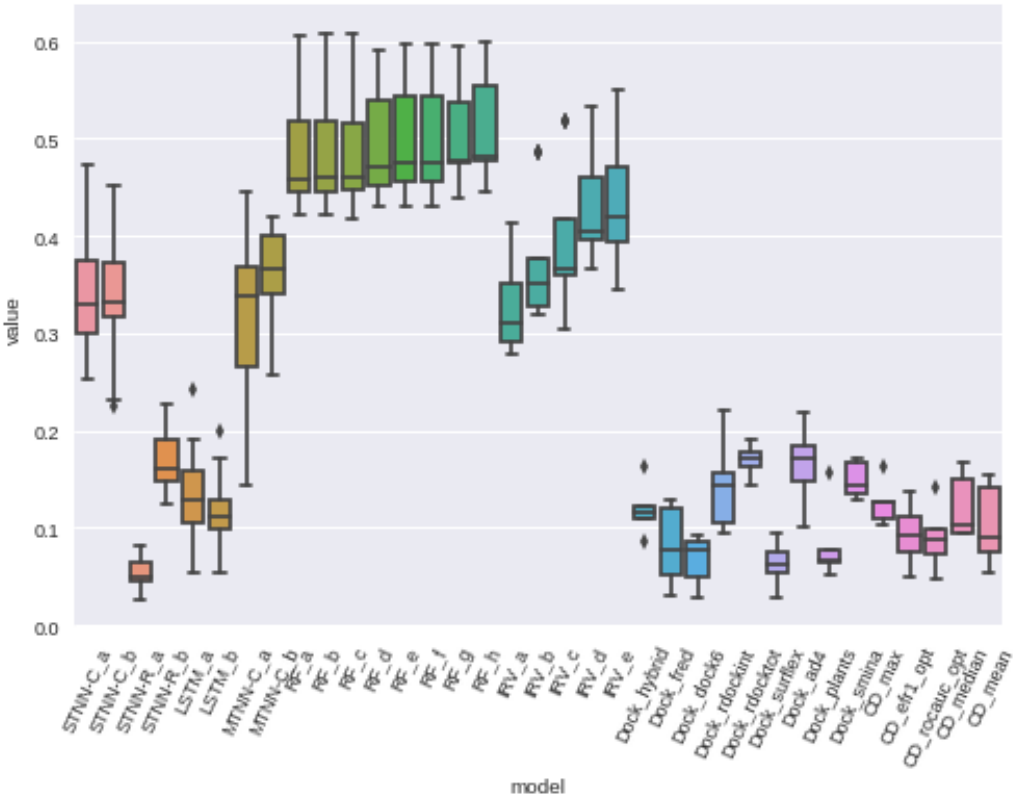


Figure A.7: Cross-validation performance with AUC[BEDROC]

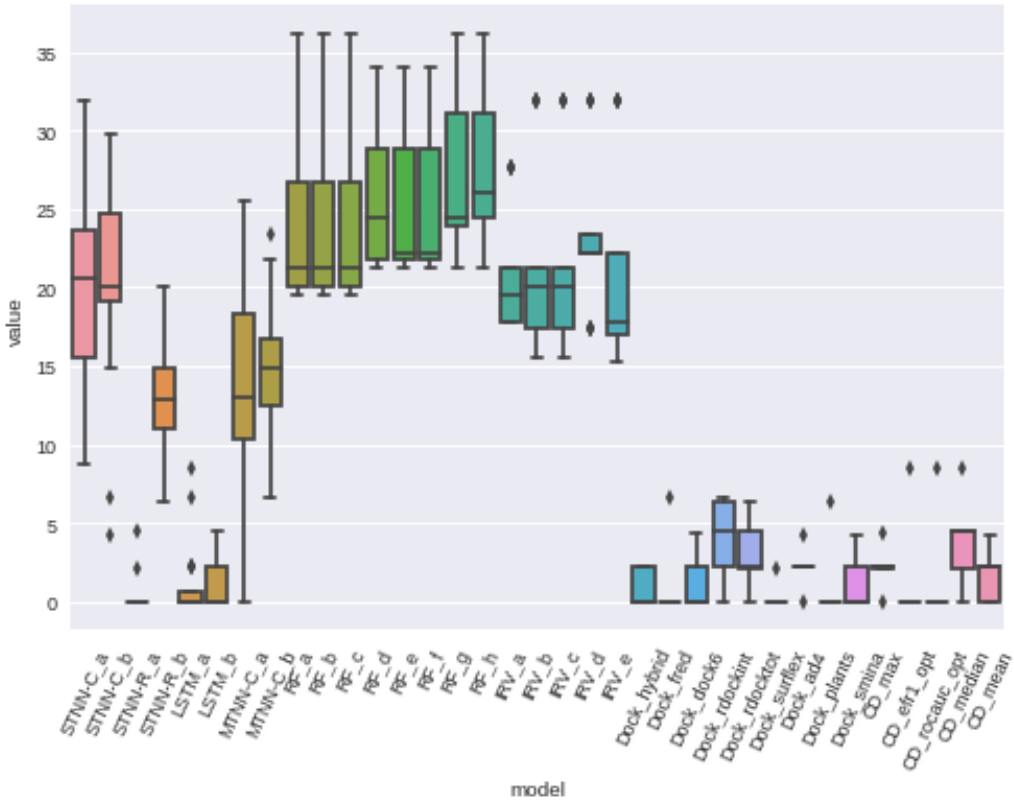


Figure A.8: Cross-validation performance with EF@1%

A.6 Prospective Screening: PriA-SSB prospective Metrics

Table A.7: On-target evaluation metrics for all models. Models were trained on **PriA-SSB AS** and evaluated on **PriA-SSB prospective**.

model	AUC[ROC]	AUC[BEDROC]	AUC[PR]	NEF@1%
Baseline	0.84937	0.67375	0.16167	0.55556
ConsensusDocking_efr1_opt	0.57953	0.11677	0.00293	0.00000
ConsensusDocking_max	0.57996	0.14810	0.00337	0.03704
ConsensusDocking_mean	0.55288	0.09588	0.00261	0.00000
ConsensusDocking_median	0.53129	0.07482	0.00246	0.00000
ConsensusDocking_rocauc_opt	0.58635	0.11949	0.00298	0.00000
Docking_ad4	0.36292	0.01643	0.00159	0.00000
Docking_dock6	0.55541	0.13279	0.00454	0.01852
Docking_fred	0.51009	0.12103	0.00301	0.03704
Docking_hybrid	0.49760	0.13474	0.00293	0.01852
Docking_plants	0.48162	0.06959	0.00223	0.01852
Docking_rdockint	0.56174	0.12492	0.00324	0.01852
Docking_rdocktot	0.68720	0.21658	0.00470	0.01852
Docking_smina	0.42361	0.03394	0.00188	0.00000
Docking_surflex	0.57940	0.15061	0.00341	0.01852
IRV_a	0.64669	0.35955	0.07617	0.29630
IRV_b	0.71961	0.48510	0.12394	0.44444
IRV_c	0.78292	0.59296	0.18787	0.51852
IRV_d	0.82602	0.65816	0.19050	0.51852
IRV_e	0.86718	0.71450	0.20442	0.53704
LSTM_a	0.58979	0.17634	0.00357	0.01852
LSTM_b	0.61639	0.18218	0.00440	0.01852
MultiClassification_a	0.83244	0.58368	0.18462	0.40741
MultiClassification_b	0.84750	0.61346	0.22199	0.50000
RandomForest_a	0.87578	0.73649	0.28165	0.66667
RandomForest_b	0.87065	0.74287	0.28530	0.66667
RandomForest_c	0.87524	0.74433	0.28648	0.66667
RandomForest_d	0.88677	0.75521	0.28425	0.64815
RandomForest_e	0.89007	0.75693	0.28200	0.66667
RandomForest_f	0.88105	0.68324	0.17308	0.44444
RandomForest_g	0.88903	0.76547	0.36893	0.66667
RandomForest_h	0.89689	0.76886	0.37933	0.66667
SingleClassification_a	0.76435	0.51959	0.11103	0.37037
SingleClassification_b	0.81857	0.61809	0.30469	0.55556
SingleRegression_a	0.92068	0.73424	0.18769	0.53704
SingleRegression_b	0.89712	0.68403	0.18575	0.50000

Table A.8: Off-target evaluation metrics for all models. As a control, the models trained on **RMI-FANCM FP** were evaluated on **PriA-SSB prospective**.

model	AUC[ROC]	AUC[BEDROC]	AUC[PR]	NEF@1%
ConsensusDocking_efr1_opt	0.39931	0.04559	0.00182	0.00000
ConsensusDocking_max	0.49919	0.07458	0.00227	0.00000
ConsensusDocking_mean	0.48110	0.07205	0.00217	0.00000
ConsensusDocking_median	0.47915	0.06370	0.00214	0.00000
ConsensusDocking_rocauc_opt	0.41457	0.04680	0.00186	0.00000
Docking_ad4	0.37911	0.02775	0.00173	0.00000
Docking_dock6	0.60630	0.12972	0.00318	0.00000
Docking_fred	0.40761	0.06924	0.00218	0.01852
Docking_hybrid	0.43895	0.04436	0.00196	0.00000
Docking_plants	0.45539	0.07053	0.00212	0.01852
Docking_rdockint	0.52785	0.08938	0.00245	0.00000
Docking_rdocktot	0.60125	0.13316	0.00313	0.00000
Docking_smina	0.44368	0.04307	0.00195	0.00000
Docking_surflex	0.56411	0.10643	0.00295	0.01852
IRV_a	0.52240	0.13901	0.00615	0.03704
IRV_b	0.51675	0.13037	0.00701	0.03704
IRV_c	0.53331	0.15181	0.00605	0.03704
IRV_d	0.52513	0.14306	0.00593	0.03704
IRV_e	0.52069	0.14026	0.00608	0.03704
LSTM_a	0.60099	0.15468	0.00341	0.00000
LSTM_b	0.55336	0.18084	0.00351	0.00000
MultiClassification_a	0.64870	0.20565	0.00551	0.03704
MultiClassification_b	0.54647	0.14914	0.00376	0.05556
RandomForest_a	0.50393	0.09951	0.00583	0.03704
RandomForest_b	0.52529	0.09898	0.00634	0.03704
RandomForest_c	0.52841	0.09705	0.00654	0.03704
RandomForest_d	0.49301	0.09502	0.00617	0.03704
RandomForest_e	0.49008	0.09053	0.00609	0.03704
RandomForest_f	0.61363	0.15680	0.01225	0.03704
RandomForest_g	0.51600	0.10026	0.00628	0.03704
RandomForest_h	0.53381	0.10430	0.00637	0.03704
SingleClassification_a	0.51852	0.11102	0.01917	0.03704
SingleClassification_b	0.53075	0.13639	0.01135	0.03704
SingleRegression_a	0.44809	0.07585	0.00224	0.01852
SingleRegression_b	0.44100	0.07796	0.00482	0.03704

A.7 Prospective Screening: Actives in Top 250 Predictions

Table A.9: Number of active compounds and unique clusters in the top 250 predictions compared to the experimental actives.

Model	Actives	Actives not in baseline	SIM clusters	MCS clusters
Experimental	54	–	27	35
Baseline	32	–	12	9
ConsensusDocking_efr1_opt	0	0	0	0
ConsensusDocking_max	2	1	2	2
ConsensusDocking_mean	0	0	0	0
ConsensusDocking_median	0	0	0	0
ConsensusDocking_rocauc_opt	0	0	0	0
Docking_ad4	0	0	0	0
Docking_dock6	1	1	1	1
Docking_fred	2	1	2	2
Docking_hybrid	1	0	1	1
Docking_plants	1	1	1	1
Docking_rdockint	1	0	1	1
Docking_rdocktot	1	0	1	1
Docking_smina	0	0	0	0
Docking_surflex	1	1	1	1
IRV_a	16	1	10	8
IRV_b	24	3	11	8
IRV_c	28	4	12	9
IRV_d	29	4	13	9
IRV_e	29	4	13	9
LSTM_a	1	0	1	1
LSTM_b	1	1	1	1
MultiClassification_a	22	1	12	8
MultiClassification_b	27	3	13	9
RandomForest_a	36	6	12	9
RandomForest_b	37	7	12	9
RandomForest_c	36	6	12	9
RandomForest_d	36	7	12	9
RandomForest_e	36	7	12	9
RandomForest_f	24	4	11	6
RandomForest_g	37	7	13	9
RandomForest_h	37	7	13	9
SingleClassification_a	21	2	13	9
SingleClassification_b	31	5	12	9
SingleRegression_a	29	4	11	8
SingleRegression_b	28	7	10	8

A.8 Prospective Screening: UpSet Plots for Active Compound Clusters

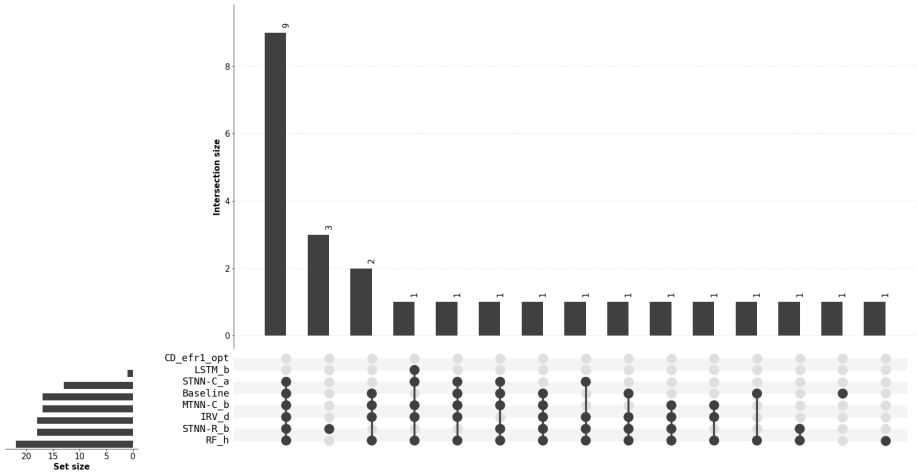


Figure A.9: An UpSet plot showing the overlap in identified MCS clusters between the selected models and the chemical similarity baseline on **PriA-SSB prospective**.

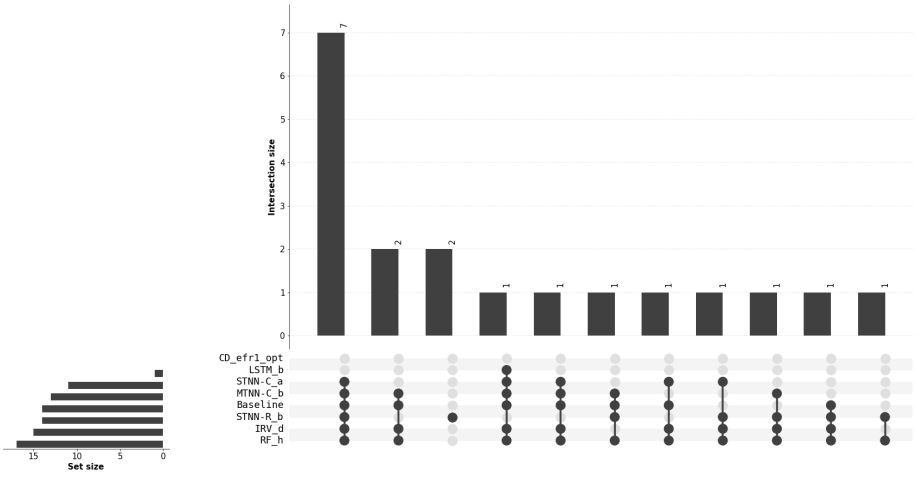


Figure A.10: An UpSet plot showing the overlap in identified SIM clusters between the selected models and the chemical similarity baseline on **PriA-SSB prospective**.

Appendix B

Appendix: Structural Representation and Learning

Co-authors of [55] contributed to this appendix chapter.

B.1 Task Specification

Table B.1: Number of active and total molecules for each task in Tox21.

Task	Num of Active	Num of Total
NR-AR	304	7332
NR-AR-LBD	237	6817
NR-AhR	783	6592
NR-Aromatase	298	5853
NR-ER	784	6237
NR-ER-LBD	347	7014
NR-PPAR-gamma	186	6505
SR-ARE	954	5907
SR-ATAD5	262	7140
SR-HSE	378	6562
SR-MMP	912	5834
SR-p53	414	6814

B.2 Node Attribute Matrix Specification

Table B.2: 42 features are divided into 8 segments.

segmentation	digit	meaning	values
0	0-9	atom symbol	[C, Cl, I, F, O, N, P, S, Br, Unknown]
1	10-16	atom degree	[0, 1, 2, 3, 4, 5, 6]
2	17-23	atom number of Hytrogeon	[0, 1, 2, 3, 4, 5, 6]
3	24-29	atom implicit valence	[0, 1, 2, 3, 4, 5]
4	30-35	atom charge	[-2, -1, 0, 1, 2, 3]
5	36-37	if atom is aromatic	[not aromatic, is aromatic]
6	38-39	if atom is acceptor	[not acceptor, is acceptor]
7	40-41	if atom donor	[not donor, is donor]

B.3 Result Classification Tasks

B.3.1 ROC on Tox21

Table B.3: This table includes three different methods on N-Gram Graph. Six combinations of n-gram and r random projection dimension are listed. For each combination, model with best performance is **bolded**.

target name	r-dimension	n-gram	XGBoost	RF	DNN
NR-AR	50	2	0.825	0.825	**0.855**
		4	0.832	0.826	**0.863**
		6	0.839	0.826	**0.864**
	100	2	0.818	**0.826**	0.816
		4	0.832	0.823	**0.847**
		6	**0.837**	0.822	0.830
NR-AR-LBD	50	2	0.835	0.851	**0.867**
		4	0.835	0.849	**0.857**
		6	0.845	0.841	**0.857**
	100	2	0.835	0.855	**0.867**
		4	0.827	0.841	**0.857**
		6	0.843	0.840	**0.845**
NR-AhR	50	2	**0.883**	0.874	0.852
		4	**0.889**	0.872	0.852
		6	**0.884**	0.870	0.851
	100	2	**0.887**	0.874	0.866
		4	**0.888**	0.873	0.861
		6	**0.886**	0.872	0.859
NR-Aromatase	50	2	0.839	**0.849**	0.826
		4	0.829	**0.849**	0.824
		6	0.833	**0.849**	0.826
	100	2	0.829	**0.853**	0.824
		4	0.833	**0.848**	0.834
		6	0.829	**0.844**	0.832

NR-ER	50	2	**0.712**	0.693	0.708
		4	**0.717**	0.695	0.708
		6	0.704	0.697	**0.708**
	100	2	0.711	0.694	**0.717**
		4	0.714	0.698	**0.719**
		6	0.704	0.699	**0.729**
NR-ER-LBD	50	2	**0.811**	0.801	0.805
		4	**0.821**	0.816	0.800
		6	**0.829**	0.812	0.799
	100	2	**0.822**	0.798	0.813
		4	**0.821**	0.818	0.801
		6	**0.822**	0.807	0.802
NR-PPAR-gamma	50	2	0.821	**0.850**	0.726
		4	0.784	**0.847**	0.728
		6	0.817	**0.826**	0.717
	100	2	0.802	**0.849**	0.751
		4	0.802	**0.835**	0.748
		6	0.792	**0.837**	0.772
SR-ARE	50	2	**0.819**	0.815	0.795
		4	**0.829**	0.824	0.807
		6	0.826	**0.827**	0.804
	100	2	**0.822**	0.815	0.799
		4	**0.837**	0.828	0.803
		6	**0.836**	0.832	0.794
SR-ATAD5	50	2	**0.853**	0.840	0.814
		4	0.846	**0.865**	0.807
		6	0.844	**0.858**	0.807
	100	2	**0.858**	0.844	0.805
		4	0.853	**0.865**	0.821
		6	0.843	**0.858**	0.808
SR-HSE	50	2	**0.785**	0.760	0.775
		4	**0.805**	0.771	0.779
		6	**0.821**	0.773	0.771
	100	2	**0.792**	0.762	0.759
		4	**0.798**	0.775	0.760
		6	**0.796**	0.771	0.764

SR-MMP	50	2	**0.897**	0.887	0.857
		4	**0.904**	0.893	0.851
		6	**0.905**	0.893	0.849
	100	2	**0.903**	0.889	0.863
		4	**0.909**	0.893	0.860
		6	**0.908**	0.893	0.862
SR-p53	50	2	**0.847**	0.826	0.778
		4	**0.864**	0.840	0.778
		6	**0.872**	0.843	0.772
	100	2	**0.855**	0.830	0.791
		4	**0.868**	0.841	0.795
		6	**0.865**	0.841	0.794
Average	50	2	**0.827**	0.823	0.805
		4	**0.830**	0.829	0.804
		6	**0.835**	0.826	0.802
	100	2	**0.828**	0.824	0.806
		4	**0.832**	0.828	0.809
		6	**0.830**	0.826	0.807

B.3.2 Generalization Performance on Tox21

Table B.4: Generalization performance is the performance (AUC[ROC]) gap between train and test set. Top three robust models are **bolded** and the most stable one is **underlined**. Though its performance is not the best (details in Table 2.2), N-gram graph with DNN is the most robust representation and model pair.

Representation	Morgan			Message-Passing Graph			N-Gram Graph		
Method	RF	XGB	DNN	NEF	GCNN	Weave	RF	XGB	DNN
NR-AR	0.179	0.162	0.249	0.269	**0.068**	**0.083**	0.174	0.161	**0.049**
NR-AR-LBD	**0.118**	0.142	0.190	0.141	**0.105**	0.141	0.159	0.155	**0.091**
NR-AhR	0.117	0.105	0.149	0.147	**0.068**	**0.065**	0.130	0.116	**0.063**
NR-Aromatase	0.171	0.209	0.307	0.238	**0.106**	**0.145**	0.151	0.167	**0.086**
NR-ER	0.282	0.219	0.291	0.277	**0.120**	**0.127**	0.302	0.296	**0.022**
NR-ER-LBD	0.162	0.161	0.204	0.188	**0.134**	**0.138**	0.186	0.171	**0.079**
NR-PPAR-gamma	0.191	0.284	0.262	0.248	0.179	**0.141**	**0.174**	0.183	**0.132**
SR-ARE	0.161	0.135	0.219	0.234	**0.087**	**0.061**	0.173	0.174	**0.039**
SR-ATAD5	0.135	0.186	0.254	0.259	**0.126**	**0.103**	0.141	0.156	**0.085**
SR-HSE	0.203	0.241	0.277	0.240	**0.168**	**0.156**	0.223	0.179	**0.093**
SR-MMP	0.126	0.122	0.130	0.140	**0.078**	**0.072**	0.107	0.095	**0.079**
SR-p53	0.125	**0.125**	0.290	0.179	0.128	**0.113**	0.157	0.128	**0.068**
Average	0.164	0.174	0.235	0.213	**0.114**	**0.112**	0.173	0.165	**0.074**

Bibliography

[1] Rdkit: Open-source cheminformatics. *http://www.rdkit.org*.

[2] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. Distributed representations of subgraphs. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pages 111–117. IEEE, 2017.

[3] William J. Allen, Trent E. Balius, Sudipto Mukherjee, Scott R. Brozell, Demetri T. Moustakas, P. Therese Lang, David A. Case, Irwin D. Kuntz, and Robert C. Rizzo. DOCK 6: Impact of new features and current docking performance. *Journal of Computational Chemistry*, 36(15):1132–1156, June 2015.

[4] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS Central Science*, 3(4):283–293, 2017.

[5] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[6] Andreas Argyriou, Andreas Maurer, and Massimiliano Pontil. An algorithm for transfer learning in a heterogeneous environment. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 71–85, Berlin, Heidelberg, 2008. Springer-Verlag.

[7] Sanjeev Arora, Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and lstm. *International Conference on Learning Representations*, 2018.

[8] Jonathan B Baell and Georgina A Holloway. New substructure filters for removal of pan assay interference compounds (pains) from screening libraries and for their exclusion in bioassays. *Journal of medicinal chemistry*, 53(7):2719–2740, 2010.

[9] Basudeb Bhattacharyya, Nicholas P. George, Tiffany M. Thurmes, Ruobo Zhou, Niketa Jani, Sarah R. Wessel, Steven J. Sandler, Taekjip Ha, and James L. Keck. Structural mechanisms

of pria-mediated dna replication restart. *Proceedings of the National Academy of Sciences*, 111(4):1373–1378, 2014.

[10] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 42–50, New York, NY, USA, 2011. ACM.

[11] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[12] Travers Ching, Daniel S. Himmelstein, Brett K. Beaulieu-Jones, Alexandr A. Kalinin, Brian T. Do, Gregory P. Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M. Hoffman, Wei Xie, Gail L. Rosen, Benjamin J. Lengerich, Johnny Israeli, Jack Lanchantin, Stephen Woloszynek, Anne E. Carpenter, Avanti Shrikumar, Jinbo Xu, Evan M. Cofer, Christopher A. Lavender, Srinivas C. Turaga, Amr M. Alexandari, Zhiyong Lu, David J. Harris, Dave DeCaprio, Yanjun Qi, Anshul Kundaje, Yifan Peng, Laura K. Wiley, Marwin H. S. Segler, Simina M. Boca, S. Joshua Swamidass, Austin Huang, Anthony Gitter, and Casey S. Greene. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, April 2018.

[13] François Chollet. Keras. *https://github.com/fchollet/keras*, 2015.

[14] Ann E. Cleves and Ajay N. Jain. Knowledge-guided docking: accurate prospective prediction of bound configurations of novel ligands using Surflex-Dock. *Journal of Computer-Aided Molecular Design*, 29(6):485–509, June 2015.

[15] Connor W Coley, Regina Barzilay, William H Green, Tommi S Jaakkola, and Klavs F Jensen. Convolutional embedding of attributed molecular graphs for physical property prediction. *Journal of chemical information and modeling*, 57(8):1757–1772, 2017.

[16] Jason B Cross, David C Thompson, Brajesh K Rai, J Christian Baber, Kristi Yi Fan, Yongbo Hu, and Christine Humblet. Comparison of several molecular docking programs: pose prediction and virtual screening accuracy. *Journal of chemical information and modeling*, 49(6):1455–1474, 2009.

[17] George Dahl. Deep learning how i did it: Merck 1st place interview. *Online article available from http://blog. kaggle. com/2012/11/01/deep-learning-how-i-did-it-merck-1st-place-interview*, 2012.

[18] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for QSAR predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[19] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[20] John S. Delaney. ESOL: Estimating Aqueous Solubility Directly from Molecular Structure. *Journal of Chemical Information and Computer Sciences*, 44(3):1000–1005, May 2004.

[21] Charles W. Dunnett. Pairwise Multiple Comparisons in the Unequal Variance Case. *Journal of the American Statistical Association*, 75(372):796–800, December 1980.

[22] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, July 2015.

[23] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[24] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. pages 2224–2232, 2015.

[25] Spencer S Ericksen, Haozhen Wu, Huikun Zhang, Lauren A Michael, Michael A Newton, F Michael Hoffmann, and Scott A Wildman. Machine learning consensus scoring improves performance across targets in structure-based virtual screening. *Journal of Chemical Information and Modeling*, 57(7):1579–1590, 2017.

[26] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[27] Francisco-Javier Gamo, Laura M. Sanz, Jaume Vidal, Cristina de Cozar, Emilio Alvarez, Jose-Luis Lavandera, Dana E. Vanderwall, Darren V. S. Green, Vinod Kumar, Samiul Hasan, James R. Brown, Catherine E. Peishoff, Lon R. Cardon, and Jose F. Garcia-Bustos. Thousands of chemical starting points for antimalarial lead identification. *Nature*, 465(7296):305–310, May 2010.

[28] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 2016.

[29] Keilwagen J. Grau J, Grosse I. Prroc: computing and visualizing precision-recall and receiver operating characteristic curves in r. *Bioinformatics*, 31(15):2595–2597, 2015.

[30] Shengbo Guo, Onno Zoeter, and Cédric Archambeau. Sparse bayesian multi-task learning. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1755–1763. Curran Associates, Inc., 2011.

[31] Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S. Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M. Brockway, and Alán Aspuru-Guzik. The Harvard Clean Energy Project: Large-Scale Computational Screening and Design of Organic Photovoltaics on the World Community Grid. *The Journal of Physical Chemistry Letters*, 2(17):2241–2251, September 2011.

[32] Paul C. D. Hawkins, A. Geoffrey Skillman, Gregory L. Warren, Benjamin A. Ellingson, and Matthew T. Stahl. Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database. *Journal of Chemical Information and Modeling*, 50(4):572–584, April 2010.

[33] Paul CD Hawkins, A Geoffrey Skillman, and Anthony Nicholls. Comparison of shape-matching and docking as virtual screening tools. *Journal of medicinal chemistry*, 50(1):74–82, 2007.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[35] Laurent Jacob, Jean philippe Vert, and Francis R. Bach. Clustered multi-task learning: A convex formulation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 745–752. Curran Associates, Inc., 2009.

[36] Natasha Jaques, Sara Taylor, Ehimwenma Nosakhare, Akane Sano, and Rosalind Picard. Multi-task learning for predicting health, stress, and happiness. In *NIPS Workshop on Machine Learning for Healthcare*, 2016.

[37] Stanisław Jastrzębski, Damian Leśniak, and Wojciech Marian Czarnecki. Learning to smile (s). *arXiv preprint arXiv:1602.06289*, 2016.

[38] Steven Kearnes, Brian Goldman, and Vijay Pande. Modeling industrial admet data with multitask networks. *arXiv preprint arXiv:1606.08793*, 2016.

[39] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[40] David Ryan Koes, Matthew P. Baumgartner, and Carlos J. Camacho. Lessons learned in empirical scoring with smina from the CSAR 2011 benchmarking exercise. *Journal of Chemical Information and Modeling*, 53(8):1893–1904, August 2013.

[41] Oliver Korb, Thomas Stützle, and Thomas E. Exner. Empirical Scoring Functions for Advanced Protein-Ligand Docking with PLANTS. *Journal of Chemical Information and Modeling*, 49(1):84–96, January 2009.

[42] Alexandru Korotcov, Valery Tkachenko, Daniel P. Russo, and Sean Ekins. Comparison of Deep Learning With Multiple Machine Learning Methods and Metrics Using Diverse Drug Discovery Data Sets. *Molecular Pharmaceutics*, 14(12):4462–4475, December 2017.

[43] Dennis M Krüger and Andreas Evers. Comparison of structure-and ligand-based virtual screening protocols considering hit list complementarity and enrichment factors. *ChemMedChem*, 5(1):148–158, 2010.

[44] Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*, 2012.

[45] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[46] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*, 2017.

[47] David Lagorce, Olivier Sperandio, Jonathan B Baell, Maria A Miteva, and Bruno O Villoutreix. Faf-drugs3: a web server for compound property calculation and chemical library design. *Nucleic acids research*, 43(W1):W200–W207, 2015.

[48] M.K. Lau. Dtk: Dunnett-tukey-kramer pairwise multiple comparison test adjusted for unequal variances and unequal sample sizes. *R package*, 3, 01 2013.

[49] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

[50] Eelke B. Lenselink, Niels ten Dijke, Brandon Bongers, George Papadatos, Herman W. T. van Vlijmen, Wojtek Kowalczyk, Adriaan P. IJzerman, and Gerard J. P. van Westen. Beyond the hype: deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set. *Journal of Cheminformatics*, 9(1):45, December 2017.

[51] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister. UpSet: Visualization of Intersecting Sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1983–1992, December 2014.

[52] Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. Self-paced multi-task learning. In *AAAI*, pages 2175–2181, 2017.

[53] Evanthia Lionta, George Spyrou, Demetrios K Vassilatis, and Zoe Cournia. Structure-based virtual screening for drug discovery: principles, applications and recent advances. *Current topics in medicinal chemistry*, 14(16):1923–1938, 2014.

[54] Shengchao Liu, Moayad Alnammi, Spencer S Ericksen, Andrew F Voter, James L Keck, F Michael Hoffmann, Scott A Wildman, and Anthony Gitter. Practical model selection for prospective virtual screening. *bioRxiv*, 2018.

[55] Shengchao Liu, Thevaa Chandereng, and Yingyu Liang. N-gram graph, a novel molecule representation. *arXiv preprint arXiv:1806.09206*, 2018.

[56] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *arXiv preprint arXiv:1611.05377*, 2016.

[57] Alessandro Lusci, David Fooshee, Michael Browning, Joshua Swamidass, and Pierre Baldi. Accurate and efficient target prediction using a potency-sensitive influence-relevance voter. *Journal of cheminformatics*, 7(1):63, 2015.

[58] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

[59] Kelly A Manthei and James L Keck. The blm dissolvasome in dna replication and repair. *Cellular and molecular life sciences*, 70(21):4067–4084, 2013.

[60] Matthew K. Matlock, Na Le Dang, and S. Joshua Swamidass. Learning a Local-Variable Model of Aromatic and Conjugated Systems. *ACS Central Science*, 4(1):52–62, January 2018.

[61] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

[62] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. DeepTox: Toxicity Prediction using Deep Learning. *Frontiers in Environmental Science*, 3(80), 2016.

[63] Mark McGann. FRED and HYBRID docking performance on standardized datasets. *Journal of Computer-Aided Molecular Design*, 26(8):897–906, August 2012.

[64] Merck. Merck molecular activity challenge. *https://www.kaggle.com/c/MerckActivity*, 2012.

[65] John B. O. Mitchell. Machine learning methods in chemoinformatics. *Wiley Interdisciplinary Reviews. Computational Molecular Science*, 4(5):468–481, September 2014.

[66] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[67] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[68] Garrett M. Morris, Ruth Huey, William Lindstrom, Michel F. Sanner, Richard K. Belew, David S. Goodsell, and Arthur J. Olson. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *Journal of Computational Chemistry*, 30(16):2785–2791, December 2009.

[69] Keerthiram Murugesan and Jaime Carbonell. Self-paced multitask learning with shared knowledge. *arXiv preprint arXiv:1703.00977*, 2017.

[70] Michael M. Mysinger, Michael Carchia, John. J. Irwin, and Brian K. Shoichet. Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking. *Journal of Medicinal Chemistry*, 55(14):6582–6594, July 2012.

[71] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning Distributed Representations of Graphs. *arXiv:1707.05005 [cs]*, July 2017. arXiv: 1707.05005.

[72] Anthony Nicholls. What do we know and when do we know it? *Journal of Computer-Aided Molecular Design*, 22(3-4):239–255, March 2008.

[73] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023, 2016.

[74] Patrice Nordmann, Gaelle Cuzon, and Thierry Naas. The real threat of klebsiella pneumoniae carbapenemase-producing bacteria. *The Lancet infectious diseases*, 9(4):228–236, 2009.

[75] 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*, 12:2825–2830, 2011.

[76] Anastasia Pentina, Viktoriia Sharmanska, and Christoph H Lampert. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5492–5500, 2015.

[77] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

[78] Bharath Ramsundar, Bowen Liu, Zhenqin Wu, Andreas Verras, Matthew Tudor, Robert P Sheridan, and Vijay S Pande. Is multitask deep learning practical for pharma? *Journal of Chemical Information and Modeling*, 2017.

[79] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[80] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[81] Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*, June 2017. arXiv: 1706.05098.

[82] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[83] Sergio Ruiz-Carmona, Daniel Alvarez-Garcia, Nicolas Foloppe, A. Beatriz Garmendia-Doval, Szilveszter Juhos, Peter Schmidtke, Xavier Barril, Roderick E. Hubbard, and S. David Morley. rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids. *PLOS Computational Biology*, 10(4):e1003571, April 2014.

[84] Thomas Scior, Andreas Bender, Gary Tresadern, José L. Medina-Franco, Karina Martínez-Mayorga, Thierry Langer, Karina Cuanalo-Contreras, and Dimitris K. Agrafiotis. Recognizing Pitfalls in Virtual Screening: A Critical Review. *Journal of Chemical Information and Modeling*, 52(4):867–881, April 2012.

[85] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864, 2017.

[86] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[87] S Joshua Swamidass, Chloé-Agathe Azencott, Kenny Daily, and Pierre Baldi. A croc stronger than roc: measuring, visualizing and optimizing early retrieval. *Bioinformatics*, 26(10):1348–1356, 2010.

[88] S Joshua Swamidass, Chloé-Agathe Azencott, Ting-Wan Lin, Hugo Gramajo, Shiou-Chuan Tsai, and Pierre Baldi. Influence relevance voting: an accurate and interpretable virtual high throughput screening method. *Journal of chemical information and modeling*, 49(4):756–766, 2009.

[89] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

[90] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, May 2016.

[91] Roberto Todeschini and Viviana Consonni. *Molecular descriptors for chemoinformatics: volume I: alphabetical listing/volume II: appendices, references*, volume 41. John Wiley & Sons, 2009.

[92] Tox21 Data Challenge. Tox21 data challenge 2014. *https://tripod.nih.gov/tox21/challenge/*, 2014.

[93] Jean-François Truchon and Christopher I Bayly. Evaluating virtual screening methods: good and bad metrics for the "early recognition" problem. *Journal of chemical information and modeling*, 47(2):488–508, 2007.

[94] Yufeng J. Tseng, Anton J. Hopfinger, and Emilio Xavier Esposito. The great descriptor melting pot: mixing descriptors for the common good of QSAR models. *Journal of Computer-Aided Molecular Design*, 26(1):39–43, January 2012.

[95] Thomas Unterthiner, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter. Deep learning as an opportunity in virtual screening. *Advances in neural information processing systems*, 27, 2014.

[96] Vishwesh Venkatraman, Violeta I Pérez-Nueno, Lazaros Mavridis, and David W Ritchie. Comprehensive comparison of ligand-based virtual screening tools against the dud data set reveals limitations of current 3d methods. *Journal of chemical information and modeling*, 50(12):2079–2093, 2010.

[97] Andrew F Voter, Michael P Killoran, Gene E Ananiev, Scott A Wildman, F Michael Hoffmann, and James L Keck. A high-throughput screening strategy to identify inhibitors of ssb protein–protein interactions in an academic screening facility. *SLAS DISCOVERY: Advancing Life Sciences R&D*, page 2472555217712001, 2017.

[98] Andrew F Voter, Kelly A Manthei, and James L Keck. A high-throughput screening strategy to identify protein-protein interaction inhibitors that block the fanconi anemia dna repair pathway. *Journal of biomolecular screening*, 21(6):626–633, 2016.

[99] Izhar Wallach and Abraham Heifets. Most Ligand-Based Classification Benchmarks Reward Memorization Rather than Generalization. *Journal of Chemical Information and Modeling*, April 2018.

[100] Yanli Wang, Stephen H. Bryant, Tiejun Cheng, Jiyao Wang, Asta Gindulyte, Benjamin A. Shoemaker, Paul A. Thiessen, Sigian He, and Jian Zhang. Pubchem bioassay: 2017 update. *Nucleic Acids Research*, 45(D):D955–D963, 2017.

[101] David Weininger, Arthur Weininger, and Joseph L Weininger. Smiles. 2. algorithm for generation of unique smiles notation. *Journal of Chemical Information and Computer Sciences*, 29(2):97–101, 1989.

[102] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[103] Yongxin Yang and Timothy M. Hospedales. Trace norm regularised deep multi-task learning. *CoRR*, abs/1606.04038, 2016.

[104] Yi Zhang and Jeff Schneider. Learning multiple tasks with a sparse matrix-normal penalty. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, NIPS'10, pages 2550–2558, USA, 2010. Curran Associates Inc.

[105] Tian Zhu, Shuyi Cao, Pin-Chih Su, Ram Patel, Darshan Shah, Heta B. Chokshi, Richard Szukala, Michael E. Johnson, and Kirk E. Hevener. Hit Identification and Optimization in Virtual Screening: Practical Recommendations Based on a Critical Literature Analysis. *Journal of Medicinal Chemistry*, 56(17):6560–6572, September 2013.