Advancing Machine Learning for Small Molecule Property Prediction

by

Paul Glidden Francoeur

Master of Science, Grand Valley State University, 2016

Submitted to the Graduate Faculty of the

School of Medicine

in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2023

UNIVERSITY OF PITTSBURGH

KENNETH P. DIETRICH SCHOOL OF ARTS AND SCIENCES

This dissertation was presented

by

Paul Glidden Francoeur

It was defended on

08-21-2023

and approved by

David R. Koes, Associate Professor, Department of Computational & Systems Biology, School of Medicine

Olexandr Isayev, Assistant Professor, Department of Chemistry, Carnegie Mellon University

Junmei Wang, Associate Professor, School of Pharmacy

Patrick Walters, Chief Data Officer, Relay Therapeutics

Dissertation Director:
Ivet Bahar, Director of the Lauger Center for Physical and Quantitative Biology, Stony Brook University.

Copyright © by Paul Glidden Francoeur

2023

This work is released under a Creative Commons Attribution 4.0 International License.

Advancing Machine Learning for Small Molecule Property Prediction

Paul Glidden Francoeur, PhD

University of Pittsburgh, 2023

Recently, machine learning (ML) models have rapidly become the state of the art at various molecular property prediction tasks. The speed of ML models, without sacrificing accuracy, makes them especially attractive in screening contexts, where a large number of potential molecules need to reduced to a number feasible for experimental testing. However, the black box nature and rapid advancement of ML models has resulted in a proliferation of input representations and model architectures. This makes selection of the "best" model architecture and input representation for a given task difficult. Additionally, while ML models thrive on having large datasets for training, the amount of labeled structures for properties like receptor-ligand binding affinity is small.

This work sets out to help address these two problems with ML models for molecular property prediction. First, a wide variety of molecular input representations and ML model architectures were trained to predict calculated molecular properties. The characterization of both the performance of these models, and how well they utilize the training data, yields suggestions on how to best select a ML approach for more realistic property prediction tasks, given the amount of compute resources and training data available. Next, in order to address the lack of labeled structural data, a new dataset, CrossDocked2020, was created to expand

the PDBbind dataset to expand the available binding pose classification data. By docking ligands into non-cognate, but similar, receptors we were able to expand the 200,000 poses available from the PDBbind General set into 22.5 million poses in CrossDocked2020. Various data imputation techniques were then explored to see if they could improve the binding affinity regression of a convolutional neural network (CNN) on CrossDocked2020. The utilization of an ensemble of CNN models to impute the missing binding affinity labels of complexes in CrossDocked2020 had a small, but significant improvement on model performance. Lastly, in order to give further support that the knowledge from this work is applicable in the real world, the CNN developed in this work was utilized to identify a small molecule to disrupt the actin-profilin1 protein-protein binding complex.

Table of Contents

List of Tables

List of Figures

Preface

Before starting my PhD at Pitt, I knew that I wanted to work on merging computational approaches to solve biological problems. This fascination with combining computer science, mathematics, chemistry, and biology started brewing during my undergraduate degree thanks to a research project with Dr. Agnieszka Szarecka at Grand Valley State University. That project introduced me to work done by Dr Ivet Bahar and drove my interest into Structural Biology as a whole. Unfortunately, GVSU did not have the support for a proper education at that time. Eventually, I ended up on a roundabout path that involved getting a Master's degree and working in a cancer research lab for a year before starting at Pitt. Coming into this program, with barely any coding experience, I did not expect that I would wind up doing heavy computational work and spending all my time on training machine learning models. It was incredibly fortuitous that I met David during orientation and was able to do a rotation project with him, and working through this degree with him has completely changed my life trajectory and allowed me to develop skills that I thought would be totally out of reach if you asked me a year before I started.

Aside from David, I would also like to thank the other members of my committee, Dr Ivet Bahar, Dr Junmei Wang, Dr Olexandr Isayev, and Dr Patrick Walters. Their guidance, suggestions, and confirmation about my work means the world, and has given me the confidence that I need to move on to the next chapter of my life. I also want to extend my thanks to the other members of the Koes lab for listening to me practice my defense, prepare for job interviews, teach me new things in our journal clubs, and share in our publication successes. It has been wonderful working with all of you, and being a part of the wonderful joint CPCB program between Pitt and CMU.

I also want to extend my gratitude to my mother, Mary Ellen Glidden, who has completely supported me throughout my entire life. Her checking in, being willing to listen to me gripe when I needed to vent, and being a generally warm and kind presence has been invaluable. Lastly, I wish to thank my father, Yves Francoeur. He died right before I started my PhD, but I know how proud he would be if he was here to see me finish. It was through the work that both of my parents put into their lives that set me up to follow this path through life. I will be forever grateful for the opportunities that they both enabled for me. Thank you.

Pittsburgh, August 2023 P. Francoeur

1.0 Introduction

The number of molecules defined as drug-like, obeying Lipinski's rule of five for oral bioavailability, is estimated to be 10^{60}.[1–3] With such a large number of possible molecules, it is likely that there exists a molecule that would satisfy all the necessary requirements for any given task. However, our knowledge about the full chemical space is quite limited. ChEMBL currently contains 2.3 million compounds that cover 1.5 million assays, which is a tiny fraction of the estimated space.[4] It is also not feasible to physically screen every compound in chemical space, since the current rate of 100,000 compounds a day[5] would take about 10^{51} years to complete. This process is further limited to only being possible for compounds which currently exist. Thus, the number of possible molecules remains too large to screen.

With experimental enumeration impossible to obtain, the next step is to calculate the properties of the molecules which we do not have measurements for. However, this class of potential molecules is both too large and contains molecules with too many atoms to be analyzed via quantum mechanics. Thus we turn to modeling based approaches such as scoring with Force Fields (FF) or other semi-empirical methods to calculate properties. Notably, these methods are still too slow to calculate the properties of every molecule in chemical space, but they scale to much larger molecules than can be currently handled via direct computation of quantum mechanics. Advancements in machine learning (ML) methods have resulted in a boom of predictors for a variety of chemical properties, e.g. there are 38 published ML models in the Journal of Chemical Information and Modeling when searching for "binding affinity" from 2022 alone. Importantly, ML methods, once trained, are very fast and parallelisable, which allows them to efficiently screen large libraries of available compounds. Additionally, as evidenced by ML methods' success in other fields, as we obtain larger and larger amounts of labeled chemical space, it stands to reason that the ML methods will continue to improve over time.

However, ML based methods are not without flaws. Increasingly complex ML models have become more successful, but as this complexity increases it becomes harder to interpret the reasoning behind model behavior. Ultimately, this results in treating the ML methods as black boxes, with the users having no idea how a method resolved to a particular prediction. This treatment of models also results in difficulty in comparing the various types of ML models available, as there are numerous differences in architecture and molecular input representation between the models. There is no indication as to which type of model is best for a given molecular property prediction task, nor is there any justification as to what type of input representation will provide the model with the best results.

Additionally, ML methods are prone to fitting to biases in the dataset and not being able to generalize to new data outside of their training distribution. This is a significant drawback, as chemical datasets are full of experimental error (different techniques, different labs, etc.), human bias (e.g. selecting compounds for measurement) and are a small subset of all of chemical space. ML models using 3D structural data as input compound these problems, as there is even more limited data available to these types of models. For example, the PDBbind version 2020 only contains 23,496 structures with labeled binding affinity data.[6] We know that molecular properties depend on the interaction of atoms in 3D space, so using an input representation that explicitly encodes this information makes logical sense for an ML model. However the lack of available training data limits the effectiveness of models which utilize this input representation. There is an opportunity to expand the limited data to train better models.

Lastly, papers that introduce new ML models compare to other existing models to show their improvements. These comparisons rely on performance on benchmark datasets, which can have a variety of problems. The most important problem is that given the limited amount of data available for training, good performance on a benchmark dataset does not necessarily imply good performance of the model in a real-world setting. So, while there are many new models being released which improve performance on these benchmarks, it is unclear if the

better result is due to fitting better to the benchmark, or the model better fitting to chemical space as a whole. Thus, there is an opportunity to verify a model being useful in some new wet-lab based experimental setting.

This work addresses the following knowledge gaps in applying ML models to predicting molecular properties: 1) What is the relationship between molecule input representations and model performance? 2) How can we expand the available structural data for training ML models to classify binding poses and predict protein-ligand binding affinity?, and 3) In a real drug discovery campaign, will ML models identify new potential binders?

1.1 Machine Learning for Molecular Property Prediction

In order for ML models to operate on molecules, we must represent said molecule in a machine interpretable format. There are a few excellent reviews on this topic, but to summarize, representations can be broadly broken into three categories based on their dimensionality: 1D, 2D, and 3D.[7,8] 1D descriptors are scalar values that describe something about the molecule and do not contain any information about how the atoms are interconnected in the molecule. These include molecular weight, atomic numbers, and atom type counts. In particular, since 1D descriptors do not contain information on functional groups or connectivity, there are clashes between different molecules sharing the same set of descriptors. For example, 1-Butene and 2-Butene have the same molecular weight, the same number of double bonds, 4 carbon atoms, and 8 hydrogen atoms, yet they are distinct molecules.

2D descriptors are vectors of values to describe a molecule and can encode functional groups and connectivity. The most common type of 2D descriptors are molecular fingerprints. These fingerprints are binary vectors, where each dimension of the fingerprint can represent the presence or absence of a particular subgroup of the molecule. Fingerprints represent atom connectivity and molecular topology through a variety of methods. The most common

of which are 1) keyed fingerprints such as molecular access system (MACCS) keys[9] , 2) path-based fingerprints - Daylight fingerprints[10] and RDKit[11] , and 3) Circular fingerprints - Morgan fingerprints[12]. Lastly, 3D descriptors depend on the orientation of the molecule's atoms in 3D space, such as steric properties, surface area, volume, and more.

Models can then fit these input descriptors to perform the desired task. These types of models are called scoring functions and are classified into 3 groups: force-field based[13–16], empirical[17,18], or knowledge based[19,20]. Force-field based methods utilize parameters estimated experimental/simulated data and aim to model the intermolecular potential energies of the molecule(s). Empirical scoring functions are constructed from interaction terms such as hydrophobicity and hydrogen bonding, which are parameterized to the available data. Lastly, knowledge-based methods are constructed from non-physical statistical potentials derived from the known complexes. Each of these scoring approaches utilize a linear fit of their input features to the target prediction.

ML approaches allow nonlinear fits of input features to target predictions and improve model performance. For example, RF-score, released in 2010, is a random forest model to predict protein-ligand binding affinity, and outperformed classical scoring functions at the time.[21] RF-score utilizes counts of interacting atom types, defined by two atom types being within a distance threshold, for a total of 36 features. A more recent example from 2017 is SVM-SP, a support vector regression model fit to 146 pair potentials calculated between 17 different atom types, and was more successful than traditional scoring functions to screen compounds against HIV-protease.[22]

While these ML approaches are improvements, their reliance on molecular descriptors comes with the drawback that the descriptors themselves are fixed and are not learnable to improve a model's performance. More advanced ML methods can learn their own molecular embedding space from the training data. The two most common input representations for models with this capability are: 1) grids of atomic density, and 2) molecular graphs. Grids of atomic density treat a molecule as an image, where instead of channels of red, blue, or green

pixels, one instead utilizes a channel for each atomic type and projects the molecule's 3D coordinates onto a multidimensional grid[23]. Molecular graphs treat the atoms of the molecule as nodes and the bonds of the molecule as the edges of a graph. These graphs can include other atomic/bond features such as atom type, formal charge, bond type, and stereoisomers as additional node or edge features.

The first application of an ML model learning its own descriptors was in 2017 with the first CNN for binding pose classification.[24] This network utilizes 3D atomic density grids as input, and during training identifies its own set of features to predict the score of a protein-ligand complex. After this initial paper, there have been considerable advances in the use of CNN models to predict the classification of the binding pose or the binding affinity of a protein ligand complex.[25–27] Each of these models utilizes different channels of atomic features and / or represents the atomic grid at different resolutions, while also having different architectures for the CNN. For example, Pafnucy[26] utilizes 19 channels to represent an atom including features for atom type, hybridization, number of bonds, partial charge, and others, whereas gnina[27] utilizes 28 channels that represent atom types only. This means that the performance differences between the models is entangled with architecture changes and input representation changes, making it difficult to infer which areas of the model should be evaluated for further improvements.

There has also been considerable advancements in utilizing graph based models.[28–30] DGraphDTA[28] utilizes a two-pronged approach to its input, where the molecule is represented by a graph where the node features include the element type, the number of total bonds, the number of bound hydrogens, the number of implicit hydrogens, and if the atom is aromatic, resulting in a 78 dimensional vector. A 54-dimensional feature vector which includes the residue name, position-specific scoring matrix, and a variety of other properties (e.g. pH of isoelectric point, hydrophobicity, charge, etc.) represents the protein. DGraphDTA then processes the two graphs separately and combines the output of the two processes to obtain the final prediction. MedusaGraph[29] takes a simpler approach where a node's feature is

the atom type (duplicated for receptor and ligand) and the 3D coordinate, for a total of 21 features. This singular graph is then processed to produce the model's output. Lastly, SS-GNN[30], released in 2022, uses a single graph consisting of all ligand atoms and receptor atoms that are within 5Å of any ligand atom as input. An 11-dimensional feature vector containing the atom type, charge, hybridization, valence, degree, number of hydrogens, 3D coordinates, chirality, mass, aromaticity, and whether it is a protein or ligand atom represents the graph nodes. SS-GNN then processes this single graph to produce the binding affinity prediction. Again, while these methods are all graph neural networks, the differences in architecture are compounded by differences in input representation, making comparisons between the methods difficult.

The wide variety of models introduces a new problem to users: Which model is best for my particular task? Although the papers include comparisons to other similar models within them, it is often unclear what exact data splits or cleaning procedure was utilized for each method. There are relatively few papers comparing between various model types.[31–35] Typically, the comparison paper is focused on a large number of datasets, rather than a variety of complex ML models. Others avoid more complex models and only utilize simpler ML approaches, such as k-nearest neighbors, random forests, gradient-boosted trees, and support vector regression, while eschewing the more complex deep neural nets.[34] Finally, while comparisons can contain all the relevant model types, sometimes training sets are set up such that activity classes only have up to 1000 molecules for training, which is not the environment where large/complex ML methods excel.[35] Additionally, a number of ML method development papers rely on improving performance on benchmark datasets, such as PDBbind or DUD-E. However, there have been problems reported with available benchmarks. For example, DUD-E overestimates the performance of ML models[36]. In a similar fashion, the PDBbind core set mimics the distribution of the refined set utilized in training.[6] This provides an overestimation of the ML model's performance, and commonly ML models trained on PDBbind data fail to generalize in a crossdocking setting.[27] This has led to the creation of

other more rigorous datasets, such as the maximum unbiased validation (MUV) dataset[37], but these more rigorous benchmarks have not had widespread adoption among the community.

Throughout the rise of ML based methods, advancements in different model architectures often go hand in hand with changes in input representations. Justification for these changes is typically reporting performance on benchmarks or a post-hoc analysis of the model presented in the paper. Taking protein-ligand binding affinity as a case study, many different methods achieve similar performance, with a Person's R correlation of around 0.8 on the PDBbind core set being typical of a variety of models and input representations.[24–26,30,38–45] The performance plateau indicates that almost any type of model and combination of input representations can be successful in this task. This raises the question: what is the relationship between input representations and model type on performance for predicting a given molecular property?

In order to address this, I propose an extensive analysis of different molecular representations and model types to asses how various combinations of inputs and model types affect a model's ability to predict various computed properties of molecules, e.g. molecular weight, longest path through the molecular graph, UFF energy[46] etc. By utilizing computed properties I can eliminate human sources of error present in various benchmark datasets and utilize the larger number of molecules available in ChEMBL for my analysis. Additionally, we can compute properties that we know should be difficult or impossible for certain input representations. Thus, by experimental design, if a model performs well on such a task, we can infer that there are additional sources of information in the training data from which the model is learning (e.g. human selection bias). Lastly, once we have identified the best possible input and model combinations, we can enable three analyses: 1) Are more complex models better when fully optimized, 2) How variable are these more complex models, and 3) How efficiently are these models utilizing the available training data. These analysis will result in better guidance for selecting an appropriate molecular input representation and ML model for a given amount of available training data.

1.2 Limitations of Structural Data for Machine Learning

Molecular properties are the direct result of atoms interacting in space; therefore, it makes intuitive sense to use some representation of the structure of the molecule as input. However, the scale of the available data is significantly smaller than the scale of other successful ML applications such as that used in language comprehension. Take predicting protein-ligand binding affinity as an example. PDBbind version 2020 only contains 23,496 structure[6], which both is a small portion of the total PDB (140,000 ligand bound structures) and significantly less than the 2 million assay entries of ChEMBL that lack structural information. Thus there is an opportunity to expand the available data for an ML model's training.

There are two ways in which we can expand the data for training: 1) we can generate new *binding poses* to train on and 2) we can impute *binding affinity labels* for unlabeled complexes in the PDB. Each of these approaches have problems. While it is common to utilize a docking algorithm such as AutoDock Vina[47] to generate putative binding poses for training ML models, generating even more poses for a given receptor-ligand complex has diminishing returns. Namely, it is trivially easy to generate poses of poor quality that do not reflect the interactions between the receptor and the ligand that generate the binding affinity. Adding a large number of such poses to your training data has the potential to drown out the signal from the true interactions. For the imputation of missing labels, it is unclear if such an approach will help performance, as there is a large amount of unlabeled data. For example, it is entirely possible that the features learned from the small set of labeled examples introduce systematic errors into the model calculated the imputed labels. Those errors then can then result in the imputed labels not actually being helpful for model training.

To address the limitations of generating additional poses, we make an additional assumption. We assume that a molecule will bind to a similar receptor with the same potency. This assumption allows us to match one known receptor-ligand binding interaction with many other receptors, resulting in a combinatorial explosion of data for training. This process is

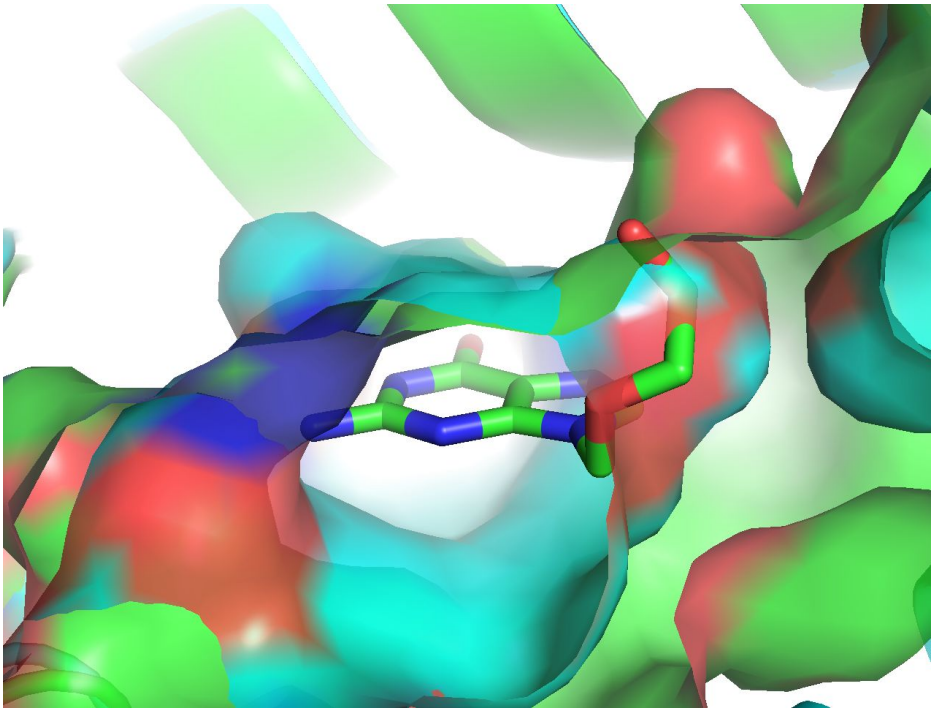


Figure 1.1: A sample overlap between 2 structures in the DCK_HUMAN_2_260 binding pocket. The cognate receptor (3mjr) and native ligand pose is shown in green, and the aligned crossdocked receptor (1p5z) is shown in cyan. Notice that the ligand contains a carbon tail which intersects into the polar density created by an oxygen in the crossdocked receptor. This clash is absent in the cognate receptor. Thus, utilizing the native ligand pose as the ground truth for the crossdocked receptor is likely incorrect.

known as crossdocking, and has the benefit of resulting in a measure of a model's ability to predict drugs in a non-cognate receptor context. By training on a majority of non-cognate receptor data, the resulting model is more applicable to real-world drug discovery scenarios where it is unknown if a drug will bind a target receptor or not.

While the aforementioned properties of crossdocking are nice, utilizing such a setup introduces a new potential problem for training ML models. It is not trivial to determine what the ground-truth binding pose for a ligand is in a crossdocking context. The non-cognate receptors can have different orientations of similar residues, or entirely different residues present in the binding pocket. This can result in the ligand pose for the cognate receptor producing steric clashes with the new receptor (Figure 1.1). Additionally it is also possible

for the ligand to have a different binding pose in the non-cognate receptor, even if the ligands cognate pose does not produce steric clashes with the new receptor. Each of these situations could hurt the ML model's performance during training, and it is unclear if that is enough of a detriment to outweigh the potential performance gains of including more data to train on.

The second approach of imputing labels for ligand bound structures expands available training data by providing additional training examples to learn from, at the cost of said examples having noisy labels. Broadly speaking, there are 4 classes of missing data: Structurally Missing Data (SMD), Missing Completely At Random (MCAR), Missing At Random (MAR), and Missing Not At Random (MNAR). Data SMD refers to data where a given entry is not supposed to have a value in a field (e.g. the age of a first child for a person with no children). Data MCAR describes when missing values are independent from the observed and unobserved entries. Notably, data MCAR affects statistical power, but does not introduce bias into the sample. Data MAR, however, describes data whose probability of being missing depends on some known property. Lastly, data MNAR refers to data whose probability of being missing depends on unobserved measurements. Data MNAR is especially problematic since future observations cannot be predicted without bias from the model. Due to the numerous biases present in our small samples of chemical space and further reductions in the amount of structural information available, we are dealing with data MNAR.

Lately, ML based approaches have become more popular for data imputation, with k nearest neighbors and iterative random forests being especially successful on a variety of missing data situations[48,49]. While these methods are successful, they are relatively primitive approaches to ML. In essence, these models are learning some latent representation of the training data, and utilizing this representation to guess labels for the unlabeled data. We hypothesize that similar to utilizing sophisticated models to learn their own representations of chemical space in tasks like protein-ligand binding affinity, we can utilize the same model to impute labels for missing data. This has a small compute benefit, as the model that we are evaluating is the same as the one that is fitting to the data, which means that a separate

"data imputation model" does not need to be fit. Additionally, this setup allows for an iterative approach to the imputation where we can successively impute the missing labels and retrain the prediction model. This results in an easy to implement workflow that can be applied to any ML model fitting any dataset.

Recently, there has been considerable development into new ML architectures and input representations to predict protein-ligand binding affinity. While these advancements are significant, we are approaching limitations in absolute model performance due to the lack of available structural data for training. We investigate two ways to expand the available structural data: 1) Expanding the available binding complexes by utilizing crossdocking to generate putative poses of ligands in related proteins where we assume that they can bind, and 2) imputation of binding affinity for unlabeled complexes in the PDB. Crossdocking allows for both a combinatorial expansion of available binding pose data, since instead of one protein and one ligand to generate poses, we have many proteins per ligand, and it also serves as a better indicator of model performance. With a crossdocking dataset, the majority of poses are from ligands in non-cognate receptors. Notably, predicting properties for a ligand in non-cognate receptors is the task in drug discovery, as you are not searching for a known protein-ligand interaction but something new. Imputation, on the other hand, serves to provide pseudo-labels for training a new model. Neither of these approaches has been utilized in predicting protein-ligand binding affinity, and we will demonstrate both strategies are effective at improving the performance of an ML model.

1.3 The Profilin1 - Actin Complex

The prior sections of this work focused on methods and evaluations to improve ML model training. While these experiments are useful, it is also important to show that the models generated are applicable to real-world drug discovery campaigns. We demonstrate this using

our deployed models to assist in an ongoing drug discovery campaign targeting the Profilin1 (PFN1) protein's interaction with actin.

PFN1 is a key protein in the angiogenesis pathway, and aberrant angiogenesis is a component of many disease states. Notably, upregulated angiogenesis is a hallmark of clear-cell renal cell carcinoma (ccRCC) tumor microenvironment.[50] Furthermore, PFN1 is a known regulator of actin dynamics and actin-based cellular processes such as migration and proliferation[51–56] and has been identified as a marker of late stage ccRCC.[57,58] Additionally, upregulation of PFN1 in vascular endothelial cells is implicated in proliferative diabetic retinopathy (PDR), one of the leading causes of blindness worldwide[59]. In the progression of PDR, up-regulation of PFN1 results in an increase in PFN1-actin binding, which in turn upregulates processes related to cell mobility and angiogenesis. Thus, the PFN1-actin binding interaction is an attractive clinical target. Notably, this interaction in particular is attractive as a testing ground for our ML methods, as the protein-protein binding surface is flat and thus devoid of traditional binding pockets (Figure 1.2). Ergo, finding a small molecule that mimics a protein-protein binding interaction is difficult for traditional modeling methods.

Gau et al.[60] identified a small molecule, C2, which could disrupt the PFN1-actin binding interaction *in vivo*. Although C2 was effective in disrupting the PFN1-actin interaction, the concentration necessary to do so was 50-100uM[60]. This concentration is too high for a commercial drug and, while we know that C2 disrupts the protein-protein interaction, its mechanism of binding is unknown. This adds additional difficulty in trying to optimize C2 for better potency, as we do not know what interactions C2 is making with PFN1 or actin to achieve its function, nor do we know what potential interactions are available at the binding site to improve binding efficiency.

Improving upon C2 provides an excellent research challenge, as its important molecular features are unknown, its binding site is unknown, and protein-protein interactions are a difficult problem for traditional modeling methods. Therefore, the success of ML methods on identifying a molecule to target this protein-protein interaction would be strong support that

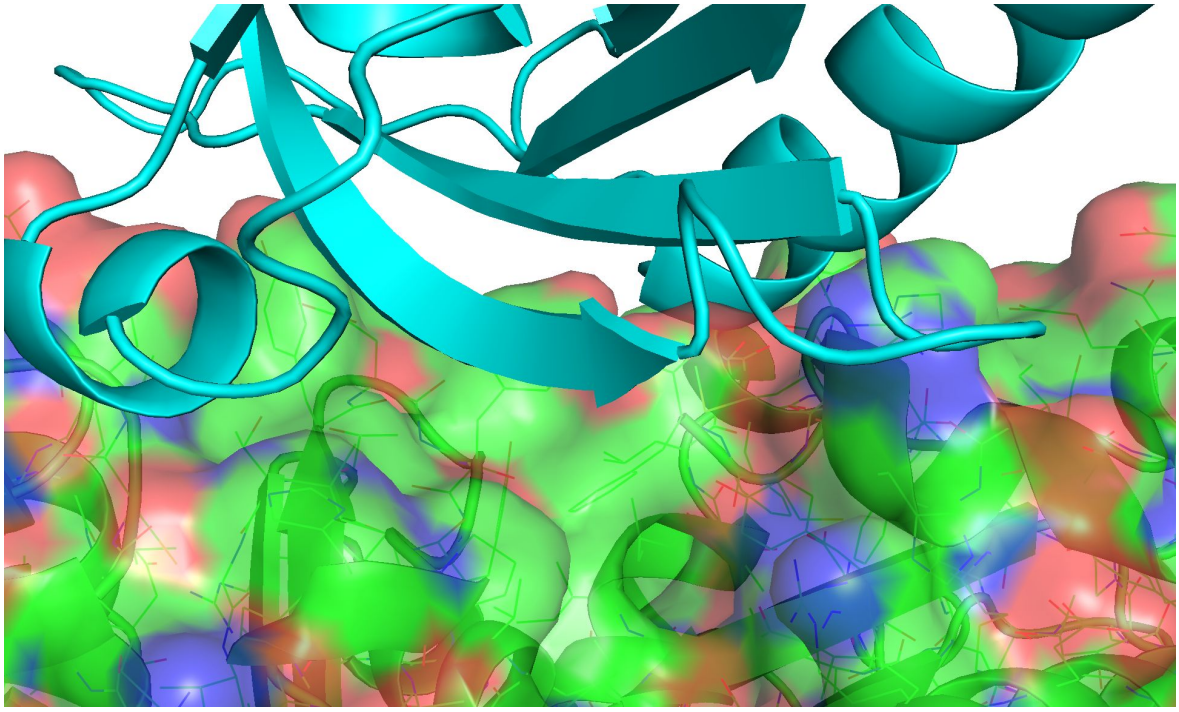


Figure 1.2: The PFN1-actin binding interface from 2BTF in the PDB. Actin is shown in green. PFN1 is shown in Cyan. Note that the PFN1-actin interface is relatively smooth and devoid of traditional ligand binding pockets.

such *in silica* methods are valuable in difficult experimental settings. We propose utilizing a structure-based virtual screen to identify new candidate molecules to target the PFN1-actin binding interaction, using the models developed during the previous sections of this thesis to score the candidates.

1.4 Dissertation Overview

ML methods for molecular property prediction have seen rapid advancements over the past decade. However, changes in model architecture are often accompanied by different input representations, making it difficult to determine the best model class or what the most appropriate input representation is for a given property prediction task is. Additionally for structure based ML models there is a general convergence of performance on benchmark datasets like the PDBbind. The sizes of the datasets used for these benchmarks pale in comparison to those used in other tasks where ML is successful, such as image recognition. Lastly, ML models are compared by their performance on these benchmark datasets, which are unrepresentative of the use case of these models in actual drug discovery campaigns. This work seeks to address these problems through three sections of work. First, we evaluated a variety of ML model types and input representations on computed molecular properties to investigate the relationship between the input representation of a molecule and the ML models that are successful at predicting various properties. Second, we expand the available data for training structure based ML models through crossdocking and the imputation of missing labels. Lastly, we demonstrate the viability of these techniques by utilizing them to screen for new compounds to target the challenging PFN1-actin protein-protein binding complex in a drug discovery campaign.

2.0 Investigating the Relationship Between Molecule Input Representations and Model Performance

2.1 Summary

Various machine learning (ML) models have been developed for molecular property prediction tasks. These models span a large variety of general architectures and input representations, often both at the same time. This in turn makes the relationship between input representation and model architecture unclear. Here we characterize the performance of ML models on several computed properties in order to investigate the relationship between input representation and model performance. Fitting power laws to expanding versions of the training data also allow us to estimate how ML models will improve as more data becomes available. We demonstrate that more complex ML models both empirically outperform the simpler models, and also utilize training data more efficiently. Additionally, our analysis of fitting the UFF energy of conformers demonstrates the importance of analyzing model behavior, as our 3D model's did not perform as expected. Lastly, fitting the UFF energy was our hardest task and our fits of successively increasing training sets showed that fitting random forests is the best strategy for good model performance until we hit 1 million training points. This suggests that simpler methods are more effective in challenging, data limited settings and warrants further investigation.

2.2 Introduction

The large size of chemical space, estimated at 10^{60} [1–3], is too vast to label experimentally. Thus, in drug discovery or other screening campaigns there is a need for fast and effective chemical property predictors to narrow down the search space in order to find the subset of molecules most likely to succeed at some task. Traditionally this is performed by modeling molecules in some way such as scoring with force fields or other semi-empirical methods, as the space is far to vast for quantum mechanics calculations. Machine learning (ML) models have started to dominate the field of chemical property predictors with a plethora of models being published. For example, there are 38 ML models published in the Journal of Chemical Information and Modeling in 2023 alone when searching for "binding affinity". ML methods are very attractive, as once trained they are fast, parallelizable, and very accurate in the regime of their training data. This allows trained ML models to efficiently screen large libraries of available compounds. Finally, ML methods tend to perform better as the volume of training data increases. As more experimental measurements of different molecules are verified and stored, it stands to reason that ML methods will continue to become better in the future.

However, these approaches are not without their flaws. ML methods, due to their nonlinear modeling, are prone to fit to biases in the training data.[36,61–64] This is exasperated for small molecule property prediction as we are very data limited. ChEMBL is a database containing 2.3 million compounds covering 1.5 million assays, and only represents a tiny fraction of all of chemical space.[4]

Problematic training data aside, there is another fundamental problem when attempting to compare between two ML models. Namely, that performance differences between two different models could be due to differences in their architecture OR differences in their input representation. Often, when new methods are released, both the architecture and the input representations are different from other established models. Thus, when comparing to other

reported numbers on a benchmarking dataset it is unclear if the new model is better because of the new architecture, or if it found a better way to represent molecules to a computer. There have been very few comparisons between various models in a rigorous fashion.[31–35] These comparisons also tend to focus on a large number of datasets.[31–33], can eschew complex models[34], or have very limited training data[35]. All of these issues mean that the results of the comparisons tend to be rather generic and do not offer meaningful information about what type of model is best for a given task.

The other type of comparison happens when a new model is published and the authors compare the performance metrics (root mean squared error, Pearson R, etc.) of their model with the published numbers of other models on a particular dataset. Better metrics are used to justify or ad-hoc explain why differences in model architecture or input representation for the published work are better than in the prior models. However, there are potential caveats to this approach as well. The values of certain metrics depend on the underlying data distribution, which can be the result of a random data generation process (e.g. the generation of docked poses), which could be different between the two publications.

In addition, advances in model architectures often go hand in hand with different input representations of the molecules. This means that the post-hoc justification of better performance metrics is insufficient to explain if the different model architecture or a better representation is the cause of the improvement. Model architecture differences can be probed with ablation studies, but this practice is not as applicable to input representations. In total, these issues make it so that if one is given a set of data and a desired property prediction task, there is no way to tell what model type or input representation is the best for the desired task.

We can partially address this overall knowledge gap through the investigation of three questions: 1) Are more complex models better when fully optimized, 2) How variable are more complicated models, and 3) How efficiently are models utilizing the available training data. In order to remove as much bias as possible from our analysis, we will investigate

computed properties of molecules in the ChEMBL dataset. We eliminate sources of error due to experimental measurement by having a completely deterministic label generation procedure for the computed properties. With this dataset we can then train and evaluate a variety of models ranging from the relatively simple ridge regression model, to the much more complex molecule attention transformer[65], which also span a variety of input representations (e.g. chemical fingerprints, grids of atomic density, and graphs). Optimizing the various models' hyperparameters will allow us to answer the first two questions by comparing the absolute performance of the best performing model in each class, and analyzing the variability in model performance during the hyperparameter sweep.

Kaplan et al.[66] demonstrate that the cross entropy loss of large language models empirically follow power laws with model size, compute budget, and dataset size. A power law relationship is described by the following equation:

$$f(x) = a \cdot x^k \tag{2.1}$$

where a is a scaling factor, and k describes the proportional change in the other quantity. Importantly, power laws are scale invariant.

$$\begin{aligned} f(x) &= a \cdot x^k \\ f(cx) &= a(cx)^k \\ f(cx) &= c^k(ax^k) \\ f(cx) &= c^k f(x) \end{aligned} \tag{2.2}$$

This means that all power laws with a particular scaling exponent are equivalent up to a constant factor. This means that fitting a power law between dataset size and a model's test set performance can yield a framework to detect diminishing returns. When the power law is true, we can expect a fixed proportional increase in test set performance by doubling the

amount of training data. However, if the power law overestimates the performance gains of adding training data, then we have evidence that the model has reached diminishing returns where the addition of more training data results in fewer performance gains. Additionally, if the power law fits well, then we can compare the rates of performance gain as data is added (the exponent), which will tell us how efficiently a model is utilizing the available training data. That is, a model which lowers the test set error faster (has a more negative exponent) improves more from the same amount of training data. This is possible as due to the scale invariance of power laws, we can ignore the scaling factor a during our comparisons. Lastly, a consistent power law across tasks can give insight into how well a particular model and input representation combination will generalize its performance to new tasks.

This work seeks to investigate the relationship between input representation and model type on the performance of ML models on a variety of computed property prediction tasks. Through the training and evaluation of these models we will be better able to suggest input representation and model combinations for a particular property prediction task given the amount of training data available and how well we expect said model to scale as more data becomes available. By training on predicted property labels we know that the sources of variation between the models trained in this work is due to intrinsic qualities of the model and input representation, as the labels were generated through a defined process. This eliminates the possibility of the more complex models achieving better performance through a better propensity to fit to noise inherent in experimental labels. In this work we examine ridge regression, support vector regression (SVR), random forests (RF), and feed-forward neural networks trained on molecular fingerprints, a convolutional neural network (CNN) trained on grids of atomic density, and a molecule attention transformer trained on 2D and 3D molecular graphs.

2.3 Methods

Here we describe the ChEMBL dataset preparation, input representations, and model types.

2.3.1 ChEMBL dataset preparation

The SMILES strings for ChEMBL30 were downloaded from the ChEMBL website's download service. We then calculated the labels for the 2D property dataset consisting of the molecular weight (MolWT), number of rings (NumRings), cLogP, and the longest path through the molecular graph (LongestPath) with rdkit for each molecule. We then removed the molecules with a molecular weight under 100Da or over 1000Da. We also removed molecules where the LongestPath calculation failed. This resulted in a total of 1,984,674 molecules being retained. We refer to this grouping of molecules as the "2D dataset". For each molecule we then calculated the default rdkit fingerprint with bitsize 2048 (shortened to rdkitFP), the Morgan fingerprint[67] with radius 2 and bitsize 2048, and every descriptor in rdkit's Descriptors module (shortened to Feature). Note that the rdkit Descriptor vector contains the MolWT, NumRings, and cLogP. Since these are 3 out of 4 labels of our dataset, the Feature vector serves as a trivial baseline for information content of a representation for the MolWT, NumRings, and cLogP prediction tasks. The data was then randomly split into a single 75% training and 25% testing set. A histogram of the train-test split for each calculated property of the 2D dataset is shown in Figure 2.1.

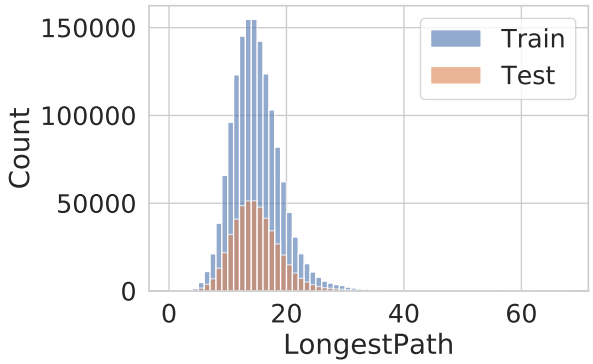
The 3D property datasets were similarly generated. First, rdkit was utilized to generate up to 10 conformers for each molecule, which were then saved as sdf files. We then calculated the radius of gyration and UFF energy for each conformer, which were then independently filtered. For the radius of gyration, we removed poses where the calculation failed, resulting in a total of 26,060,341 conformers with each molecule having an average of 8.75 conformers. We refer to this grouping of molecules as the "RadGyr dataset". For the UFF energy, we removed

conformers with over 1000kcal/mol as the extreme values were on the order of 10^6kcal/mol and were interfering with model training. In total 24,112,889 conformers were retained, with each molecule having an average of 8.36 conformers. We refer to this grouping of molecules as the "UFF dataset". We then randomly split both groupings of molecules into a single 75% training and 25% testing set each. A histogram of the train-test split of the RadGyr and UFF datasets is shown in Figure 2.1e,f.

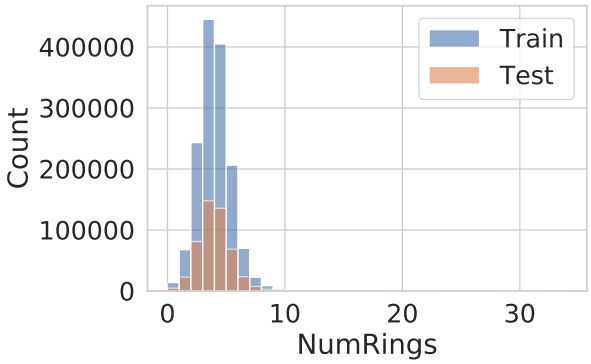
Lastly, we generated a successively growing version of each training set. This was done by randomly selecting an initial small version of the training set, and then successively growing the dataset through the addition of more randomly selected training data. For the 2D dataset this corresponded to 50k, 100k, 200k, 500k, and 1 million molecules being selected for the training set. For the 3D datasets, instead of dealing with entire molecules we are dealing with conformers of molecules. So, when we apply the same random selection procedure to the 3D datasets, we are acting on conformers. Thus the same molecule is potentially split across different sizes of the training set. For the RadGyr and UFF datasets, this random growth procedure resulted in training set sizes of 100K, 200K, 500K, 1 million, 2 million, and 10 million **poses** being generated.

2.3.2 Input Representation and Model setup

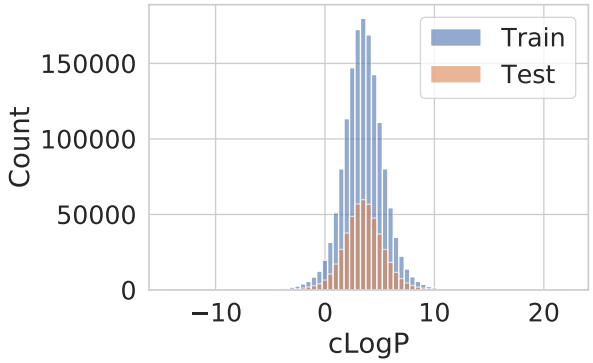
We evaluated several model types: linear regression, ridge regression, support vector regression (SVR), random forest regression (RF), feed-forward neural networks (F.F.), convolutional neural networks (CNN), and transformers. The linear, ridge, SVR, RF, and F.F. models were all trained on either rdkit's path-based fingerprint, the Morgan fingerprint, or rdkit's descriptor fingerprint as described above. The CNN is grid based and uses voxels of atomic density as described by Ragoza et al.[68]. Lastly, the transformers are the molecule attention transformer as described by Maziarka et al.[65]. The linear, ridge, SVR, and RF models were all implemented via the scikit-learn package[69]. The F.F., CNN, and transformer were all



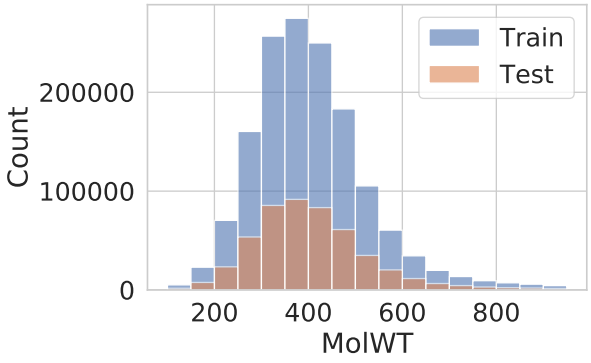
(a) Histogram of the LongestPath prediction task. Bin size is 1.



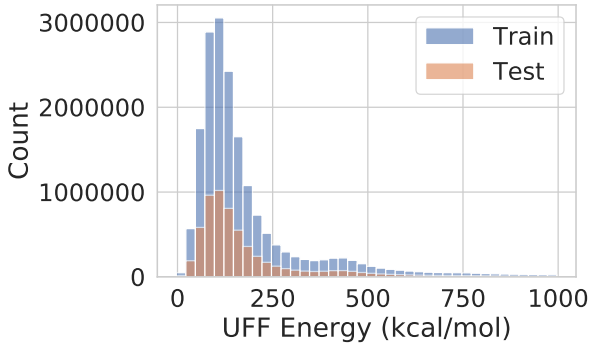
(b) Histogram of the Number of Rings prediction task. Bin size is 1.



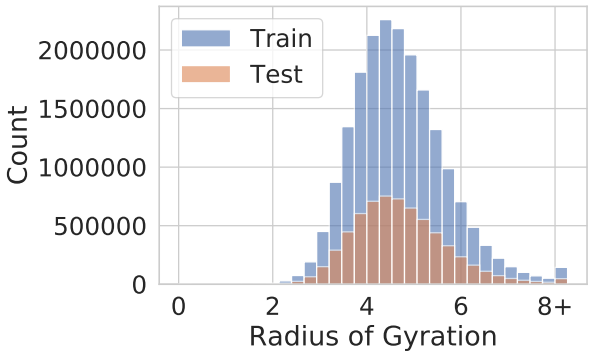
(c) Histogram of the cLogP prediction task. Bin size is 0.5.



(d) Histogram of the Molecular Weight prediction task. Bin size is 50Da.



(e) Histogram of the UFF Energy prediction task. Bin size is 25kcal/mol.



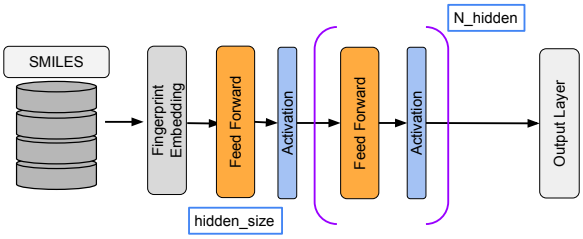
(f) Histogram of the Radius of Gyration prediction task. Bin size is 0.26Å.

Figure 2.1: Histograms of the train-test split for each of the predictive tasks. Notably, for the Radius of Gyration histogram, all radii greater than 8 were put into the same bin for visual clarity.

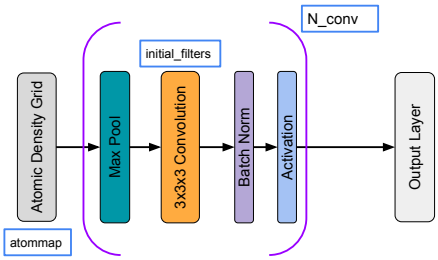
implemented in PyTorch[70].

The general architectures for the F.F., CNN, and transformer are shown in Figure 2.2. The F.F. network is a series of a series of blocks. A block consists of a single fully connected layer mapping its input to a defined hidden dimension followed by a non-linear activation function. The F.F. network consists of an initial block going from the input dimension to a defined hidden dimension, then some number of additional blocks (see n_hidden in Figure 2.12), and a final fully connected layer to produce the output. Similarly, the CNN also consists of a series of blocks. Here each block is a Max Pool, followed by at 3x3x3 convolution, batch norm, and non-linear activation function. However in this setup, the first convolution's output number of channels is defined (initial_filters) and each subsequent block doubles the number of output channels for each convolutional layer. The output layer is a single fully connected layer to produce the predicted label.

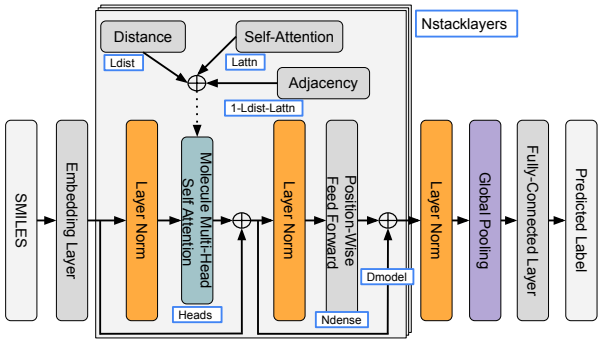
Lastly, the transformer is the same molecule attention transformer defined by Maziarka et al.[65]. Briefly, this model works on molecular graphs where each node is an atom, and each edge is a bond. Each node has an associated feature vector: a one hot encoding of the atom's identity (B, C, N, O, F, P, S, Cl, Br, I, or Other), the number of bonds [0,5], the number of bonded hydrogens [0,4], the formal charge (-1,0,1), if the atom is in a ring, and if the atom is aromatic. Additionally, the pairwise distance matrix between all atoms is computed. The network consists of a number of blocks (Nstacklayers) where each block consists of self attention building up to a final internal representation vector of a specified size (Dmodel), which is then fed into a fully connected layer to produce the predicted label. In the Molecule Multi-Head Self Attention layer, we utilize a weighted combination of self-attention on the graph's node features (Self-Attention in Figure 2.2c), adjacency matrix, and distance matrix as our input into the self-attention process. The weight applied to each of these matrices is defined by the Ldist and Lattn hyperparameters, with the remaining weight for the adjacency matrix being defined by $1 - Ldist - Lattn$. In total these different model architectures were selected to represent a sliding scale of complexity, from linear regression to the transformer



(a) General Feed-Forward Architecture



(b) General Convolutional Neural Network Architecture



(c) General Transformer Architecture

Figure 2.2: General architectures used during hyperparameter sweeps for the feed-forward (F.F.), convolutional (CNN), and transformer neural networks. The F.F. networks utilize a molecular fingerprint as input, the CNN utilized grids of atomic density, and the transformer utilizes a molecular graph. Each of the blue boxes is a hyperparameter that was tuned during the sweep.

neural network architecture.

In order to meaningfully compare between these difference architectures, we first had to optimize each model's hyperparameters for every prediction task. This was done via a two stage process utilizing the tools from Weights and Biases.[71] First, a grid or random search was performed over a broad range of hyperparameter values. Then, the results of the first search was utilized to set the ranges of hyperparameters to explore with a Bayesian hyperparameter search. As implemented by Weights and Biases, the Bayesian hyperparameter search utilizes a Gaussian Process[72] to model the relationship between the other hyperparameters and a specified test set metric. For all models we set the hyperparameter optimization to minimize

the root mean square error (RMSE) of the test set. The fingerprint based models utilized the full training and test set setup during hyperparameter optimization. However, due to the computational demands of the CNN and transformer model, we only utilized the reduced training set of 1 million poses for the training set to speed up the hyperparameter search. The ridge regression, SVR, RF and F.F. models were tuned for each of the cLogP, MolWT, LongestPath, and NumRings predictive tasks as these are 2D calculated properties and the input to each of these models is a 2D fingerprint. Similarly, the CNN and transformers were tuned to the UFF Energy predictive tasks, as these models incorporate 3D conformer information and the UFF Energy is a conformer specific property.

For the ridge regression model we explored the strength of the L2 regularization and the number of iterations utilized by sklearn in the conjugate gradient solver. The general search space of the SVR was to cover the regularization parameter and the tolerance for the stop criterion specified by the sklearn python package. The RF model sweep covered the number of trees in the forest, how the leaves were split and pruned, and how deep each tree was allowed to be. For the F.F. models the sweep generally covered which nonlinear activation function was utilized, the size of the hidden dimension, and the number of hidden layers in the network. The CNN model generally explored different atom type definitions (covering additional types for aromatic/aliphatic carbon, donor/acceptor Nitrogen, polar/apolar Hydrogen, and donor/acceptor Oxygen), the initial dimension of the first convolution layer, the number of convolutions, and the nonlinear activation function. Lastly, the Transformer model swept over the number of dense layers in the self-attention blocks, the number of attention blocks, the number of heads in each self-attention layer, the hidden dimension of the model, and the weighting of the distance matrix, adjacency matrix, and the graph node's feature vector as input to the self-attention layer. The full details of each hyperparameter sweep are shown in Figure 2.12. Notably, we stopped the two stage process for the F.F. models on the cLogP and MolWT predictive tasks after the Bayesian stage failed to improve the F.F. models on the LongestPath and NumRings tasks. Similarly, to save on computational expense, only the

random hyperparameter sweep was performed on the CNN and transformer models. The top models were selected based on their RMSE on the test set.

2.4 Results

Here we present the results of our extensive hyperparameter sweep, the evaluation of the best performing model on each predictive task, and the fit of power laws to our growing training set sizes.

2.4.1 More complex models are better

The RMSE of the best performing model during the hyperparameter sweep is shown in Table 2.1. For the fingerprint based models we observe that the Morgan fingerprint based models outperform the rdkit fingerprint for all of the 2D predictive tasks. This implies that the connectivity based Morgan fingerprint is a better representation that the path based rdkit fingerprint. However, we note that we selected a single bit size for the fingerprint vector and did not perform any hyperparameter sweeps over the fingerprints themselves. Thus the differences we observe could potentially be removed by optimizing our fingerprint selection. We also note that the rdkit feature vector contains the Number of Rings, cLogP, and Molecular Weight as part of the vector, which means that models trained with this input should be able to achieve perfect predictions.

Interestingly, this near zero performance is not observed with the SVR and F.F. models on the Molecular Weight predictive task. While the RMSE is not zero, it is less than the weight of a Hydrogen atom for each of these cases. Still, the nonzero performance is surprising as both the SVR and F.F. models have L2 regularization on the model weights. A perfect fit is possible though the assignment of 0 weight to all non-molecular weight input features

(207 total) in the input vector, and a weight of 1 to the remaining molecular weight input feature. However, it is possible that a combination of a small amount of weight on the other 207 features would lower the weight assigned to the molecular weight input feature. This situation would be preferred by the implementation of the regularization for the SVR and F.F. models, and would lead to more error. In general, the regularization of these more complex models is designed to avoid the situation where the model hyper-focuses on a small set of input features to determine the output, which is precisely the behavior we desire in this case. For the RF models, this behavior is less impactful as for a given decision tree, the optimal information yield is to split nodes along this singular input feature (or other input features that are highly correlated with the molecular weight). Thus, the error is likely due to having to group the data into bins for regression, where a leaf's prediction is the mean molecular weight of the members of said leaf. Lastly, since 3/4 properties are contained in the rdkit feature vector, and our observation that Morgan fingerprint based models outperformed the rdkit fingerprint models, we utilized the Morgan fingerprint as input for the 2D models for the remaining analyses.

Additionally, for the 2D property prediction tasks, we observe that as the model complexity increases, we observe better performance across both the Morgan and rdkitFP input representations (Table 2.1). This implies that the more complicated models are better able to fit the information present in the fingerprint representations. However, this trend of more complicated models having better performance was not necessarily observed with the 3D tasks. We expected that the Transformer would outperform the CNN, but this was only true for the UFF energy predictive task (Table 2.1). For the Radius of Gyration predictive task, the CNN outperformed the Transformer. We suspect that this is due to the grid representation of the CNN completely representing the radius of the molecule, whereas the 3D pairwise distance matrix of the Transformer does not explicitly represent the center of mass of the molecule and its corresponding radius.

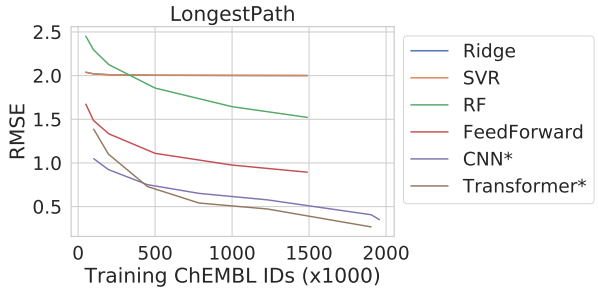
The final model endpoints shown in Table 2.1 do not tell the full story. It is also important

Task	Input	Linear	Ridge	SVR	RF	F.F.	CNN	Transformer
Longest Path*	Feature	1.669	1.275	1.278	0.763	0.417		
	Morgan	1.981	2.001	2.001	1.522	0.894		
	rdkitFP	2.971	2.989	2.989	1.828	1.285		
Number of Rings*†	Feature	$6.41e^{-14}$	$2.31e^{-6}$	$4.1e^{-4}$	$2.45e^{-3}$	$6.53e^{-3}$		
	Morgan	0.570	0.570	0.570	0.402	0.199		
	rdkitFP	0.725	0.725	0.725	0.437	0.301		
cLogP*†	Feature	$8.08e^{-14}$	$8.443e^{-6}$	$9.6e^{-4}$	$3.02e^{-3}$	$8.41e^{-3}$		
	Morgan	0.916	0.921	0.921	0.732	0.366		
	rdkitFP	1.181	1.186	1.186	0.758	0.516		
Molecular Weight*†	Feature	$1.66e^{-11}$	$1.372e^{-5}$	0.159	0.00372	0.458		
	Morgan	50.136	50.364	50.364	43.085	21.738		
	rdkitFP	72.168	72.439	72.439	43.801	29.851		
UFF Energy‡	Grid						115.838	
	Graph							114.516
Radius of Gyration‡	Grid						0.1634	
	Graph							0.2265

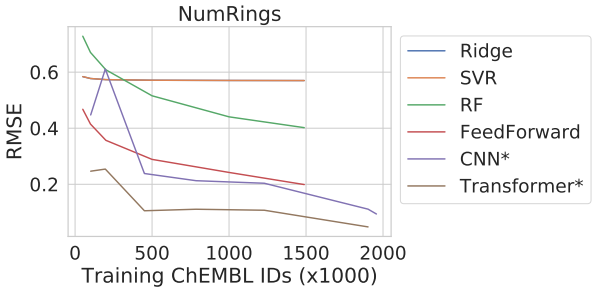
* Training and testing on the 75%-25% random split of the 2D dataset.
† rdkit feature vector contains label as part of the input
‡ Train - 1million subset of the corresponding dataset, Test - full 25% random split of the dataset.

Table 2.1: Test set RMSE for the best performing model of each model type during the hyperparameter sweep for each calculated molecular property task.

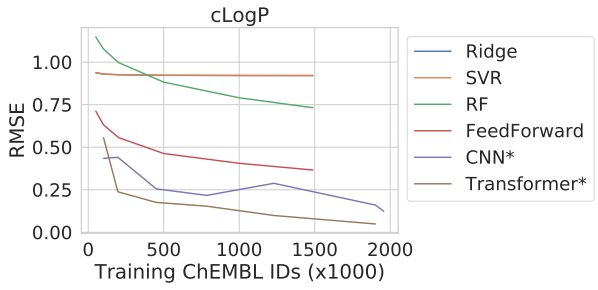
to determine how performance scales as a function of the amount of training data available. To determine this we trained the best performing (lowest test set RMSE) ridge, SVR, RF, F.F., CNN and Transformer model on our successively growing training sets (Figure 2.3). We utilized the Morgan fingerprint for our input representation for the ridge, SVR, RF, and F.F. models. For all tasks, the ridge and SVR model achieved the same performance. We also observe the general trend of SVR performing worse than the RF, which does worse than the F.F., which does worse than the CNN, which does worse than the Transformer across nearly every task, given a sufficient amount of training data is available. The exception to this general trend is the UFF energy regression task where the CNN outperforms the Transformer and on 10million data points the F.F. neural net also outperforms the transformer (Figure 2.3e).



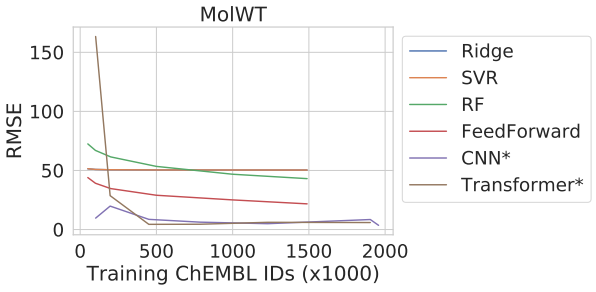
(a) Longest Path test set RMSE



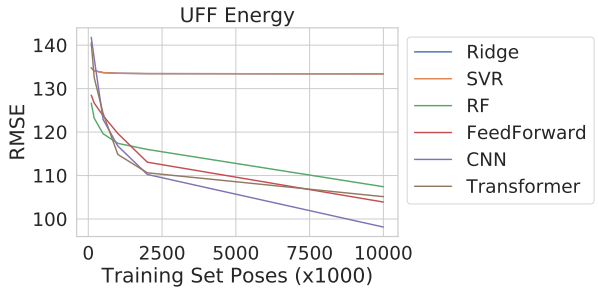
(b) Number of Rings test set RMSE



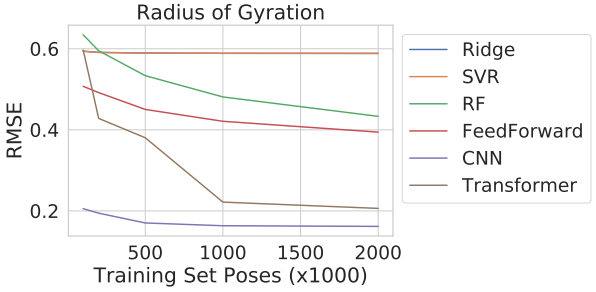
(c) cLogP test set RMSE



(d) Molecular Weight test set RMSE (Da)



(e) UFF Energy test set RMSE (kcal/mol)



(f) Radius of Gyration test set RMSE

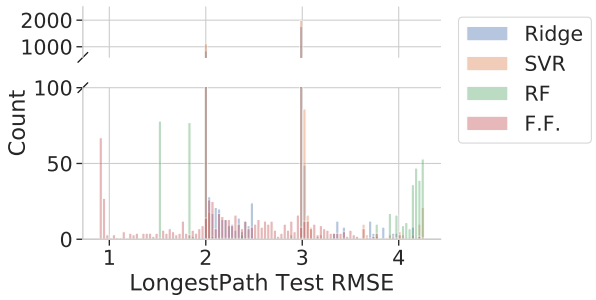
Figure 2.3: Best performing models on each of the prediction tasks. The Ridge and SVR models are completely overlapping. For (a)-(d) the CNN and Transformer models utilized the 3D RadGyr dataset with the corresponding calculated property, while the other models utilized the 2D dataset. This is indicated with an asterisk, as the dataset for the Ridge, SVR, RF, and FeedForward models are distinct form the dataset used for the CNN and Transformer. For (e) all models utilized the UFF dataset, and for (f) all models utilized the RadGyr dataset. F) only goes to 2 million poses due to computational complexity preventing the completion of the 10 million, and full dataset. Similarly, the UFF energy set did not have the full dataset completed.

2.4.2 More complex models require more hyperparameter tuning effort

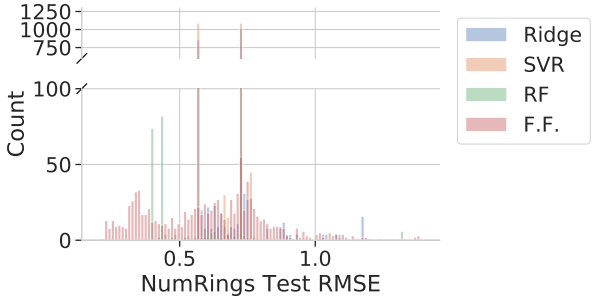
In the prior section we compared the performances of the best model, by test set RMSE, for each model type on each predictive task. While useful, it is also relevant how much computational effort goes into finding the best model, especially if you are compute limited. We can quantitatively assess the effort needed to find the best performing model by analyzing the variability of the test set performance across the hyperparameters analyzed during the sweep. We also only care about models which exhibit "good" test set performance. Here, we defined this to be performing better (lower test set RMSE) than a model which simply predicts the mean of the training set for every molecule. We also note that we do not really care about the hyperparameter sweeps that utilized the rdkit Feature vector as their input. Figure 2.4 shows the histograms of the test set performance of the Ridge, SVR, RF, and F.F. models on the 2D dataset, and the Transformer and CNN models on the UFF and RadGyr datasets.

It is clear that both the Ridge and SVR models have a very narrow distribution over their hyperparameters (Figure 2.4a-d). This implies that generally, a small hyperparameter sweep is sufficient to cover the possible performance space for these models. On the other hand, the RF, F.F., Transformer, and CNN models all demonstrate a much wider range of test set performance given a set of hyperparameters. Thus, for these more complex methods, a larger hyperparameter sweep is necessary to cover the optimal hyperparameter space.

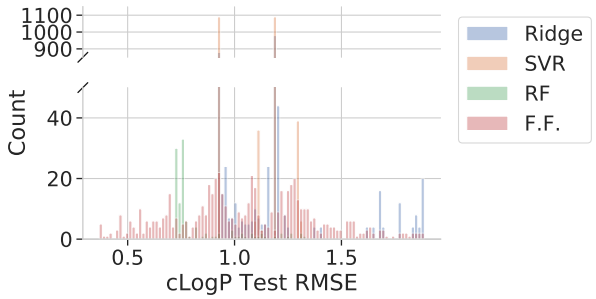
Table 2.2 shows the number of models that were excluded from the histograms in Figure 2.4. Notably, going along with their narrow distributions, the Ridge and SVR models were unlikely to have a combination of hyperparameters that resulted in a model worse than one that predicts the mean of the training set. Interestingly, there were a large number of RF models that exhibited this poor performance. This is due to the combination of two hyperparameters explored in the initial search, ccp_alpha and min_impurity_decrease. If both of these hyperparameters were not set to 0, then the resulting RF model collapsed to a fit worse than



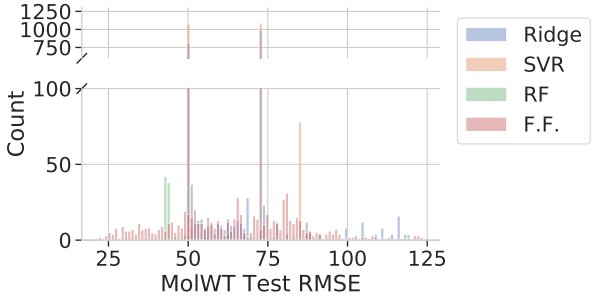
(a) LongestPath test set RMSE histogram. Mean Training set predictor RMSE: 4.2999.



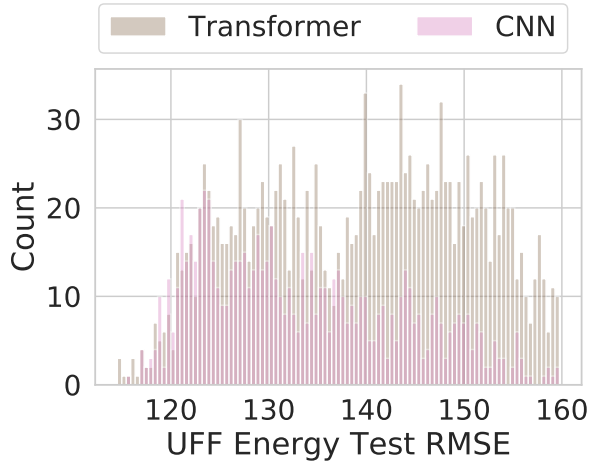
(b) Number of Rings test set RMSE histogram. Mean Training set predictor RMSE: 1.4106.



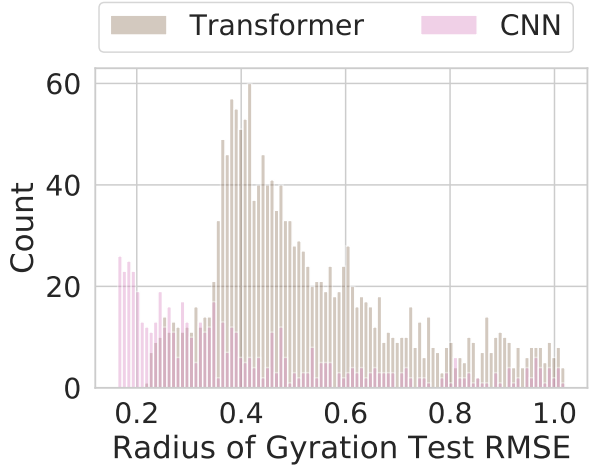
(c) cLogP test set RMSE histogram. Mean Training set predictor RMSE: 1.9033.



(d) Molecular Weight test set RMSE histogram. Mean Training set predictor RMSE: 124.8209.



(e) UFF energy test set RMSE histogram. Mean Training set predictor RMSE: 160.2122.



(f) Radius of Gyration test set RMSE histogram. Mean Training set predictor RMSE: 1.0202.

Figure 2.4: Test set RMSE of each model trained during the hyperparameter sweep. We removed models trained with the rdkit Feature vector, and models whose test set RMSE was equivalent or worse than predicting the training set mean for each molecule.

Property	Ridge	SVR	RF	F.F.	CNN	Transformer
LongestPath	25/2958	127/3449	748/865	693/1321		
Number of Rings	8/2160	181/2451	692/923	650/1683		
cLogP	2/2158	179/2449	748/865	693/1321		
Molecular Weight	0/2160	183/2450	943/1058	589/1320		
UFF Energy					207/999	308/2000
Radius of Gyration					354/945	311/2000

Table 2.2: Number of models not shown in the histograms of Figure 2.4, due to the test set performance being equivalent or worse than predicting the mean of the training dataset for every molecule. Each column is a model type, and for each task the numerator is the number of models removed, and the denominator is the total number of models evaluated in the hyperparameter sweep.

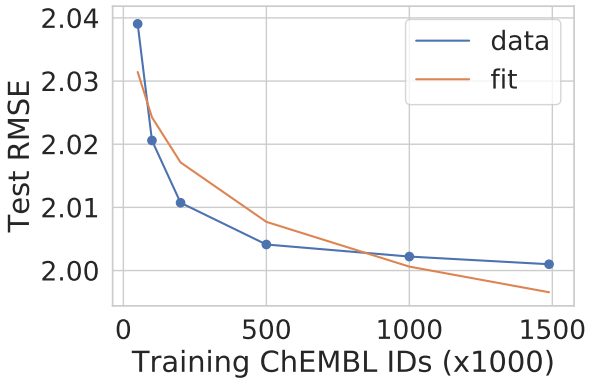
the mean training set predictor.

As we get to the other more complicated ML models, F.F., CNN, and Transformer, the relationship of how certain hyperparameters affect the resulting model's test set RMSE become more complex. It thus becomes unclear what specific combinations of hyperparameters will result in a collapsed fit. This further reinforces the idea that these more complex methods require a more extensive hyperparameter sweep in order to determine an optimal set of hyperparameters for a given task.

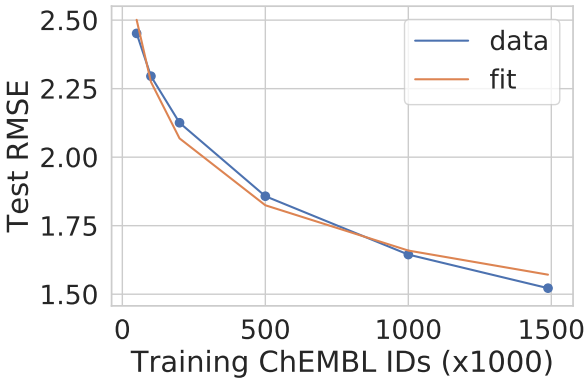
2.4.3 More complex models utilize training data more efficiently

The results shown in Figure 2.3 show comparisons of absolute performance between the various models. However, it is also important to consider how efficiently the models are utilizing the available training data. Not only does this give us an estimate of how much we can expect these models to improve, but it will also give insight into whether or not the models have reached diminishing returns and it would be better to focus research efforts elsewhere. In order to investigate this we fit power laws to the data utilized in Figure 2.3 for each predictive task (Figures 2.5-2.10).

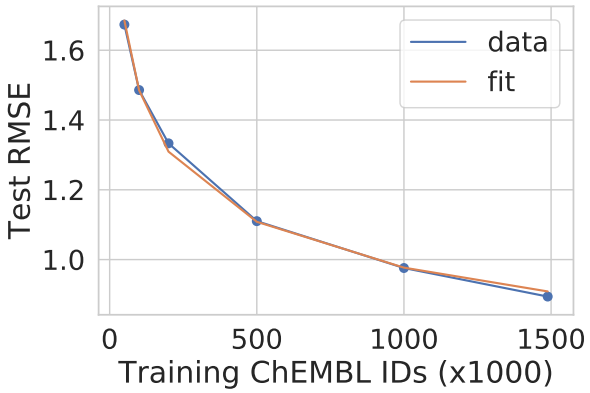
In general, a power law describes the relationship of training set size to test set RMSE



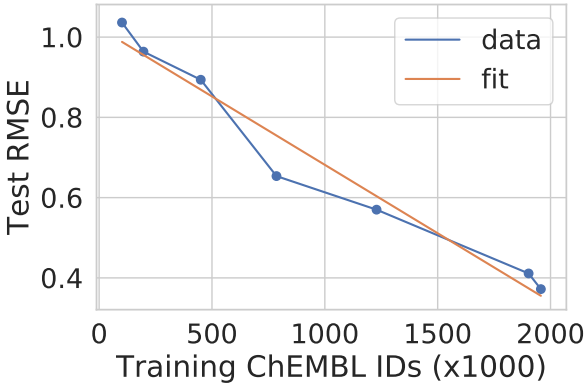
(a) SVR – Longest Path (R^2 0.864)



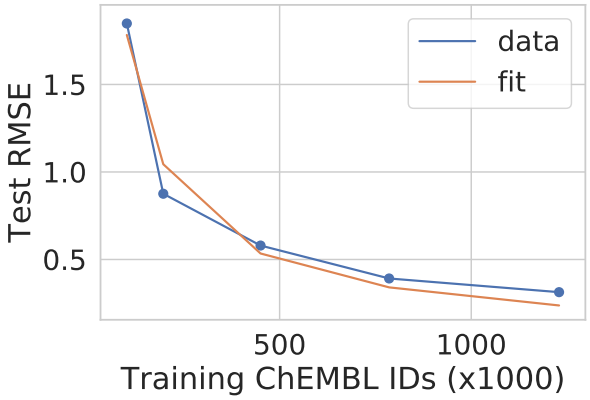
(b) Random Forest – Longest Path (R^2 0.985)



(c) Feed Forward – Longest Path (R^2 0.998)



(d) CNN linear fit - Longest Path (R^2 0.963)

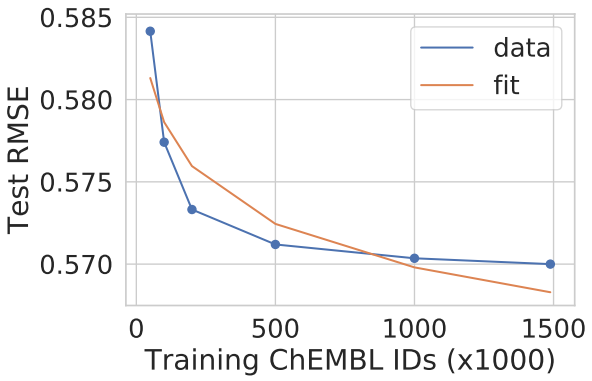


(e) Transformer – Longest Path (R^2 0.973)

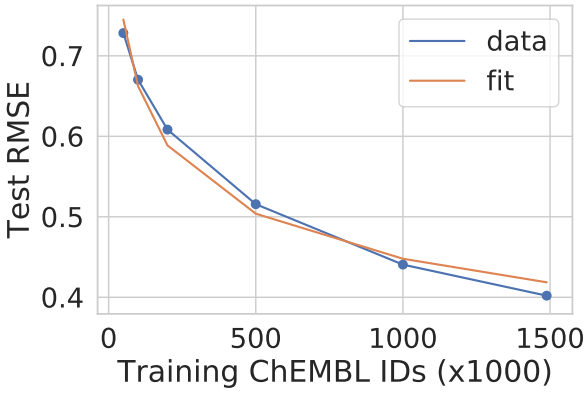
Model	R^2	Power Law Exponent
SVR	0.864	-0.00510
RF	0.985	-0.137
F.F.	0.998	-0.182
CNN	*0.963*	*-0.000341*
Transformer	0.973	-0.808

(f) Table summarizing the power law fit. Note: CNN is a linear fit, so the slope is reported indicated with *italics*.

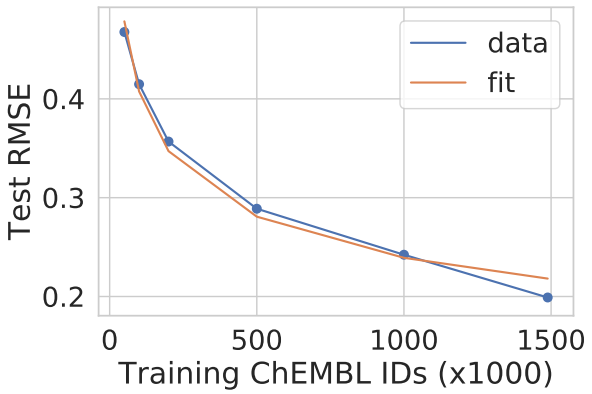
Figure 2.5: Fitting power laws mapping the training set size to model performance for the Longest Path prediction task. We only fit up the first 5 growing training sets for the Transformer model, as the computational demands were too high.



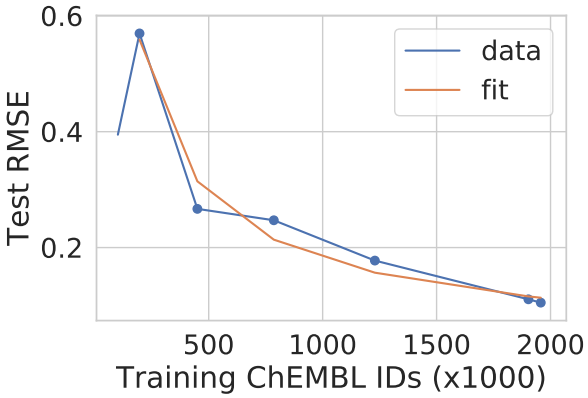
(a) SVR – Number of Rings (R^2 0.859)



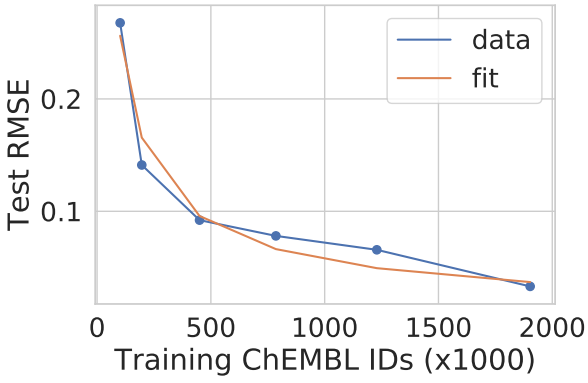
(b) Random Forest – Number of Rings (R^2 0.986)



(c) Feed Forward – Number of Rings (R^2 0.987)



(d) CNN – Number of Rings (R^2 0.973)

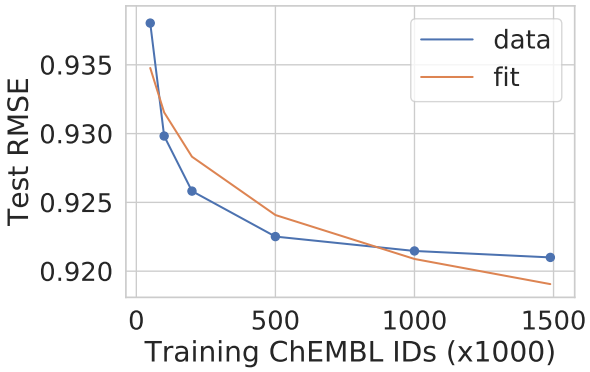


(e) Transformer – Number of Rings (R^2 0.967)

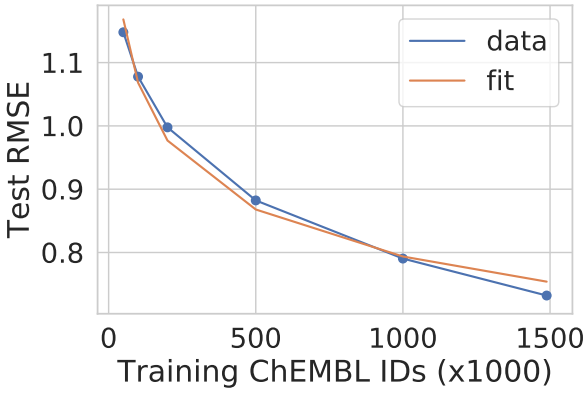
Model	R^2	Power Law Exponent
SVR	0.859	-0.00668
RF	0.986	-0.170
F.F.	0.987	-0.231
CNN	0.973	-0.693
Transformer	0.967	-0.660

(f) Table summarizing the power law fit

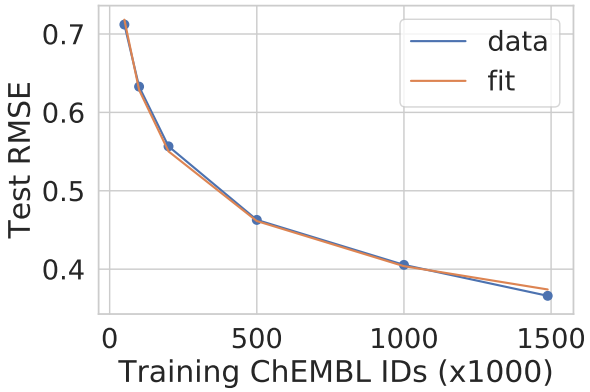
Figure 2.6: Fitting power laws mapping the training set size to model performance for the Number of Rings prediction task. We only fit up the first 6 growing training sets for the Transformer model, as the computational demands were too high. Additionally, we excluded the first data point of the CNN from the power law fit due to it being an outlier



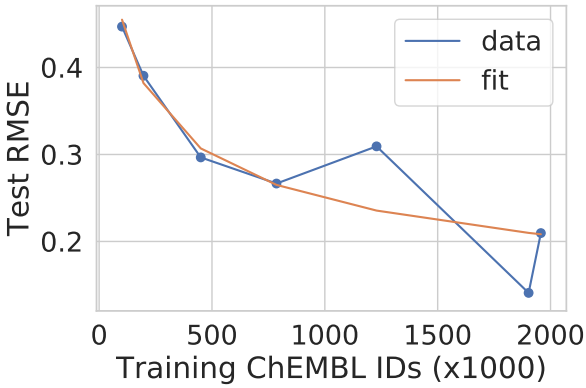
(a) SVR – cLogP (R^2 0.878)



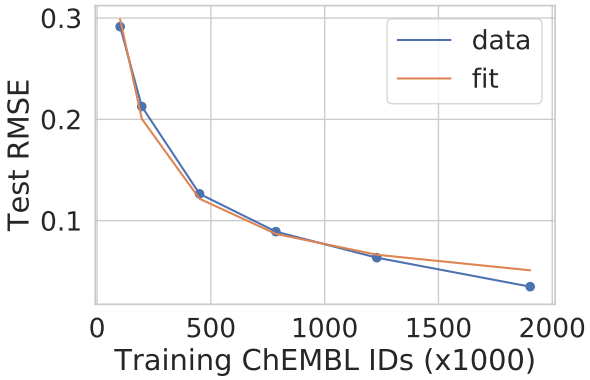
(b) Random Forest – cLogP (R^2 0.988)



(c) Feed Forward – cLogP (R^2 0.998)



(d) CNN – cLogP (R^2 0.837)

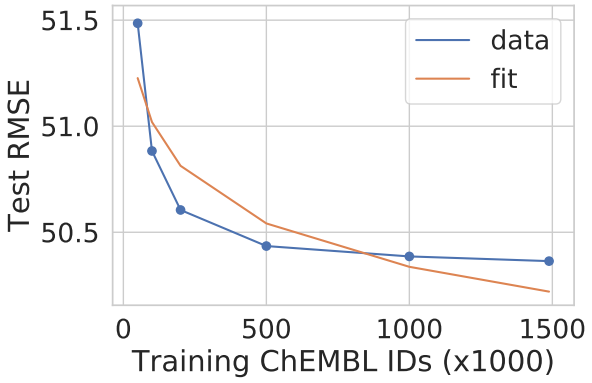


(e) Transformer – cLogP (R^2 0.990)

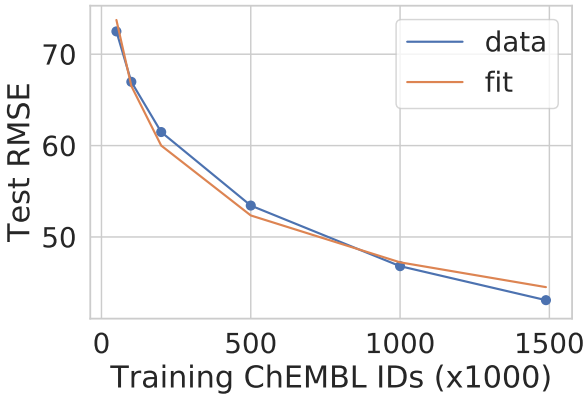
Model	R^2	Power Law Exponent
SVR	0.878	-0.00500
RF	0.988	-0.129
F.F.	0.998	-0.192
CNN	0.837	-0.264
Transformer	0.990	-0.603

(f) Table summarizing the power law fit

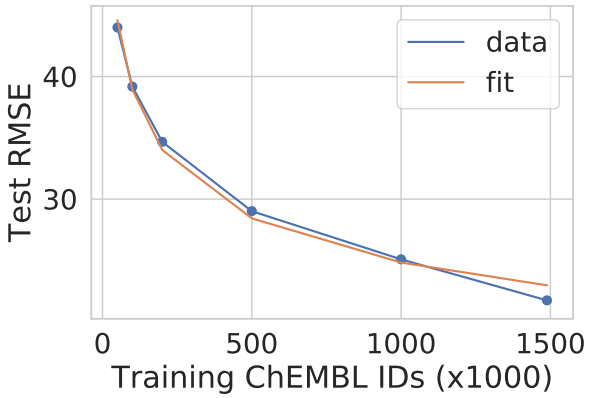
Figure 2.7: Fitting power laws mapping the training set size to model performance for the cLogP prediction task. We only fit up the first 6 growing training sets for the Transformer model, as the computational demands were too high.



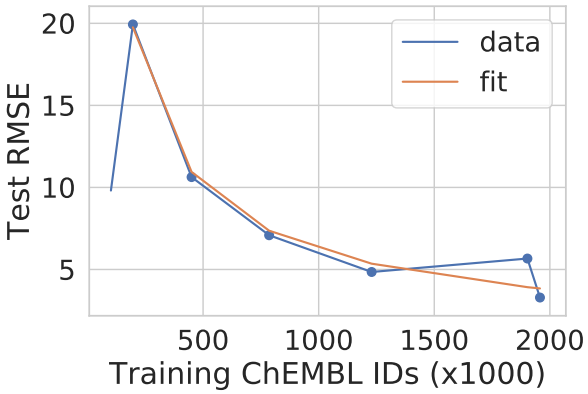
(a) SVR – Molecular Weight (R^2 0.827)



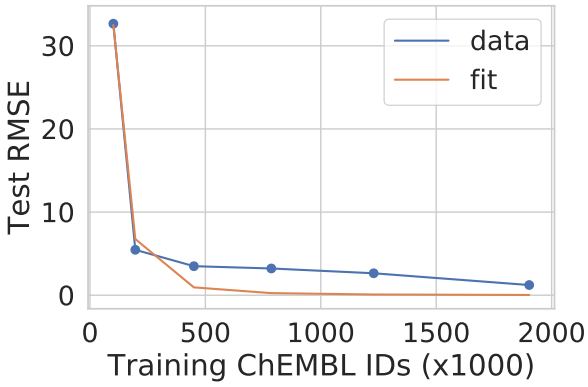
(b) Random Forest – Molecular Weight (R^2 0.989)



(c) Feed Forward – Molecular Weight (R^2 0.992)



(d) CNN – Molecular Weight (R^2 0.979)

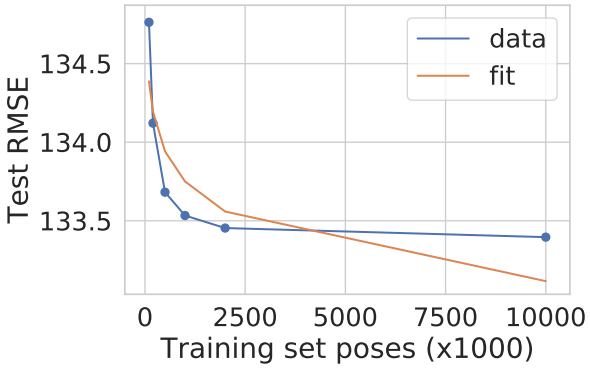


(e) Transformer – Molecular Weight (R^2 0.966)

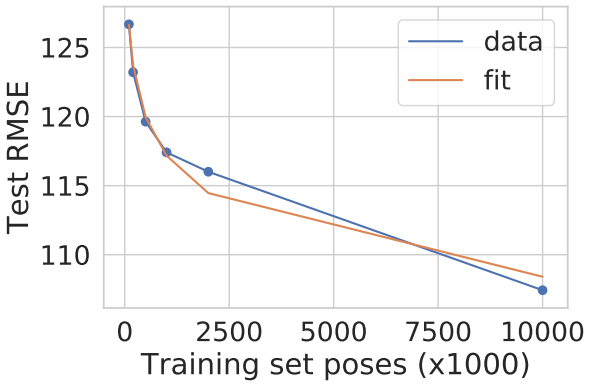
Model	R^2	Power Law Exponent
SVR	0.827	-0.00585
RF	0.989	-0.149
F.F.	0.992	-0.196
CNN	0.979	-0.712
Transformer	0.966	-2.37

(f) Table summarizing the power law fit

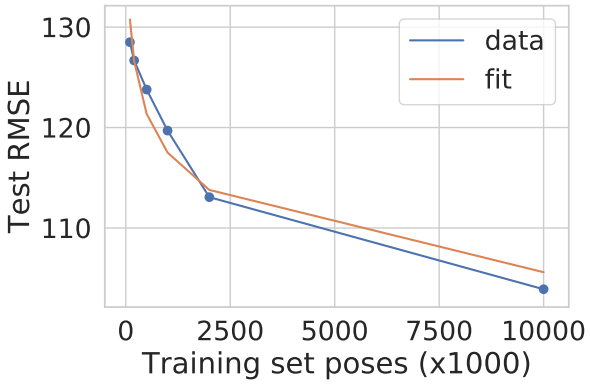
Figure 2.8: Fitting power laws mapping the training set size to model performance for the Molecular Weight prediction task. We only fit up the first 6 growing training sets for the Transformer model, as the computational demands were too high. Additionally we excluded the first data point from the CNN power law fit due to it being an outlier.



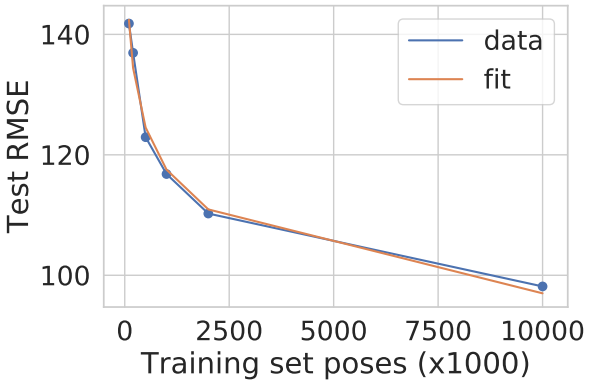
(a) SVR – UFF energy (R^2 0.749)



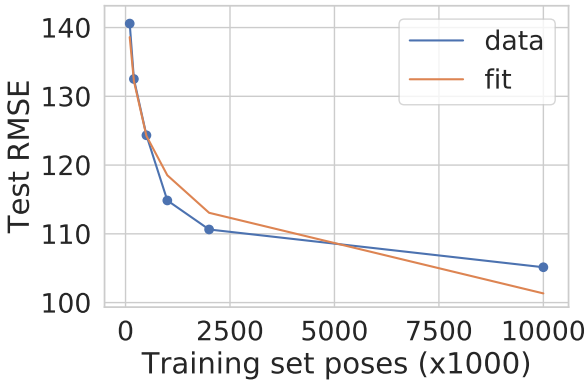
(b) Random Forest – UFF energy (R^2 0.983)



(c) Feed Forward – UFF energy (R^2 0.956)



(d) CNN – UFF energy (R^2 0.991)

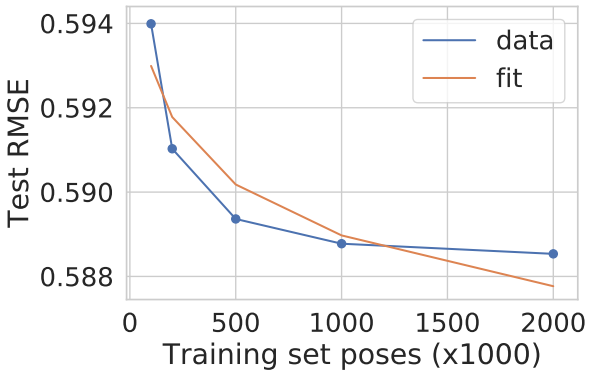


(e) Transformer – UFF energy (R^2 0.959)

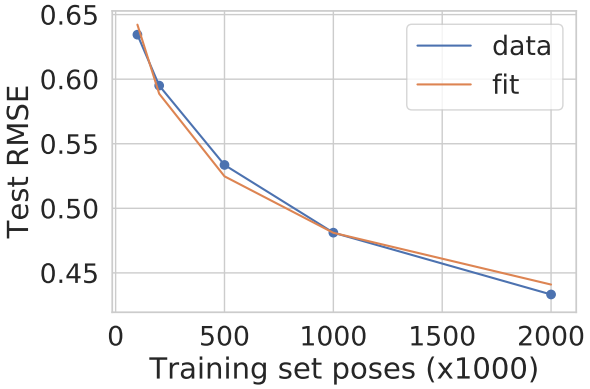
Model	R^2	Power Law Exponent
SVR	0.749	-0.00206
RF	0.983	-0.0338
F.F.	0.956	-0.0464
CNN	0.991	-0.0834
Transformer	0.959	-0.0680

(f) Table summarizing the power law fit

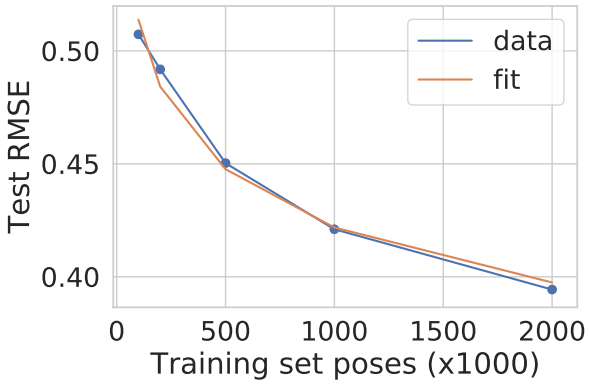
Figure 2.9: Fitting power laws mapping the training set size to model performance for the UFF energy prediction task. We only fit up the first 6 growing training sets for the Transformer model, as the computational demands were too high.



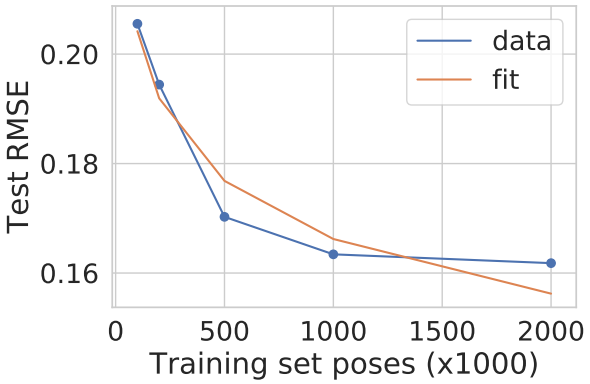
(a) SVR – Radius of Gyration (R^2 0.860)



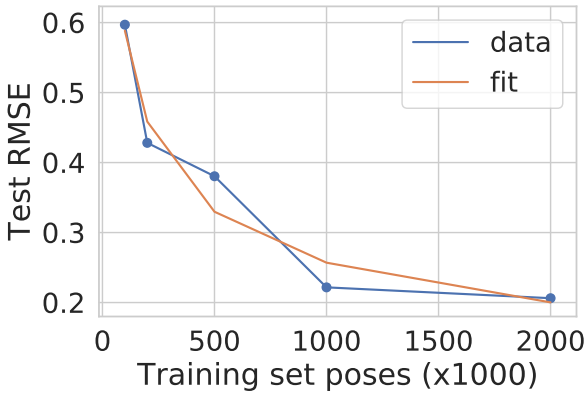
(b) Random Forest – Radius of Gyration (R^2 0.991)



(c) Feed Forward – Radius of Gyration (R^2 0.987)



(d) CNN – Radius of Gyration (R^2 0.942)



(e) Transformer – Radius of Gyration (R^2 0.953)

Model	R^2	Power Law Exponent
SVR	0.860	-0.00295
RF	0.991	-0.126
F.F.	0.987	-0.0857
CNN	0.942	-0.0894
Transformer	0.953	-0.360

(f) Table summarizing the power law fit

Figure 2.10: Fitting power laws mapping the training set size to model performance for the Radius of Gyration prediction task. We only fit up the first 5 growing training sets for the Transformer model, as the computational demands were too high, and similarly restricted our fit of the CNN to keep comparisons between the 3D methods more fair.

well ($R^2 > 0.9$). Notable exceptions are the CNN on the Longest Path prediction task, which was better described with a linear fit, the CNN model on the cLogP predictive task, and the SVR model on every task. It is clear from the plots (Figures 2.5-2.10(a)) that the SVR model's performance has diminishing returns, and that the power law is overestimating the test set RMSE improvements. Interestingly, for the CNN model's performance on cLogP the relatively poor R^2 value is driven by 2 data points, the 2nd and 3rd from the final point, that poorly fit the power law (Figure 2.7(d)). It is especially interesting that the 10 million pose training dataset for the CNN outperformed the full training dataset as this was the only time such behavior was observed in our experiment.

By comparing the exponents listed in Figures 2.5-2.10 (f) we can determine which models are utilizing the available training data most efficiently (e.g. more negative exponents mean a faster reduction in test RMSE). We observe that across all tasks, except the Radius of Gyration prediction, the F.F. model has a more negative exponent in the fit power law than the RF model. Thus not only were the F.F. models' performance better across these tasks (Figure 2.3 (a)-(e)), but we expect as more training data becomes available that the difference in improvement will widen. In the Radius of Gyration prediction task, the opposite is true. For this task, the RF model has a more negative exponent (-0.126) than the F.F. model (-0.0857). So, while the performance of the F.F. model is currently better than the RF model (0.354 vs 0.433 RMSE), we expect that the RF model will surpass the F.F. model with enough training data. Assuming that our power laws will hold as more training data becomes available, the RF model will surpass the F.F.at approximately 27.5 million available training poses.

This trend of more complex models utilizing data more efficiently than the less complex models breaks down for the 3D methods. The CNN has a more negative power law exponent for the Number of Rings and UFF energy predictive tasks, and exhibited a linear fit on the LongestPath predictive task. The Transformer on the other hand had a more negative power law fit on the cLogP, Molecular Weight, and Radius of Gyration predictive task. We also

note that the CNN model may be exhibiting signs of diminishing returns on the Radius of Gyration task as the power law is over-estimating the test set RMSE, and the performance of the CNN is flattening out (Figure 2.10c).

Model	2D task mean exponent	3D task mean exponent
SVR	-0.00566 ($6.7e^{-4}$)	-0.00251 ($4.4e^{-4}$)
RF	-0.146 (0.015)	-0.0794 (0.046)
F.F.	-0.200 (0.019)	-0.0660 (0.020)
CNN	-0.557 (0.21)*	-0.0864 (0.0030)
Transformer	-1.11 (0.73)	-0.214 (0.15)

Table 2.3: Mean power law exponent for each model across the various input tasks. The standard deviation of the fits is shown in parenthesis. There are 4 tasks in the 2D task, except for CNN for which the LongestPath task is excluded due to it being a linear fit (indicated with *). There are only 2 tasks in the 3D task.

We then looked at how consistent our power law fits for a given model were by comparing the fit exponent for a given model across the tasks.Table 2.3 shows the mean exponent of our power law fits per model, split between the 2D and 3D predictive tasks. We observe that across both the 2D tasks, the SVR, RF and F.F. models essentially have the same exponent for their power law fit as evidenced by having a standard deviation between the fits an order of magnitude smaller than the mean fit. This was also true of the SVR and CNN model on the 3D tasks. However, it was not the case for both the CNN and Transformer models on the 2D tasks, and not the case for the RF, F.F, and Transformer on the 3D task. Notably, there are only 2 3D tasks, so the standard deviation here is fairly unreliable. These findings indicate that for more complex models, while a power law fit can describe the relationship between training data size and model performance, the exact nature of this fit is task specific.

We show the datapoints utilized in Figures 2.9 and 2.10 in Table 2.4. We expected that the RF and F.F. models would both be poorly suited to the UFF energy and Radius of Gyration predictive tasks, as they utilize the 2D Morgan fingerprint as input and thus cannot utilizes the necessary 3D conformer information to predict the correct answer. For the Radius of Gyration task, this is exactly what plays out with the CNN being the best performer no

Train Size	RF	F.F.	CNN	Transformer
100k	**126.68**	128.50	141.80	140.58
200k	**123.21**	126.68	136.93	132.52
500k	**119.63**	123.78	*122.93*	124.33
1 mil	117.41	119.70	116.81	***114.85***
2 mil	116.01	113.07	**110.25**	110.63
10 mil	107.43	103.90	**98.15**	105.14

(a) UFF Energy Test set Error

Train Size	RF	F.F.	CNN	Transformer
100k	0.634	0.507	***0.206***	0.597
200k	0.595	0.492	**0.194**	*0.428*
500k	0.534	0.450	**0.170**	0.380
1 mil	0.481	0.421	**0.163**	0.222
2 mil	0.433	0.394	**0.162**	0.206

(b) Radius of Gyration Test Set Error

Table 2.4: Model Test RMSE comparisons for the 3D property prediction tasks. **Bold** indicates the best performing model in the row. *Italics* indicates when the CNN or Transformer surpasses the feed-forward neural network's performance.

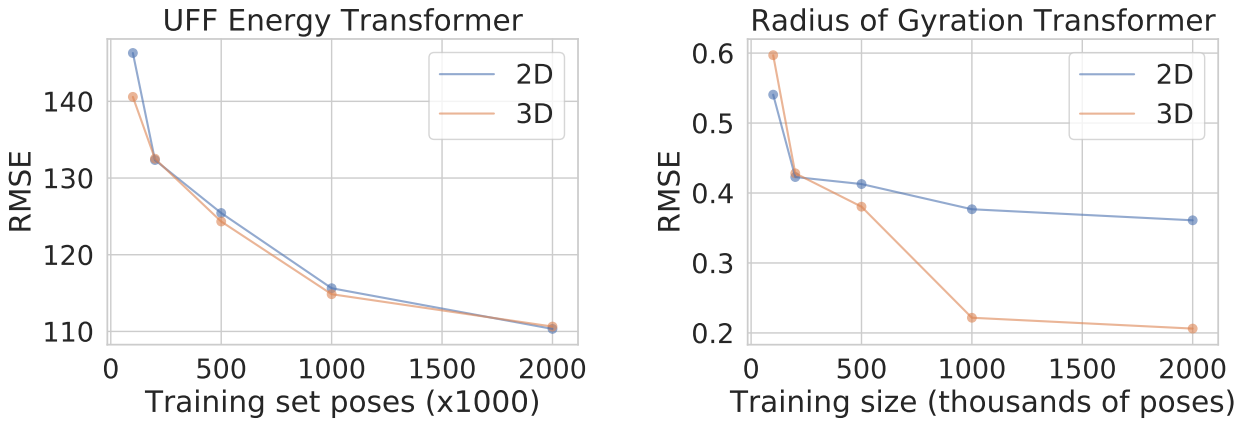
matter the amount of training data. The Transformer also quickly surpasses the fingerprint based models once 200,000 training poses are available. This is not necessarily surprising since the CNN is able to "see" the entire size of the molecule in three dimensions, making it well suited to the task.

Conversely, in the Transformer model the 3D information needed to calculate the Radius of Gyration (the 3D coordinates of each atom) are more hidden from the model. First, unless there exists an atom at the molecule's center of mass, the pairwise distance matrix does not explicitly contain the Radius of Gyration. So first the model would have to figure out the math (and the assumptions) to compute a set of coordinates that corresponds with the input distance matrix. Then, the model has to calculate the center of mass for that computed set of coordinates, followed by calculating the distance to each coordinate to locate the solution. There is an easy to compute approximation, being half of the largest value in the pairwise distance matrix, but it is prone to error if the molecule is not globular. Additionally, the only

place that the distance matrix enters the model is in each head of the self-attention layers of the transformer (Figure 2.2c). Within the code, the distance matrix is pushed through an exponential kernel ($g(d) = \exp(-d)$) during the self-attention block. Thus, the model does not have direct access to the distances within the distance matrix. Which in turn, makes the Transformer architecture, as implemented here, less suited to predicting the Radius of Gyration than the CNN.

However, this was not the case with the UFF energy task. The R.F. model was our best performing model until 1 million training poses were available. It also took 500,000 training poses before the CNN was able to overtake the F.F. model, and 1 million training poses before the Transformer was able to do the same. Additionally, even though the F.F. and RF models utilized the same input, it took 2 million training poses before the F.F. network outperformed the random forest. This is particularly interesting as since all of the models performed relatively poorly, it indicates that if the problem is hard or not suitable to our input representations. We can conclude that in such a case if there is not a lot of training data available, then the best approach is to utilize the random forest. But once a sufficiently large amount of data is available, then the more complex models start to outperform the random forest.

Lastly, we wish to examine how the 3D information is impacting our 3D predictive tasks. We cannot simply compare between the 2D and 3D models, since the differences between them can be due to both the ability to process 3D information as well as differences in the model type. Thus, it is better to compare between a 2D conformer and 3D conformer as input to our 3D methods. This must be done with care. For example, the CNN architecture requires 3D information to be present in the input. Simply flattening the conformer to 2 dimensions allows some data leakage of the conformer into the 2D representation as the locations of the atoms will be different. It also either changes the dimensionality of the feature maps learned by the CNN as you remove a coordinate dimension from the molecular grid, or introduces a lot of zero value grids due to the 3D grid being mostly empty with



(a) 2D versus 3D distance matrix input for Transformer models on UFFenergy.

(b) 2D versus 3D distance matrix inputs for Transformer models on Radius of Gyration.

Figure 2.11: Visualizing the difference of using a 2D instead of 3D pairwise distance matrix as input to the transformer models. We expected that the 3D models would outperform the 2D counterparts as there is extra information available to the same number of model parameters. This is the case for the Radius of Gyration, but not the case for the UFF energy.

everything squashed into 2 dimensions. The Transformer model does not suffer from this problem. 3D information is incorporated into the graph structure through the pairwise distance matrix. Said matrix can be generated utilizing the positions of the 3D conformer, or through calling the Compute2DCoords() function in rdkit on 2D molecular graph itself. Notably, the dimensionality of the pairwise distance matrix is always $N_{atoms} \times N_{atoms}$. Thus, the data leakage can be avoided by having each conformer have the same 2D pairwise distance matrix, and thus, the same input into the neural network.

In Figure 2.11 we show the results of utilizing a 2D or 3D pairwise distance matrix as input to our Transformer model on the UFF energy and Radius of Gyration predictive tasks. The results on the Radius of Gyration task were generally what was expected; that the 3D version of the input was better than the 2D version due extra information on the 3D conformer being present in the input. Notably, this effect only starts to appear after 500,000 input poses are included in the training set. This was not the case for the UFF energy predictive task. Instead, we observe that as the training set size grows, the two versions of

the Transformer model collapse to the same performance. This indicates that the conformer information is not being utilized in the predictive performance on this task.

2.5 Conclusion

In order to investigate the relationship between input representation and performance of various ML models, we trained several models on various computed molecular property prediction tasks. We observe that the more complex models empirically outperform the simpler models (Figure 2.3). We compared our best performing model across tasks by fitting power laws to describe the relationship between training set size and the resulting test set error. In general, these fits explained the relationship very well (commonly $R^2 > 0.95$) and allows for relatively easy detection of diminishing returns if the power law is overestimating the model's test set performance (e.g. the SVR model on every task). We also observed remarkable consistency between the fit exponents of the power law across tasks for our simpler models, the SVR, RF, and F.F., except for the RF model on the 3D tasks (Table 2.3). This implies that our simpler models utilizing molecular fingerprints fit training data in a similar fashion regardless of the task. As this behavior was consistent across 3 model types, it suggests that this property could be due to the input Morgan fingerprint representation rather than some property of a model type. This is further suggested by the CNN and Transformer models, which both utilize different input representations, having generally different power law exponent fits across the tasks.

We also demonstrate that more complex models tend to utilize data more efficiently, as the more complex models tend to have more negative exponents in their power law fit (Figures 2.5-2.10f, Table 2.3). In the 2D tasks, it is clear that the SVR handles new data worse than the RF, which in turn is worse than the F.F. models as expected. Things become less clear with the CNN, due to the large standard deviation. However, this is due to both

the CNN having only 3 tasks to fit, since the Longest Path was better explained by a linear fit, and relative outlier fit exponent of -0.264 on the cLogP task (Figure 2.7f) as compared to -0.712 on Molecular Weight (Figure 2.8f) and -0.693 on Number of Rings (Figure 2.6f). We also note that this worst fit exponent for the CNN (-0.264 for the cLogP task) is higher than any observed exponent for the F.F. model. So, while our fit exponents are much noisier for the CNN on the 2D predictive tasks, they are all higher than the F.F. model and thus can support the claim that more complex models utilize the data more efficiently. There is a similar story with the Transformer model, where an outlier exponent fit of -2.37 on the Molecular Weight predictive task (Figure 2.8f) is much higher than the other exponent fits of -0.808, -0.660, and -0.603 on the LongestPath, Number of Rings, and cLogP tasks respectively (Figure 2.5f, 2.6f, 2.7f). Again, the smallest exponent fit for the Transformer model, -0.603 on cLogP (Figure 2.7f), is higher than any observed in the F.F. model. So again, we can conclude that this more complicated model utilizes data more efficiently than a F.F. model. However, due to the large standard deviations in the fit exponents, we cannot make conclusions about the ability to utilize the training data between the CNN and Transformer model for the 2D predictive tasks. For the 3D tasks, this relationship is harder to tease as there are only 2 tasks and the CNN is the only model that achieved consistent results between the two tasks.

An interesting exception to the power law fit is the CNN on the Longest Path prediction task, where a linear model had a much better fit to the data (linear R^2 of 0.963 versus the power law's R^2 of 0.876). This indicates that this particular input representation and model architecture is well suited to the task, as we are seeing consistent linear improvements. It also indicates that we have not started to reach the performance limit of the model, where diminishing returns of additional training data would start to form. This is an interesting contrast to the Transformer model, which does follow a power-law relationship. The Transformer exhibits better performance than the CNN on the Longest Path prediction task. The Transformer had a test set RMSE of 0.313 with 1.23 training million molecules versus 0.372 RMSE with 1.96 million training molecules for the CNN. However, we expect

that the CNN will surpass the transformer at ≈2.6 million total training molecules, assuming that both the linear fit for the CNN and the power law for the Transformer continue to hold.

One of our initial hypotheses was that the extra 3D input for the UFF energy predictive task would allow networks with 3D input to outperform their 2D counterparts. That is, we expected that the CNN and Transformer architectures would outperform the RF and F.F. networks, which was not necessarily the case (Table 2.4a). Notably, it took 500,000 training poses for the CNN to outperform the F.F. network, and 1 million poses before the 3D methods outperformed the 2D ones. It is also important to note that the RMSE of over 100 kcal/mol is a poor performance to recapitulate the energy defined by the UFF for a given conformer. To investigate this further, we utilized the Transformer model as it is easy to change the pairwise distance matrix to utilize 2D distances based on the molecular graph instead of 3D distances of the conformer without modifying other parts of the network architecture (Figure 2.11). We expected to see that the 3D method would always outperform the 2D method as it has an additional source of information about the 3D conformer. However, we observe that as the training size grows the two versions of the model achieve the same performance. We speculate that this is due to the large variance in the UFF energy of the dataset (Figure 2.1e). The variance of UFF energies between molecules is much greater than the variance between conformers of the same molecule. Ergo, in a learning context, a quick way to reduce the overall error of a model would be to distinguish between molecules rather than learn the actual force field parameters to distinguish between conformers of the same molecule. Thus, we suspect that the models are learning to map molecular identity to the UFF energy rather than the actual parameters of the force field to determine the energy from the positions of the atoms in a conformer.

The difficulty of our models on the UFF energy predictive task can better mimic the challenging property prediction use cases (such as receptor-ligand binding affinity) for these models. Table 2.4 shows that when the available training data is limited, a random forest based on molecular fingerprints is likely the best modeling option. It takes over a million

training poses to be available before the CNN or Transformer architectures can outperform the random forest. Additionally when training with 10 million poses the performance difference between the RF, F.F., CNN, and Transformer models is not very large. This could help explain the phenomenon observed with the PDBbind benchmark[6], where a large variety of different architectures and input representations achieve similar performance on the core set. With the small amount of available data, only 19,443 structures, there is simply not enough data present to meaningfully outperform simpler approaches on the PDBbind. Thus, it is imperative that training data expansion methods be developed to help the advancement of ML models in these structure-based molecular property predictive tasks.

```
2d fingerprints -- rdkitFP 2048, Morgan radius 2 & 2048, rdkit Feature vector
    Ridge Regression
         Initial search: grid (n=240)
             --alpha = 1, 100, 10000, 1000000, 100000000  Constant that multiplies the L2 term
             --fit_intercept = 0,1                        Bool on fitting the intercept
             --max_iter = 1000,10000,15000,20000          Maximum iterations for conjugate gradient solver.
             --positive = 0,1                             Bool to force coefficients to be positive
             --solver = auto                              Automatically choose the solver method
             --training = rdkitFP, Morgan, rdkit feature vector    Input representation selection
         Second search: bayes (n=100)
             --alpha: uniform [0,100000]
             --fit_intercept = 1
             --max_iter = 20000
             --positive = 0
             --solver = auto
    Linear SVR
        Initial search: grid (n=675)
             --c = 100, 10, 1, 0.1, 0.01               Regularization parameter. Strength is inversely proportional
             --epsilon = 0, 0.1, 1, 10, 1000           Tolerance for stopping criteria
             --intercept_scaling = 1, 0.1, 10          enable intercept_scaling*synthetic feature weight.
             --max_iter = 1000, 10000, 20000           Limit on iterations within solver
             --loss = squared_epsilon_insensitive      Use the L2 loss function
             --training = rdkitFP, Morgan, Feature    Input representation selection
        Second search: bayes (n=100) Num Rings, MolWT, cLogP, LongestPath Morgan
             --c = 2
             --epsilon: uniform [0,0.1]
             --intercept_scaling = 1
             --loss = squared_epsilon_insensitive
             --max_iter = 30000
        Second search LongestPath rdkitFP: bayes (n=100)
             --c: uniform [1,100]
             --epsilon: uniform [0,10]
             --intercept_scaling: uniform [0,10]
             --loss squared_epsilon_insensitive
             --max_iter 20000
        Third search LongestPath rdkitFP: bayes (n=100)
             --c = 2
             --epsilon: uniform [0,0.1]
             --intercept_scaling = 1
             --loss = squared_epsilon_insensitive
             --max_iter: int_uniform [20000,30000]
    Random Forest
         Initial search: random (n=1000)
             --ccp_alpha = 0, 0.25,0.5,0.75                Complexity parameter for Minimal Cost-Complexity Pruning
             --criterion = squared_error, poisson          Function to measure quality of a split
             --max_depth = None, 100, 1000, 10000          Maximum depth of the tree
             --max_features = auto, sqrt, log2             Number of features to consider when looking for the best split
             --max_samples = 1, 0.75, 0.5, 0.25            Fraction of samples to draw from training set for each tree.
             --min_imputity_decrease = 0, 0.25, 0.5,0.75   Impurity decrease threshold for splitting nodes
             --min_samples_leaf = 1,2,5,10,20              Minimum number of samples required to be in a leaf
             --min_samples_split = 2,5,10,100              Minimum number of samples to split an internal node
             --n_estimators = 50, 100, 1000                Number of trees in the forest
             --training = rdkitFP, Morgan, Feature        Input representation selection
         Second search: (bayes, n=100)
             --ccp_alpha = 0
             --criterion = squared_error
             --max depth: int_uniform [1000,20000]
             --max features = auto
             --min impurity decrease = 0
             --min samples leaf = 1
             --min samples split = 2
             --n_estimators: int_uniform [50,5000]
```

(a) Hyperparameter sweeps for scikit learn models utilizing fingerprints as input

```
2d fingerprints -- rdkitFP 2048, Morgan radius 2 & 2048, rdkit Feature vector
    Feed Forward Neural Netowork
        Wide-Shallow Initial search: random (n=1000)
            --activation = lrelu, sigmoid, tanh, relu    Nonlinear activation function
            --epochs = 1,2,10,20                         Number of training epochs
            --hiddensize = 50, 100, 1000, 10000          Dimension of layers of the model
            --lr = 0.0001, 0.001, 0.00001, 0.1           Learning rate for Adam Optimizer
            --n_hidden = 0,1                             Number of hidden layers
            --weight_decay = 0, 0.01, 0.1, 0.2           Weight decay for Adam Optimizer
            --training = rdkitFP, Morgan, Feature      Input representation selection
        Wide-Shallow Num Rings rdkitFV & Feature & Morgan Second search: bayes (n=100)
            --activation = lrelu
            --epochs: int_uniform [10,30]
            --hidden_size: int_uniform [8000, 20000]
            --lr: uniform [0.0001, 0.001]
            --n_hidden: int_uniform [0,5]
            --weight_decay = 0
        Wide-Shallow LongestPath Morgan Second search: bayes (n=100)
            --activation = lrelu, tanh
            --epochs: int_uniform [10,40]
            --hidden_size: int_uniform [1000,12000]
            --lr: uniform [0.00001, 0.001]
            --n_hidden: 0,1
            --weight_decay = 0
        Wide-Shallow LongestPath rdkitFV Second search: bayes (n=100)
            --activation = sigmoid, tanh
            --epochs: int_uniform [1,40]
            --hidden_size: int_uniform [8,256]
            --lr: uniform [0.00001, 0.001]
            --n_hidden: 0,1
            --weight_decay: uniform [0,0.01]
        Narrow-Deep Initial search: random (n=1000)
            --activation = lrelu, sigmoid, tanh, relu
            --epochs = 1,2,10,20
            --hiddensize: 50, 100, 200
            --lr = 0.0001, 0.001, 0.00001, 0.1
            --n_hidden = 5, 10, 15, 20
            --weight_decay = 0, 0.01, 0.1, 0.2
            --training = rdkitFP, Morgan, Feature
        Narrow-Deep Num Rings rdkitFV & Morgan Second search: bayes (n=100)
            --activation = lrelu
            --epochs: int_uniform [10,30]
            --hidden_size: int_uniform [200, 500]
            --lr: uniform [0.0001,0.001]
            --n_hidden: int_uniform [10,30]
            --weight_decay = 0
        Narrow-Deep LongestPath Morgan & rdkitFV Second search: bayes (n=100)
            --activation = lrelu
            --epochs: int_uniform [1,40]
            --hidden_size: int_uniform [32,512]
            --lr: uniform [0.00001, 0.001]
            --n_hidden: int_uniform [20,100]
            --weight_decay: uniform [0.05,0.4]
```

(b) Hyperparameter sweeps for feed forward neural networks utilizing fingerprints as input

```
3D grid based CNN Initial search: random (n=1000)
     --atommap = basic, C, Caromatic, CN, CNO, CNOH, H   Atom Type definitions for libmolgrid
     --batchsize = 64,128,256                            Number of examples per batch
     --lr  = 0.1,0.01,0.001,0.0001                       learning rate for Adam optimizer
     --initial filters = 32,64,128                       Number of output channels in first convolution
     --n_conv = 2,3,4                                    Number of convolutions in network
     --weight_decay = 0, 0.01, 0.1, 0.2                  Weight decay for Adam optimizer
     --activation = lrelu,sigmoid,elu,relu               Non-linear activation function following each convolution
Graph transformer Initial search: random (n=1000)
     --Ndense = 1,2,3,5                                   Number of position-wise feed forward layers
     --batch size = 64,128,256                            Number of examples per batch
     --delta = 0.1,0.2,0.5,1,2,10                         Delta for the Huber loss function
     --dmodel = 64,128,256,512,1024                       Hidden dimension size of the model
     --dropout = 0,0.01,0.1,0.2                           Probability to zero elements in an input tensor
     --heads = 2,4,8,16,32                                Number of self attention heads
     --lattn = 0.1,0.2,0.33,0.5                           Weight of the self-attention matrix
     --ldist = 0.1,0.2,0.33,0.5                           Weight of the distance matrix
     --nstacklayers = 2,4,6,8,10,16                       Number of attention blocks
     --lr = 0.1,0.01,0.001,0.0001                         Learning rate for SGD optimizer
     --momentum = 0.6,0.7,0.8,0.9                         Momentum for SGD optimizer
```

(c) Hyperparameter sweeps for convolutional neural networks utilizing grids of atomic density as input, and molecule attention transformer networks utilizing molecular graphs as input

Figure 2.12: Hyperparameters explored during the search.

3.0 Expanding Training Data

3.1 Summary

Machine learning methods have become increasingly popular for protein-ligand scoring. In particular, several new structure-based methods all achieve similar performance on the PDBbind dataset[6]. Machine learning methods' success in other fields is due in part to a large volume of available training data, which is not the case for receptor-ligand structures. There are two vectors to approach expanding the available structural data for machine learning models: expanding the available binding pose data, and expanding the available binding affinity data. We developed the CrossDocked2020 dataset which expands the available binding pose data through docking ligands into similar receptors. This results in the expansion of the 200,000 poses available in the PDBbind General set into over 22.5 million poses in CrossDocked2020. We then demonstrate that training convolutional neural networks on CrossDocked2020 results in models that perform better on the redocking and crossdocking tasks, are more pose sensitive, and yield more informative gradients. Lastly, we demonstrate that utilizing imputed labels generated from these networks on the missing binding affinity data in CrossDocked2020 results in improved performance for both pose classification and binding affinity regression via simple imputation techniques. The release of CrossDocked2020 and the data splits utilized to train our model allows for the direct comparison of other models to the ones generated in this study. This study also provides the proof of concept for further investigating imputation as an *in silica* method to improve binding affinity regression for structure-based models.

3.2 Introduction

A key component in the drug discovery pipeline is protein-ligand scoring. It provides a method to narrow the scope of all of chemical space down into a more reasonably sized set of compounds for experimental testing. Given that the interactions of atoms in space determine the properties of a given protein's interactions with a ligand, it is common to utilize a structure-based method to score these molecules.[73–77] In these structure-based methods, the scoring function is responsible for evaluating the correctness of the pose and predicting the affinity of a given complex. Traditionally, scoring functions fall into one of three categories: force-field based[13–16], empirical[17,18], or knowledge-based[19,20].

Force-field based methods model the intermolecular potential energies through bonded and nonbonded parameters estimated from experimental and simulated data.[78] Empirical scoring functions, in contrast, are constructed from manually selected interaction terms (e.g. hydrophobicity, hydrogen bonding, etc.) parameterized to available data. Lastly, knowledge-based methods are constructed from entirely non-physical statistical potentials derived from available protein-ligand complexes. Each of these approaches commonly utilize a linear fit of their input features to the target prediction. Machine learning (ML) models have, relatively recently, emerged as their own class of scoring function, and are particularly attractive as they fit a non-linear function of their input to the target prediction[21,79–83].

ML approaches to scoring require an input representation of the complex, which is often calculated using a predefined set of features to characterize the protein-ligand binding. This overt featurization possibly limits the performance of ML scoring functions by imbuing them with extra sources of bias from the human selected features. This limitation can be avoided by using a direct representation of the protein-ligand structure as input. One such representation is a 3D grid where the only features are the choice of atom types and how atom occupancy is represented in the grid. There have been several recent efforts at utilizing convolutional neural networks (CNNs) on these atomic grids for scoring receptor-ligand

complexes.[25,68,84–86] CNNs are particularly attractive as they allow the model to determine its own representations/features in order to determine what makes a low RMSD pose or strong binder for a given receptor-ligand complex. In addition, there have also been considerable advancements in utilizing graph-based representations with other ML architectures to predict receptor-ligand binding affinity.[87,88]

It is also important to note that the available structural data for receptor-ligand binding is inherently biased and does not span all of chemical space. Importantly, all of the available data is the result of specific human design choices, e.g. drug campaigns for a specific receptor.[89] Cleves and Jain[89] demonstrated that there are different inductive biases present for ligand-based modeling methods depending on if the method was 2D or 3D. There are three common biases present in virtual screening datasets: 'analogue bias' (highly similar active compounds), 'artificial enrichment' (poor property matching between actives and decoys leading to easier classification), and 'false negatives' (assumed decoys that were later experimentally verified to be active).[90] The recent success of ML methods has renewed interest in controlling for biases in the available datasets.[36,61–64] This is especially relevant as ML-based methods tend to fit to the initial biases of their training data.[61] For example, the DUD-E benchmark[91] for virtual screening has been shown to have numerous biases present in it, with ligand-only models able to achieve comparable performances to CNNs despite their lack of receptor information.[36]

ML models for pose selection and affinity prediction largely rely on the PDBbind dataset[6] which curates the Protein Data Bank (PDB) for high quality receptor-ligand structures with published binding affinities. Unfortunately this dataset is small by ML standards, containing a total of 19,443 protein-ligand entries in version 2020. This is a far cry from the scale of data utilized for large deep learning models. As an example, AlphaFold is a 93 million parameter model to predict a protein's 3D structure from its sequence, and was trained on all 204,104 structures available in the PDB and 355,993 unlabeled sequences from Uniclust30[92]. Part of the success of AlphaFold is in the supplementation of the structures available in the PDB with the unlabeled sequence data. Thus, it is desirable for us to similarly expand the PDBbind

based data to become more on scale with the datasets used for AlphaFold. There are two approaches to scaling the PDBbind data: we can expand the poses of every receptor-ligand complex to generate more pose data, and we can expand the binding affinity data. *In silica* methods to perform these expansions are particularly attractive as it is time-consuming, expensive, and difficult difficult to generate both a crystal structure and binding affinity measurement for a given receptor-ligand pair.

It is theoretically trivial to expand the number of poses for a given receptor-ligand complex through molecular docking. By simply sampling more and more poses, it is theoretically possible to generate an infinite amount of training data. However, it is unclear if generating data in this fashion would be useful to the model as most poses generated in this fashion are of poor quality. Ultimately, the goal of receptor-ligand scoring is not to recapitulate the binding pose of a known receptor-ligand complex (redocking), but to predict the poses of novel ligands in a given structure (crossdocking).

We can neatly address this problem by utilizing crossdocking itself to expand the binding pose data. The assumption that similar receptors bind similar ligands is common in virtual screening tasks. This same assumption would allow us to group the PDB into clusters of similar receptors and then expand the poses by docking every ligand in the cluster to every other receptor in the cluster. This both approximates the crossdocking use case better by generating poses of ligands in non-cognate receptors, but also serves to combinatorially expand the number of poses within each cluster.

A shortcoming of this approach is that it does not expand the amount of binding affinity data present in the dataset. In particular, the PDB contains a large number of complexes with unknown binding affinity, but known structure. It is non-trivial to incorporate outside sources of binding information into a structure-based dataset. A simple approach is to take the binding affinity label from a receptor-ligand complex with an unknown structure. However, it has been shown that training ML models on such data by using the top ranked docked pose of such a complex results in entirely pose-insensitive models, defeating the point

of using structure-based modeling at all.[68] Template-based docking docking could provide better results for such a setup, but it is unclear if it be successful.[93]

A different *in silica* solution for this problem is to assign a binding affinity value to known receptor-ligand complexes in the PDB through imputation.[94,95]. There are several approaches to imputing missing data, the simplest of which is to delete or ignore data points with missing labels (i.e. not perform imputation). This approach is problematic as it can introduce extra bias into the models, especially if the missing data is not randomly distributed.[96] We know that this is a large fraction of our available data, so this approach is unsuitable.

The next easiest method is known as "simple imputation" and entails replacing the missing values by a single quantifiable attribute of the non-missing values (e.g. mean, median, mode). However, it is known that these methods produce extra bias or unrealistic results on high-dimensional data sets.[97] This is also unsuitable for our task as we define our labels as a function of our high-dimensional data input (atom positions in 3D space). Thus, regression imputation is the most attractive option to explore.

In this method of imputation a model is fit to the known data labels and is then used to assign the imputed labels. There are several approaches to this type of imputation, from the statistical, such as a weighted quantile regression, to more ML inspired approaches like k-nearest neighbors, support vector machines, random forests, etc.[95,96] ML-based imputation approaches have been successful across the medical field, showing success in Medical Expenditure Panel Surveys[98], or being utilized in clinical decision making.[99]. Rubinsteyn et al.[100] examined a variety of imputation methods for imputing binding affinities for peptide-Major Histocompatibility Complex (MHC) interactions. The methods examined were not ML-based and only predicted a singular class of binding affinity interactions (e.g. only against MHC as a receptor).

The most common ML models employed in imputation are k-nearest neighbor (k-NN) models and random forest models.[95] Nearest neighbor algorithms require a meaningful similarity metric. This is challenging for molecules, as it has been shown that fingerprint-based

similarity metrics can incorrectly measure the similarity between molecules.[101] Additionally, for receptor-ligand binding affinity regression, a ligand similarity is insufficient as the interactions between the receptor and ligand are important to correct modeling. It is unclear what similarity metric could capture each of these unique features, making these ML models unsuitable to our task. Instead, we propose utilizing our CNN models directly to impute the missing labels.

This work is broken into two halves: first generating an expanded pose dataset through crossdocking and, second, expanding the binding affinity data through the utilization of our CNN models to impute the missing values. We extend the CNN developed by Ragoza et al.[68] to jointly train for pose classification, i.e. classifying if a given ligand pose has a low root mean squared deviation (RMSD) to the true crystal pose, and binding affinity regression. This CNN is the rigorously benchmarked on the traditional PDBbind dataset in order to determine its general efficacy at these two tasks. Then, we can utilize this CNN in order to give an initial benchmark for the CrossDocked2020 dataset and determine if expanding the available poses through crossdocking is beneficial to model performance. We can then utilize this CNN as our ML model to perform regression imputation on the receptor-ligand complexes in CrossDocked2020 which are missing binding affinity labels. The evaluation of a newly trained CNN on the imputed labels will then inform if this imputation approach to generating missing binding affinity values is effective at increasing model performance.

3.3 Expanding Pose Data

Our first approach to expanding the pose data is a combinatorial expansion through the use of crossdocking. We assume that similar ligands will bind to similar receptors, which allows us to generate new training complexes by docking ligands into non-cognate receptors. In this subaim we create a new dataset, CrossDocked2020, and rigorously evaluate the performance

Model	Parameters	Forward ± SD (ms)	Backward ± SD (ms)
Def2017	383,616	1.110 ± 0.0259	1.151 ± 0.0286
Def2018	388,736	1.147 ± 0.0334	1.369 ± 0.0363
HiRes Affinity	1,106,560	10.375 ± 0.181	20.640 ± 0.360
HiRes Pose	964,224	5.452 ± 0.0597	8.381 ± 0.918
Dense	684,640	8.116 ± 1.550	15.712 ± 0.180

Table 3.1: Number of parameters and time for a forward pass and backwards pass on a NVIDIA TITAN Xp for each model. The reported time is the average time per a single input complex averaged across 10 runs where each run consisted of 1000 iterations of batch size 50.

of our CNN's on this dataset to provide a solid benchmark for the community to fairly compare their new model's performance against.

3.3.1 Model Architectures and Input Representations

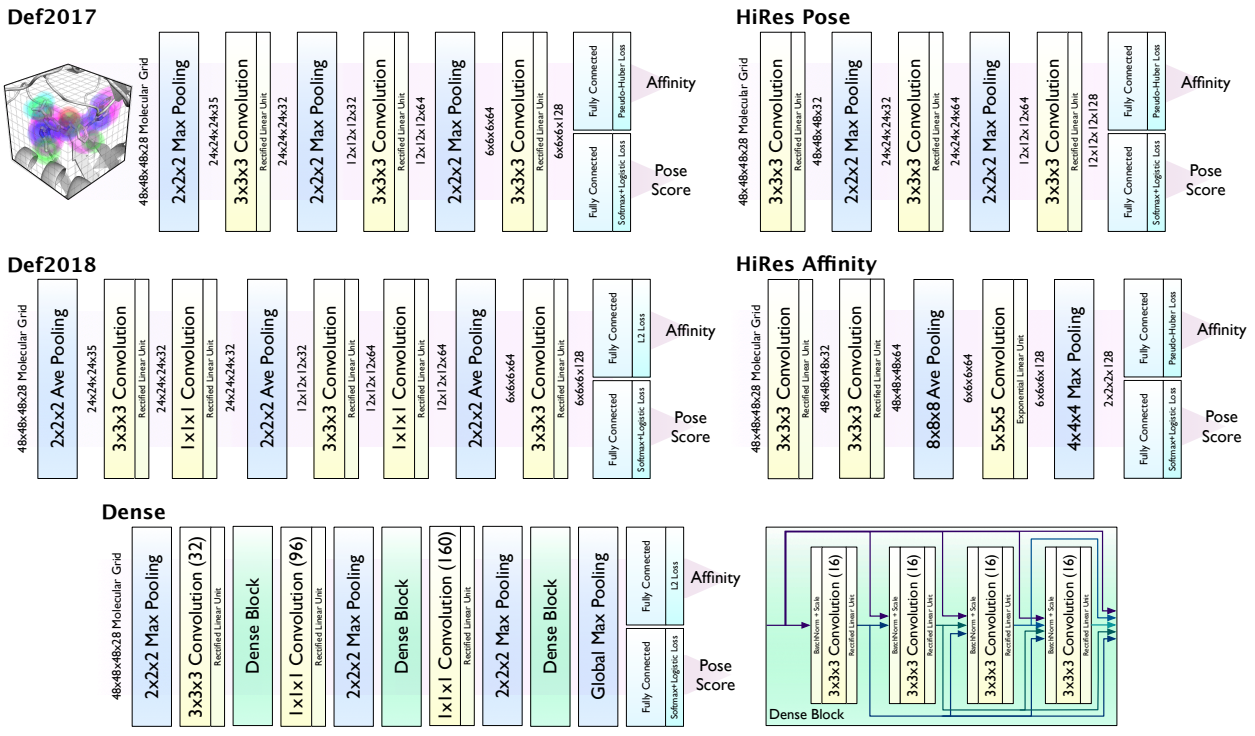


Figure 3.1: CNN model architectures. Code is available at `http://github.com/gnina`.

In this subaim, we evaluate five distinct CNN model architecture variations shown in

Figure 3.1. The number of parameters and timing for the forwards and backwards passes of each network is listed in Table 3.1. All of our models utilize the same input representation: a 3D grid of Gaussian-like atom type densities as generated by `libmolgrid`[23]. The grid is a 23.5Å cube with 0.5Å resolution centered on the center of mass of the ligand. Each grid point contains 14 ligand atom type channels and 14 receptor atom type channels, including distinct types for oxygen/nitrogen hydrogen donor/acceptors and aliphatic/aromatic carbons.

The "Default 2017" (Def2017) architecture is the architecture originally developed by Ragoza et al.[68]. The remaining architectures were the result of an extensive hyperparameter search on clustered cross-validated splits of the PDBbind refined set. The "HiRes" models were the best performing models on either the binding affinity regression or the binding pose classification task. In contrast, the "Default 2018" (Def2018) architecture was selected based on its combined performance on both affinity regression and pose classification, and its evaluation time. Lastly, "Dense" is a densely connected CNN[102] that is partially derived from a model previously utilized for virtual screening[86].

All models consist of a series of 3D convolutional and/or pooling layers followed by two separate fully connected layers whose outputs are the pose score and binding affinity prediction. Pose selection is a classification task to distinguish between low RMSD (< 2Å) and high RMSD (> 2Å) poses. This section of the network utilizes a logistic loss function. Conversely, the affinity prediction network is a regression task to correct predict the binding affinity for the receptor-ligand complex. It is trained with a custom L2-like pseudo-Huber loss that is hinged when evaluating high RMSD poses. That is, if the input is a low RMSD pose then the model is penalized for predicting either a too high or too low binding affinity. But, if the input is a high RMSD pose then the model is only penalized for predicting too high of a binding affinity for the complex.

3.3.2 PDBbind dataset preparation

The PDBbind is one of the most common benchmarking sets for predicting receptor-ligand binding affinity for models that take 3D structural data as input. It consists of an expansive (General) set, a curated (Refined) set, and a predefined 'Core' set. The 'Core' set is selected such that it matches the overall distribution of the rest of the data. In order to train our models we created several partitions of PDBbind v2016 for training and evaluation: Refined\Core, General\Core, clustered cross-validated (CCV) Refined, and CCV General. Complexes were discarded if the ligand molecular weight was greater than 1000Da, or if the ligand name was ambiguous. Each receptor and ligand was downloaded directly from the PDB as an SDF through the `downloadLigandFiles` service to avoid ambiguities in bond orders and protonation states present in the full PDB file. Waters and all atoms identified by the HETATM tag were stripped from the receptor via the ProDy python package[103].

Up to 20 docked poses were generated by docking ligands into their cognate receptor with smina[104] by defining a box around the crystal ligand with the `autobox` feature and keeping the rest of the options at the default value. Additionally, we energy minimized the crystal ligand using the UFF force-field[46] via `rdkit`[11], and then used the Vina scoring function to minimize this UFF conformer of the crystal ligand with respect to its cognate receptor. This ultimately gives up to 21 generated poses per receptor-ligand pair: 20 docked poses and 1 energy-minimized crystal pose.

This filtering and docking process resulted in the Refined set containing 3,805 complexes with 66,953 poses, the General set containing 11,324 complexes and 201,839 poses, and the Core set containing 280 complexes with 4,618 poses. The binding affinity labels were taken using the pK reported in the PDBbind. The binding poses were labeled as good if they had under 2Å RMSD to the crystal pose, and poor otherwise. Dataset comparisons and information are shown in Table 3.2.

In addition to the more typical train on General/Refined and test on Core setup, we

Dataset	Pockets	Complexes	Poses	Ligands	Affinity Data %
PDBbind Core	–	280	4,618	280	100
PDBbind Refined	–	3,805	66,953	2,972	100
PDBbind General	–	11,324	201,839	8,757	100
ReDocked2020	2,916	18,369	786,960	13,780	32.7
CrossDocked Iteration 0	2,922	18,450	10,691,929	13,839	39.9
CrossDocked Iteration 1	2,922	18,450	19,182,423	13,839	41.3
CrossDocked Only	2,767	18,293	21,797,142	13,786	42.2
CrossDocked2020	2,922	18,450	22,584,102	13,839	41.9

Table 3.2: Composition of the datasets used in this work. ReDocked2020 and CrossDocked2020 both have model-generated counterexample. CrossDocked Iteration 0 is the CrossDocked2020 set without any counterexamples added. ReDocked2020 and CrossDocked Only form a non-overlapping partition of CrossDocked2020 into redocked and cross-docked poses. Affinity Data refers to the percentage of poses with associated binding affinities from the PDBbind.

created CCV splits of the General+Refined+Core set (CCV General) and the Refined+Core set (CCV Refined). The clusters were created by grouping together receptors with over 50% sequence similarity or over 40% sequence similarity and 90% ligand similarity as computed with rdkit's FingerprintMols[11]. This results in complexes with highly similar ligands only being placed in distinct clusters if the receptors have less than 40% sequence similarity. Clusters were then randomly assigned to folds for 3-fold cross-validation.

3.3.3 CrossDocked 2020 dataset preparation

In order to expand the available binding poses, we rely on the assumption that similar ligands bind similar receptors. Protein sequence similarity is not a perfect metric to determine if two receptors are similar in this context, as we really only care about the properties of the binding site. To address this discrepancy we utilized the Pocketome v17.12 database.[105] Pocketome groups structures from the PDB based on the similarity of their ligand binding sites into "pockets" which contain the identified receptors and ligands. Thus for every pocket specified in Pocketome we downloaded the receptor and ligand files from the PDB. Similarly to the

PDBbind data, ligands with over 1000Da molecular weight were removed and the receptor structures were stripped of water and aligned to the Pocketome identified binding site. Unlike the PDBbind data, ions as identified by ProDy were retained and assigned as receptor atoms. Then for each ligand-receptor pair in a given pocket, we generated up to 20 docked poses and a singular UFF energy minimized pose as described with the PDBbind data. Finally, the binding data (pK) of a particular ligand was taken from PDBbind v2017 and assigned to all poses containing that ligand. This assumes that the binding affinity of a ligand is constant for all members of a given pocket. We also assume that the original crystal pose is the correct pose for a ligand with every receptor in a pocket. Notably, these assumptions are commonly made during structure-based virtual screening, but are not always valid. Thus, the labels of the data in CrossDocked2020 in inherently noisier.

We then had to generate CCV splits of the dataset. Importantly, we wanted to retain the nature of grouping pockets based on features of their binding sites. Due to the grouping by Pocketome, we already have a clustering of receptors and ligands within a pocket and only need a method to cluster between pockets. We performed this cross-pocket clustering by utilizing the ProBiS[106] algorithm with the z-score parameter set to 3.5 on the Pocketome identified cluster centers of each pocket. These clusters were then randomly assigned to the folds for cross-validation. In total, CrossDocked2020 version 1.0 contains 13,780 unique ligands, 41.9% of which have a binding affinity label, and is grouped into 2,922 pockets containing 18,450 pocket-ligand complexes. There are 22,584,102 receptor-ligand poses, 11,892,137 of which are generated in our counterexample generation procedure (outlined in the next paragraph). A ReDocked subset was created by only including poses where the ligand was docked into its cognate receptor. The ReDocked set contains the same pockets and ligands as the CrossDocked2020 set, but only has 18,369 complexes and 786,960 poses of which 391,137 are counterexamples (Table 3.2).

It has been shown that an iterative approach to the generation of training data improves the robustness of the trained model.[107] In order to do this, we first train a model on all

of the available training data then use it to optimize every pose in the training data *with respect to the newly trained model*. This results in the generation of new poses that the model considers as improvements to their starting pose. Since we know the correct answer (the crystal ligand pose), we can identify the newly generated poses that the model struggles with. That is, we update the training set with the newly generated poses that score high (above 0.9) while being more than 2Å RMSD away from the crystal pose (confidently wrong), or scored low (below 0.5) while being less than 2Å RMSD away from the crystal pose (unconfidently correct). The new poses that are identified in this way provide a set of *counterexamples* that are designed to confuse the model.[108] We filter these counterexamples to ensure that we only add poses that are more than 0.25Å away from any other pose in the training set. Each iteration added fewer poses (Table 3.2) and becomes computationally more demanding, so this process was performed twice for the creation of the CrossDocked2020 dataset.

3.3.4 Training procedure

All models were trained using a custom fork of the Caffe deep learning framework[109] with `libmolgrid` integration[23] using the `train.py` script available at `https://github.com/gnina/scripts` with a batch size of 50. Training examples were randomly shuffled, batches were balanced with respect to the pose label (low vs high RMSD poses), and examples were stratified with respect to their receptor so that targets are sampled uniformly during training regardless of the number of docked poses per target. In order to overcome the coordinate frame dependency of grids, input structures were randomly rotated and translated up to 6Å (provided the ligand did not leave the box) every time an example grid was generated during training. This approach was shown to be successful by Ragoza et al.[68].

Models were optimized with the stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.01, momentum 0.9, and with a weight decay of 0.01. We implemented an early stopping criteria to dynamically reduce the learning rate and terminate training when

Selected Training Hyper Parameters

Data Set	step_when	step_end_cnt	percent_reduced
PDBbind Refined (Crystal)	5	4	100
PDBbind Refined (Core)	25	4	100
PDBbind Refined (CCV)	18	4	100
PDBbind General (Crystal)	5	4	100
PDBbind General (Core)	88	4	100
PDBbind General (CCV)	88	4	100
ReDocked (CCV)	200	3	3.82
CrossDocked (CCV)	200	3	0.132

Table 3.3: Training Hyper Parameters

the model converges. Early stopping hyperparameters for each training set are provided in Table 3.3. Every 1000 iterations of the training set the early stopping criteria is evaluated on a reduced version of the training set. The size of this set is determined by the *percent_reduced* parameter in `train.py`. If there is no reduction in the training loss during the last *step_when* evaluations, then the learning rate is lowered by a factor of 10. This lowering of the learning rate can occur *step_end_cnt* times, after which training will cease. We select the *step_when* parameter such that the network will see the entire training set or 200,000 examples, whichever is smaller, before updating the learning rate.

For each dataset we trained five models with five different random seeds for evaluation. Additionally, for the CCV PDBbind data, each seed utilized a different 3-fold split of the data. This was not the case for models trained with CrossDocked2020 or ReDocked2020, where only a single 3-fold split is considered due to the computational cost and time required to create splits of this much larger dataset.

3.3.5 Evaluation metrics

All of our models output a pose score for binding pose classification and predict the binding affinity of the receptor-ligand complex for binding affinity regression. We evaluate the binding pose classification task with the area under the curve of the receiver operating characteristic

curve (AUC), and the 'Top1' fraction. The AUC indicates how well the model separates low RMSD from high RMSD poses and is a measure of inter-target ranking power. Conversely, Top1 is the fraction of low-RMSD (< 2Å) poses among the top-ranked poses and is a measure of intra-target ranking power (i.e. how often docking is successful). Notably, the meaning of Top1 significantly depends on the underlying ratio of generated poses. As an example, if not all complexes have a low RMSD pose, then the best possible Top1 is less than 1.0. Additionally, the expected Top1 of a random classifier will vary depending on the number of low RMSD poses that were sampled during docking. In order to provide context for our Top1 results, we provide the best possible Top1 and the performance of a random classifier. Finally, when evaluating crossdocked poses, we consider all docked poses of a ligand across all receptors in a given pocket (pocket-ligand pairs) as a single set, emulating ensemble docking[110].

In order to evaluate the quality of the binding affinity, we must first select which docked pose of the ligand we are evaluating. This is done to avoid having multiple instances of the same complex with the same label in our metrics. Unless stated otherwise, we select a pose for a given complex (receptor:ligand for PDBbind, or pocket:ligand for Pocketome) by taking the pose with the highest pose score (the same pose used to generate the 'Top1' statistic) or best Vina score when evaluating the Vina scoring function. The predicted affinity for this selected pose is then used to calculate the Pearson's R and root mean squared error (RMSE) with the experimental binding affinity data in pK units. We also analyzed the effect of selecting our singular pose by the highest predicted affinity, the best pose (lowest RMSD to the crystal), the worst pose (highest RMSD to the crystal), or a random pose for these affinity metrics.

Our general base line is the Autodock Vina[47] scoring function. In order to compare the binding affinities from the PDBbind to the Vina scores, we need to convert the Vina score

from kcal/mol to pK. This is done via the formula:

$$pK = -\log_{10}(e^{\frac{vina}{T \cdot R}})$$

Where $T = 295$K and $R = 1.98720 \cdot 10^{-3}$kcal mol^{-1}K^{-1} is the ideal gas constant.

With all of the datasets and metrics set up, we then performed a series of experiments. The first group of experiments served to characterize our various CNN models on the PDBbind data in order to compare them with other state of the art models. We then demonstrate that the results of our (and other's) models on the PDBbind Core test set are overly optimistic, and argue for the adoption of CCV splits by the community going forward. This leads into the second group of experiments where we rigorously evaluate the performance of the Def2018 and Dense architectures on the CrossDocked2020 dataset. Lastly, we present the results of our best networks to serve as a benchmark for the community and provide the exact poses and data splits used to train our models.

3.3.6 Characterizing CNN performance on the PDBbind data

In order to show the benefits of training a model on CrossDocked2020, we first need to compare our CNN models to contemporary methods. This means a rigorous evaluation of our models on the PDBbind data. We first compare our models with other networks that all trained on the PDBbind Refined/General set and tested on the Core set in Table 3.4. In particular, Pafnucy[85] and KDeep[25] are both CNNs based on grids similar to our models, RF-Score[21] is a random forest, and 1D2D CNN[88] is a CNN with a distinct input representation based off of the topology of the input. We include Vina as a representative of a traditional scoring function. Our best performing models show similar performance to the previous grid-based CNN methods and RF score, although a precise comparison is not possible due to differences in the training and test sets (and not having the exact docked pose distributions used to

Model	RMSE	R
Def2018 Refined Crystal	1.50	0.73
Def2018 Refined	1.50	0.72
Def2018 General	1.38	0.79
Def2018 General Ensemble	1.37	0.80
Dense General	1.49	0.73
Dense General Ensemble	1.35	0.79
Pafnucy[85*]	1.42	0.78
KDeep[25†]	**1.27**	0.82
RF Score[21‡]	1.39	0.80
1D2D CNN[88†]	1.64	**0.848**
Vina	2.22	0.41

[*] Train: PDBbind General and Refined v2016 crystal structures (N=11,906). Removed Nucleic Acid+Protein, Protein+Protein, and Nucleic Acid+Ligand from all sets. Test: remaining Core set (N=290).
[†] Train: PDBbind Refined v2016 crystal structures (N=3767). Test: PDBbind Core set crystal structures (N=290)
[‡] Train: PDBbind Refined v2007 crystal structures (N=1300). Test: PDBbind Core set crystal structures (N=195)

Table 3.4: Affinity prediction performance on PDBbind Core (N=280) for a variety of models.

train other models). This is reassuring, as it indicates that further analysis of our models on PDBbind and CrossDocked2020 should produce a comparable effect on the other models rather than being a quirk of our particular CNN architectures.

In order for our CrossDocking method to be appropriate, we need to know the effect of training on docked poses rather than the crystal poses for affinity prediction. We investigated this by training on two versions of the PDBbind Refined data: one with only the crystal poses (Crystal) and another with only docked poses (Docked). When training with the Crystal set, the pose score layer of the model is omitted and the only loss computed is the L2-like loss on the affinity prediction. The Docked set includes both low (< 2Å) and high (> 2Å) RMSD poses, and models trained with this follow the training procedure outlined in the Methods section.

We measure the performance of our models on predicting receptor-ligand binding affinity for these two sets in Figure 3.2. All four models achieve comparable performance on both the

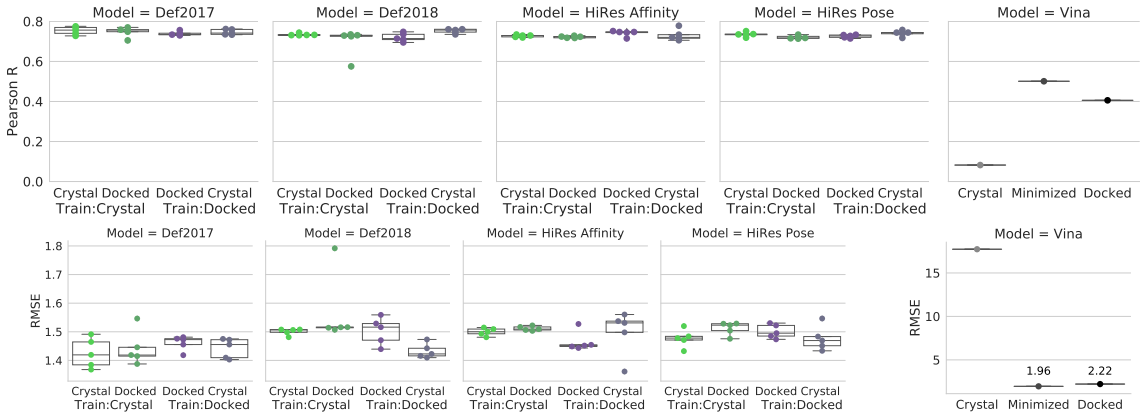
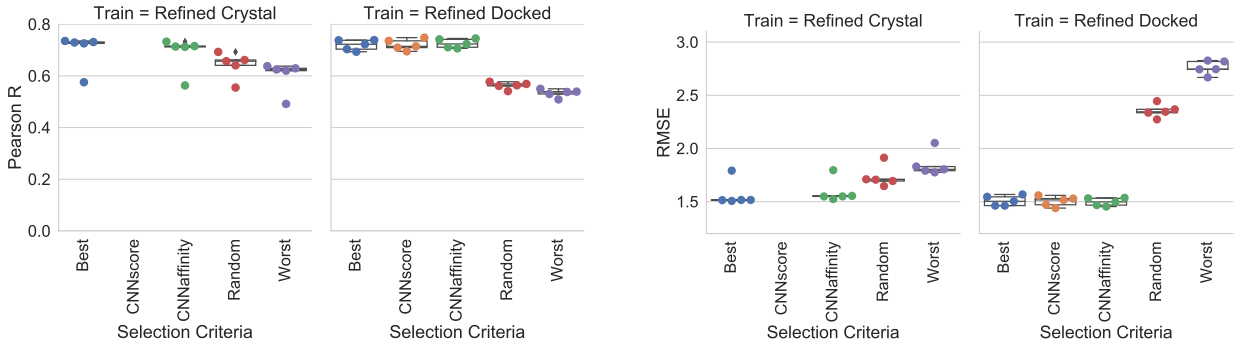


Figure 3.2: Affinity prediction correlation and RMSE on PDBbind Core set for models trained using crystal or docked poses from the Refined Set. Autodock Vina was used as a baseline. The test set consisted of either crystal or docked poses. Note: there is an increased scale for the Autodock Vina RMSE results plot

Crystal and Docked datasets, with average R in the range 0.72 to 0.75. This demonstrates that the inclusion of docked poses does not reduce binding affinity prediction performance, despite the inclusion of low quality poses. We also demonstrate that a model trained with only Crystal data and evaluated on Docked poses achieves a similar result to a model trained on Docked poses and evaluated on Crystal data. This indicates that our CNN models are insensitive to small perturbations of ligand positions (e.g. a low RMSD pose is scored similarly to a crystal pose as desired). Notably, this is in contrast to the AutoDock Vina scoring function, which performed poorly on the Crystal data. This is due to the presence of clashes in the Crystal data, which result in very large repulsion terms.

As shown in Figure 3.2, our new models behave similarly to the Def2017 model. The HiRes models are the best at the task and dataset (Refined) they were optimized for, but this pattern is not conserved across different training and test sets, suggesting that the models may have been selected for their ability to overfit the Refined set. Since Def2018 was selected for its generally solid performance and fast run time, it has fewer parameters which may have had the effect of muting the problems of the HiRes models. Since all four models demonstrated similar trends, and the Def2018 generally performed best, all further evaluations were only



(a) Affinity prediction correlation

(b) Affinity prediction error

Figure 3.3: Affinity prediction performance for Def2018 model with different pose selection methods when trained on Crystal or Docked poses of PDB Refined and tested on Core. Best is the lowest RMSD pose to the crystal pose, CNNscore is the highest predicted scoring pose (not applicable for Crystal trained models), CNNaffinity is the highest predicted affinity, Worst is the highest RMSD pose to the crystal pose, and Random is taking a pose at random.

for the Def2018 model. This limited improvement motivated the substantially different Dense model architecture, which is evaluated in Figure 3.14.

Given that training on docked poses had little effect on binding affinity prediction (Figure 3.2), we then evaluated if our models were pose-sensitive at all and how the choice of binding pose affected performance. There are five different pose selection methods: Best (selecting the pose with the lowest RMSD to the crystal pose), CNNscore (selecting the pose with the highest predicted pose score, the default), CNNaffinity (selecting the pose with the

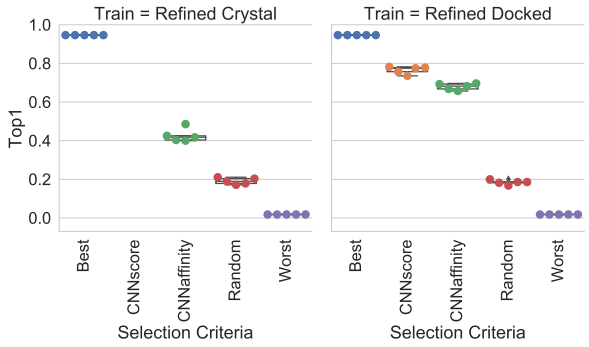


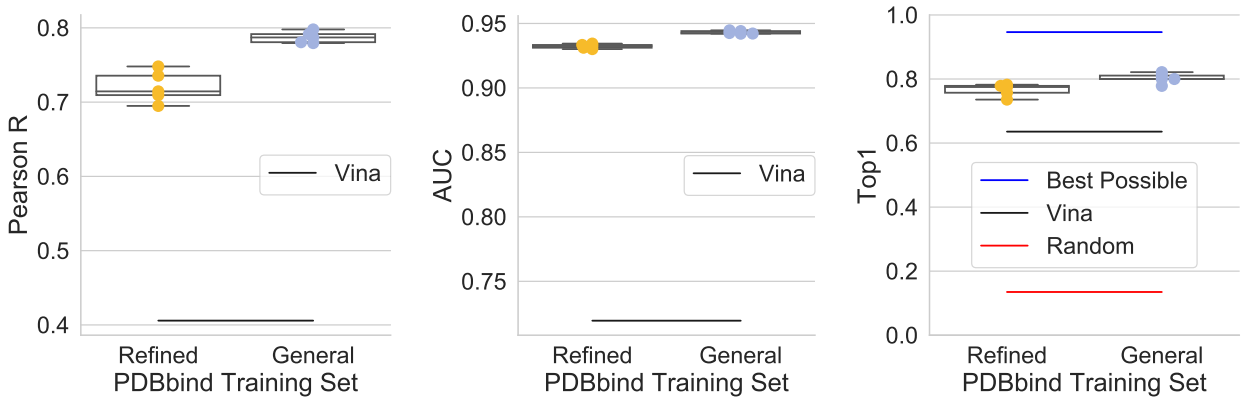
Figure 3.4: Intra-target pose ranking performance of various pose selection methods with the Def2018 model when trained on Crystal or Docked poses of PDB Refined and tested on Core.

highest predicted affinity), Random (selecting a random pose), and Worst (selecting the pose with the highest RMSD to the crystal pose). For each of these pose selection methods we evaluated the Def2018 model trained on the Refined Crystal or Refined Docked set and tested on the Core set made up of docked poses (Figures 3.3 and 3.4).

As the quality of the selected pose decreases, both the correlation and RMSE of the predicted affinity worsen. The effect is more pronounced for the models trained with Docked data (Figure 3.3). Interestingly, while using the highest RMSD pose reduces affinity prediction performance, the Crystal trained Def2018 model still achieves an R of 0.60 compared to 0.70 with the best possible pose. This suggests that the model trained with the Crystal data is making minimal use of the protein-ligand interactions in the affinity prediction task. However, models trained withe Docked poses exhibit affinity prediction quality better correlated with pose quality (Figure 3.3), and the affinity prediction by itself is significantly better at selecting low RMSD poses (Figure 3.4).

All of the previous analysis was performed on the PDBbind Refined set, which is filtered from the PDBbind General set. The General set is composed of data that is of dubious quality[111]. We investigated the effect of adding more, but lower quality, data to training our models by comparing models trained with the General set to models trained with the Refined set, with both tested on the Core set (Figure 3.5). For all of our models and metrics, training on the PDBbind General set improves Core set predictions. This suggests that the quality controls utilized in the creation of the Refined set can be overly strict, and imply that training on a larger quantity of data can outweigh the data being lesser quality.

These past analyses trained on the Core set are problematic, as the Core set, by design, mimics the distribution of values of the General/Refined set. Thus, taking your model performance metrics by their result on the Core set is similar to the fit of the training set as the test set is being drawn from the same distribution of values. However, in the drug discovery space, one is more concerned with the ability of the model to generalize to unseen chemistry. We can better mimic this through the evaluation on our 3-fold CCV splits of



(a) Affinity prediction correlation (b) Inter-target pose ranking (c) Intra-target pose ranking

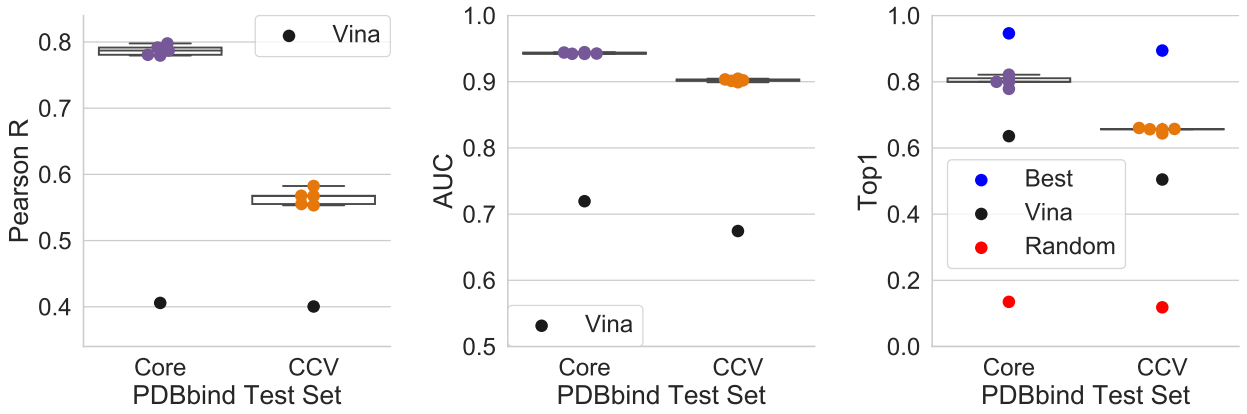
Figure 3.5: Performance on Core when the training set is expanded from PDB Refined to General.

the General set. For each of our three metrics, the clustered cross-validated models perform substantially worse. Pearson R drops from 0.78 to 0.56, AUC from 0.94 to 0.89, and Top1 from 0.77 to 0.62 (Figure 3.6).

The most likely explanation for this performance difference is that the Core set is a poor measure of a model's ability to generalize. The training size of the CCV models is higher than model's trained on the Refined set (Table 3.2), and the CCV metrics are also substantially worse than the Refined set performance on the Core set (Figure 3.5). By design, the CCV splits measure the performance of models on new target classes, whereas the Core set is constructed to that there is a low/medium/high affinity example of each target class. This results in a different distribution of affinity values that produces artificially high correlations (Figure 3.6). These factors suggest that a significant portion of the performance measured by testing on the Core set is attributable to overfitting the training set.

3.3.7 CNN performance on CrossDocked2020

In the prior subsection we demonstrated that our CNN models achieve comparable performance to other methods and that expanding the available training data through both docking AND the inclusion of more lower quality data improves model performance. This further motivated



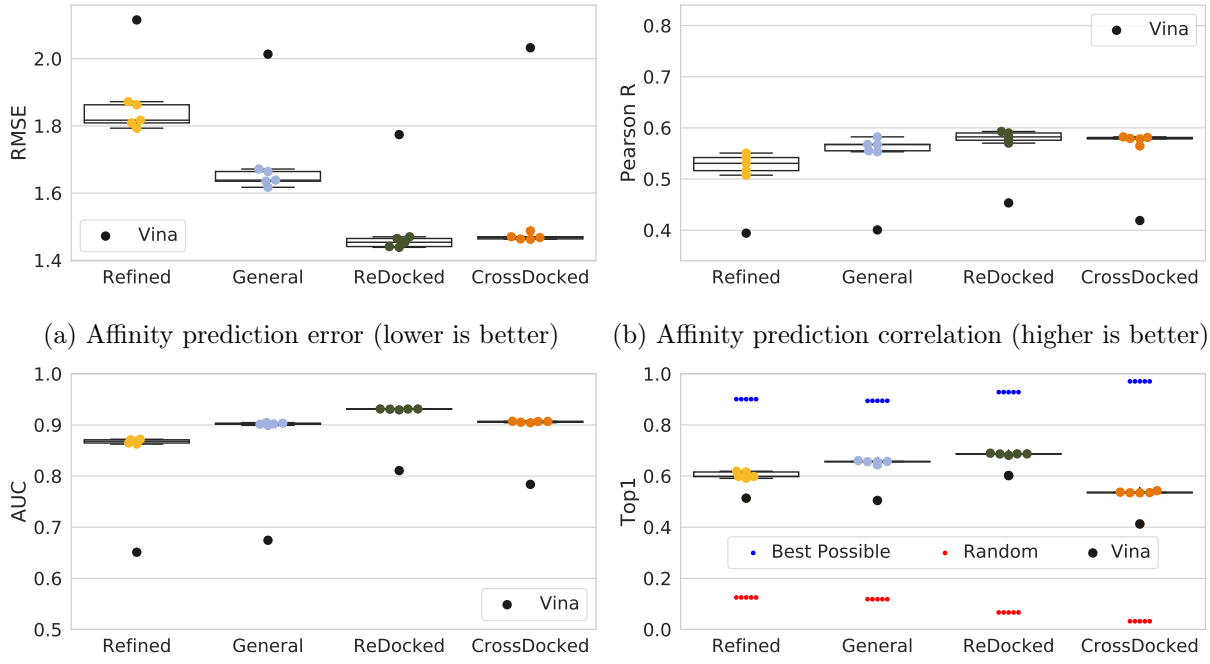
(a) Affinity prediction correlation (b) Inter-target pose ranking (c) Intra-target pose ranking

Figure 3.6: Performance when utilizing different train/test splits. Models were either trained on PDBbind General and tested on PDBbind Core (Core) or trained with clustered cross-validation splits of the PDBbind General. Note the same data is in both sets, but is divided differently among train and test.

the creation of the CrossDocked2020 dataset, which greatly expands the available pose data by including cross-docked poses, complexes that lack affinity data, and counterexamples. We compare the performance of the Def2018 model on CCV splits of the PDBbind Refined set, General set, the ReDocked subset of CrossDocked2020, and CrossDocked2020 itself in Figure 3.7.

We generally observe that as more redocked poses are added to the training set (Refined < General < ReDocked Figure 3.7), model performance increases for all metrics. Interestingly, we also note that the affinity metrics improve even with the inclusion of training complexes with unknown binding affinity labels. However, we caution that as the underlying data distributions of the different CCV sets are different, it is not possible to definitively conclude that the improvement is due to the additional volume of data. In fact, Vina also sees improvement on the ReDocked2020 dataset.

The notable exception from above is that the performance of the Def2018 model at pose selection for the CrossDocked2020 dataset is substantially reduced. This is not necessarily surprising, as crossdocked poses are inherently noisier and there are many more poses to



(a) Affinity prediction error (lower is better)

(b) Affinity prediction correlation (higher is better)

(c) Inter-target pose ranking (higher is better)

(d) Intra-target pose ranking (higher is better)

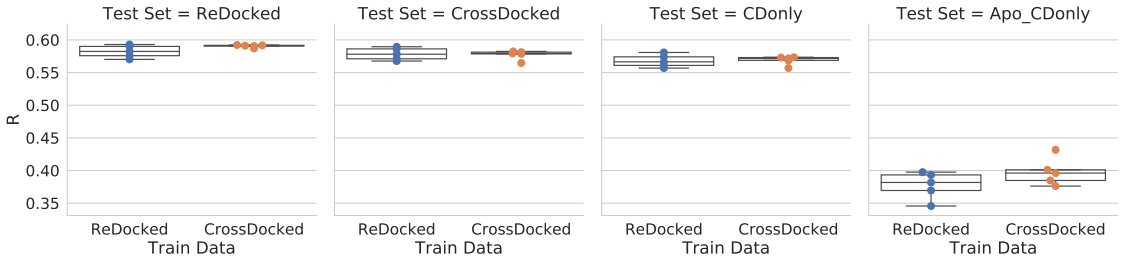
Figure 3.7: Clustered cross-validation performance of the Def2018 model trained with our various datasets. Training and testing set size increases along the horizontal axis. Note, as each test set is distinct the performance of each method cannot be directly compared. Instead compare with performance relative to Vina

select from. It is simply a much more challenging task, while also being a more realistic assessment of a model's performance in a prospective docking experiment. Notably, the drop in docking accuracy for the CNN model is less than the drop exhibited by Vina, which is reassuring.

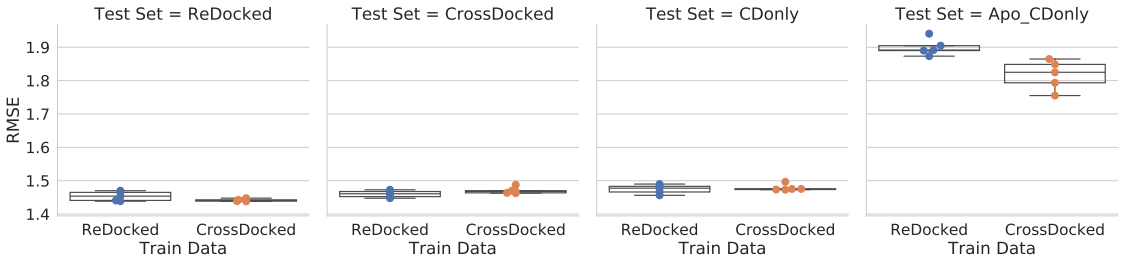
In contrast, binding affinity regression performance of the Def2018 model is similar on the ReDocked and CrossDocked2020 datasets. This suggests that the inclusion of extra negative examples and noisier pose labels does not affect the affinity prediction capabilities of our model. We investigated this further by evaluating models trained on ReDocked2020 and tested on CrossDocked 2020 and vice-versa in Figure 3.8. The difference in Pearson R between the models in not statistically significant ($p > 0.05$, Student's t-test) and the CrossDocked2020 trained model has a better AUC and worse Top1 than the ReDocked2020 model. Importantly, models trained with CrossDocked2020 see a performance boost when evaluated on ReDocked2020 as compared to a performance drop the other way around. This suggests that models trained with the crossdocked poses are more robust.

We then characterized the impact of training with crossdocked poses by training models with either the CCV PDBbind General set or CrossDocked2020, and then evaluating them on their matching test set, the other test set, or a subset of CrossDocked2020 without the counterexamples (it0). We included the it0 version of CrossDocked2020 to have a more fair comparison to the PDBbind General set data, which does not have counterexamples present in it. Additionally, since the data splits of the PDBbind General set are different per seed, each corresponding swapped test set has a different amount of data removed to avoid test-on-train. Figure 3.9 shows the results of this training schema. Since each test set is unique, we cannot directly compare the results of each column and can only comment on the observed trends.

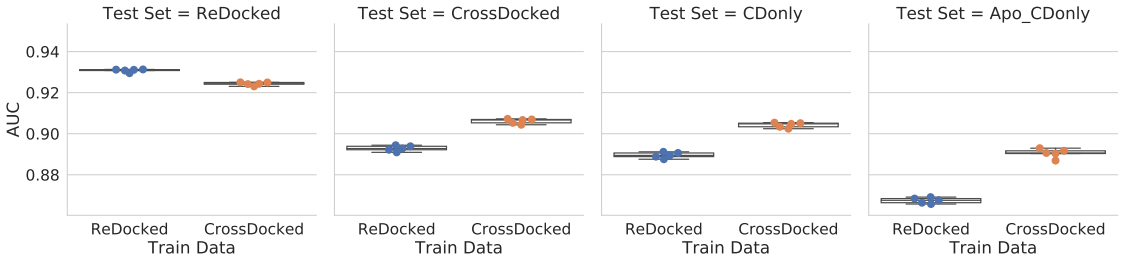
Models trained with PDBbind data alone are unsurprisingly fooled by the counterexamples present in CrossDocked2020, whereas models trained on CrossDocked2020 generalize well to the PDBbind data. Removing the counterexamples is enough to rescue the PDBbind



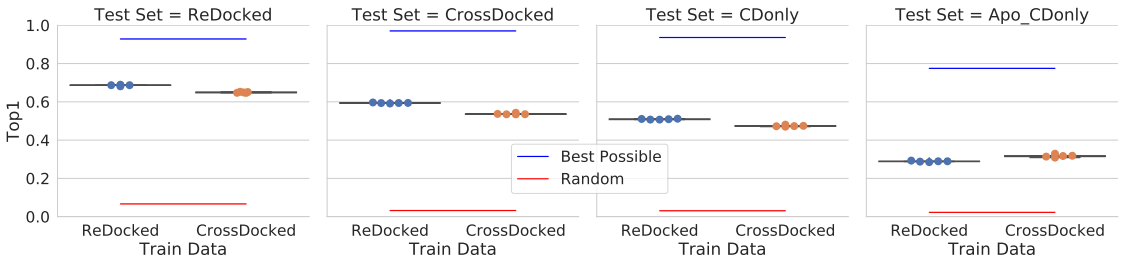
(a) Affinity prediction correlation



(b) Affinity prediction RMSE

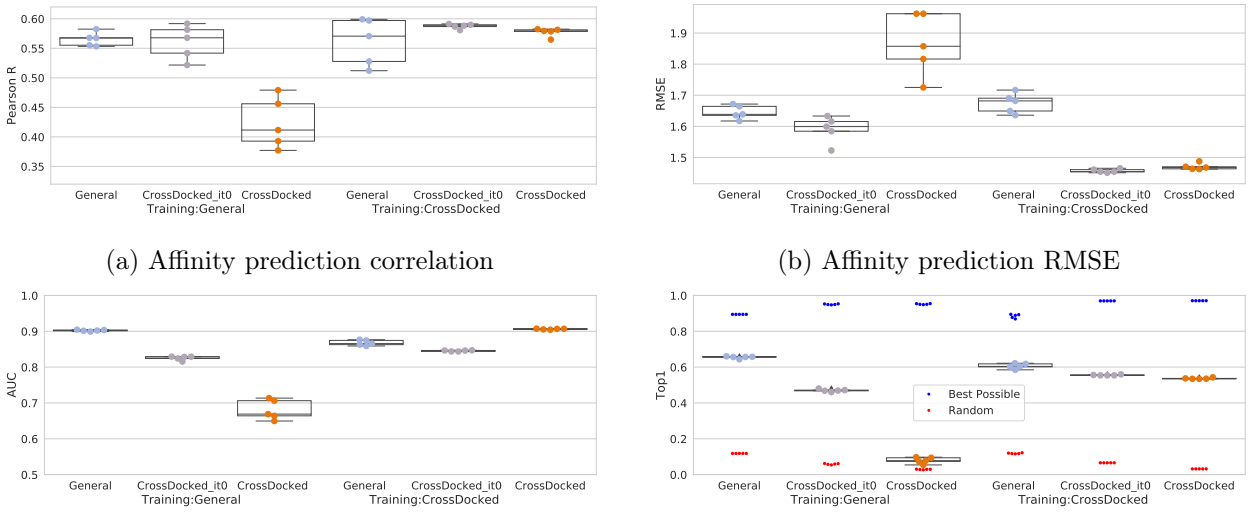


(c) Inter-target pose ranking



(d) Intra-target pose ranking

Figure 3.8: Performance of training and testing with and without cross-docked poses. Def2018 models were trained on either the ReDocked2020 set or the CrossDocked2020 set. They were then evaluated on either the ReDocked2020 set, the CrossDocked2020 set, only the cross-docked poses in the CrossDocked2020 set (CDonly), or only the apo receptors of the CDonly set.



(a) Affinity prediction correlation

(b) Affinity prediction RMSE

(c) Inter-target pose ranking

(d) Intra-target pose ranking

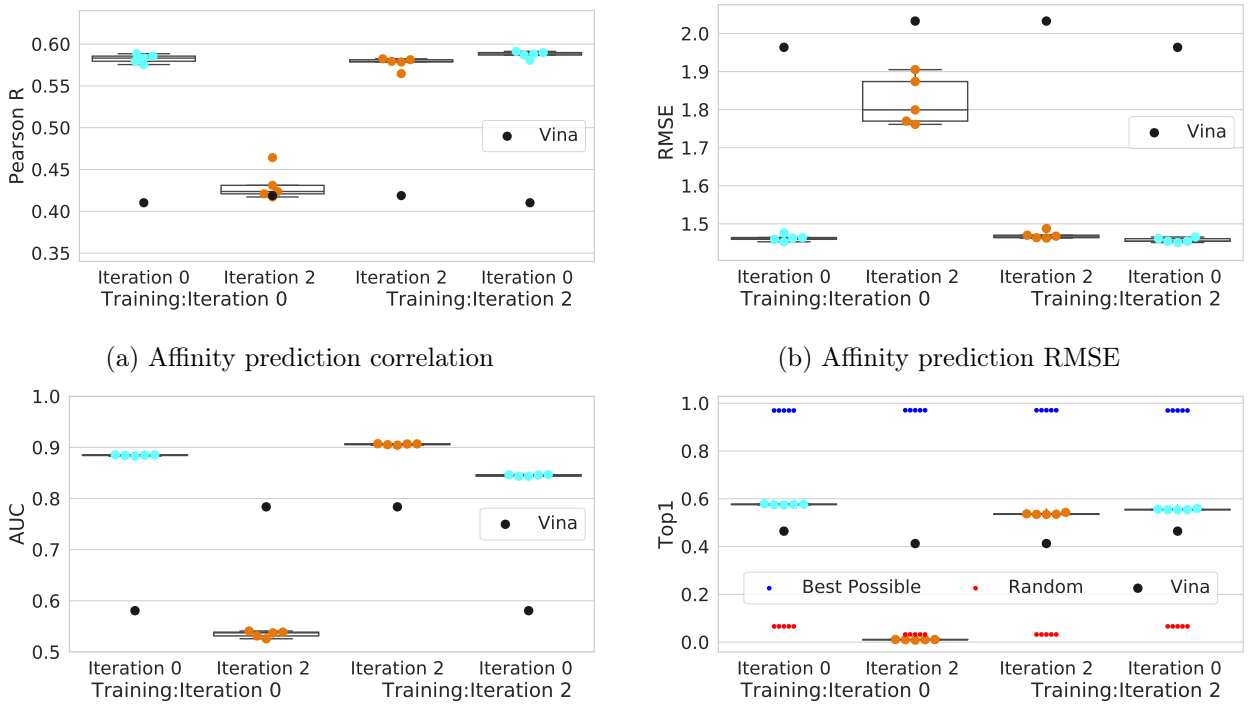
Figure 3.9: Performance of training and testing with and without cross-docked poses. Def2018 models were trained on either the clustered cross-validated PDBbind General set or the CrossDocked2020 set. They were then evaluated on either the PDBbind General set, the CrossDocked2020 set without counterexamples, or only the full CrossDocked2020 set. Note that each test set here is unique, due to varying splits of PDBbind General having different overlap with CrossDocked2020

trained model's performance on the crossdocked poses for binding affinity prediction, but is insufficient to rescue the performance on binding pose classification. This reinforces that CrossDocked2020 measures the results on a more challenging binding pose classification problem. Lastly, when comparing the CrossDocked2020 test set without the counterexample poses (the grey columns) between both training sets, models trained with CrossDocked2020 data exhibit a small performance gain on the AUC, Pearson R and Top1, along with a substantial improvement on RMSE. This suggests that models trained with only the redocked PDBbind data are not as equipped to handle crossdocking tasks.

In turn, this questions if the performance differences observed in Figure 3.9 are due to the inclusion of crossdocked poses, or just from other differences between the PDBbind data and the data in CrossDocked2020. We investigated this by training models on either the ReDocked subset of CrossDocked2020 or all of CrossDocked2020 and report the performance of these models on test sets in escalating order of receptor deviation from the cognate receptor (Figure 3.8). We expect that the apo structures present in CrossDocked2020 represent the most challenging examples, as we are trying to fit a ligand into a receptor with no ligand present. This is also reflective of a common use case scenario in the drug discovery pipeline.

Figure 3.8 shows that training on CrossDocked2020 allows for a small performance boost on affinity prediction for the apo structures (R from 0.378 to 0.398, and RMSE from 1.90 to 1.82), AUC (0.867 to 0.891), and Top1 (0.289 to 0.317). Interestingly, on all tasks and test sets models trained on CrossDocked2020 generally performed about the same or better than models trained on the ReDocked2020 data. Notable exceptions are the AUC when testing on ReDocked, and all of the Top1 metrics except the apo test sets. This suggests that training with CrossDocked data generally does not hurt model performance, and helps in the hardest tasks.

We then investigated the effect of the two iterations of counterexample generation on model performance for CrossDocked2020. This was done by training models on our initial "Iteration 0" CrossDocked2020 set (no counter examples) and our full CrossDocked2020 set,



(a) Affinity prediction correlation

(b) Affinity prediction RMSE

(c) Inter-target pose ranking

(d) Intra-target pose ranking

Figure 3.10: Effect of counterexamples on Def2018 clustered cross-validated performance. The models were trained on the CrossDocked2020 set either without counterexamples (Iteration 0) or with counterexamples (Iteration 2). They were then evaluated on the test set without or with the counterexamples. Note same colors indicates the same test set.

"Iteration 2", and evaluating them on both test sets (in Figure 3.10). As expected models trained without counterexamples were completely fooled by counterexamples existing in the test set (orange dots in Figure 3.10). Counter examples hurt a model's ability to perform binding pose selection on datasets without counterexamples in them, dropping the AUC from 0.885 to 0.845 and dropping the Top1 from 0.577 to 0.566 (blue dots in Figure 3.10C,D). The opposite effect is observed on binding affinity predictions with Pearson R going from 0.577 to 0.587 and RMSE going from 1.463 to 1.457 (blue dots in Figure 3.10A,B). Taken together, these conflicting results suggest that adding counterexamples into the training regime does not strictly improve model performance on the original data.

However, the motivation for including counterexamples was not to improve model perfor-

mance; rather, they are include to improve the model's robustness to out-of-distribution poses and give the model more meaningful gradients. In order to test for this, we analyzed the results of a docking run of the PDBbind Core set where our Def2018 and Dense networks were utilized in the energy minimization process. We plot the distribution of RMSD to the Crystal pose for every pose generated during docking for models trained without counterexamples (It0) or with counterexamples (It2) in Figure 3.11. The inclusion of counterexamples during training resulted in more low RMSD poses being sampled for both models. The effect is more dramatic for the Def2018 model (mean RMSD of 6.03 to 4.62) than the Dense models (mean RMSD 6.35 to 5.33). This is likely due to the counterexamples being generated for the Def2018 architecture specifically. It is reassuring that they still provided a benefit to the Dense architecture.

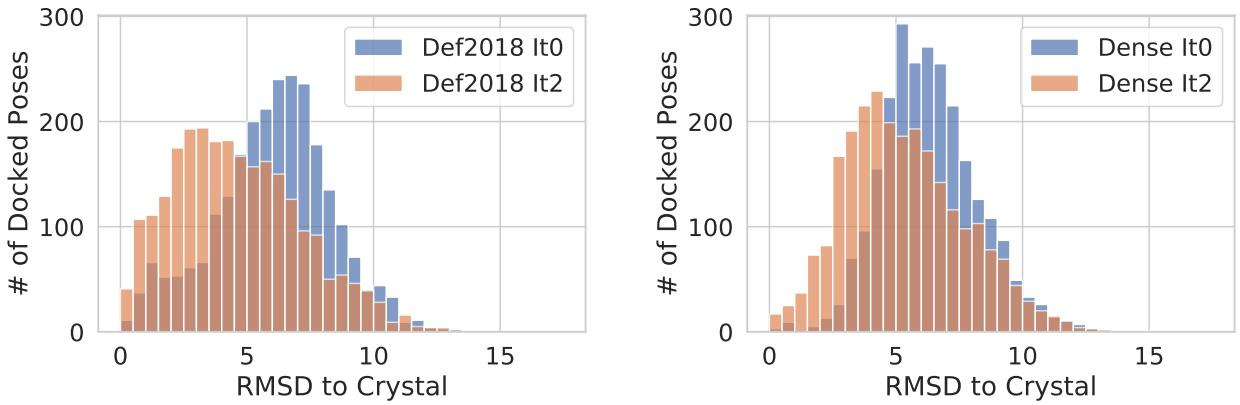


Figure 3.11: A Histogram of the RMSD of poses minimized using the Def2018 or DenseNet models trained with or without our counterexamples. While not as impressive as the Def2018 model from which the counterexamples are generated, they still yield a positive benefit for the DenseNet.

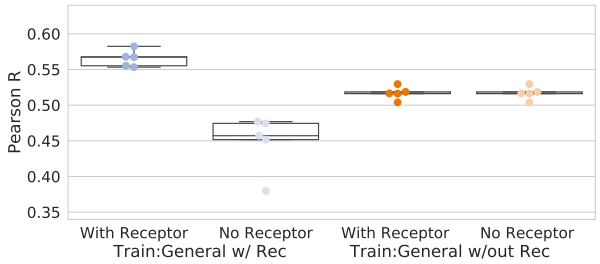
The expectation for a structure-based ML model is that its output is primarily a function of the receptor-ligand interactions, which is the case with classical scoring functions. However, it has been demonstrated that ligand-only, cheminformatic information can explain much of the performance of structure-based ML models[36,61,112]. We investigate this effect with the Def2018 architecture on both the PDBbind General set and the CrossDocked2020 set in Figure 3.12, by comparing the results between versions of the model trained with only ligand

informations and models trained with the full complex.

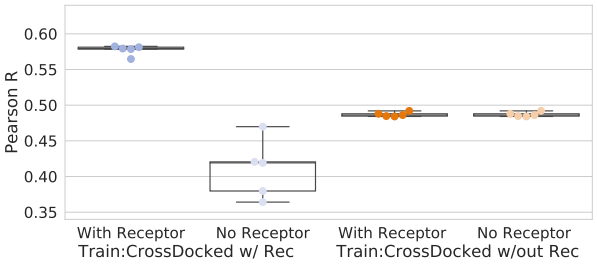
Unsurprisingly, models trained without receptor information have unchanged performance when tested on a set with receptors. This is due to the weight regularization during training setting the weights that would deal with the receptor channels to zero. Models trained with a receptor perform worse on the ligand-only test set than models trained with only ligand information. This indicates that some amount of receptor information is being utilized in our model's predictions.

Consistent with the previous observations[36,63,112], a model trained without receptor information is able to achieve a significant correlation on predicting binding affinity with a Pearson R of 0.52 on the PDBbind General set and 0.49 on the CrossDocked2020 set. This is a performance drop from training on the entire complex (Pearson R of 0.56 and 0.58 on the General set and CrossDocked2020 respectively), but it is not a major drop. This suggests that protein ligand interactions play more of a role in affinity prediction for models trained with CrossDocked2020. In order to investigate this further we trained a variety of simpler models with cheminformatic descriptors as input on the General set and CrossDocked2020 (Table 3.5. These simpler models are able to achieve a Pearson R of 0.51 on average for the PDBbind General set, but only a Pearson R of 0.27 on CrossDocked2020.

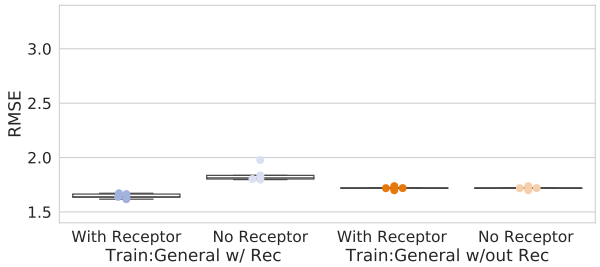
In contrast to the affinity results, since pose selection is a inherently a function of the pose of the ligand relative to the receptor, models trained without a receptor have a Top1 metric equal to random performance (Figure 3.12G,H). Similarly, models trained with a receptor exhibit close to random AUCs when evaluated on the ligand-only test set (Figure 3.12E,F). Surprisingly, the AUC of the ligand-only models on the General set and CrossDocked2020 are both significantly higher than the expected 0.5 of a random classifier. Since there is no receptor, this non-random performance must be due to differences in the ligand conformation or some general cheminformatic descriptor of the ligand. Our initial hypothesis is that the native crystal pose could have lower energy than other poses, which could be identified by our model. In Figure 3.13 we show that scoring each pose in the PDBbind General set by its



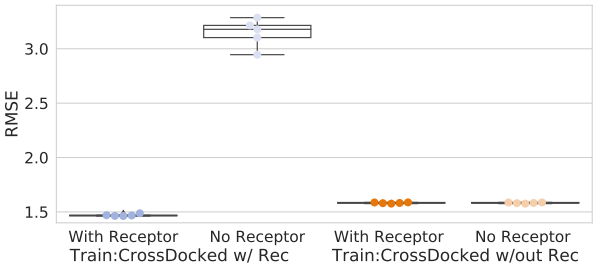
(a) PDBbind General affinity prediction correlation



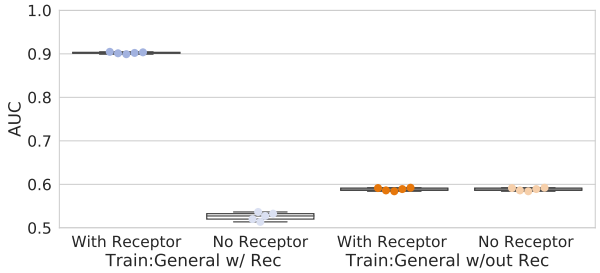
(b) CrossDock affinity prediction correlation



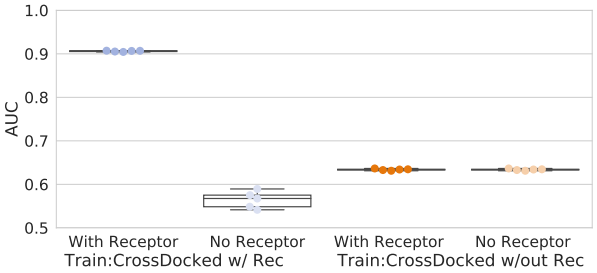
(c) PDBbind General affinity prediction RMSE



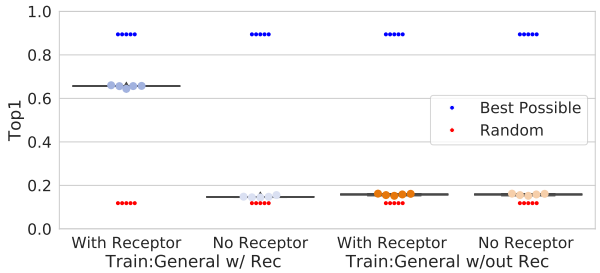
(d) CrossDock affinity prediction RMSE



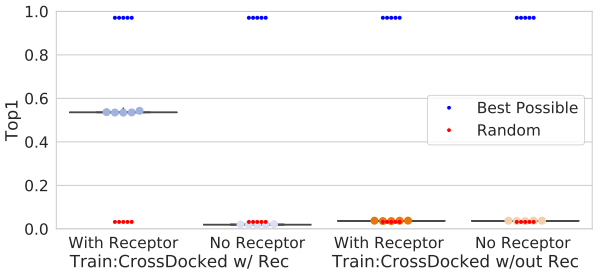
(e) PDBbind General inter-target pose ranking



(f) CrossDock inter-target pose ranking



(g) PDBbind General intra-target pose ranking



(h) CrossDock intra-target pose ranking

Figure 3.12: Ligand-only model performance. Def2018 models were trained with or without receptors (w/ Rec or w/out Rec) and evaluated on test sets with or without receptors (With Receptor or No Receptor).

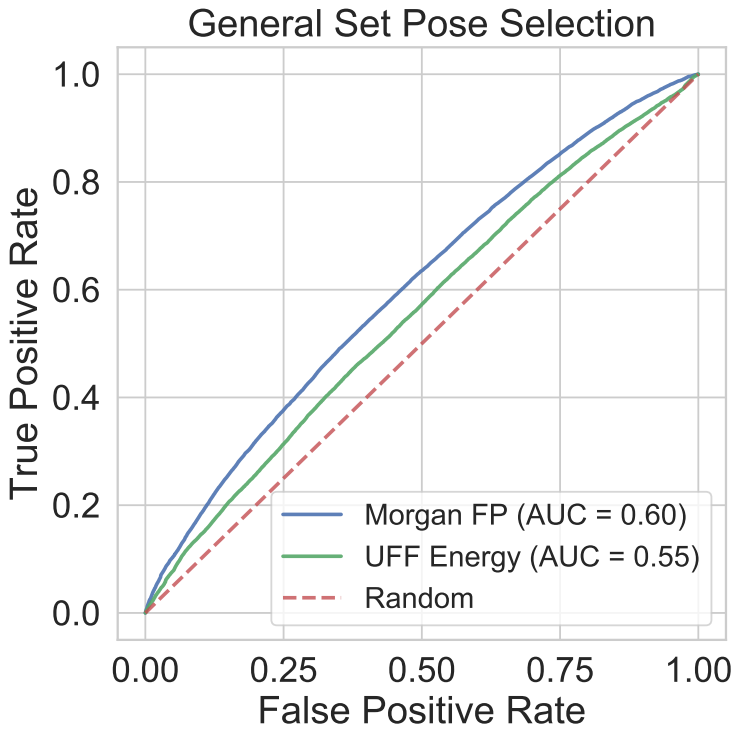
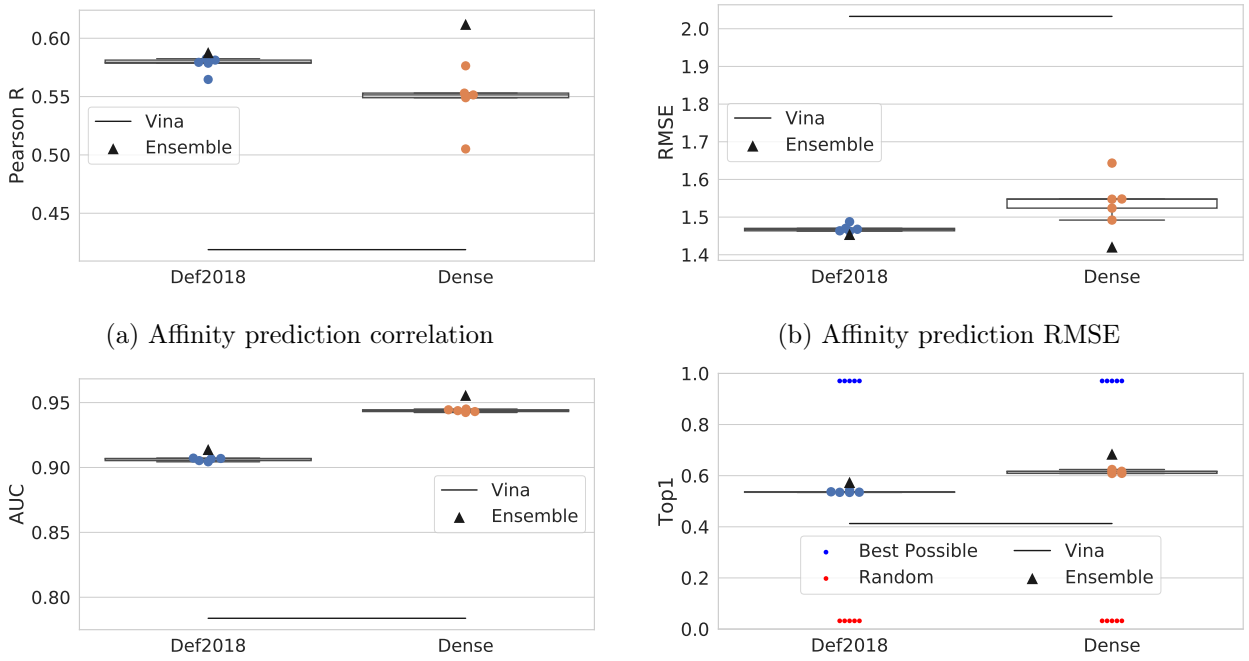


Figure 3.13: Investigating performance of using simple ligand-only classifiers to distinguish good from bad poses for the PDBbind General and Core sets.

internal energy as calculated by the UFF force field achieves and AUC of 0.55. This is similar, but worse, AUC performance than our ligand only CNN models which achieve an AUC of 0.59. Thus, the energy of the ligand pose is unable to account for the extra enrichment we observe with the ligand only models.

We then fit a linear regression model to the 2D-only Morgan fingerprint of the ligands in the General set (Figure 3.13). This results in an AUC of 0.60, which is a much closer match to the performance of our ligand-only CNN model (AUC 0.59). Since these fingerprints are independent of the ligand conformation, this result is achieved despite different poses of the same ligand producing identical scores. The reason this does not result in an AUC of 0.5 is that not all ligands have the same fraction of low RMSD poses. As an example, a rigid molecule which binds to a fully enclosed protein pocket would have fewer high RMSD poses as the steric constraints of the system would prevent them being sampled during docking. It



(a) Affinity prediction correlation

(b) Affinity prediction RMSE

(c) Inter-target pose ranking

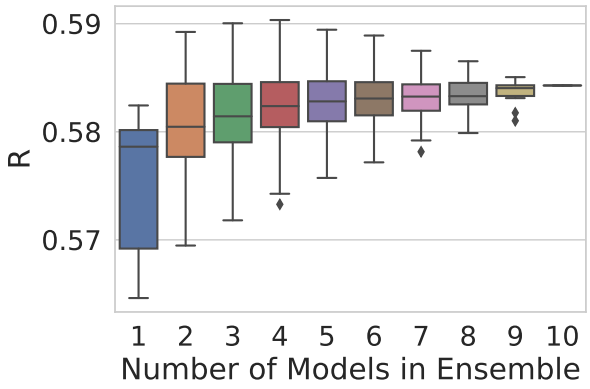
(d) Intra-target pose ranking

Figure 3.14: Dense model compared to Def2018 on the CrossDocked2020 set. The performance of the ensemble of both sets of five models is also shown.

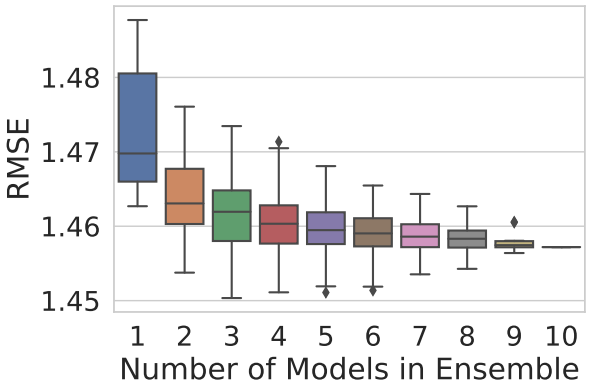
appears that the model can identify these 'highly dockable' chemotypes, which results in the non-random AUC. This artificial enrichment is not present in the Top1 metric, as it strictly evaluates the ordering of poses of the same ligand.

Finally, we evaluated our more computationally demanding Dense model on the CrossDocked2020 dataset in Figure 3.14. The Dense architecture has nearly twice as many parameters as Def2018 and it takes an order of magnitude longer to evaluate (Table 3.1). This extra computation results in an improvement at pose selection, with Dense having a Top1 of 0.615 compared to the 0.537 of Def2018. Interestingly, the Dense nets actually performed worse at binding affinity regression than the Def2018 architecture (average R of 0.55 instead of 0.58).

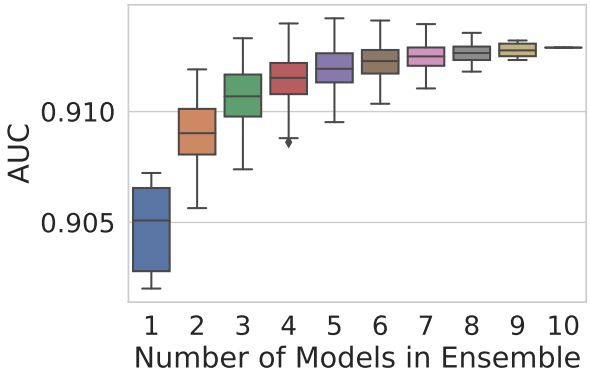
Our final experiment concerned the use of model ensembles to produce better results on CrossDocked2020. We first evaluated taking the ensemble mean of up to 10 differently seeded Def2018 models on CrossDocked2020 as the predicted score and binding affinity of



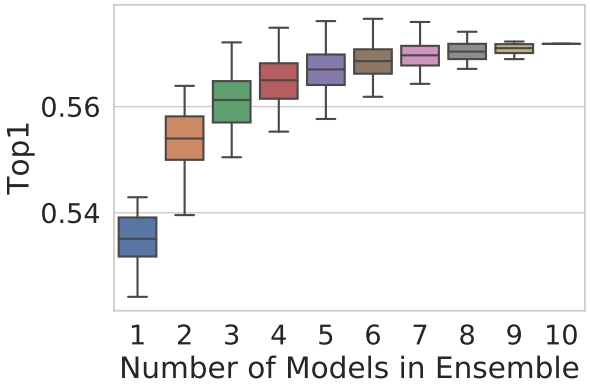
(a) CrossDocked2020 Pearson R from various model ensembles



(b) CrossDocked2020 RMSE from various model ensembles



(c) CrossDocked2020 AUC from various model ensembles



(d) CrossDocked2020 Top1 from various model ensembles

Figure 3.15: Ensembles of Default 2018 models trained on CrossDocked2020. There are diminishing returns after an ensemble of 5 models is used.

each pose (Figure 3.15). We observed diminishing returns after five models were utilized in the ensemble. As such we then evaluated a five model ensemble for both the Def2018 and Dense architectures on CrossDocked2020, the PDBbind General set, and the PDBbind Refined set.

We observed a small performance gain for both our Def2018 and Dense architectures, as shown in Table 3.6 and Figure 3.14. In all cases, an ensemble of models has equal or superior performance to the individual model. Interestingly, the best performance gain was observed with the Dense architecture on CrossDocked2020. A potential explanation is that the much

larger number of parameters of the Dense net could have been overfit to the training data, as ensembling is an effective method for compensating for overfitting.[113] This actually achieved our best performing model on CrossDocked2020, as the ensemble of Dense models achieved a Top1 of 0.684, AUC of 0.956, Pearson R of 0.612, and RMSE of 1.42.

3.4 Expanding Binding Affinity Data

The previous section of work demonstrated the utility of expanding the available binding pose data through the creation of the CrossDocked2020 dataset. However, this only addresses half of the problem. In this subaim, we address the other half of the problem: expanding the available binding affinity data through the imputation of the missing labels with our CNN.

3.4.1 Model Architecture, Dataset, and Training Procedure

We utilize the Def2018 model architecture that was described in the previous subaim for all experiments in this subaim. Contrary to the data presented above, these experiments were performed with version 1.3 of the CrossDocked2020 dataset. This version upgrade fixed several receptor structures that were flattened in the original dataset (version 1.1), and fixed several ligands that had the double bonds removed from their aromatic rings (version 1.2). Lastly, version 1.3 fixed an issue where multiple ligands on different chains would be downloaded into the same file, several misaligned receptor and ligand structures, and culled redundant entries in the Pocketome database. In total CrossDocked2020v1.3 contains 2,900 binding pockets, consisting of 17,815 pocket-ligand pairs, and a total of 22,566,449 poses and is the latest version of CrossDocked2020. We utilized the same pocket-ligand clustering procedure described in subaim 1 in order to perform 3-fold CCV splits for all of our experiments. Finally, we also utilized the exact same model training procedure as subaim

Dataset	Model	RMSE	R
General (CCV)	CNN (With Receptor)	1.65	0.56
General (CCV)	Gradient Boosted Trees (Descriptors)	1.63	0.54
General (CCV)	Random Forest (Descriptors)	1.65	0.52
General (CCV)	CNN (Without Receptor)	1.72	0.52
General (CCV)	Decision Tree (Descriptors)	1.69	0.50
General (CCV)	KNN (Descriptors)	1.70	0.50
General (CCV)	SVM (Descriptors)	1.69	0.49
General (CCV)	Lasso (Descriptors)	1.70	0.48
CrossDocked (CCV)	CNN (With Receptor)	1.47	0.58
CrossDocked (CCV)	CNN (Without Receptor)	1.58	0.49
CrossDocked (CCV)	Gradient Boosted Trees (Descriptors)	1.82	0.31
CrossDocked (CCV)	Random Forest (Descriptors)	1.82	0.30
CrossDocked (CCV)	SVM (Descriptors)	1.92	0.28
CrossDocked (CCV)	KNN (Descriptors)	1.86	0.25
CrossDocked (CCV)	Lasso (Descriptors)	1.86	0.24
CrossDocked (CCV)	Decision Tree (Descriptors)	1.93	0.23

Table 3.5: Comparison of CNN models trained with and without receptor information and a variety of models trained with simple chemical descriptors. R and RMSE values are the mean across the ensemble.

Train	Test	Model	Evaluation	RMSE	R	AUC	Top1	BP	Rand
CrossDock	CCV	Dense	Average	1.55	0.547	0.944	0.615	0.970	0.0321
			Ensemble	**1.42**	**0.612**	**0.956**	**0.684**	0.970	0.0321
CrossDock	CCV	Def2018	Average	1.47	0.577	0.906	0.537	0.970	0.0321
			Ensemble	1.45	0.587	0.914	0.574	0.970	0.0321
General	Core	Dense	Average	1.490	0.733	0.942	0.788	0.946	0.135
			Ensemble	**1.348**	0.788	**0.960**	**0.836**	0.946	0.135
General	Core	Def2018	Average	1.383	0.787	0.943	0.802	0.946	0.135
			Ensemble	1.368	**0.796**	0.946	0.814	0.946	0.135
Refined	Core	Def2018	Average	1.503	0.720	0.932	0.766	0.946	0.135
			Ensemble	1.438	0.749	0.941	0.800	0.946	0.135

Table 3.6: Effect of using an ensemble of models compared to average of individual model performance. BP: Best possible fraction of low RMSD poses; Rand: expected fraction of randomly sampled low RMSD poses.

Imputation Type	Pocket-Ligand Grouping	Only Low RMSD Poses
Individual	No	No
Individual Ensemble	No	No
Median Ensemble	Yes	No
Median Good Only Ensemble	Yes	Yes
Max Ensemble	Yes	No
Max Good Only Ensemble	Yes	Yes
Min Ensemble	Yes	No
Min Good Only Ensemble	Yes	Yes

Table 3.7: This table shows how a given experimental approach imputes a given pocket-ligand complex. The imputation types marked with "Individual" create an imputed binding affinity label for every pose. The other imputation types select a single imputed label for every pose for a given pocket-ligand complex. The label selection is done by taking the median, maximum, or minimum of the imputed labels for either all poses or only the low RMSD poses of the pocket-ligand complex.

1 in order to train the models for the experiments in this subaim.

3.4.2 Experimental setup

For each experiment we trained 5 models with different random seeds on the 3-fold CCV splits. The general training schema is: 1) Train and evaluate an initial model ignoring any missing labels, 2) Use the trained model and a selected imputation type to impute the missing labels, and 3) Train and evaluate a new model on all of the data (including the imputed labels). Steps 2 and 3 of this training scheme can be repeated as many times as you like. For our Individual and Individual Ensemble imputation types, we repeated steps 2 and 3 until the performance on the test set no longer improved. Each of the imputation types is listed in Table 3.7.

The first imputation type that we used is the simplest: treat each binding pose as a different example and utilize the raw predictions of the trained model (Individual). Notably, this results in each seeded model having a distinct training set from one another. Thus we end up with five different sets of distinct labels for each pose that required imputation, with

each subsequent model only seeing 1 of these label sets. The second imputation type, similar to Individual, treats each binding pose as a separate example. However, this time we take the ensemble mean of our five model's predictions as the imputed label (Individual Ensemble). This allows for the same training set to be utilized for each of our random seeds on training subsequent generations of models and produces a singular distinct binding affinity label for each pose requiring imputation. It was unclear whether this behavior of having a singular pocket-ligand complex having many imputed labels was desirable, so we also investigated other imputation types.

For the remaining imputation types, we stored the predicted label of every pose for a given pocket-ligand complex (rows with pocket-ligand grouping in Table 3.7). We then took as our imputation label either the max, median or minimum of these stored labels. In contrast to the second imputation type, this approach results in a single value being utilized for every pose of a particular pocket-ligand complex. This matches how we treat the binding affinity labels in CrossDocked2020, where every pose of a pocket-ligand complex has the same binding affinity label.

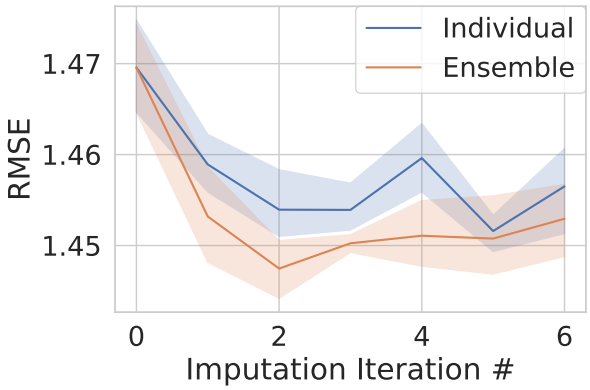
There is a potential flaw in this approach as well. Namely, we are storing the predicted binding affinity values for poses that we know are low quality (> 2Å RMSD to the crystal pose). As demonstrated in the prior subaim, we know that our CNN models produce binding affinity predictions that are pose-sensitive. Ergo, it could be that including the predicted labels from poor quality poses could have a negative effect on our imputed label selection. Thus, we also investigated only storing the predicted binding affinities from poses that we know are < 2Å RMSD from the crystal pose, and then taking the median, max, or minimum of these "Good Only" poses as our imputed label (rows with "Only Low RMSD Poses" in Table 3.7). We evaluate all of our models on the same test set folds, which have no imputed binding affinity labels, utilizing the same metrics as described in subaim 1.

3.4.3 Imputation Improves Model Performance

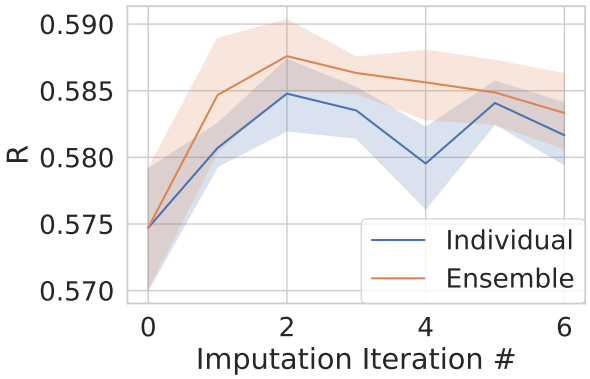
In our first experiment, we sought to determine if imputing missing labels improves our CNN's ability to predict receptor-ligand binding affinity. For this experiment we selected the first two imputation types (Individual and Individual Ensemble) for their simplicity in producing imputed labels. Figure 3.16 shows that utilizing imputed labels during training indeed improves the model's performance on binding affinity prediction. Maximal performance gains were achieved after 2 rounds of imputation. Notably, even though the binding pose classification training data remains unchanged, we observed that the imputed labels also provided a small improvement at the binding pose classification tasks (Figure 3.16C,D).

In subaim 1 we demonstrate that the Def2018 CNN architecture produces binding affinity predictions that are pose dependent. For this experiment, while the pose label is unchanged, we are now supplying a unique imputed binding affinity label to many poses in the training data. During the training procedure, the loss is a combination of the affinity loss and the classification loss. So by adding the imputed binding affinity labels, we supply more information to the model as now *every* pose has both the binding affinity loss and the classification loss.

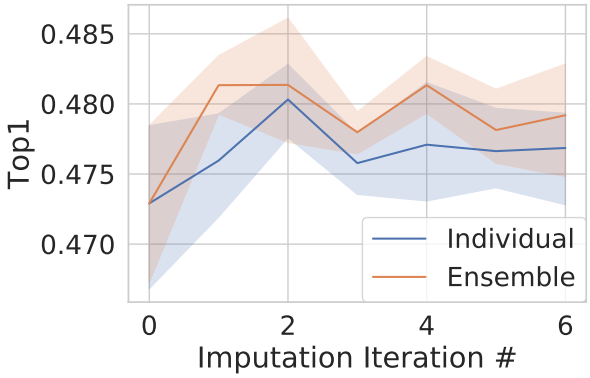
However, it is unclear if utilizing a different imputed label for every pose is the best approach for performance on the binding affinity regression task. In contrast to this experiment, when training with experimental binding affinities, every pose has the same binding affinity label. If we trust our model's ability to predict protein-ligand binding affinity accurately, then it does not make sense to introduce extra noise in the form of label variation when training on imputed labels.



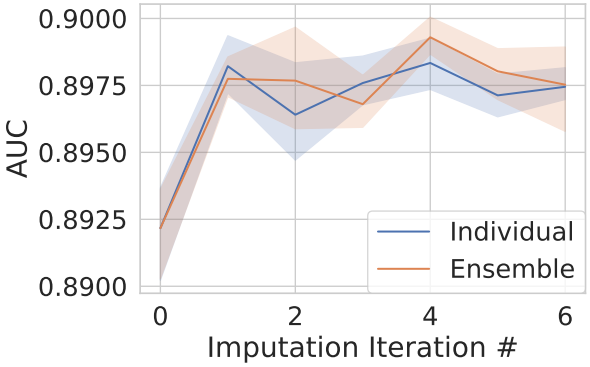
(a) Binding affinity RMSE improves with the addition of imputed labels. Lower is better.



(b) Binding affinity Pearson's R improves with the addition of imputed labels. Higher is better.



(c) Binding pose Top1 improves with the addition of imputed labels. Higher is better.



(d) Binding pose AUC improves with the addition of imputed labels. Higher is better.

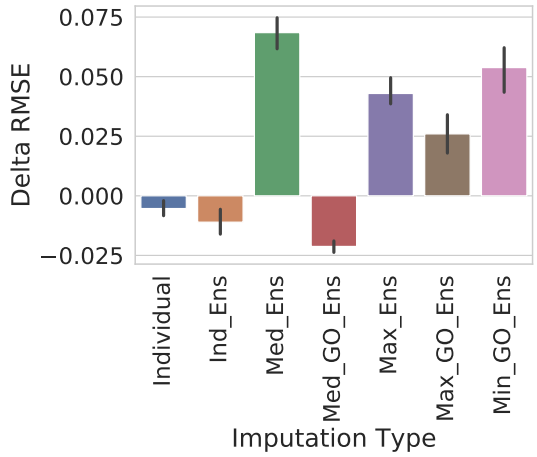
Figure 3.16: Adding imputed binding affinity labels to the training set provides a small improvement to all predictive tasks. We show the results of six iterations of data imputation and model retraining on affinity labels from CrossDocked2020v1.3. At each data point we plot the mean of 5 models trained with different random seeds. The colored area is the 95% confidence interval around the mean calculated via bootstrapping in *seaborn*. The blue line shows the results of five different random seeds (Individual Table 3.7). The orange line shows an ensemble approach, taking the mean of the five models as the imputed label of every pose (Individual Ensemble Table 3.7).

3.4.4 Restricting Imputation to Low RMSD Poses Further Improves Model Performance

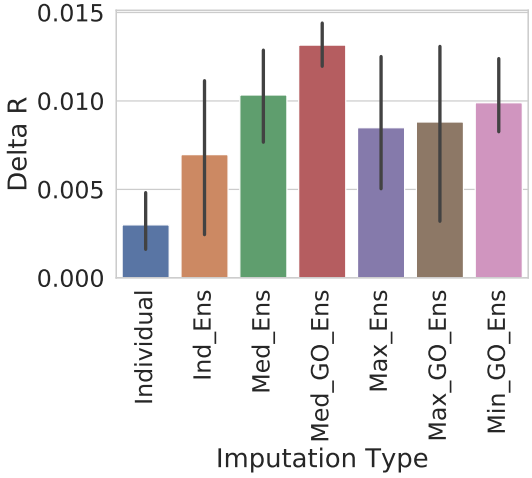
In order to address this question we investigated three different approaches to selecting a single imputed label for all poses of a pocket-ligand pair: the median, maximum, or minimum. We also investigated the effect of this calculation using all possible poses, or only considering the imputed labels from low RMSD poses. This resulted in 6 different imputation types for analysis. In Figure 3.16 we demonstrate that an ensemble mean for the imputed labels outperformed individual imputation, so we only investigated the ensemble mean of 5 models to generate the imputation labels for each of these new imputation types. After generating the new training data, we trained five new models with different seed on the new dataset. We then compared the difference between models trained with this singular round of imputation and models trained without imputed data in Figure 3.17.

All of these new imputation types improved the model's performance on binding affinity Pearson's R, but only the "Median Good Only Ensemble" improved the binding affinity RMSE (Figure 3.17A,B). This imputation type also had the best performance gain on the binding affinity regression task. So, we performed another round of imputation label generation using the "Median Good Only Ensemble" imputation type, similar to the previous experiment. The results of this extra round of imputation are shown in Figure 3.18. Again, we observe a small additional performance gain from this second round of imputation label generation. Due to the performance gain being small and the results of Figure 3.16, no additional rounds of imputation were performed.

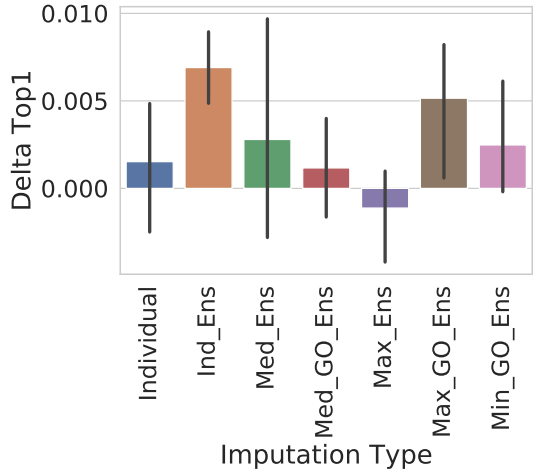
We note that while the "Median Good Only Ensemble" provided the best results for the binding affinity prediction task, it performed relatively poorly on the binding pose classification task (Figure 3.17). The "Individual Ensemble" performed the best at having the top-ranked pose be low RMSD (Top1) and was the only approach to achieve a statistically significant improvement between the models trained with imputation and those without



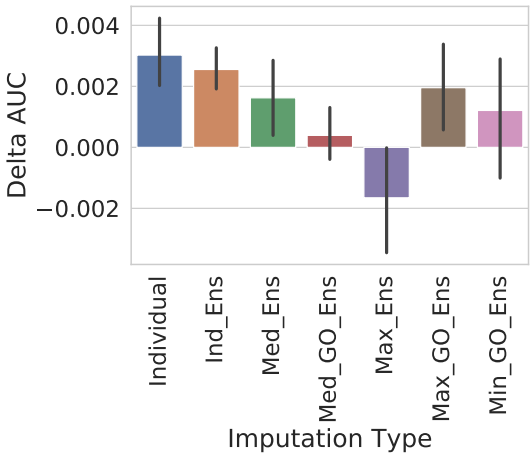
(a) Change in binding affinity RMSE relative to no imputation with a variety of imputation styles. Lower is better.



(b) Change in binding affinity Pearson's R relative to no imputation with a variety of imputation styles. Higher is better.



(c) Change in pose classification Top1 relative to no imputation with a variety of imputation styles. Higher is better.



(d) Change in pose classification AUC relative to no imputation with a variety of imputation styles. Higher is better.

Figure 3.17: Comparing different binding affinity imputation styles. The performance metrics for five models with different seeds trained with each imputation style were subtracted from the mean performance of training without imputation. The error bar is the 95% confidence interval calculated via bootstrapping in *seaborn*. For each plot, a bar corresponds to a singular imputation style. *Ind* is short for Individual, and *Ens* is short for Ensemble. The first two styles (blue and orange) are the same as used in Figure 3.16. For the rest of the styles, we select one number for each pocket-ligand pair, either by the median (Med), maximum (Max), or minimum (Min). Styles marked with *_GO* only utilize the imputations from good poses. The Min_Ens results were omitted, due to performing so poorly that they re-scaled the plots (Delta RMSE 1.674, Delta R -0.157, Delta Top1 -0.0175, and Delta AUC 0.00219). The Student's T test for each of these values is reported in Table 3.8.

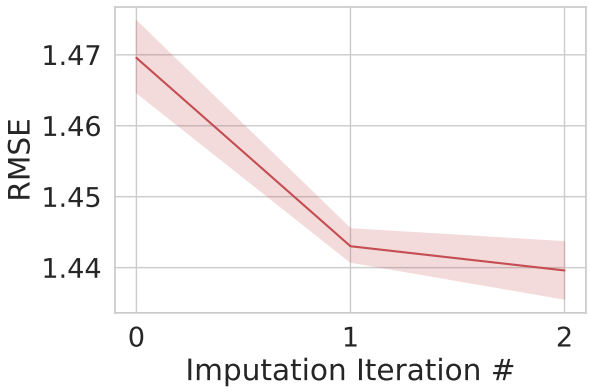
Imputation Type	RMSE	R	AUC	Top1
Individual	**0.0144**	0.0595	**0.000747**	0.459
Individual Ensemble	**0.00482**	0.253	**0.000755**	**0.0433**
Median Ensemble	**1.07e-6**	**0.00234**	**0.00477**	0.391
Median Good Only Ensemble	**3.43e-5**	**0.000304**	**0.0131**	0.485
Max Ensemble	**2.29e-5**	**0.00863**	0.352	0.914
Max Good Only Ensemble	**0.00712**	**0.0144**	**0.00429**	0.130
Min Ensemble	**3.14e-13**	**1.54e-10**	**0.00488**	**0.00358**
Min Good Only Ensemble	**4.27e-5**	**0.00202**	**0.0218**	0.324

Table 3.8: Student's T-test p-values for the difference between 0 and 1 round of imputation for each of the methods. Numbers in bold are < 0.05. This table corresponds to the data utilized to generate Figure 3.17. Notably, for the binding affinity RMSE, every method results in a statistically significant difference as compared to not performing the imputation. This is not true for all the other metrics, though generally imputation results in a statistically significant difference for binding affinity Pearson's R and AUC while generally failing to produce a statistically significant difference for Top1.

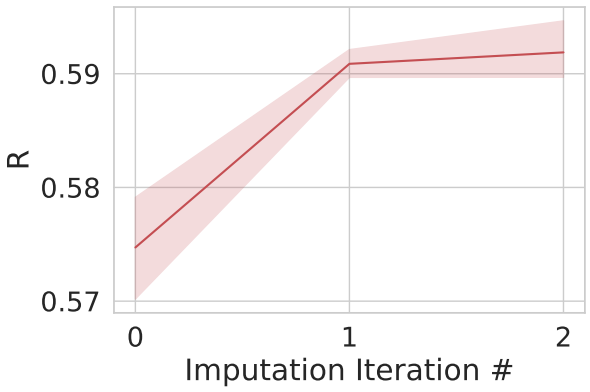
(Table 3.8). Both the "Individual" and "Individual Ensemble" imputation types resulted in the best performance gain for AUC.

3.4.5 Balancing Imputed and Known Labels Maximizes Model Learning

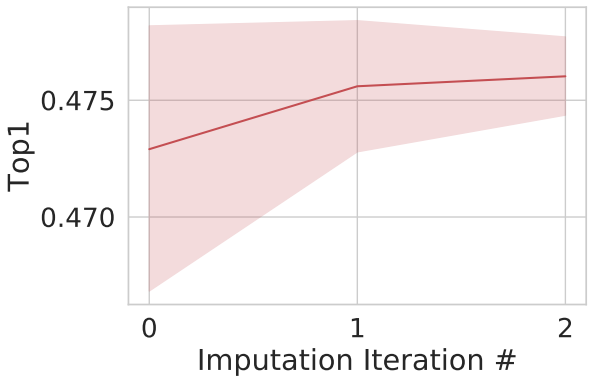
The majority of CrossDocked2020 (60%) is missing a binding affinity label. Thus, we theorize that it is potentially harmful to have a majority of the data available for training have an imputed label. We characterized the effect of gradually adding more imputed labels to the training set for the "Median Good Only Ensemble" imputation type. This was performed by first randomly splitting the imputed labels into 5 chunks (e.g. each chunk had 20% of the imputed labels). We then created a series of training sets by randomly selecting a chunk to add without replacement. This resulted in a total of five sets with growing numbers of imputed labels containing 20%, 40%, 60%, 80%, and then 100% of the imputed labels respectively. For each of these training sets, we again trained and evaluated five models with



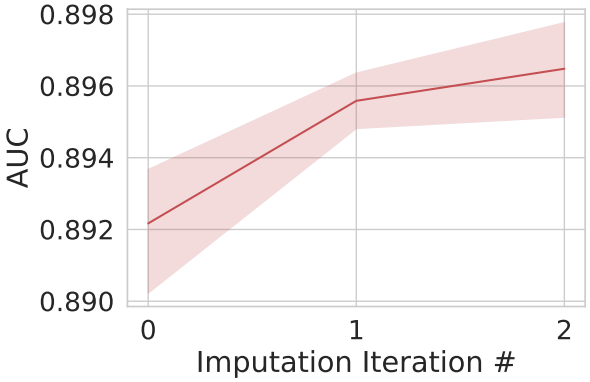
(a) Binding affinity RMSE improves when utilizing an ensemble of the median predicted affinity for imputation. Lower is better.



(b) Binding affinity Pearson's R improves when utilizing an ensemble of the median predicted affinity for imputation. Higher is better.



(c) Binding pose Top1 improves when utilizing an ensemble of the median predicted affinity for imputation. Higher is better.



(d) Binding pose AUC improves when utilizing an ensemble of the median predicted affinity for imputation. Higher is better.

Figure 3.18: Performance metrics of our best imputation approach, taking the ensemble mean of the median predicted binding affinity for each pocket-ligand complex good pose, for binding affinity regression. The 0th point on the line is the model results after training on the original dataset. We then used that model to generate the imputed labels, and utilized them to train the model for the 1st data point. Said model was then used to generate the imputed labels for the second datapoint's model's training. For each point five models with different seeds were trained from scratch. The shaded area is the 95% confidence interval of the mean calculated via bootstrapping in *seaborn*.

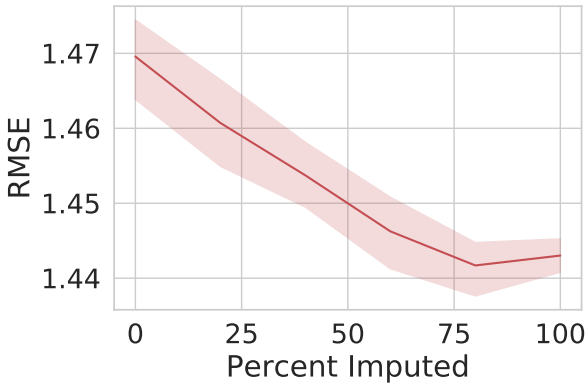
different random seeds (Figure 3.19).

We observe a general trend of improvement as more imputed data is added, with improvement plateauing at the inclusion of 80% of the imputed labels. This plateau is especially interesting as 80% of the imputed binding affinity labels corresponds to having 47.2% of the data having an imputed label and 41.1% having a known binding affinity label (with the remaining data being unlabeled). Though not explicitly tested for, this result implies that maximal improvement is achieved with a balance of imputed and known labels during training.

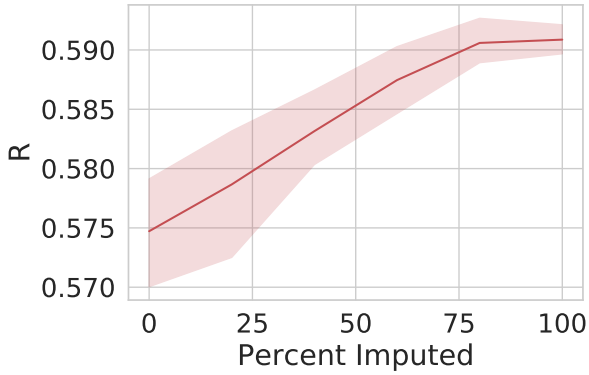
3.5 Conclusion

In subaim 1 we present the CrossDocked2020 dataset for training structure-based machine learning models and rigorously evaluate the performance of several CNN architectures on CrossDocked2020 to serve as a benchmark for the community. We first demonstrate that our CNN architectures achieve similar performance to other published methods on the common PDBbind dataset (Table 3.4), although exact comparisons are complicated by differences in test set selection. In particular, our best performing single model (Def2018 trained on the General set) achieves a Pearson R of 0.79 which is similar to another grid-based CNN KDeep which achieved a Pearson R of 0.82[25]. This is especially impressive because KDeep has 1,340,769 parameters, which is about triple the amount of parameters we use in Def2018.

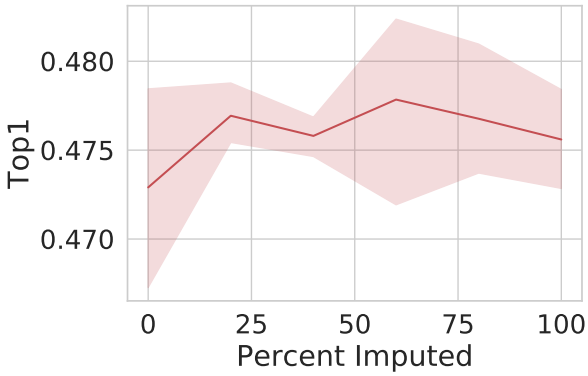
We further demonstrate, consistent with conventional wisdom that ML models struggle to extrapolate beyond their training domain, our CNN models have trouble generalizing for both the pose classification and binding affinity regression tasks. First, for pose classification, models trained to predict binding affinity with only crystal structures as input fail to identify low RMSD docked poses (Figure 3.4), despite their good performance on crystal poses (Figure 3.2). Training on docked poses and jointly training binding affinity prediction and



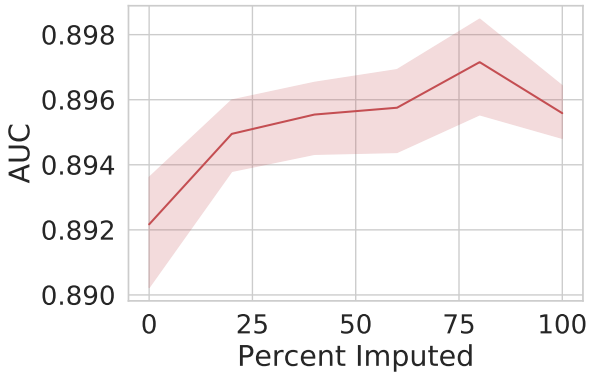
(a) Binding affinity RMSE improves as imputed data is added to the training set. Lower is better.



(b) Binding affinity Pearson's R improves as imputed data is added to the training set. Higher is better.



(c) Binding pose Top1 exhibits a small improvement with imputed data during training. Higher is better.



(d) Binding pose AUC improves as imputed data is added to the training set. Higher is better.

Figure 3.19: Effect on metrics as a function of successively adding more imputed binding affinities to the training set. Each plot is showing the results of five models with different seeds, being trained on successively more of the imputed binding affinity labels. The shaded area is the 95% confidence interval of the mean calculated via bootstrapping in *seaborn*. Shown are no imputed labels, to all of the imputed labels, in increments of 20%. The imputation generation procedure is the ensemble mean of the median predicted binding affinity from good poses only from Figure 3.17.

pose selection does not improve affinity prediction performance (Figure 3.2), but does make the affinity prediction model pose sensitive (Figure 3.3). Interestingly, we note that training upon docked poses is insufficient for a model to perform well when incorporated into a pose sampling strategy unless a counterexample generation procedure was utilized when training the models (Figure 3.10).

We demonstrate similar struggles for the binding affinity regression task. The first piece of evidence is the substantial performance drop for our binding affinity metrics when changing the train-test split from train on General/Refined and test on Core to CCV splits (Figure 3.6). Additionally, the minimal importance of the receptor structure (Figure 3.12) and the relative success of affinity prediction using high RMSD poses (Figure 3.3) strongly suggest that these models will not generalize to new chemotypes.

Traditional protein-ligand scoring functions typically struggle to balance performance on pose selection and binding affinity prediction[114–118]. Our models help to join efforts on these two tasks by having both the affinity prediction and pose selection modules share most of their computation (all but the last fully connected layer, see Figure 3.1) but remain distinctly computed outputs. Our models' affinity predictive power is unaffected by the inclusion of negative poses in the training data (Figures 3.3 and 3.9). Consistently, through our evaluation of swapped test sets, we observe that the performance losses are more severe for the pose prediction task than the affinity prediction task (Figures 3.9, 3.10, 3.2, 3.8, and 3.12). Notably, in Figure 3.12, we observe that models trained with receptor information perform worse on ligand-only test sets than the same architecture trained on ligand-only data. This demonstrates that the model predictions are dependent on the receptor input. Importantly the inclusion of receptors during training improves binding affinity prediction (Pearson R of 0.577 instead of 0.487), and our CNN ligand-only models outperformed the ligand-only models trained on simple predictors (Table 3.5). So, while it is true that ligand-only information is a substantial contribution to the affinity prediction performance, model performance on CrossDocked2020 is improved through the combination of ligand structural and receptor

information.

We were surprised by our ligand-only models achieving better than random AUC on pose classification. Unlike binding affinity prediction, upon which cheminformatic methods are routinely successful, pose classification should be entirely dependent on the receptor structure. We thought that there would be no relevant information about the pose of a ligand when the corresponding receptor structure was missing. However, due to the construction of our datasets, each ligand can generate a different number of low RMSD docked poses. So, a ligand only model could learn this 'dockability' index for the ligands in the training set and use that to group all poses of a given ligand together. It is unclear if this is a useful prior to learn, since such a prior can produce more confident scores for ligands that are easier to dock. Luckily, the effect could be eliminated by resampling the training set such that every receptor-ligand complex has the same number of low and high RMSD poses.

This type of problem underlies the importance of the underlying distribution of poses when comparing pose classification performance between two models. Both the AUC and Top1 metrics are highly dependent on the construction of the test set. For example, a random classifier will have an expected AUC of 0.5, but its Top1 will depend on the average fraction of low RMSD poses available for each ligand. It is also trivial to inflate the AUC by the inclusion of trivial-to-predict high RMSD poses, while those same poses would leave the Top1 metric unchanged. A concrete example of the difficulty is comparing our PDBbind-based models to the graph-based model of Lim et al.[87]. Their model achieves an AUC of 0.968, higher than any of our models, but also exhibits a Top1 of under 0.5 which is substantially worse than our models (Figure 3.5). Since we do not have access to the exact poses utilized by Lim et al.[87] to train their model, we do not know if this Top1 result is actually a poor result or not. It could be the case that for half of their test set a low RMSD pose was never sampled, in which case a Top1 of 0.5 is perfect performance. For this reason, we made sure to make available the exact splits utilized for training all of our models.

Lastly, when evaluating our pose selection criteria, we observed that selecting the pose

with the highest CNNaffinity can recover most of the pose predictive performance when compared to selecting a pose based on the highest CNNscore for the complex (Figure 3.4). We suspect that this behavior is due to our model training procedure. During training the binding affinity loss behaves differently depending on if the pose is low RMSD or not. Namely, low RMSD poses are penalized for *both* under and over-predicting the binding affinity. In contrast, high RMSD poses are only penalized for *over-predicting* the binding affinity. This imbues the CNN predicted binding affinity with some amount of pose predictive power, since the low RMSD poses are more strongly penalized for being incorrect.

In subaim 2 we demonstrated the imputed binding affinity labels improve model performance at predicting receptor-ligand binding affinity and potentially improve binding pose classification for the the CrossDocked2020 dataset. Through the investigation of several imputation approaches utilizing our CNN model, we can suggest best practices for imputation in different settings: ensemble-based approaches perform better (Figure 3.16), two rounds of imputation generation achieves maximal performance (Figures 3.16,3.18), and a roughly equal number of imputed labels and known labels achieves optimal performance gains (Figure 3.19).

Notably, our results only investigated utilizing a model trained with CrossDocked2020 for the imputation. We know that our training data is not representative of chemical space, and our imputed labels could just be further reinforcing the biases of CrossDocked2020 during training. A potential solution to this problem would be to utilize a different model that is trained on a different, larger dataset (e.g. ChEMBL[119]). Such a model setup could potentially provide more useful imputed binding affinity labels. However, there are several initial challenges to such an approach: 1) selecting a different training dataset without having leakage into CrossDocked2020, 2) selecting an appropriate input representation that works for both the new dataset and CrossDocked2020, and 3) selecting a new model architecture for this task. We also only investigated relatively simple approaches (median, maximum, and minimum) to generating a single imputed label for a protein-ligand pair. It is entirely possible that a more sophisticated approach, such as SICE[97] or maximum likelihood estimation, could

provide a better label. However, these approaches would require a model that can output its confidence in its predictions, which is outside the score of our current Def2018 CNN architecture.

Additionally, we observed that imputing a unique binding affinity for each pose (Individual and Individual Ensemble) resulted in a small, but statistically significant improvement on the pose classification task (Figure 3.17, Table 3.8). This is not necessarily surprising due to our model's training procedure. In the Def2018 architecture, the CNN produces a 6x6x6x128 feature tensor as input into two separate fully connected networks in order to produce the binding pose classification and binding affinity regression respectively. By including imputed binding affinities that are different for every pose we allow the binding affinity hinge loss to affect all of the poses in the training set, instead of only being applied to the 40% of the data with a known binding affinity. This can allow the CNN to gain some additional information to help in its pose-classification task.

This work provides both a more realistic and challenging dataset in CrossDocked2020, and a completely comparable baseline of CNN model performance on CrossDocked2020 through the publication of the exact poses utilized to train our datasets. Our initial work on including imputed labels provides a ground work to explore other techniques to further advance model performance. Taken together both of these data expansion techniques improve the performance of our CNNs as compared to training on more traditional datasets and provide a completely comparable baseline for which other model architectures can be fairly compared towards.

3.6 Declarations

The first subaim of this chapter is adapted from:

Francoeur, P., Masuda, T., Sunseri, J., Jia, A., Iovanisci, R., Snyder, I. and Koes,

D. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of Chemical Information and Modeling* **2020**. DOI: 10.1021/acs.jcim.0c00411

The second subaim of this chapter is adapted from:

Francoeur, Paul G and Koes, David R. Expanding Training Data for Structure-Based Receptor-Ligand Binding Affinity Regression through Imputation of Missing Labels. *ACS Omega* **2023** DOI: 10.1021/acsomega.3c05931

These were all first author papers where I performed the vast majority of the experiments, performed all of the analysis, and wrote the manuscripts with editing help from the other contributing authors. Notable exceptions are Dr. Tomohide Masuda who developed and tested the Dense architecture utilized in subaim 1, Dr. Jocelyn Sunseri who developed libmolgrid and pioneered the lab's initial CNN models for protein-ligand scoring, Richard Iovanisci and Ian Snyder who did the ensemble analysis presented in subaim 1, and Andrew Jia who performed the sub-sampling analysis of CrossDocked2020.

4.0 Real Application

4.1 Summary

The PFN1-actin binding interaction is involved in increasing angiogenesis and is implicated in both renal cell carcinoma and wet age-related macular degeneration. Previous work identified a small molecule, C2, that disrupted this protein-protein binding interaction through an unknown mechanism and required a high concentration of compound to be effective. We performed whole protein docking of C2 to actin and PFN1 to identify potential binding sites and used them to screen for new potential drugs. Our convolutional neural networks identified a novel small molecule, C74, that targets the PFN1-actin binding interaction. C74 performs similarly to C2 in renal cell carcinoma cell proliferation and migration assays at half the concentration of C2. This study provides a real world success of deploying the models developed in prior sections of this thesis on a real drug discovery problem.

4.2 Introduction

Angiogenesis is a highly regulated cellular process wherein new blood vessels are formed. This process is fundamental for vascular expansion during development and healing, and aberrations in angiogenesis is implicated in both renal cell carcinoma and wet age-related macular degeneration (AMD). Renal cell carcinoma is among the 10 most common cancers in both men and women, with an estimated incidence of 81,800 and number of deaths of 14,890.[120] The most common subtype, clear-cell renal cell carcinoma (ccRCC), occurs in over 75% of patients. Notably, about 20-30% of ccRCC patients present with metastases at

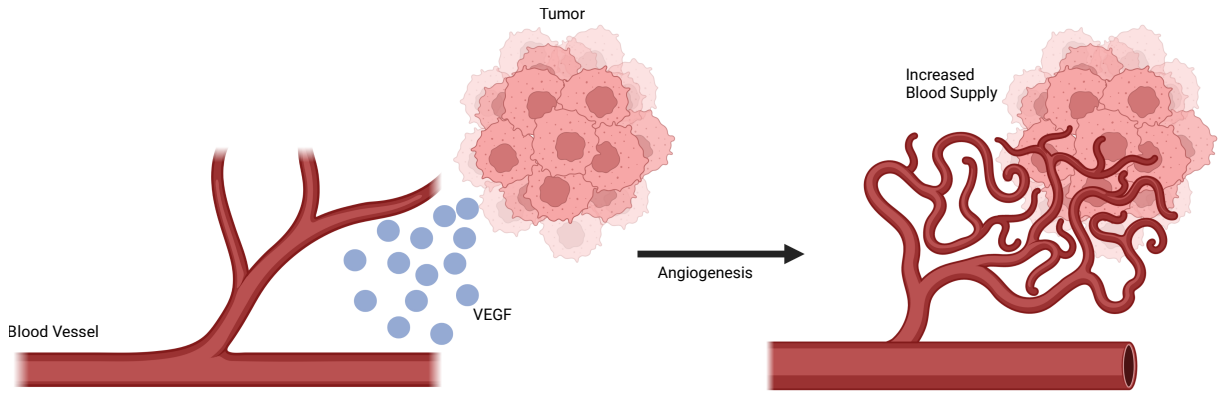


Figure 4.1: Cartoon showing that the secretion of VEGF leads to the highly vascularized tumor microenvironment of ccRCC. Figure made with biorender.com.

diagnosis, with another one-third of patients developing either local recurrence and/or distant metastases following initial treatment.[50] Additionally, the 5-year survival rate of patients with advanced-stage ccRCC is 14%.[121] ccRCC commonly presents a highly vascularized tumor microenvironment arising from the upregulation of vascular endothelial growth factor (VEGF).[50] A cartoon of VEGF secretion is shown in Figure 4.1. VEGF is a rational target for treatment of ccRCC, since blocking VEGF signalling will stop the tumor from recruiting more blood vessels through angiogenesis, and thus it will starve and/or stop growing. Indeed, many patients initially respond to therapies targeting VEGF, however virtually all of them develop progressive, drug-refractory disease.[122–124]

During normal VEGF function, PFN1 is eventually upregulated to increase actin polymerization, which in turn leads to angiogenesis (Figure 4.2). When VEGF signalling is blocked, several compensatory pathways are upregulated which also lead to the same upregulation of PFN1 and angiogenesis occurring even with VEGF signalling suppressed(Figure 4.2). In the literature PFN1 has been demonstrated to be a key protein in both the angiogenesis pathway and in regulating actin dynamics.[51–55] PFN1 upregulation has also been identified as a marker of late stage ccRCC.[57,58] Thus, there is evidence that PFN1 has potential to be a more effective target than VEGF to inhibit angiogenesis in the treatment of ccRCC.

On the other hand, wet AMD only affects 10-15% of AMD patients, but accounts for nearly

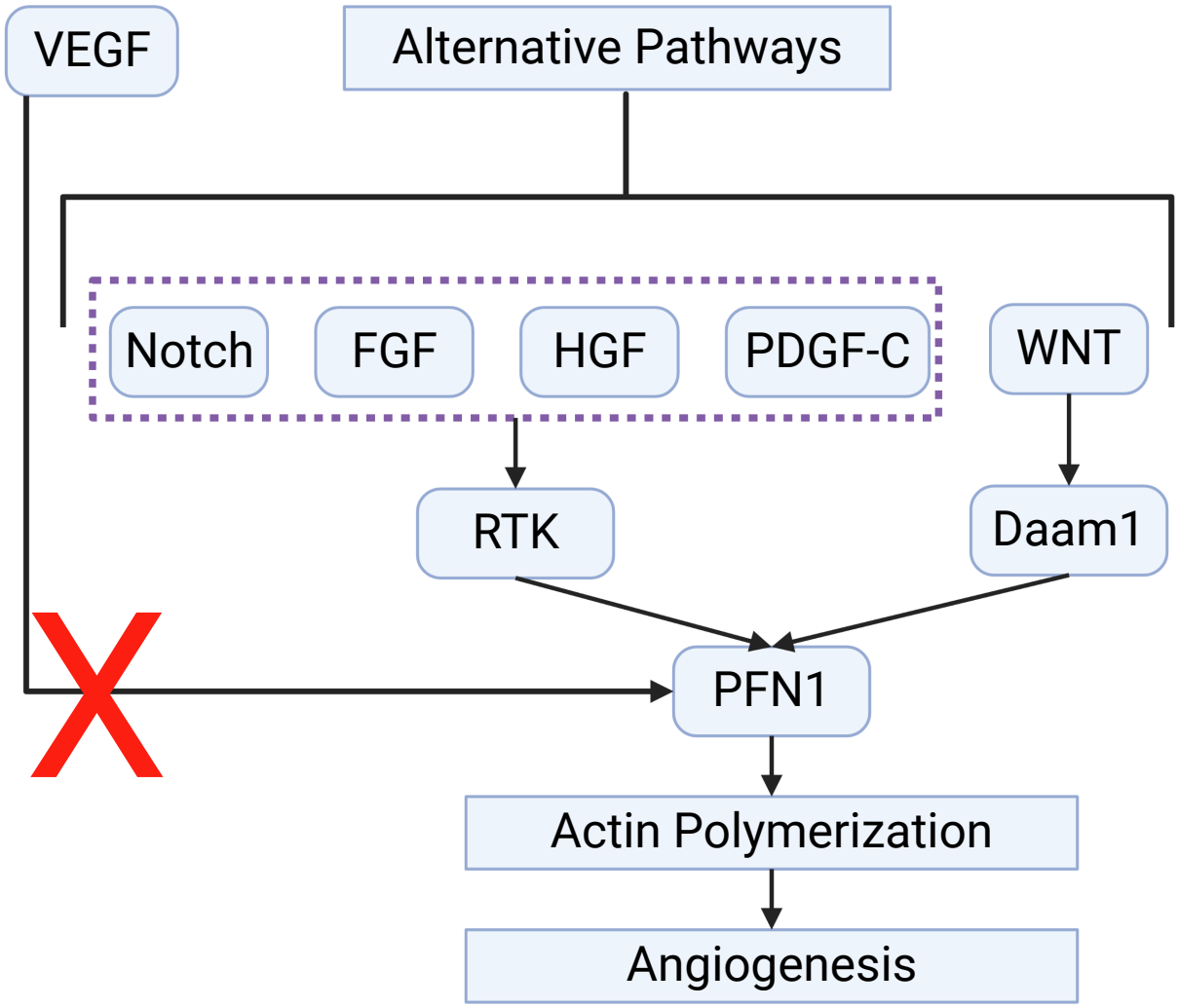


Figure 4.2: Cartoon of the compensatory pathways that induce angiogenesis when VEGF signalling is blocked. Figure made with biorender.com.

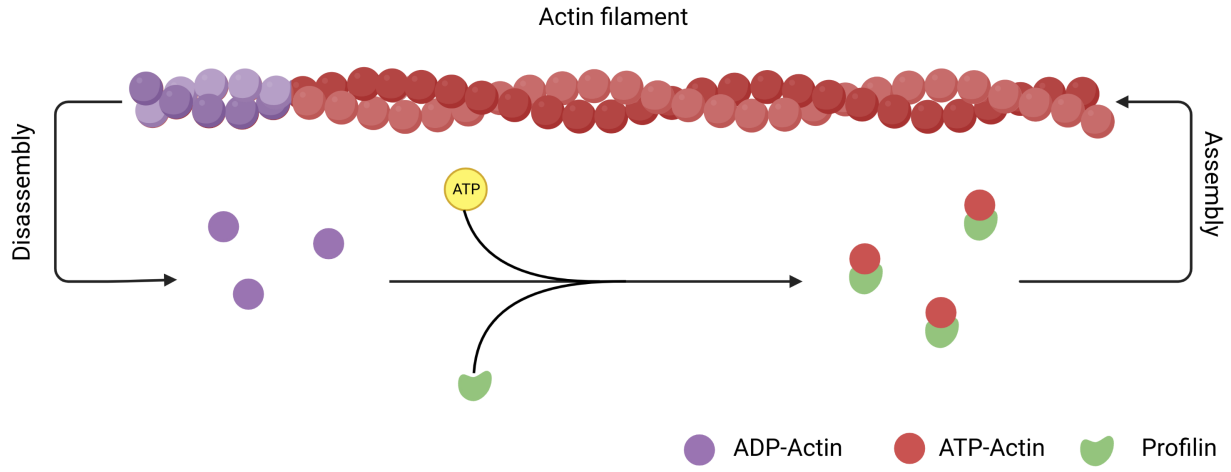


Figure 4.3: Cartoon showing PFN1's role in actin polymerization. Figure made with biorender.com

all of AMD-related vision loss[125,126]. Disease progression is characterized by the invasion of leaky choroidal neo-vessels into the retina, which results in rapid vision loss[127]. This invasion occurs when pro-angiogenic growth factors (e.g. VEGF) induce tip cell differentiation in a sub-population of vascular endothelial cells (VEC), which guide the vascular outgrowth. Intravitreal injection of anti-VEGF agents has been shown to be an effective treatment in wet AMD models by diminishing the vascular growth and leakage[128,129]. Again, a substantial number of patients acquire resistance to the VEGF-based treatment, due to the side stepping previously outlined.

PFN1 has also been shown to be transcriptionally upregulated in both the retinal VEC in proliferative diabetic retinopathy patients and the oxygen-induced retinopathy mouse model, which causes blindness in a similar mechanism to wet AMD[59]. Suppressing PFN1 has also been shown to reduce migration, proliferation, and angiogenic ability of VECs *in vitro* and *ex vivo*.[130–132] Thus, there is also evidence that PFN1 has potential as a target to treat wet AMD.

Both ccRCC and wet AMD are commonly treated by VEGF-based treatments and suffer from the developments of resistance to the treatments. There is ample evidence that PFN1 could provide the same anti-angiogenic treatment benefits while avoiding the VEGF

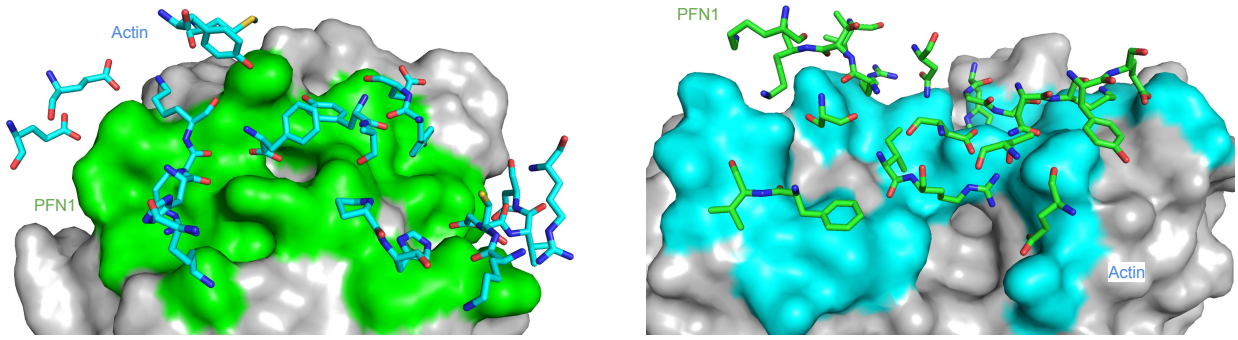
sidestepping mechanisms in both disease states. We propose targeting the PFN1-actin complex to perform this task. PFN1 binds to actin in the cytosol and recruits it to the leading edge of the actin filament to enable actin polymerization (Figure 4.3). Notably, designing a drug to target a protein-protein interaction is difficult. Luckily, Gau et al.[60] has already identified a small molecule, C2, that disrupts the PFN1-actin binding interaction *in vitro*. However, it was only effective at a concentration of 50-100uM, which is too high for a commercial drug.[60] Furthermore, the exact mechanism by which C2 disrupts PFN1-actin binding is unclear. This makes using C2 as a lead molecule challenging, as we do not know what interactions it is making with which protein. This in turn means that we do not understand which properties of the C2 molecule can be improved, nor which parts of the C2 molecule are essential to its function. So in order to improve upon the binding affinity of C2, we must first hypothesize potential binding sites for the molecule.

4.3 Methods

Here we describe the whole protein docking procedure utilized to identify potential ligand binding sites on PFN1 and actin. We also describe the pharmacophore based virtual screen that was utilized in each potential binding site.

4.3.1 Whole Protein Docking

The PFN1-actin binding interaction is along a relatively flat section of both proteins (Figure 4.4). This surface is devoid of traditional binding pockets, which in turn makes utilizing off the shelf binding pocket detection algorithms challenging. Additionally, the large area of the protein-protein interaction surface further increases the challenge of designing a small molecule to disrupt the two proteins from forming a complex, since it is unclear which



(a) PFN1 surface of the PFN1-actin complex. PFN1 residues within 5 angstrom of actin are colored.

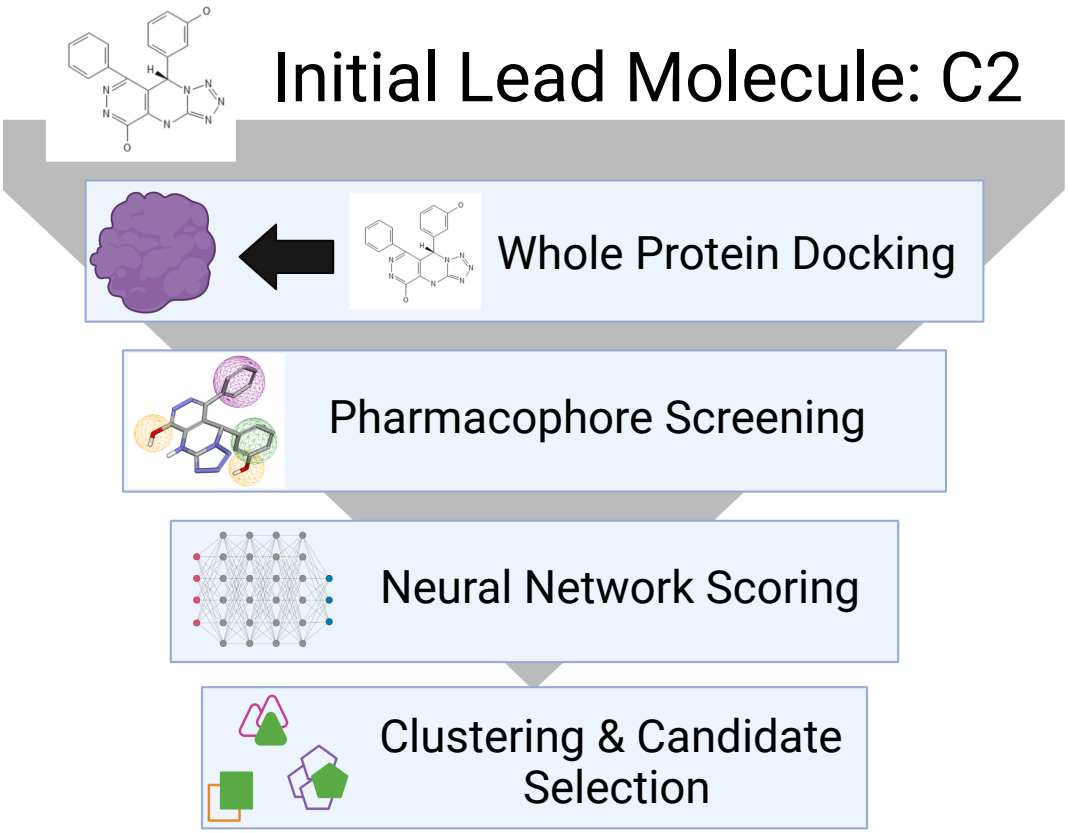
(b) Actin surface of the PFN1-actin complex. Actin residues within 5 angstrom of PFN1 are colored.

Figure 4.4: The binding surfaces of both PFN1 (green) and actin (blue) in the PFN1-actin complex.

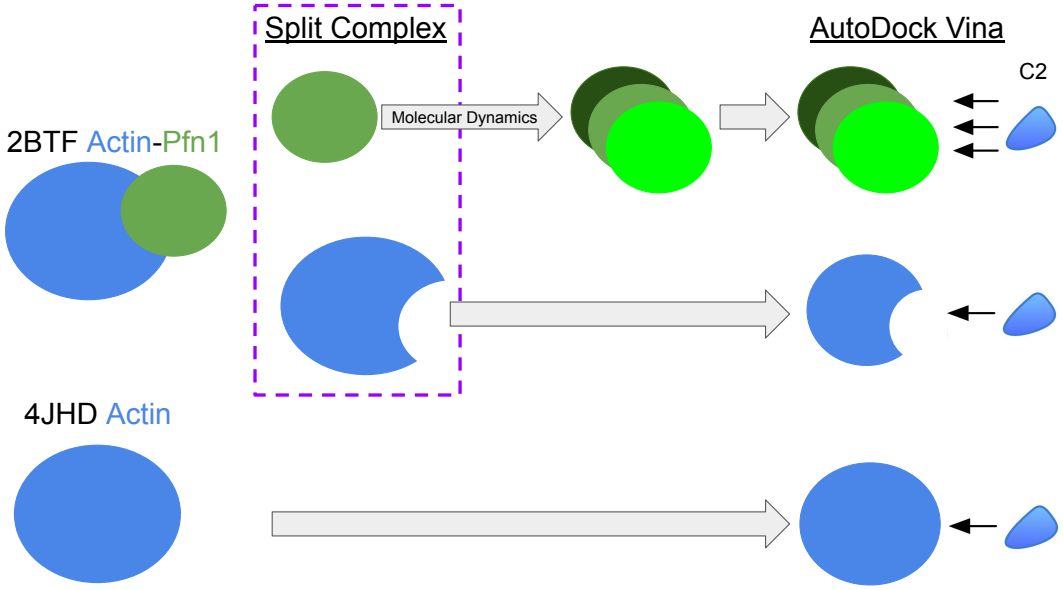
interactions are important for complex formation. However, we have a compound that is known to disrupt the binding interaction in C2. We utilized whole protein docking with C2 to identify potential binding sites on both actin and PFN1.

Molecular docking refers to a set of algorithms that are utilized to predict the binding orientation of a ligand to a target receptor. The output of a molecular docking procedure is a set of ranked conformations and, usually, the predicted binding affinity of the ligand to the given receptor.[133–135] These procedures consist of two parts: sampling new potential poses, and scoring said pose. Sampling refers to an extensive search of the conformational space of the molecules being docked. This space is large since both the receptor and the ligand are flexible, so the receptor is often kept rigid to reduce the search space. The scoring function determines the fitness of a given conformation to the receptor and is used to rank the generated conformations in order of their likelihood of being correct.

Typically, the search space of molecular docking is restricted to a known binding site. However, our binding site is unknown. Therefore we expand the search space of the docking algorithm to encompass the entire receptor. Accordingly, we must increase the exhaustiveness of the search in order to adequately sample our search space. We can identify potential binding sites via the locations on the protein surfaces where C2 scores highly in the docking procedure. A diagram of the procedure is shown in Figure 4.5b.



(a) The computational pipeline to identify small molecules to target the PFN1-actin complex.



(b) The procedure for the processing of the actin and PFN1 structures for the whole protein docking step in A).

Figure 4.5: Diagrams depicting the methodology of the drug discovery project.

There are 3 potential states where the compound could bind: 1) free actin, 2) the PFN1-actin complex, or 3) free PFN1. We obtained the PFN1-actin complex through the PDB entry 2BTF and the free actin complex through the PDB entry 4JHD. We desire our compound to bind to free PFN1, so to maximize the chances of finding a druggable pocket at the PFN1-actin binding interface, we performed a 100ns molecular dynamics simulation with Amber18[136] utilizing the ff15ipq force field and TIP3P water on the PFN1 structure extracted from 2BTF. We then selected the 3 most diverse conformations, measured by backbone RMSD, for whole protein docking. For each of our receptors, we docked C2 utilizing *smina*[104] with exhaustiveness 50 and the autobox_ligand parameter set to the entire receptor. The results from this series of whole protein docking experiments were then utilized to identify potential binding sites, from which we performed virtual screens of potential molecules.

4.3.2 Virtual Screening

Virtual screening refers to a series of *in silica* tools that filter chemical compound libraries to identify those most likely to bind to a specific target.[137] A common approach to performing virtual screening is to use pharmacophore-based models to query large chemical libraries for compounds with specific properties. The International Union of Pure and Applied Chemistry defines pharmacophores as "the ensemble of steric and electronic features that is necessary to ensure the optimal supra-molecular interactions with a specific biological target structure and to trigger (or block) its biological response."[138] The theory behind pharmacophore models is that if a new compound maintains the chemical functionalities and relative spatial arrangement of a known binder, the new compound will also have biological activity against the target.[137]

The most common pharmacophore features are: hydrogen bond acceptors; hydrogen bond donors; hydrophobic areas; positive charges; negative charges; aromatic; and metals. Additionally, it is common to include a shape or exclusion volume, which represent the

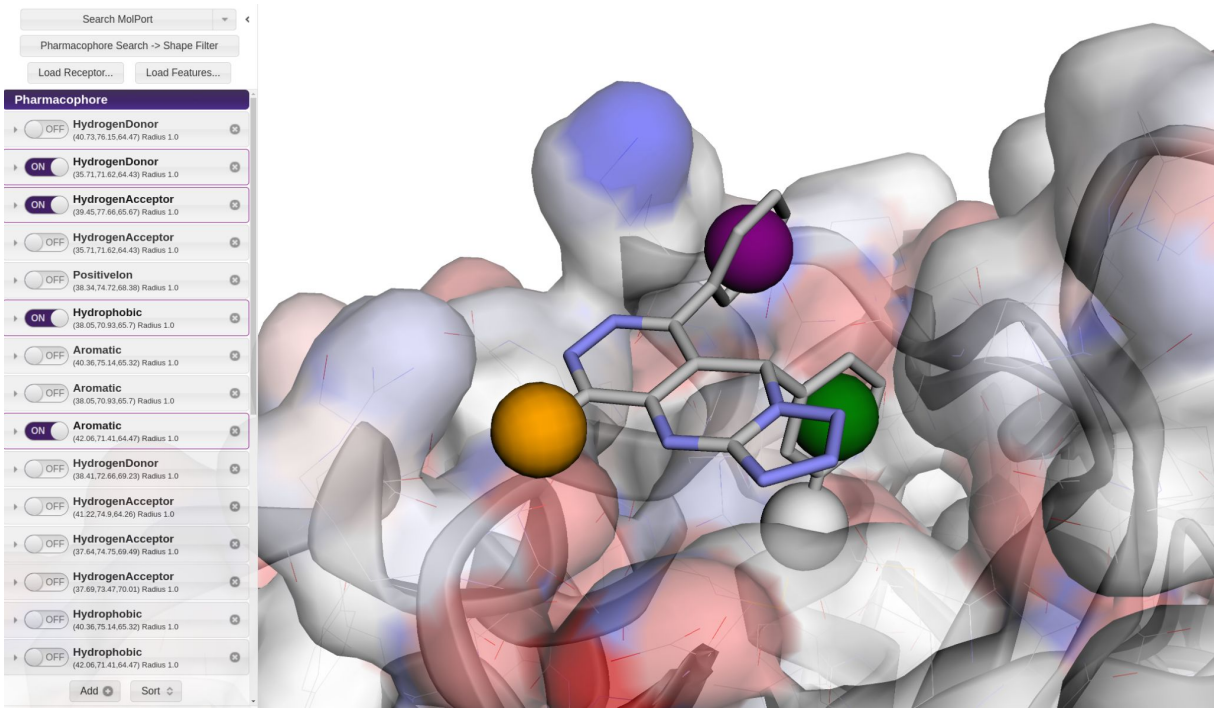


Figure 4.6: Pharmit pharmacophore query for the actin-PFN1 site 1 binding pocket. This was the input that resulted in the identification of C74.

size/shape of the binding pocket. These features are represented as geometric entities (e.g. spheres). Notably, pharmacophores can be derived from the binding pocket alone, potential ligands alone, or a combination of the two. For each of the binding sites identified in the previous section, we utilized pharmit[139] to characterize the highest scoring conformation of C2 to generate potential pharmacophores. The resulting pool of pharmacophores were then hand selected to a smaller set that were then utilized to screen MolPort for purchasable compounds as the first step in our virtual screen (Figure 4.6).

Each of these searches return a large number of potential molecules which need to be narrowed down to a reasonable number of candidates for experimental verification. To do so, we employ two methods: 1) rescoring the energy minimized hit pose from pharmit with AutoDock Vina and *gnina*, and 2) reperforming the energy minimization using *gnina*.[140] The resulting scores from these methods were then clustered to produce our candidate molecules. In particular, at this stage of the selection process, we used ML models that were developed

in the earlier part of this work. Thus, a functional hit molecule that was selected via our ML models provides a real world example where the deployment of our ML methods was successful.

4.4 Results

In order to identify potential binding sites for small molecules, we utilized whole protein docking of C2 to propose five potential small molecule binding sites targeting the PFN1-actin binding complex. Four of the sites are along the PFN1-actin binding surface, with the final site being the actin ATP binding site (Figure 4.7). For each binding site we ran a pharmacophore search of MolPort resulting in an initial pool of 128,331 molecules across the identified potential binding sites. We then scored the matches in each binding site utilizing our CNN models developed in Aim 2 and the AutoDock Vina scoring function, and clustered the candidate molecules according to their molecular similarity. We then selected 67 candidate molecules from these clusters for experimental validation. Of these candidate molecules, one, C74, was experimentally verified to disrupt PFN1-actin binding and reduce RCC cell proliferation *in vitro* and tumor growth *in vivo*.

4.4.1 Binding Site Identification

Actin monomers, stripped of non-protein atoms, were extracted from PDB entries 2BTF and 4JHD. Similarly, we extracted the PFN1 monomer from 2BTF. We then selected the 3 most diverse conformations of a 100ns molecular dynamics simulation of the PFN1 monomer as described above. We then performed whole protein docking of C2 to these 5 structures using *smina* with *exhaustiveness* set to 50. From this, we were able to identify 5 potential binding sites (Figure 4.7). Included in these potential sites is the predicted binding site of

Pocket	Number of Matching Molecule Poses	Number of Unique Molecules
Actin ATP Site	18,870	8,264
Actin-PFN1 Site	82,223	33,643
Profilin Site 1	67,639	33,281
Profilin Site 2	135,187	63,795
Profilin Site 3	19,095	11,308
Total		128,331

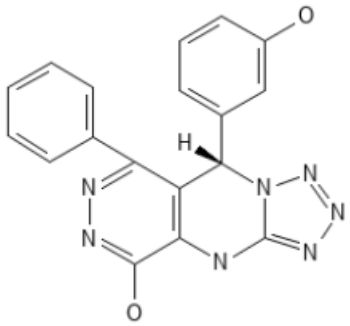
Table 4.1: Total number of MolPort molecules matching the pharmacophore search for each potential binding site. The middle column contains the total number of ligand poses identified in each pocket. The right column counts the number of unique molecules identified for each pocket.

C2[60] (PFN1-actin site 3) and the actin ATP binding site. In particular, the actin ATP site is on the other side of the protein from the PFN1-actin binding surface. The other 4 potential sites are along the PFN1-actin binding surface.

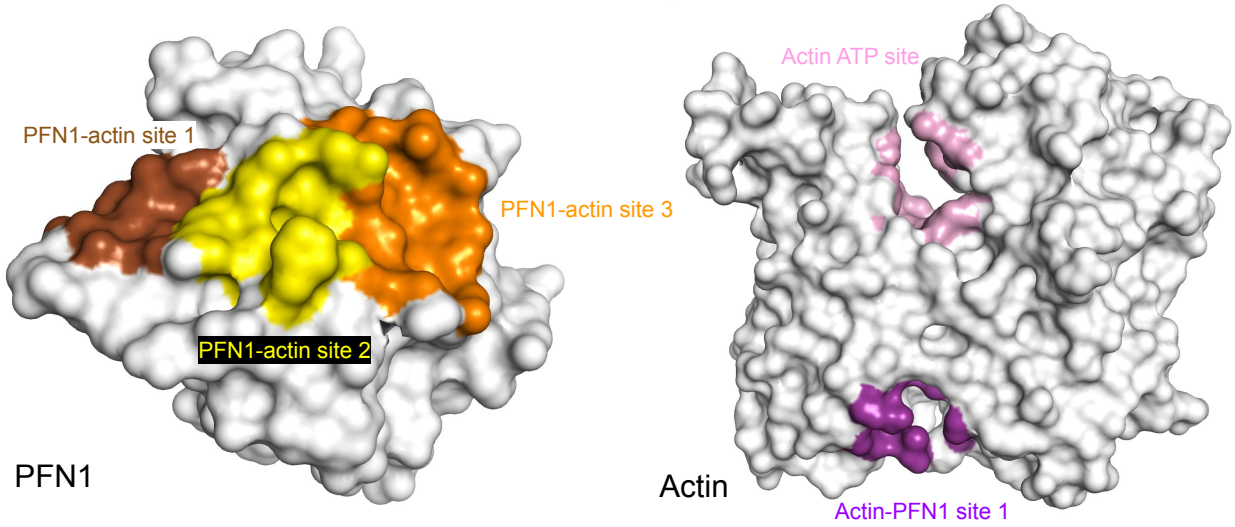
4.4.2 CNN Virtual Screening Results

After obtaining our potential binding sites, we then had to generate a pharmacophore model for each site. This was performed with the *pharmit* website, using the potential pocket as the receptor and the highest scoring C2 pose in said pocket as the ligand. Since we do not know which pharmacophores are important for C2's activity, we cannot tell which pharmacophores are important to keep. Thus, we elected to use a minimal set of broad pharmacophores to return a large number of molecules and rely on downstream scoring to identify good candidates. In order to do this we manually adjusted the selected pharmacophores until less than 200,000 molecules in MolPort matched (Figure 4.6). The total number of matching molecules are listed in Table 4.1.

The matched molecules from *pharmit* were minimized with *smina*[104] with Autodock Vina scoring to provide their initial pose. The Vina score was one of the methods we utilized to rank the molecules. We also re-ran the minimization utilizing *gnina*, our internal CNN model, which classifies the binding poses as being near native and predicts the binding affinity. The



(a) 2D representation of the docked molecule, C2.



(b) The possible binding spots on PFN1 and actin.

Figure 4.7: Identified potential binding sites on actin and PFN1 through whole protein docking of C2. The docking was performed using *smina* with *exhaustiveness* set to 50. We obtained 5 potential binding sites for further analysis in the virtual screening pipeline. Note that in the actin-PFN1 complex, the actin-PFN1 site 1 (purple) interacts with PFN1-actin site 2 (yellow).

CNN predicted binding affinity was the other metric we utilized to rank the molecules. We clustered the molecules utilizing the *cluster_mols* plugin[141] of PyMol with its default settings. For each cluster, we identified the top-ranking molecule by the Vina score or CNN predicted binding affinity. This process resulted in 67 candidate molecules for experimental validation.

4.4.3 ccRCC Experimental Validation

Each of our proposed 67 molecules was first tested with a pyrene-actin polymerization assay to determine if they would inhibit actin polymerization.[60] In the pyrene-actin polymerization assay, as time passes actin can polymerize. During this elongation process, the fluorescent intensity increases. For our purposes, PFN1 binds to actin which inhibits the polymerization of the actin filaments relative to an experiment with actin alone, since the actin is interacting with PFN1 instead of other actin to polymerize. Ergo, if we introduce a compound that inhibits PFN1-actin binding, we would expect the fluorescence curve of acting+PFN1+compound to be similar to the curve produced by actin alone. Of the 67 proposed molecules, only 7 molecules passed this first test (Table 4.2). The performance of C74 is shown in Figure 4.8. Notably, C74 was a molecule that was selected by having a high predicted CNN score in the actin-PFN1 binding site (purple site in Figure 4.7). The predicted binding pose of C74 to actin is shown in Figure 4.9). Furthermore, C74 at a dose of 100 micromolar inhibits actin polymerization mediated by PFN1 (Figure 4.8).

C74 was then compared to C2 in a cell proliferation and migration assay. The cells were treated with 50 micromolar of C2, or 10, 25, or 50 micromolar of C74. The results are shown in Figure 4.10. C74 performs at about half the concentration (25 micromolar) of C2 (50 micromolar) on both cell proliferation and cell migration assays.

Lastly, C74 was also tested in subcutaneously implanted RENCA cells *in vivo*. The results of this study are shown in Figure 4.11. After 19 days of treatment the C74 treated animals had less tumor growth than the control animals. Collectively, these results show that a small

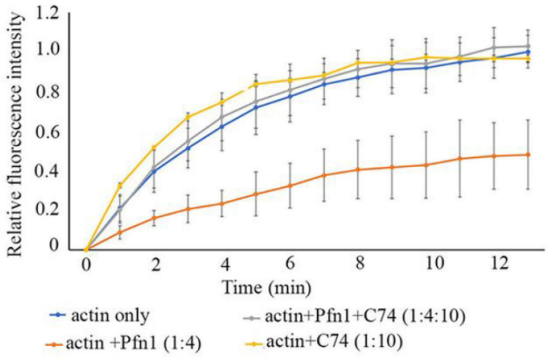


Figure 4.8: Pyrene-actin polymerization assay curves for C74. Each time point is the mean plus or minus the standard deviation of the fluorescence relative to the maximum recorded florescence of actin alone. The numbers in parentheses indicate relative concentrations of actin, PFN1, and C74. The actual concentrations of actin and PFN1 are 10 and 40 micromolar respectively. C74 was utilized at 100 micromolar. This data was generated by David Gau and Jordan Sturm.

Compound Identifier	MolPort ID	Inhibits Actin	Inhibits PFN1:Actin
C60	MolPort-007-600-121	Yes	No
C63	MolPort-000-481-426	Yes	No
C64	MolPort-000-481-100	Yes	No
C66	MolPort-002-622-882	Yes	No
C73	MolPort-000-778-708	No	Yes
C74	MolPort-000-793-534	No	Yes
C76	MolPort-019-793-213	No	Yes
C98	MolPort-004-271-775	No	Yes
C99	MolPort-002-295-702	No	Yes
C107	MolPort-029-999-390	No	Yes
C108	MolPort-028-585-829	No	Yes

Table 4.2: Table summarizing the results of the pyrene-actin polymerization assay for the proposed 67 compounds. Italics indicates that the compound inhibits actin polymerization by interacting with actin rather than profilin. Compounds in bold have the desired result of disrupting the actin-pfn1 complex while not preventing actin from polymerizing. This data was generated by David Gau.

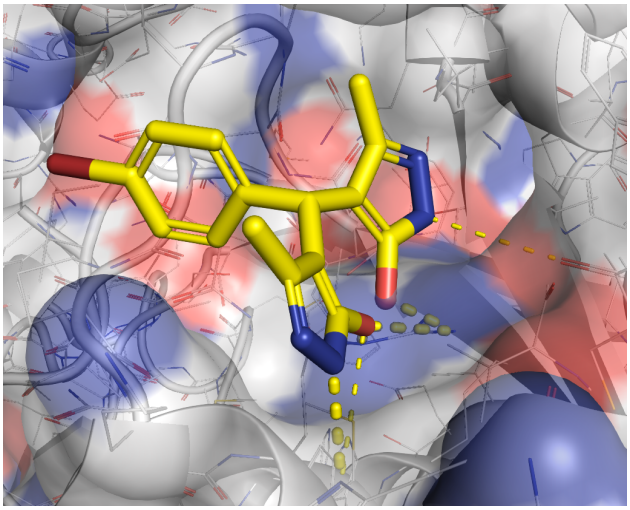


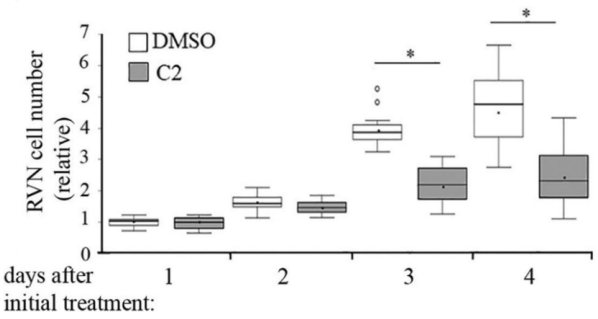
Figure 4.9: CNN predicted binding pose for C74 to actin. This is the actin-PFN1 site (purple color in Figure 4.7). Favorable polar contacts are shown in the yellow dotted lines. Notably, this predicted pose was produced by the gradients of our CNN models, and resulted in sterically clashing oxygen atoms which is almost certainly incorrect.

molecule inhibitor of the PFN1-actin interaction can decrease the aggressiveness of RCC cells *in vitro* and *in vivo*.

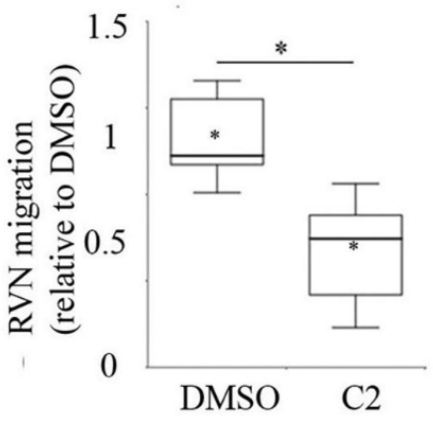
4.4.4 Eye Neovascularization Experimental Validation

C74 was additionally tested for use in preventing angiogenesis in the eye. There were two cell lines utilized in the *in vitro* experiments: immortalized human dermal micro-VEC cell lines (HdmVEC; source: ATCC, CRL-3243), and primary human retinal micro-VEC (HrmVEC; source: Cell Biologics, ACBRI181). *Ex vivo* experiments were performed with choroids harvested from 3-week old C57BL/6 mice. Lastly *in vivo* experiments were performed on 8-week old C57BL/6JRj mice.

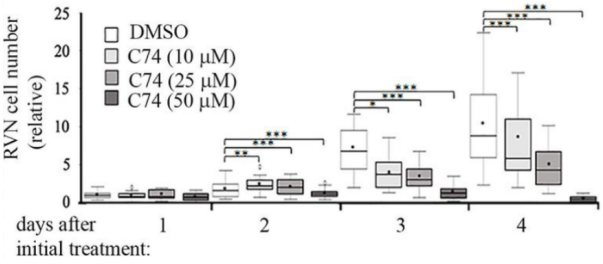
We will first describe the *in vitro* experiments. C74 was first tested with a single-cell migration assay (Figure 4.12A). HdmVEC cells were sparsely plated in a 24-well plate coated with type I collagen (Millipore) and subjected to overnight treatment with DMSO (control), 25uM C74, or 50uM C74. The next day, the cell culture was replaced and time-lapse images of



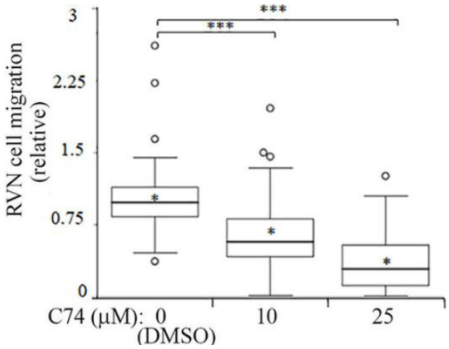
(a) A series of boxplots showing the effect of 50 micromolar dosage of C2 versus DMSO on the proliferation of VHL-negative RVN cells.



(b) Boxplot summarizing the effect of C2 versus DMSO treatment on serum-induced chemotactic migration of RVN cells. These are the results of 3 independent experiments.

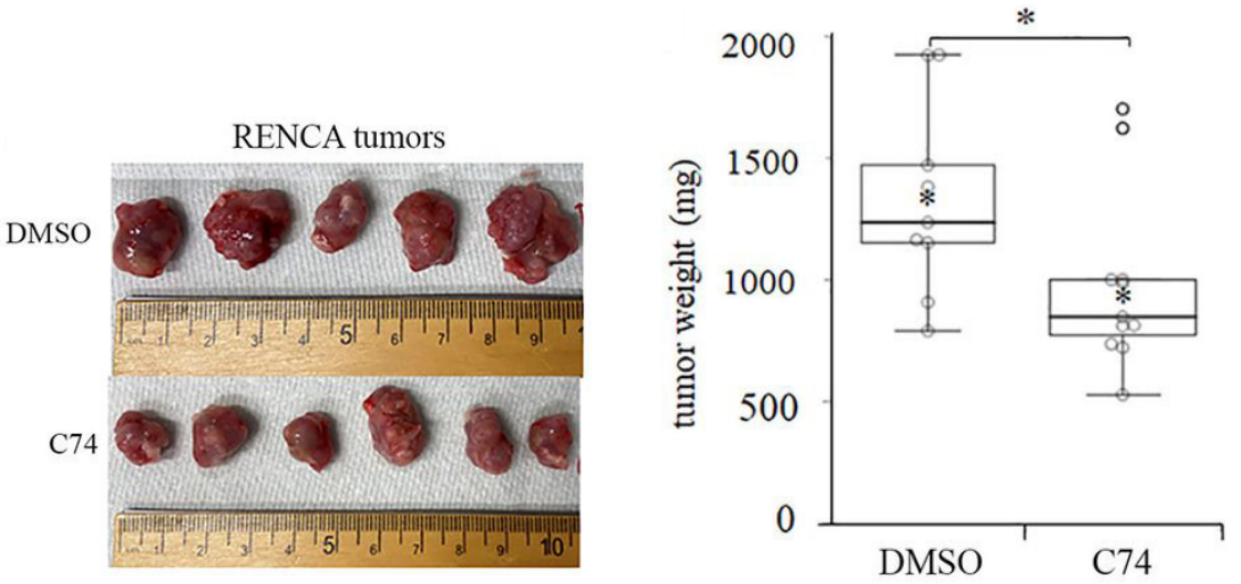


(c) Boxplots showing the effect of a series of dosages of C74 versus DMSO on the proliferation of VHL-negative RVN cells.



(d) Boxplot summarizing the effect of C74 versus DMSO treatment on serum-induced chemotactic migration of RVN cells.

Figure 4.10: A and C shows the results of cell proliferation assays upon treatment with C2 and C74 respectively. Notably, we observe that C74 shows a mild response at 10 micromolar, and a significant response at 25 micromolar. This is a considerable improvement over C2 needing 50 micromolar. B and D show the results of a cell migration assay upon treatment of C2 and C74 respectively. C74 again achieves a similar response to C2 at 25 micromolar instead of 50 micromolar. This data was generated by David Gau and Abigail Allen.



(a) Representative tumors of the subcutaneously implanted RENCA cells.

(b) Boxplot of the harvested tumors from the C74 treated mice (n=10) compared to the untreated mice receiving DMSO (n=9).

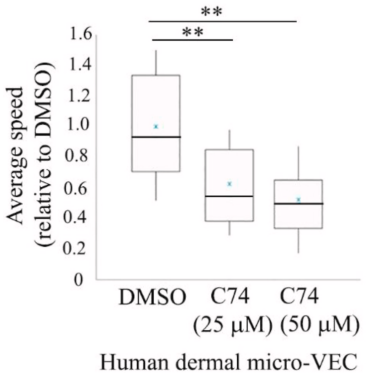
Figure 4.11: Treatment via C74 reduces RCC cell proliferation *in vivo*. Mice were treated with an intraperitoneal injection of either 16mg/kg C74 or DMSO daily over a period of 19 days. The C74 treated animals had significantly smaller tumors on average (978.8mg) than the DMSO treated mice (1327.5mg). This data was generated by David Gau.

the cells randomly migrating was taken with the 10x objective on an Olympus IX-71 inverted microscope for 120min with a 1min time interval using the CellSens software. The centroid of the cell nucleus was tracked frame-by-frame with ImageJ, and the average migration speed was calculated (total centroid distance travelled / total time). C74 treatment resulted in an 40% and 50% decrease in average speed for the 25uM and 50uM treatments respectively (Figure 4.12A).

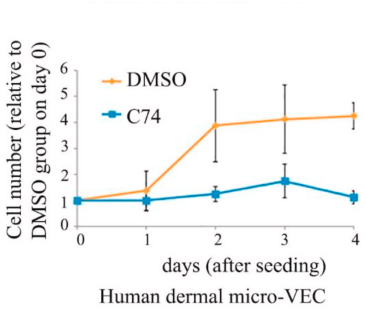
C74 was then tested for its effect on a cell proliferation assay (Figure 4.12B). 5000 HdmVEC cells were plated in duplicate on a 24-well plate and cultured overnight. After the culture, cells were subjected to DMSO (control) or 25uM C74 on day 1. The cells were then trypsinized and counted daily up to day 4 (3 days of treatment total). The medium was replenished every other day with the appropriate treatment. It is clear that C74 treatment has a robust anti-proliferative effect on the HdmVEC cell (Figure 4.12B).

C74 was also tested in a chord morphogenesis assay (Figure 4.12C). In this assay, HrmVEC are plated atop of polymerized Matrigel and allowed to adhere overnight. The following day, cells were treated with DMSO (control) or 25uM C74 for 16 hours. Phase-contrast microscopy was utilized to take pictures of the cells, where total cord length was quantified with the Angiogenesis plugin of ImageJ. 25uM treatment of C74 was sufficient to reduce the total cord length/field by 32% (Figure 4.12C). Lastly, a commercial live/dead staining kit (Life Technologies) was utilized to ensure that C74 was not cytotoxic to the cells. Virtually 100% of cells were viable even after an overnight treatment of up to 50uM of C74 (Figure 4.13).

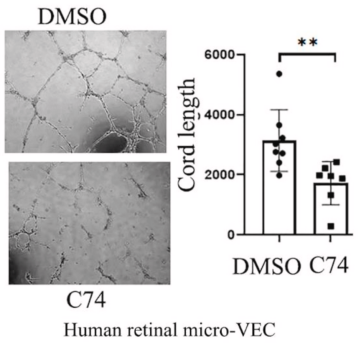
C74 was then examined if these results could be replicated *ex vivo* (Figure 4.14A). In order to do this a choroidal explant angiogenesis assay was performed according to a previously published procedure[142]. Briefly, for choroid isolation, 3-weeks old C57BL/6 mice were injected with Avertin for 5min and then euthanized by cervical dislocation. The eyes were the immediately enucleated and kept on ice-cold medium before dissection. The cornea and lens were removed from the anterior of the eye, and the peripheral choroid-scleral complex was separated from the retina and cut into 1mm by 1mm pieces. These pieces were then cultured



(a) Boxplot summarizing the average speed of HdmVEC of overnight treatments of C74 as compared to the DMSO control in single-cell migration assay. The summary is of 3 independent experiments containing a total of ≈ 60 cells.



(b) Effect of 25uM C74 treatment on proliferation of HdmVEC cells in 2D culture. The error bars are the standard deviation of three independent experiments.



(c) Representative images and quantification of Matrigel cord morphogenesis assay. The assay was performed with HrmVEC cells following overnight treatment of 25uM C74 or 0.1% DMSO.

Figure 4.12: *In vitro* anti-angiogenic activity of C74. 25uM of C74 is effective at halving the average speed and preventing the proliferation of HdmVEC cells. Overnight treatment of 25uM of C74 reduces the ability of HrmVEC cells to form chords by 33%. This data was generated by David Gau.

for 3 days. After which, the fragments were treated every alternate day with 25uM C74 or DMSO (control) until day 6. Photos of individual explants were taken on days 3 and 6, and areas of sprouting were quantified with Fiji software. The surface of day 3 was subtracted from the surface at day 6 in order to calculate the vascular outgrowth that occurred. 25uM of C74 substantially diminished vascular outgrowth (Figure 4.14A).

Lastly, C74 was tested to determine if it could inhibit choroidal neovascularization (CNV) *in vivo* though the murine laser-injury induced CNV assay (Figure 4.14B). In this model, target laser injury to the retinal pigment epithelium and Bruch's membrane induces angiogenic sprouting of chorodial VECs into the outer retinal layer, mimicking the we AMD[143]. For this assay, 8-week old C57BL/6JRj mice received four laser coagulations per eye (400mW, 50ms, 100um spot size) with a Laser Yag 532 Eyelite mounted on a slit lamp. Immediately after

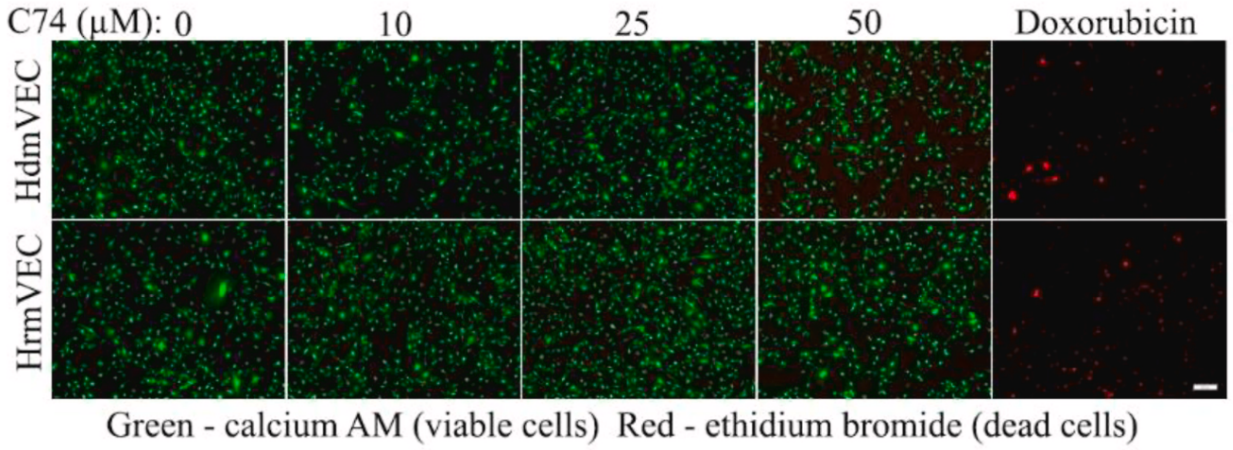
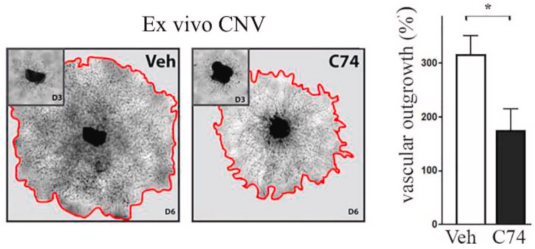


Figure 4.13: Live/dead staining of HdmVEC and HrmVEC following overnight treatment of C74 at the indicated concentrations. Doxorubicin at 8uM is the positive control for inducing cell death. Green cells are viable, red cells are dead, and the scale bar is 200um. (**: $p < 0.01$) This data was generated by David Gau.

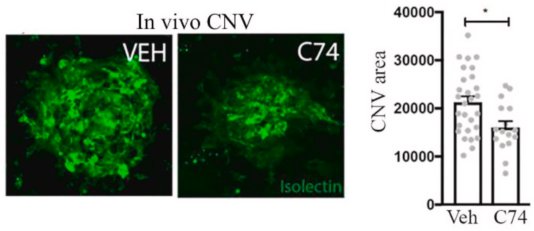
photocoagulation and then on days 4 and 7, the mice were injected intravitreally in both eyes with 2uL of C74 or DMSO. This resulted in an 25uM intravitreal concentration of C74 after the 2.5-fold dilution of the compound following entry into the 5uL intravitreal volume. On day 10, these mice were sacrificed and sub-retinal NV was assessed by lectin staining of choroidal flat-mounts according to previously described methods[142]. Treatment with C74 reduced the mean area of lectin-positive CNV by 25% (Figure 4.14B) and provides the first proof-of-concept for the ability of a pharmacological compound targeting the actin-Pfn1 binding interaction to diminish CNV.

4.5 Conclusions

Our CNN models identified C74, a novel small molecule that improves upon C2 at disrupting the PFN1-actin binding interaction. We identified C74 by first identifying potential C2 binding sites through whole protein docking, then performing a pharmacophore screen in each



(a) Representative image of choroidal explant angiogenesis culture on Day 6 under vehicle (DMSO) or treatment with 25uM C74. The red outline shows the edge of vascular outgrowth. The insert shows shows explants progress on Day 3.



(b) Representative images of isolectin-stained choroidal flat-mounts prepared from mice subjected to laser injury-induced choroidal neovascularization and treated with 0.1% DMSO (VEH) or 25uM C74. The total area of choroidal neovascularization (CNV) from both treatment groups. There are 16 lesions from 8 C74 injected eyes (4 mice), and 28 lesions from 10 DMSO injected eyes (6 mice).

Figure 4.14: Proof of concept for C74's ability to diminish choroidal neovascularization (CNV). 25uM of C74 diminishes CNV both *ex vivo* and *in vivo*. (*: $p < 0.05$) This data was generated by Lucile Vignaud.

of the five proposed binding sites, and scoring the resulting molecules with *gnina*. C74 achieves similar results to C2 in reducing ccRCC cell proliferation and migration assays with half the concentration (Figure 4.10). C74 was also shown to reduce tumor growth *in vivo* (Figure 4.11), and also diminish choroidal neovascularization *ex vivo* and *in vivo* (Figure 4.14). Thus, we have demonstrated C74's ability to disrupt angiogenesis in both ccRCC and wet AMD, which shows the promise of targeting the PFN1-actin binding interaction as a replacement target instead of VEGF. C74 has a worldwide patent - WO2022040005A1.

This drug discovery campaign shows a proof of concept for the CNN models that we developed in Aim2 in real experimental settings. The campaign is still ongoing, as C74's effective dose of about 25 micromolar can still be improved, ideally into something in the nanomolar range. We have been assisting in suggesting new analogues of C74 to be synthesized and experimentally tested by our collaborators. Furthermore, while we have a model of the potential binding site, the exact binding mechanism of C74 is still unknown. Elucidating

the actual binding modality would enable a more robust screening of analogues and a more intelligent selection of molecules to feed into our models. Our collaborators are also in the process of experimentally validating the binding mechanism of C74.

4.6 Declarations

This chapter is adapted from:

> Allen, A., Gau, D., Francoeur, P., Sturm, J., Wang, Y., Matin, R., Maranchie, J., Duensing, A., Kaczorowski, A., Duensing, S., Wu, L., Lotze, M., Koes, D., Storkus, W., Roy, P. Actin-binding protein profilin1 promotes aggressiveness of clear-cell renal cell carcinoma cells. *Journal of Biological Chemistry* **2020**. DOI: 10.1074/jbc.RA120.013963

The compound was also utilized in:

> Gau, D., Vignaud, L., Francoeur, P., Koes, D., Guillonneau, X., Roy, P. Inhibition of ocular neovascularization by novel anti-angiogenic compound. *Experimental Eye Research* **2021**. DOI: 10.1016/j.exer.2021.108861

This series of papers is a collaborative effort in which I identified the lead compound through potential binding site identification on profilin1 and actin through whole protein docking, created a pharmacophore model for each potential binding site, and performed a virtual screen where the pharmacophore models were utilized to identify potential molecules, which were then narrowed down to candidate molecules through our machine learning models. For the first manuscript, I created Figures 7A, B, and C, and wrote the “Pfn1-Actin interaction inhibitor identification" section of the Methods. For both papers, I helped edit the manuscript.

5.0 Conclusions and Future Directions

5.1 Conclusion

The immense size of chemical space is a driving force for developing better modeling methods, as it is far to vast to exhaustively search/screen. Correspondingly, the rise of computing power, especially the adoption of graphical processing units, has also allowed for the rapid advancement of machine learning (ML) for modeling molecules in recent years. This rapid growth has resulted in a large variety of model types and input representations, and it is unclear which combination is best suited to a task. Additionally, more complex ML methods require a tremendous amount of training data in order to be effective, which is especially challenging for structure-based methods. In this work we perform initial experiments to start solving both of these issues and demonstrate the effectiveness of ML in a drug discovery campaign.

For Aim 1 we characterized a wide variety of ML model architectures and 3 different input representations on computed molecular properties. We demonstrated that the more complex ML methods (transformers and convolutional neural networks) typically both outperform and utilize data more efficiently than their simpler counterparts (random forests and feed-forward neural networks). This comes with the trade off that one needs to spend more effort in hyperparameter tuning for these models. However, our UFF energy predictive task shows that this type of analysis can break down if the task is challenging. We demonstrate that until a sufficient quantity of training data (1 million poses in our experiment) is available, the 2D fingerprint based random forest model was the best performer at predicting the UFF energy of a conformer.

Additionally, the UFF energy prediction task also highlights another potential problem

with ML approaches: they do not necessarily perform the way you expect. Our transformer model utilizes the 3D information through the a pairwise distance matrix of every atom in the molecule, which can be computed from the 3D conformer or the 2D molecular graph. We hypothesized that the 3D version of the pairwise distance matrix would allow the model to outperform the 2D pairwise distance matrix as there is more information to learn from. However, we observed that as the training set grows the performance difference between a model using the 3D pairwise distance matrix and the 2D version approaches 0. This indicates that rather than utilizing the 3D information about the relative orientation of atoms, which is necessary to the correct calculation of the UFF energy, the models are instead mostly disregarding that information and determining molecule identity to predict the energy. On the other hand, the radius of gyration prediction task, which depends solely on the orientation of the atoms in 3D space, behaved as expected. So, our expectations were in line when the orientations of atoms were the sole driver of the property prediction, but when there are confounding factors it is entirely possible for the ML model to fit to those rather than what is desired by the user.

In Aim 2 we investigated two approaches to solve the issue of limited data availability for structure based models. First we created the CrossDocked2020 dataset, which both serves as a benchmark for binding pose classification and binding affinity regression, and combinatorially expands the available structures for training through docking ligands to similar non-cognate receptors. We first demonstrated that our CNN models achieve similar performance to other ML models on the often used PDBbind dataset, and then showed that the model's performance deteriorates when evaluated on clustered cross-validation splits of the PDBbind instead of the usual train on general/refined and test on core set. Our models trained utilizing our counterexample generation procedure with CrossDocked2020 yields models with similar performance to that of PDBbind trained models, but have more informative gradients. The CrossDocked2020 trained models also perform better on the hardest class of cross-docking problems (docking a ligand into an apo receptor) than models

trained only with redocking data. Lastly, our exact model performance and training splits are publicly available for CrossDocked2020, which allows for fair comparisons between the results of our model and new models as the new models can be trained and evaluated on the exact same training and testing distributions used in our work.

While CrossDocked2020 expands the available training data, it does so by expanding the number of poses. This does not actually expand the binding affinity data, which was taken from the PDBbind, and in fact only about 40% of CrossDocked2020 has a binding affinity label. We investigated utilizing simple imputation techniques to assign binding affinity labels to these unlabeled complexes. We demonstrated that utilizing the median affinity prediction of an ensemble of models on the poses that were known to be under 2Å from the crystal pose had the best effect on improving performance of our CNN on both binding affinity regression and binding pose classification. Notably, this was achieved by utilizing the same model architecture and only the data that is available in CrossDocked2020.

Finally, while this work showcases potential areas for ML model improvement, they are all *in silico* evaluations. What ultimately matters is if utilizing these models is useful in a real world drug discovery campaign. To demonstrate our CNN model's effectiveness, we incorporated them in the computational pipeline for a drug discovery campaign targeting the actin-PFN1 protein-protein binding interaction. We utilized whole protein docking to identify 5 potential small molecule binding sites on actin and PFN1 and then performed a pharmacophore screen on each site to identify 128,331 potential small molecules. Our CNN was then utilized to minimize and score each of these molecules in order to cluster the molecules in each potential site. From this pool, our collaborators experimentally tested 67 molecules, 7 of which showed activity at disrupting the actin-PFN1 binding complex in an actin polymerization assay. Of these C74 was our most promising compound, which doubled the potency of the initial C2 molecule and has a worldwide patent pending (WO2022040005A1) for small molecules targeting this protein-protein binding interaction. We demonstrate that C74 also slows tumor growth *in vivo* and can diminish choroidal neovascularization *ex vivo*

and *in vitro*.

Overall, this work has demonstrated that our current ML models are useful in a drug discovery campaign, provided a new more rigorous benchmark to the ML community for binding affinity regression and binding pose classification through CrossDocked2020, and demonstrated that imputation is a viable approach to improve binding affinity regression for structure based models. Through the first aim of this work we show that more complex ML models generally perform better, once sufficient training data becomes available, and also utilize the available data more efficiently. This indicates that while ML models are currently successful, we can expect their performance to improve as more and more structures become available in the future.

5.2 Future Directions

All of the models we tried on the UFF energy prediction task performed relatively poorly, where it appears that the models are learning a mapping of molecular identity to the UFF energy of the molecule rather than the force field parameters. We suspect that this is due in part to the much larger variance of energies across different molecules when compared to energetic differences between different poses of the same molecule. In our evaluations the training set is randomly shuffled, which means that the gradients per batch are averaged across molecules. Instead, we could provide all of the poses of some number of molecules for a batch. Then when the gradients are calculated across this batch, the information from different poses of the same molecule will affect the models' weights at the same time. This could help reduce the indexing problem that we observed.

Another approach to help this would be to normalize the UFF energy by the number of atoms in the molecule. As this is a force-field, as the number of atoms in the molecule grows, so do the number of interactions that contribute to the forces interacting on the molecule.

Thus, a type of molecular weight bias along with the identification of the molecule could achieve a greater information gain during training rather than learning the actual forces acting on the molecule. Normalizing the UFF energy by the number of atoms in the molecule could help to eliminate this source of confounding information.

Lastly, our evaluations were done utilizing a random split of ChEMBL. Thus, there is training data leakage into the test set. This was done to give an evaluation about how well our models are fitting the training data. However, since we utilized the test set performance to select our best models in the hyperparameter sweep, we could be selecting models by their ability to overfit the training data, and have no way to detect it. This could be solved by re-training the best handful of models on a molecular identity split of ChEMBL, which could more rigorously control for training data leakage.

The CrossDocked2020 dataset is currently frozen in time, as it is dependent on the Pocketome database to group similar receptors into pockets. Unfortunately, Pocketome no longer exists, which means that CrossDocked2020 cannot have new receptor structures incorporated into it. As such, it would be prudent to move to a different method to determine if two receptor binding pockets are similar. This method should be able to run stand alone, and incorporate into a pipeline that can regenerate a new version of CrossDocked2020 from the PDB.

In Aim2b, we demonstrate that our simple imputation approaches can improve model performance on both binding affinity regression and binding pose classification. There are several paths that could be viable to further improve these models: 1) modify the training procedure, 2) utilize a different model and/or training datasets to perform the imputation. A potential drawback to our study is that we did not treat imputed binding affinity labels differently from the known labels. It could be beneficial to include a hinge-type loss (like we do for binding affinity regression of poor quality poses), but only for the imputed labels.

Secondly, we utilized the same model architecture that we were training to perform the imputation. This means that our model, even with the imputed labels, is still mainly being

driven by the same binding affinity distribution present in the original dataset. We also know that there exists much larger repositories of binding affinity experiments, such as ChEMBL, which do not include structural data. It is possible that a model trained on that larger dataset could provide better imputed labels for our structure based training.

Lastly, in the actin-PFN1 drug discovery campaign, we have collaborators currently working on verifying the binding pose of C74 with actin or PFN1. Once known, this binding pose can be utilized to drive further modeling efforts for analogs, or the exploration of different kinds of chemistry to improve the potency. A different set of collaborators are also currently working on the synthesis of various molecules to provide a structure activity relationship of C74. Again, these changes can help us hone in on further improvements to C74 as its effective dosage of 25uM in our cell assays and mouse retina or 16mg/kg in the mouse tumor model is too high for a clinical setting. These two parallel tracts of information will allow us to rerun the pipeline that identified C74 with a higher quality pharmacophore screen in order to identify new, potentially better, molecules.

Bibliography

[1] Bohacek, R. S.; McMartin, C.; Guida, W. C. The art and practice of structure-based drug design: A molecular modeling perspective. *Medicinal Research Reviews* **1996**, *16*, 3–50, DOI: https://doi.org/10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6.

[2] Lipinski, C. A.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews* **1997**, *23*, 3–25.

[3] Reymond, J.-L.; van Deursen, R.; Blum, L. C.; Ruddigkeit, L. Chemical space as a source for new drugs. *Med. Chem. Commun.* **2010**, *1*, 30–38, DOI: 10.1039/C0MD00020E.

[4] Gaulton, A.; Hersey, A.; Nowotka, M.; Bento, A. P.; Chambers, J.; Mendez, D.; Mutowo, P.; Atkinson, F.; Bellis, L. J.; Cibrián-Uhalte, E.; Davies, M.; Dedman, N.; Karlsson, A.; Magariños, M. P.; Overington, J. P.; Papadatos, G.; Smit, I.; Leach, A. R. The ChEMBL database in 2017. *Nucleic Acids Research* **2016**, *45*, D945–D954, DOI: 10.1093/nar/gkw1074.

[5] Szymanski, P.; Markowicz, M.; Mikiciuk-Olasik, E. Adaptation of High-Throughput Screening in Drug Discovery - Toxicological Screening Tests. *International Journal of Molecular Sciences* **2011**, *31*, 427–452, DOI: 10.3390/ijms13010427.

[6] Liu, Z.; Su, M.; Han, L.; Liu, J.; Yang, Q.; Li, Y.; Wang, R. Forging the Basis for Developing Protein–Ligand Interaction Scoring Functions. *Acc. Chem. Res.* **2017**, *50*, 302–309, DOI: 10.1021/acs.accounts.6b00491, PMID: 28182403.

[7] David, L.; Thakkar, A.; Mercado, R.; Engkvist, O. Molecular representations in AI-driven drug discovery: a review and practical guide. *Journal of Cheminformatics* **2020**, *12*, DOI: 10.1186/s13321-020-00460-5.

[8] Deng, J.; Yang, Z.; Ojima, I.; Samaras, D.; Wang, F. Artificial Intelligence in Drug Discovery: Applications and Techniques. 2021; `https://arxiv.org/abs/2106.05386`.

[9] Systems, M. I. MACCS keys.

[10] Daylight Theory Manual. https://www.daylight.com/dayhtml/doc/theory/, 2011; `https://www.daylight.com/dayhtml/doc/theory/`.

[11] RDKit: Open-Source Cheminformatics. http://www.rdkit.org, accessed November 6, 2017.

[12] Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling* **2010**, *50*, 742–754, DOI: 10.1021/ci100050t, PMID: 20426451.

[13] Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R.; Jr., K. M. M.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *J. Am. Chem. Soc.* **1995**, *117*, 5179–5197, DOI: 10.1021/ja00124a002.

[14] Jr., A. D. M.; Bashford, D.; Bellott, M.; Jr., R. L. D.; Evanseck, J. D.; Field, M. J.; Discher, S.; Gao, J.; Guo, H.; Ha, S.; Joseph-McCarthy, D.; Kuchnir, L.; Kuczera, K.; Lau, F. T. K.; Mattos, C.; Michnick, S.; Ngo, T.; Nguyen, D. T.; Prodhom, B.; III, W. E. R.; Rouc, B.; Schlenkrich, M.; Smith, J. C.; Stote, R.; Straub, J.; Watanabe, M.; Wiorkiewixz-Kuczera, J.; Yin, D.; Karplus, M. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *J. Phys. Chem. B* **1998**, *102*, 3586–3616, DOI: 10.1021/jp973084f.

[15] Simonsen, T.; Archontis, G.; Karplus, M. Free Energy Simulations Come of Age: Protein-Ligand Recognition. *Acc. Chem. Res.* **2002**, *35*, 430–437, DOI: 10.1021/ar010030m.

[16] Koes, D. R.; Baumgartner, M. P.; Camacho, C. J. Lessons Learned in Empirical Scoring with smina from the CSAR 2011 Benchmarking Exercise. *J. Chem. Inf. Model.* **2013**, *53*, 1893–1904, DOI: 10.1021/ci300604z.

[17] Friesner, R. A.; Banks, J. L.; Murphy, R. B.; Halgren, T. A.; Klicic, J. H.; Mainz, D. T.; Repasky, M. P.; Knoll, E. H.; Shelley, M.; Perry, J. K.; Shaw, D. E.; Francis, P.; Shenkin, P. S. Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy. *J. Med. Chem.* **2004**, *47*, 1739–1749, DOI: 10.1021/jm0306430.

[18] Wang, R.; Lai, L.; Wang, S. Further development and validation of empirical scoring functions for structure-based affinity prediction. *J. Comput.-Aided Mol. Des.* **2002**, *16*, 11–26, DOI: 10.1023/A:1016357811882.

[19] Muegge, I. A knowledge-based scoring function for protein-ligand interactions: Probing the reference state. *Perspect. Drug Discovery Des.* **2000**, *20*, 99–114, DOI: 10.1023/A:100872900.

[20] Gohlke, H.; Hendlich, M.; Klebe, G. Knowledge-based scoring function to predict protein-ligand interactions. *J. Mol. Biol.* **2000**, *295*, 337–356, DOI: 10.1006/jmbi.1999.3371.

[21] Ballester, P. J.; Mitchell, J. B. O. A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. *Bioinformatics* **2010**, *26*, 1169–1175, DOI: 10.1093/bioinformatics/btq112.

[22] Li, L.; Wang, B.; Meroueh, S. O. Support Vector Regression Scoring of Receptor–Ligand Complexes for Rank-Ordering and Virtual Screening of Chemical Libraries. *Journal of Chemical Information and Modeling* **2011**, *51*, 2132–2138, DOI: 10.1021/ci200078f, PMID: 21728360.

[23] Sunseri, J.; Koes, D. R. libmolgrid: Graphics Processing Unit Accelerated Molecular Gridding for Deep Learning Applications. *J. Chem. Inf. Model.* **2020**, *60*, 1079–1084, DOI: 10.1021/acs.jcim.9b01145, PMID: 32049525.

[24] Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. Protein–Ligand Scoring with Convolutional Neural Networks. *Journal of Chemical Information and Modeling* **2017**, *57*, 942–957, DOI: 10.1021/acs.jcim.6b00740, PMID: 28368587.

[25] Jiménez, J.; Škalič, M.; Martínez-Rosell, G.; De Fabritiis, G. KDEEP: Protein–Ligand Absolute Binding Affinity Prediction via 3D-Convolutional Neural Networks. *Journal of Chemical Information and Modeling* **2018**, *58*, 287–296, DOI: 10.1021/acs.jcim.7b00650, PMID: 29309725.

[26] Stepniewska-Dziubinska, M. M.; Zielenkiewicz, P.; Siedlecki, P. Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. *Bioinformatics* **2018**, *34*, 3666–3674, DOI: 10.1093/bioinformatics/bty374.

[27] Francoeur, P. G.; Masuda, T.; Sunseri, J.; Jia, A.; Iovanisci, R. B.; Snyder, I.; Koes, D. R. Three-Dimensional Convolutional Neural Networks and a Cross-Docked Data Set for Structure-Based Drug Design. *Journal of Chemical Information and Modeling* **2020**, *60*, 4200–4215, DOI: 10.1021/acs.jcim.0c00411, PMID: 32865404.

[28] Jiang, M.; Li, Z.; Zhang, S.; Wang, S.; Wang, X.; Yuan, Q.; Wei, Z. Drug–target affinity prediction using graph neural network and contact maps. *RSC Adv.* **2020**, *10*, 20701–20712, DOI: 10.1039/D0RA02297G.

[29] Jiang, H.; Wang, J.; Cong, W.; Huang, Y.; Ramezani, M.; Sarma, A.; Dokholyan, N. V.; Mahdavi, M.; Kandemir, M. T. Predicting Protein–Ligand Docking Structure with Graph Neural Network. *Journal of Chemical Information and Modeling* **2022**, *62*, 2923–2932, DOI: 10.1021/acs.jcim.2c00127, PMID: 35699430.

[30] Zhang, S.; Jin, Y.; Liu, T.; Wang, Q.; Zhang, Z.; Zhao, S.; Shan, B. SS-GNN: A Simple-Structured Graph Neural Network for Affinity Prediction. 2022; `https://arxiv.org/abs/2206.07015`.

[31] Yang, K.; Swanson, K.; Jin, W.; Coley, C.; Eiden, P.; Gao, H.; Guzman-Perez, A.; Hopper, T.; Kelley, B.; Mathea, M.; Palmer, A.; Settels, V.; Jaakkola, T.; Jensen, K.; Barzilay, R. Analyzing Learned Molecular Representations for Property Prediction. *Journal of Chemical Information and Modeling* **2019**, *59*, 3370–3388, DOI: 10.1021/acs.jcim.9b00237, PMID: 31361484.

[32] Feinberg, E. N.; Joshi, E.; Pande, V. S.; Cheng, A. C. Improvement in ADMET Prediction with Multitask Deep Featurization. *Journal of Medicinal Chemistry* **2020**, *63*, 8835–8848, DOI: 10.1021/acs.jmedchem.9b02187, PMID: 32286824.

[33] Deng, J.; Yang, Z.; Wang, H.; Ojima, I.; Samaras, D.; Wang, F. Taking a Respite from Representation Learning for Molecular Property Prediction. 2022; `https://arxiv.org/abs/2209.13492`.

[34] van Tilborg, D.; Alenicheva, A.; Grisoni, F. Exposing the Limitations of Molecular Machine Learning with Activity Cliffs. *Journal of Chemical Information and Modeling* **2022**, *62*, 5938–5951, DOI: 10.1021/acs.jcim.2c01073, PMID: 36456532.

[35] Janela, T.; Bajorath, J. Simple nearest-neighbor analysis meets the accuracy of compound potency predictions using complex machine learning tools. *Nature Machine Intelligence* **2022**, *4*, 1246–1255, DOI: 10.1038/s42256-022-00581-6.

[36] Chen, L.; Cruz, A.; Ramsey, S.; Dickson, C. J.; Duca, J. S.; Hornak, V.; Koes, D. R.; Kurtzman, T. Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening. *PLOS ONE* **2019**, *14*, 1–22, DOI: 10.1371/journal.pone.0220113.

[37] Rohrer, S. G.; Baumann, K. Maximum Unbiased Validation (MUV) Data Sets for Virtual Screening Based on PubChem Bioactivity Data. *Journal of Chemical Information and Modeling* **2009**, *49*, 169–184, DOI: 10.1021/ci8002649, PMID: 19161251.

[38] Hassan-Harrirou, H.; Zhang, C.; Lemmin, T. RosENet: Improving Binding Affinity Prediction by Leveraging Molecular Mechanics Energies with an Ensemble of 3D Convolutional Neural Networks. *Journal of Chemical Information and Modeling* **2020**, *60*, 2791–2802, DOI: 10.1021/acs.jcim.0c00075, PMID: 32392050.

[39] Rana, M. M.; Nguyen, D. D. EISA-Score: Element Interactive Surface Area Score for Protein–Ligand Binding Affinity Prediction. *Journal of Chemical Information and Modeling* **2022**, *62*, 4329–4341, DOI: 10.1021/acs.jcim.2c00697, PMID: 36108270.

[40] Yang, C.; Zhang, Y. Delta Machine Learning to Improve Scoring-Ranking-Screening Performances of Protein–Ligand Scoring Functions. *Journal of Chemical Information and Modeling* **2022**, *62*, 2696–2712, DOI: 10.1021/acs.jcim.2c00485, PMID: 35579568.

[41] Wee, J.; Xia, K. Ollivier Persistent Ricci Curvature-Based Machine Learning for the Protein–Ligand Binding Affinity Prediction. *Journal of Chemical Information and Modeling* **2021**, *61*, 1617–1626, DOI: 10.1021/acs.jcim.0c01415, PMID: 33724038.

[42] Li, Q.; Zhang, X.; Wu, L.; Bo, X.; He, S.; Wang, S. PLA-MoRe: A Protein–Ligand Binding Affinity Prediction Model via Comprehensive Molecular Representations. *Journal of Chemical Information and Modeling* **2022**, *62*, 4380–4390, DOI: 10.1021/acs.jcim.2c00960, PMID: 36054653.

[43] Jones, D.; Kim, H.; Zhang, X.; Zemla, A.; Stevenson, G.; Bennett, W. F. D.; Kirshner, D.; Wong, S. E.; Lightstone, F. C.; Allen, J. E. Improved Protein–Ligand Binding Affinity Prediction with Structure-Based Deep Fusion Inference. *Journal of Chemical Information and Modeling* **2021**, *61*, 1583–1592, DOI: 10.1021/acs.jcim.0c01306, PMID: 33754707.

[44] Meli, R.; Anighoro, A.; Bodkin, M. J.; Morris, G. M.; Biggin, P. C. Learning protein-ligand binding affinity with atomic environment vectors. *Journal of Chemical Information and Modeling* **2021**, *13*, 1758–2946, DOI: 10.1186/s13321-021-00536-w.

[45] ET-score: Improving Protein-ligand Binding Affinity Prediction Based on Distance-weighted Interatomic Contact Features Using Extremely Randomized Trees Algorithm. *Molecular informatics* **2021**, *40*, 1868–1751, DOI: 10.1002/minf.202060084.

[46] Rappe, A. K.; Casewit, C. J.; Colwell, K. S.; Goddard, W. A. I.; Skiff, W. M. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *Journal of the American Chemical Society* **1992**, *114*, 10024–10035, DOI: 10.1021/ja00051a040.

[47] Trott, O.; Olson, A. J. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.* **2010**, *31*, 455–461, DOI: 10.1002/jcc.21334.

[48] Emmanuel, T.; Maupong, T.; Mpoeleng, D.; Semong, T.; Mphago, B.; Tabona, O. A survey on missing data in machine learning. *Journal of Big Data* **2021**, *8*, DOI: 10.1186/s40537-021-00516-9.

[49] Jager, S.; Allhorn, A.; Bießmann, F. A Benchmark for Data Imputation Methods. *Frontiers in Big Data* **2021**, *4*, DOI: 10.3389/fdata.2021.693674.

[50] Pichler, R.; Heidegger, I. Novel concepts of antiangiogenic therapies in metastatic renal cell cancer. *Magazine og European Medical Oncology* **2017**, *10*, 206–212, DOI: 10.1007/s12254-017-0344-2.

[51] Shaked, Y.; Henke, E.; Roodhart, J. M.; Mancuso, P.; Langenberg, M. H.; Colleoni, M.; Daenen, L. G.; Man, S.; Xu, P.; Emmenegger, U.; Tang, T.; Zhu, Z.; Witte, L.; Strieter, R. M.; Bertolini, F.; Voest, E. E.; Benezra, R.; Kerbel, R. S. Rapid Chemotherapy-Induced Acute Endothelial Progenitor Cell Mobilization: Implications for Antiangiogenic Drugs as Chemosensitizing Agents. *Cancer Cell* **2008**, *14*, 263–273, DOI: https://doi.org/10.1016/j.ccr.2008.08.001.

[52] Rini, B. I. New strategies in kidney cancer: therapeutic advances through understanding the molecular basis of response and resistance. *Clinical Cancer Research* **2010**, *16*, 1348–1354.

[53] Pircher, A.; Jöhrer, K.; Kocher, F.; Steiner, N.; Graziadei, I.; Heidegger, I.; Pichler, R.; Leonhartsberger, N.; Kremser, C.; Kern, J.; Untergasser, G.; Gunsilius, E.; Hilbe, W. Biomarkers of evasive resistance predict disease progression in cancer patients treated with antiangiogenic therapies. *Oncotarget* **2016**, *7*, 20109–20123, DOI: https://doi.org/10.18632/oncotarget.7915.

[54] Rini, B. I. Biomarkers: hypertension following anti-angiogenesis therapy. *Clinical advances in hematology & oncology: H&O* **2010**, *8*, 415–416.

[55] de Bazelaire, C.; Alsop, D. C.; George, D.; Pedrosa, I.; Wang, Y.; Michaelson, M. D.; Rofsky, N. M. Magnetic Resonance Imaging–Measured Blood Flow Change after Antiangiogenic Therapy with PTK787/ZK 222584 Correlates with Clinical Outcome in Metastatic Renal Cell Carcinoma. *Clinical Cancer Research* **2008**, *14*, 5548–5554, DOI: 10.1158/1078-0432.CCR-08-0417.

[56] Nikolinakos, P. G.; Altorki, N.; Yankelevitz, D.; Tran, H. T.; Yan, S.; Rajagopalan, D.; Bordogna, W.; Ottesen, L. H.; Heymach, J. V. Plasma Cytokine and Angiogenic

Factor Profiling Identifies Markers Associated with Tumor Shrinkage in Early-Stage Non–Small Cell Lung Cancer Patients Treated with Pazopanib. *Cancer Research* **2010**, *70*, 2171–2179, DOI: 10.1158/0008-5472.CAN-09-2533.

[57] Yang, C.-Y.; Lin, M.-W.; Chang, Y.-L.; Wu, C.-T.; Yang, P.-C. Programmed cell death-ligand 1 expression is associated with a favourable immune microenvironment and better overall survival in stage I pulmonary squamous cell carcinoma. *European Journal of Cancer* **2016**, *57*, 91–103, DOI: https://doi.org/10.1016/j.ejca.2015.12.033.

[58] Messai, Y.; Gad, S.; Noman, M. Z.; Le Teuff, G.; Couve, S.; Janji, B.; Kammerer, S. F.; Rioux-Leclerc, N.; Hasmim, M.; Ferlicot, S.; Baud, V.; Mejean, A.; Mole, D. R.; Richard, S.; Eggermont, A. M.; Albiges, L.; Mami-Chouaib, F.; Escudier, B.; Chouaib, S. Renal Cell Carcinoma Programmed Death-ligand 1, a New Direct Target of Hypoxia-inducible Factor-2 Alpha, is Regulated by von Hippel–Lindau Gene Mutation Status. *European Urology* **2016**, *70*, 623–632, DOI: https://doi.org/10.1016/j.eururo.2015.11.029.

[59] Gau, D.; Vignaud, L.; Allen, A.; Guo, Z.; Sahel, J.; Boone, D.; Koes, D.; Guillonneau, X.; Roy, P. Disruption of profilin1 function suppresses developmental and pathological retinal neovascularization. *Journal of Biological Chemistry* **2020**, *295*, 9618–9629, DOI: https://doi.org/10.1074/jbc.RA120.012613.

[60] Gau, D.; Lewis, T.; McDermott, L.; Wipf, P.; Koes, D.; Roy, P. Structure-based virtual screening identifies a small-molecule inhibitor of the profilin 1–actin interaction. *Journal of Biological Chemistry* **2018**, *293*, 2606–2616, DOI: https://doi.org/10.1074/jbc.M117.809137.

[61] Sieg, J.; Flachsenberg, F.; Rarey, M. In Need of Bias Control: Evaluating Chemical Data for Machine Learning in Structure-Based Virtual Screening. *J. Chem. Inf. Model.* **2019**, *59*, 947–961, DOI: 10.1021/acs.jcim.8b00712.

[62] Lopez-del Rio, A.; Nonell-Canals, A.; Vidal, D.; Perera-Lluna, A. Evaluation of Cross-Validation Strategies in Sequence-Based Binding Prediction Using Deep-Learning. *J. Chem. Inf. Model.* **2019**, *59*, 1645–1657, DOI: 10.1021/acs.jcim.8b00663.

[63] Wallach, I.; Heifets, A. Most Ligand-Based Classification Benchmarks Reward Memorization Rather than Generalization. *J. Chem. Inf. Model.* **2018**, *58*, 916–932, DOI: 10.1021/acs.jcim.7b00403.

[64] Tran-Nguyen, V.-K.; Jacquemard, C.; Rognan, D. LIT-PCBA: An Unbiased Data Set for Machine Learning and Virtual Screening. *J. Chem. Inf. Model.* **2020**,

[65] Maziarka, L.; Danel, T.; Mucha, S.; Rataj, K.; Tabor, J.; Jastrzebski, S. Molecule Attention Transformer. *arXiv* **2020**,

[66] Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; Amodei, D. Scaling Laws for Neural Language Models. 2020; `https://arxiv.org/abs/2001.08361`.

[67] Morgan, H. L. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of chemical documentation* **1965**, *5*, 107–113.

[68] Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. Protein–Ligand Scoring with Convolutional Neural Networks. *J. Chem. Inf. Model.* **2017**, *57*, 942–957, DOI: 10.1021/acs.jcim.6b00740.

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

[70] Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc., 2019; pp 8024–8035.

[71] Biewald, L. Experiment Tracking with Weights and Biases. 2020; `https://www.wandb.com/`, Software available from wandb.com.

[72] Williams, C. K.; Rasmussen, C. E. *Gaussian processes for machine learning*; MIT press Cambridge, MA, 2006; Vol. 2.

[73] Gau, D.; Lewis, T.; McDermott, L.; Wipf, P.; Koes, D.; Roy, P. Structure-based virutal screening identifies a small-molecule inhibitor of the profilin 1-actin interaction. *J. Biol. Chem.* **2018**, *293*, 2606–2616, DOI: 10.1074/jbc.M117.809137.

[74] Fradera, X.; Babaoglu, K. Overview of Methods and Strategies for Conducting Virtual Small Molecule Screening. *Curr. Protoc. Chem. Biol.* **2017**, *9*, 196–212, DOI: 10.1002/cpch.27.

[75] Bajusz, D.; Ferenczy, G. G.; Keseru, G. M. Structure-Based Virtual Screening Approaches in Kinase-directed Drug Discovery. *Curr. Trends Med. Chem.* **2017**, *17*, 2235, DOI: https://doi.org/10.2174/1568026617666170224121313.

[76] Cheng, T.; Li, Q.; Zhou, Z.; Wang, Y.; Bryant, S. H. Structure-based virtual screening for drug discovery: a problem-centric review. *AAPS J* **2012**, *14*, 133–41, DOI: 10.1208/s12248-012-9322-0, [PubMed:22281989] [PubMed Central:PMC3282008] [doi:10.1208/s12248-012-9322-0].

[77] Ripphausen, P.; Nisius, B.; Peltason, L.; Bajorath, J. Quo vadis, virtual screening? A comprehensive survey of prospective applications. *J. Med. Chem.* **2010**, *53*, 8461–8467.

[78] Kitchen, D. B.; Decornez, H.; Furr, J. R.; Bajorath, J. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat Rev Drug Discov* **2004**, *3*, 935–49, [PubMed:15520816] [doi:10.1038/nrd1549].

[79] Durrant, J. D.; McCammon, J. A. NNScore 2.0: A Neural-Network Receptor-Ligand Scoring Function. *J. Chem. Inf. Model.* **2011**, *51*, 2897–2903, DOI: 10.1021/ci2003889.

[80] Hassan, M. M.; Mogollon, D. C.; Fuentes, O.; Sirimulla, S. DLSCORE: A Deep Learning Model for Predicting Protein-Ligand Binding Affinities. *ChemRxiv* **2018**, DOI: 10.26434/chemrxiv.6159143.v1.

[81] Wojcikowski, M.; Kikielka, M.; Stepniwska-Dziubinska, M. M.; Siedlecki, P. Development of a protein-ligand extended connectivity (PLEC) fingerprint and its application for binding affinity predictions. *Bioinformatics* **2018**, *bty757*, DOI: 10.1093/bioinformatics/bty757.

[82] Shen, C.; Ding, J.; Wang, Z.; Cao, D.; Ding, X.; Hou, T. From machine learning to deep learning: Advances in scoring functions for protein–ligand docking. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2020**, *10*, e1429, DOI: 10.1002/wcms.1429.

[83] Li, H.; Sze, K.-H.; Lu, G.; Ballester, P. J. Machine-learning scoring functions for structure-based drug lead optimization. *Wiley Interdiscip. Rev.: Comput. Mol. Sci. n/a*, e1465, DOI: 10.1002/wcms.1465.

[84] Ozturk, H.; Ozgur, A.; Ozkirimli, E. DeepDTA: deep drug-target binding affinity prediction. *Bioinformatics* **2018**, *34*, 821–829, DOI: 10.1093/bioinformatics/bty593.

[85] Stepniewska-Dziubinska, M. M.; Zielenkiewicz, P.; Siedlecki, P. Development and evaluation of a deep learning model for protein-ligand binding affinity prediction. *Bioinformatics* **2018**, *34*, 3666–3674, DOI: 10.1093/bioinformatics/bty374.

[86] Imrie, F.; Bradley, A. R.; van der Schaar, M.; Deane, C. M. Protein Family Specific Models Using Deep Neural Networks and Transfer Learning Improve Virtual Screening and Highlight the Need for More Data. *J. Chem. Inf. Model.* **2018**, *58*, 2319–2330, DOI: 10.1021/acs.jcim.8b00350.

[87] Lim, J.; Ryu, S.; Park, K.; Choe, Y. J.; Ham, J.; Kim, W. Y. Predicting drug-target interaction using 3D structure-embedded graph representations from graph neural networks. *J. Chem. Inf. Model.* **2019**, *59*, 3981–3988, DOI: https://doi.org/10.1021/acs.jcim.9b00387.

[88] Cang, Z.; Mu, L.; Wei, G.-W. Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening. *PLoS Comput. Biol.* **2018**, *14*, 1–44, DOI: 10.1371/journal.pcbi.1005929.

[89] Cleves, A.; Jain, A. Effects of inductive bias on computational evaluations of ligand-based modeling and on drug discovery. *J. Comput.-Aided Mol. Des.* **2008**, *22*, 147–159, DOI: 10.1007/s10822-007-9150-y.

[90] Xia, J.; Tilahun, E. L.; Reid, T.-E.; Zhang, L.; Wang, X. S. Benchmarking Methods and Data Sets for Ligand Enrichment Assessment in Virutal Screening. *Methods* **2015**, *71*, 146–157, DOI: 10.1016/j.ymeth.2014.11.015.

[91] Mysinger, M. M.; Carchia, M.; Irwin, J. J.; Shoichet, B. K. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *J. Med. Chem.* **2012**, *55*, 6582–6594.

[92] Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **2021**, *596*, 1476–4687, DOI: 10.1038/s41586-021-03819-2.

[93] Ye, Z.; Baumgartner, M. P.; Wingert, B. M.; Camacho, C. J. Optimal strategies for virtual screening of induced-fit and flexible target in the 2015 D3R Grand Challenge. *J. Comput.-Aided Mol. Des.* **2016**, *30*, 695–706.

[94] Emmanuel, T.; Maupong, T.; Mpoeleng, D.; Semong, T.; Mphago, B.; Tabona, O. A survey on missing data in machine learning. *Journal of Big Data* **2021**, *8*, DOI: 10.1186/s40537-021-00516-9.

[95] Hasan, M. K.; Alam, M. A.; Roy, S.; Dutta, A.; Jawad, M. T.; Das, S. Missing value imputation affects the performance of machine learning: A review and analysis of the literature (2010–2021). *Informatics in Medicine Unlocked* **2021**, *27*, 100799, DOI: https://doi.org/10.1016/j.imu.2021.100799.

[96] Little, R. J. A.; Rubin, D. B. *Statistical analysis with missing data.*, 2nd ed.; Wiley Ser. Probab. Stat.; Chichester: Wiley, 2002.

[97] Khan, S. I.; Hoque, A. S. M. L. SICE: an improved missing data imputation technique. *Journal of Big Data* **2020**, *7*, DOI: 10.1186/s40537-020-00313-w.

[98] McClellan, C.; Mitchell, E.; Anderson, J.; Zuvekas, S. Using machine-learning algorithms to improve imputation in the medical expenditure panel survey. *Health Services Research* **2023**, *58*, 423–432, DOI: https://doi.org/10.1111/1475-6773.14115.

[99] Wang, H.; Tang, J.; Wu, M.; Wang, X.; Zhang, T. Application of machine learning missing data imputation techniques in clinical decision making: taking the discharge assessment of patients with spontaneous supratentorial intracerebral hemorrhage as an example. *BMC Medical Informatics and Decision Making* **2022**, *22*, DOI: 10.1186/s12911-022-01752-6.

[100] Rubinsteyn, A.; O'Donnell, T.; Damaraju, N.; Hammerbacher, J. Predicting Peptide-MHC Binding Affinities with Imputed Training Data. *bioRxiv* **2016**, DOI: 10.1101/054775.

[101] Garcia-Hernandez, C.; Fernández, A.; Serratosa, F. Ligand-Based Virtual Screening Using Graph Edit Distance as Molecular Similarity Measure. *Journal of Chemical Information and Modeling* **2019**, *59*, 1410–1421, DOI: 10.1021/acs.jcim.8b00820, PMID: 30920214.

[102] Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K. Q. Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017; pp 4700–4708.

[103] Bakan, A.; Dutta, A.; Mao, W.; Liu, Y.; Chennubhotla, C.; Lezon, T. R.; Bahar, I. Evol and ProDy for bridging protein sequence evolution and structural dynamics. *Bioinformatics* **2014**, *30*, 2681–2683, DOI: 10.1093/bioinformatics/btu336.

[104] Koes, D. R.; Baumgartner, M. P.; Camacho, C. J. *J. Chem. Inf. Model.* **2013**, DOI: 10.1021/ci300604z, [PubMed:23379370] [PubMed Central:PMC3726561] [doi:10.1021/ci300604z].

[105] Kufareva, I.; Ilatovskiy, A. V.; Abagyan, R. Pocketome: an encyclopedia of small-molecule binding sites in 4D. *Nucleic Acids Res.* **2012**, *40*, 535–540, DOI: 10.1093/nar/gkr825.

[106] Konc, J.; Janežič, D. ProBiS algorithm for detection of structurally similar protein binding sites by local structural alignment. *Bioinformatics* **2010**, *26*, 1160–1168.

[107] Ragoza, M.; Turner, L.; Koes, D. R. Ligand Pose Optimization with Atomic Grid-Based Convolutional Neural Networks. Machine Learning for Molecules and Materials NIPS 2017 Workshop. 2017; arXiv preprint arXiv:1710.07400.

[108] Dreossi, T.; Ghosh, S.; Yue, X.; Keutzer, K.; Sangiovanni-Vincentelli, A.; Seshia, S. A. Counterexample-guided data augmentation. *arXiv preprint arXiv:1805.06962* **2018**,

[109] Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* **2014**,

[110] Huang, S.-Y.; Zou, X. Ensemble docking of multiple protein structures: considering protein structural variations in molecular docking. *Proteins: Struct., Funct., Bioinf.* **2007**, *66*, 399–421.

[111] Li, Y.; Liu, Z.; Li, J.; Han, L.; Liu, J.; Zhao, Z.; Wang, R. Comparative assessment of scoring functions on an updated benchmark: 1. Compilation of the test set. *J. Chem. Inf. Model.* **2014**, *54*, 1700–1716.

[112] Boyles, F.; Deane, C. M.; Morris, G. M. Learning from the ligand: using ligand-based features to improve binding affinity prediction. *Bioinformatics* **2020**, *36*, 758–764, DOI: https://doi.org/10.1093/bioinformatics/btz665.

[113] Sollich, P.; Krogh, A. Learning with ensembles: How overfitting can be useful. Advances in neural information processing systems. 1996; pp 190–196.

[114] Li, H.; Leung, K.-S.; Wong, M.-H.; Ballester, P. J. Correcting the impact of docking pose generation error on binding affinity prediction. *BMC Bioinf.* **2016**, *17*, DOI: https://doi.org/10.1186/s12859-016-1169-4.

[115] Ashtawy, H. M.; Mahapatra, N. R. Task-Specific Scoring Functions for Predicting Ligand Binding Poses and Affinity and for Screening Enrichment. *J. Chem. Inf. Model.* **2018**, *58*, 119–133, DOI: 10.1021/acs.jcim.7b00309, PMID: 29190087.

[116] Wang, C.; Zhang, Y. Improving scoring-docking-screening powers of protein–ligand scoring functions using random forest. *J. Comput. Chem.* **2017**, *38*, 169–177, DOI: 10.1002/jcc.24667.

[117] Damm-Ganamet, K. L.; Smith, R. D.; Dunbar, J. B.; Stuckey, J. A.; Carlson, H. A. CSAR Benchmark Exercise 2011–2012: Evaluation of Results from Docking and Relative Ranking of Blinded Congeneric Series. *J. Chem. Inf. Model.* **2013**, *53*, 1853–1870, DOI: 10.1021/ci400025f, PMID: 23548044.

[118] Yuriev, E.; Ramsland, P. A. Latest developments in molecular docking: 2010-2011 in review. *J. Mol. Recognit.* **2013**, *26*, DOI: https://doi.org/10.1002/jmr.2266.

[119] Mendez, D.; Gaulton, A.; Bento, A. P.; Chambers, J.; De Veij, M.; Félix, E.; Magariños, M.; Mosquera, J.; Mutowo, P.; Nowotka, M.; Gordillo-Marañón, M.; Hunter, F.; Junco, L.; Mugumbate, G.; Rodriguez-Lopez, M.; Atkinson, F.; Bosc, N.; Radoux, C.; Segura-Cabrera, A.; Hersey, A.; Leach, A. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research* **2018**, *47*, D930–D940, DOI: 10.1093/nar/gky1075.

[120] Siegel, R. L.; Miller, K. D.; Wagle, N. S.; Jemal, A. Cancer statistics, 2023. *CA: A Cancer Journal for Clinicians* **2023**, *73*, 17–48, DOI: https://doi.org/10.3322/caac.21763.

[121] Howlander, N.; Noone, A.; Krapcho, M.; Miller, D.; Brest, A.; Yu, M.; Ruhl, J.; Tatalovich, Z.; Mariotto, A.; Lewis, D.; Chen, H.; Feuer, E.; Cronin, K. SEER Cancer Statistics Review, 1975-2016, National Cancer Institute. 2019; `https://seer.cancer.gov/csr/1975_2016/`.

[122] Bergers, G.; Hanahan, D. Modes of resistance to anti-angiogenic therapy. *Nature Reviews Cancer* **2008**, *8*, 592–603.

[123] Goel, S.; Duda, D. G.; Xu, L.; Munn, L. L.; Boucher, Y.; Fukumura, D.; Jain, R. K. Normalization of the vasculature for treatment of cancer and other diseases. *Physiological reviews* **2011**, *91*, 1071–1121.

[124] Welti, J.; Loges, S.; Dimmeler, S.; Carmeliet, P., et al. Recent molecular discoveries in angiogenesis and antiangiogenic therapies in cancer. *The Journal of clinical investigation* **2013**, *123*, 3190–3200.

[125] Gehrs, K. M.; Anderson, D. H.; Johnson, L. V.; Hageman, G. S. Age-related macular degeneration—emerging pathogenetic and therapeutic concepts. *Annals of medicine* **2006**, *38*, 450–471.

[126] Stewart, M. W. Clinical and differential utility of VEGF inhibitors in wet age-related macular degeneration: focus on aflibercept. *Clinical Ophthalmology* **2012**, *6*, 1175–1186, DOI: 10.2147/OPTH.S33372, PMID: 22973088.

[127] Al-Zamil, W. M.; Yassin, S. A. Recent developments in age-related macular degeneration: a review. *Clinical interventions in aging* **2017**, 1313–1330.

[128] Brown, D. M.; Regillo, C. D. Anti-VEGF agents in the treatment of neovascular age-related macular degeneration: applying clinical trial results to the treatment of everyday patients. *American journal of ophthalmology* **2007**, *144*, 627–637.

[129] Ammar, M. J.; Hsu, J.; Chiang, A.; Ho, A. C.; Regillo, C. D. Age-related macular degeneration therapy: a review. *Current opinion in ophthalmology* **2020**, *31*, 215–221.

[130] Ding, Z.; Lambrechts, A.; Parepally, M.; Roy, P. Silencing profilin-1 inhibits endothelial cell proliferation, migration and cord morphogenesis. *Journal of cell science* **2006**, *119*, 4127–4137.

[131] Ding, Z.; Gau, D.; Deasy, B.; Wells, A.; Roy, P. Both actin and polyproline interactions of profilin-1 are required for migration, invasion and capillary morphogenesis of vascular endothelial cells. *Experimental cell research* **2009**, *315*, 2963–2973.

[132] Gau, D.; Veon, W.; Capasso, T. L.; Bottcher, R.; Shroff, S.; Roman, B. L.; Roy, P. Pharmacological intervention of MKL/SRF signaling by CCG-1423 impedes endothelial cell migration and angiogenesis. *Angiogenesis* **2017**, *20*, 663–672.

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

[134] Leach, A. R.; Shoichet, B. K.; Peishoff, C. E. Prediction of protein- ligand interactions. Docking and scoring: successes and gaps. *Journal of medicinal chemistry* **2006**, *49*, 5851–5855.

[135] Lyu, J.; Wang, S.; Balius, T. E.; Singh, I.; Levit, A.; Moroz, Y. S.; O'Meara, M. J.; Che, T.; Algaa, E.; Tolmachova, K., et al. Ultra-large library docking for discovering new chemotypes. *Nature* **2019**, *566*, 224–229.

[136] Case, D. et al. *University of California, San Francisco*.

[137] Giordano, D.; Biancaniello, C.; Argenio, M. A.; Facchiano, A. Drug Design by Pharmacophore and Virtual Screening Approach. *Pharmaceuticals (Basel)* **2022**, *15*, 646, DOI: 10.3390/ph15050646.

[138] Wermuth, C.; Ganellin, C.; Lindberg, P.; Mitscher, L. Glossary of terms used in medicinal chemistry (IUPAC Recommendations 1998). *Pure and applied Chemistry* **1998**, *70*, 1129–1143.

[139] Sunseri, J.; Koes, D. R. Pharmit: interactive exploration of chemical space. *Nucleic Acids Research* **2016**, *44*, W442–W448, DOI: 10.1093/nar/gkw287.

[140] McNutt, A. T.; Francoeur, P.; Aggarwal, R.; Masuda, T.; Meli, R.; Sunseri, J.; Koes, D. R. GNINA 1.0: molecular docking with deep learning. *Journal of Cheminformatics* **2021**, *13*, DOI: 10.1186/s13321-021-00522-2.

[141] Baumgartner, M. *Doctoral Dissertation, University of Pittsburgh*.

[142] Shao, Z.; Friedlander, M.; Hurst, C. G.; Cui, Z.; Pei, D. T.; Evans, L. P.; Juan, A. M.; Tahir, H.; Duhamel, F.; Chen, J., et al. Choroid sprouting assay: an ex vivo model of microvascular angiogenesis. *PloS one* **2013**, *8*, e69552.

[143] Montassar, F.; Darche, M.; Blaizot, A.; Augustin, S.; Conart, J.-B.; Millet, A.; Elayeb, M.; Sahel, J.-A.; Réaux-Le Goazigo, A.; Sennlaub, F., et al. Lebecetin, a C-type lectin, inhibits choroidal and retinal neovascularization. *FASEB Journal* **2017**, *31*, 1107–1119.