3D Convolutional Neural Networks for Computational Drug Discovery

by

Jocelyn Sunseri

B.S., University of Pittsburgh, 2014

B.A., Carnegie Mellon University, 2009

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

2021

UNIVERSITY OF PITTSBURGH

SCHOOL OF MEDICINE

This dissertation was presented

by

Jocelyn Sunseri

It was defended on

October 29th, 2020

and approved by

3D Convolutional Neural Networks for Computational Drug Discovery

Jocelyn Sunseri, PhD

University of Pittsburgh, 2021

This thesis describes aspects of the implementation and application of voxel-based convolutional neural networks (CNNs) to problems in computational drug discovery. It opens by justifying the novelty of this approach by presenting a more mainstream approach to the common tasks of virtual screening and binding pose prediction, augmented with more simplistic machine learning methods, and demonstrating their suboptimal performance when applied prospectively. It then describes my contributions to our group's development of voxel-based CNNs as we honed their implementation and training strategy, and reports our library that facilitates featurization and training using this approach. It continues with a prospective assessment of their performance, analogous to the first prospective evaluation, with the addition of a novel CNN-based pose sampling strategy. Next it makes a foray into model explanation, first in an oblique fashion, by examining the transferability of models to tasks that are distinct from but related to the tasks for which they were trained, and by a comparison with an approach based on exploiting dataset bias using other machine learning methods. Finally it describes the implementation of a more direct approach to model explanation, by using a trained network to perform optimization of inputs with respect to the network as a whole or individual nodes and analyzing the content of the result as well as its utility as a pseudo-pharmacophore.

Table of Contents

List of Tables

List of Figures

Preface

Before I begin, I'd like to thank my thesis advisor, David Koes, and our entire research group, especially Matthew Ragoza, Jonathan King, Paul Francoeur, and Andrew McNutt, as well as the visiting scholars and collaborators who have substantially contributed to the work we've done over the past six years.

I'd like to thank my family, especially my parents, my three siblings (Jeff, Nicole, and Danielle), and all their children and grandchildren, for supporting, challenging, and inspiring me throughout my life.

Finally, I'd like to thank my partner, Sasha, and my lifelong friends near and far, especially Sanjana Gupta, Matthew Marks, Justin Hultman, Elizabeth Pearce, Jeremy Northup, Gabriel Jasmin, Valerie Herrerra, Daniel Papasian, David Alfasi Siffert, Evan Pavloff, Kalynn Ziegler, and anyone else I've inadvertently forgotten; these relationships have made life worth living.

I dedicate this work to the memory of my cousin Terry - wish you were here.

1.0 Introduction

Developing new drugs is time-consuming and expensive[126]. The process of navigating a drug through testing and trials to clinical application involves several phases, with failures at each phase deriving from different causes and late-stage failure resulting in significant sunk cost [45]. Identifying or designing truly novel drugs is scientifically challenging and commercially risky. Employment of computational approaches to rapidly identify and optimize promising (and perhaps novel) candidates and eschew late-stage failure therefore tantalizes both the pharmaceutical manufacturer and consumer, though the accuracy and efficiency of these approaches are still lacking [169]. This thesis motivates and describes the implementation of a radical new method in computational drug discovery that attempts to leverage deep learning to significantly improve the accuracy of small molecule drug candidate identification. It assesses this method prospectively, in concert with preexisting state-of-the-art methods. It also considers how machine learning approaches to drug discovery should be evaluated given known problems with available data and attempts to rigorously do so. Finally, it develops a novel approach to model explanation that has potential applications to other aspects of the drug development process as well.

The field of structure-based drug discovery encompasses several conceptually related tasks. These include predicting whether or not each among a set of compounds will bind strongly to a selected protein target (virtual screening), ranking compounds by the strength of their binding activity, accurately predicting the experimental binding affinity of those that do bind, predicting how a compound and binding site interact and reconfigure during binding (binding pose prediction), and predicting how changes to the structure of the protein or compound will modify the way they interact [14]. Structure-based approaches are contrasted with ligand-based approaches; the former class is based on and makes predictions about the relative atomic locations of the protein and compound during binding, while the latter makes predictions based on compound similarity to a query ligand compound known to bind to the target of interest [46]. Structure-based approaches are therefore attractive due to their potential to identify truly novel compounds that bind to a particular target, and they have a broader domain of applicability in terms of the targets for which they can be used (e.g. those that have no currently known drugs) and the information they provide about the intermolecular interactions from which their predictions derive.

Structure-based approaches are based on one or more functions used to map input atomic locations for a protein and compound to predictions about their binding (e.g. a measure of interaction strength, including both direct predictions of binding affinity as well as more general scores that may just produce compound rankings intended to correlate with binding quality). These scoring functions are commonly grouped into four major classes - force fields, empirical, knowledge-based (i.e. statistical), and machine learning. Force fields [68, 205, 25, 32, 55, 19, 112, 92, 91] derive from physics and directly model interatomic potential energy with terms representing electrostatics parametrized using experimental data and quantum mechanical calculations. Empirical scoring functions [98, 187, 53, 15, 195, 100, 58] feature terms expected to underlie or correlate with binding, including protein-ligand interaction terms as well as terms that represent solvation and entropy; these terms may be included in a weighted sum with weights and other tunable parameters fit to experimental data. Knowledge-based statistical potentials [78, 125, 63, 210, 123, 10, 77] use existing structural data to identify conditional probabilities of observing specific features, for example pairwise interatomic distances, torsion angles, or hydrogen bond geometry, which can be used to evaluate an input structure by comparing its values for the relevant features to the experimental distributions. Modern machine learning scoring functions [111, 50, 11, 69, 203, 161, 110, 145] may use a wide variety of input features, including any of the feature classes used by the other types of scoring functions. They tend to impose fewer restrictions on the final functional form compared with other methods, often making predictions that are a nonlinear function of the input features; function parameters are fit to data.

Machine learning scoring functions may combine many types of input features, and this includes starting from precomputed features or attempting to learn relevant features directly from data. Starting from existing precomputed features yields models that are easily interpretable, but that may be limited in their ability to outperform conventional methods since they rely on extracting additional performance from the same basic information conventional methods use. The great promise of these methods is arguably their ability to start directly from a representation of structural data and derive potentially novel informative features during model training (i.e. when their parameters are fit).

When attempting to learn features from input data, there are still choices to be made about how that data will be represented. There are many formats commonly used for molecular representation, including strings [200, 102], fingerprints[84, 190, 51, 120, 41], graphs [95, 64, 159, 207, 29], summaries of the atoms and bonds that are neighbors of each input atom (i.e. "atomic environments") [171, 157], and more general numerical grids (which might include vectors of coordinates and atom types or discrete samples of underlying continuous atomic or electrostatic densities) [145, 90, 81, 174, 139, 153, 8, 164]. Most of these methods have undergone significant innovation in tandem with the machine learning revolution of the 2010s, including a particular interest in bringing a structural component to previously ligand-only representations (e.g. developing protein-ligand interaction fingerprints rather than relying on ligand-only fingerprinting, and developing graph representations based on spatial adjacency rather than exclusively based on bond connectivity). Distinctions remain between methods that impose specific constraints on the kinds of features that may be learned (e.g. in order to guarantee energy conservation or equivariance to natural symmetries) and those that perform minimal featurization in order to provide maximum freedom in learning informative features.

Our group was the first to report a complete, public implementation of voxel-based convolutional neural networks (CNNs) for drug discovery (initially focusing on pose prediction and virtual screening) [145], for which we take the latter approach and perform as little overt featurization as possible. Starting from a continuous representation of atomic positions, we create a grid-based input by sampling the input at discrete locations on a regular three-dimensional lattice, with separate "channels" for each permitted atom type. Work I performed at the beginning of my PhD (Chapter 2.1) helped justify this approach by demonstrating the low ceiling on expected performance improvement imposed by restricting the space of possible input features to those already in wide use in the field. Next we worked on developing our own novel approach(Chapter 2.2), with my contributions including an algorithm efficiently implementing conversions to and from our chosen grid-based input representation. As we gradually honed our training strategy and imagined new applications that would benefit from access to libraries available for machine learning in the Python programming language [189], I performed an important role in developing a general-purpose library, `libmolgrid`, which exposes fundamental data re-sampling, augmentation, and grid conversion capabilities for individuals interested in using grid-based approaches for molecular modelling in their own work, especially when that involves machine learning. `libmolgrid` features generic functionality, efficient performance, interoperability with several deep learning libraries, and native provision of data resampled and augmented in the manner we have determined is necessary for effective machine learning from molecular data. These virtues enable our group and others who have adopted it to address new research questions, including novel gridding and data representation approaches (which may learn distinct classes of features) and machine learning from binding *dynamics*.

The expressiveness afforded by a representation that imposes few limits on the features on which it bases its predictions is a virtue that comes at a price. Of course all scoring functions fit to data require some care when their performance is assessed, at least if the goal is to assess how they might be expected to perform prospectively rather than merely how well they were fit to their training data. Cross validation is typically used to assess performance on a held-out subset of the data not used for training, with clustered cross validation serving as the best way to reduce leakage of information from the test set to the training set when performing fitting and evaluation on a single data corpus [115]. The utility of cross validation and indeed benchmarking datasets as a whole is limited by their own bias [82, 26]. Biases may include lack of diversity in protein classes included in the dataset, high similarity among active compounds provided per-target as well as across targets, and significant dissimilarity between actives and inactives in terms of simplistic features that may correlate with activity due to historical artifacts deriving from the drug development process, among other problems. Benchmarks in wide use for training and validation in the virtual screening literature (including DUD [76], DUD-E [128], and MUV [151]) are known to have biases that seriously undermine their utility in estimating generalization performance [192, 184, 30, 165]; their continued use at the present time is assured due to a lack of alternatives and a need to compare with prior work reported in the literature, but better baselines are required for performance estimates of machine learning methods to control for this bias. Additionally, since this bias in some cases derives from historical bias in the underlying data, it is likely that the bias problem is present in *all* available drug discovery datasets unless they have been explicitly debiased. This thesis demonstrates that the problem is larger than was previously determined, with bias affecting binding affinity benchmarks as well as virtual screening benchmarks, and that it is possible to fit models that exploit bias in one benchmark and make accurate predictions on another benchmark for a distinct drug discovery *task*. This potentially undermines the basis of machine learning performance reported previously in the literature which does not appear to outperform the performance achievable via exploitation of dataset bias, including in areas like binding affinity prediction that were previously untouched by this dataset bias controversy (Chapter 3.2). This thesis proposes an improved performance baseline motivated by exploiting bias as an alternative to baselines based on more sophisticated ligand-based methods, which due to their less expressive features may be unable to learn biases and therefore underperform compared with methods that are secretly learning something even more naive than a fingerprint.

Real-world performance involves making predictions about completely new data with answers blinded when predictions are made. This happens during a real drug discovery campaign, but it can be simulated using blinded community benchmarking challenges that invite predictions about real drug discovery tasks for unpublished data and reveal answers at the end of the challenge period. Examples in various subfields of drug discovery include CASP [124] (protein structure prediction), CSAR [134] (virtual screening, binding affinity prediction, and pose prediction), and D3R [177] (virtual screening, binding affinity prediction, and pose prediction). These benchmarks were also used to evaluate our novel voxel-based CNN scoring functions, and our progress in these blinded benchmarks as we moved from predefined features to those learned from data is also discussed in this work (Chapter 3.1).

Finally, as suggested above, model interpretation is a fundamental problem with more expressive "black box" machine learning models that may learn features from data but do not immediately reveal what those features are. Without a direct indication of what a model has learned, it is possible for the model to have learned something that is trivial or not expected to generalize and this reduces the value of the resulting models in higher cost scenarios. Our group has explored various model interpretation methods in the past [72], mostly focused on mapping a prediction back onto regions of the input that contribute to it. This type of approach can be limited since it does not directly indicate what the network is detecting, only that it was positively or negatively influenced by certain structures in the input, and in some cases that positive or negative influence may be weak and spatially delocalized in a way that provides very little explanatory power. Direct visualization of the structures that maximally activate a convolutional filter is possible, and this approach is implemented and explored in the final chapter of this thesis (Chapter 4). That approach involves using a trained network to optimize its own input with respect to any of its internal neurons, and can also be used (conditioned on a binding site of interest) to generate the network's depiction of the ligand that would optimally bind a given protein target. These "optimal binder" ligand densities can be used to perform a virtual screen over a conformer library in a manner that is distinct from (and faster than) using the network to rescore the conformers directly. This input optimization approach is novel in the domain of machine learning for drug discovery and could be useful for black-box model interpretation and rapid virtual screening.

1.1 Significance

The major contributions of this thesis are:

1. Chapter 2.1 examines how much performance can be gained over conventional scoring functions if machine learning (ML) is used to fit new functions using the same set of features used by the conventional methods. The result motivated our development of a novel ML scoring function that learns its own features from data.
2. Chapter 2.2 describes `libmolgrid`, our library that provides gridding and input batch composition functionality that distills what we have learned about training these machine learning models as we have iterated on training and constructing molecular machine learning scoring functions using voxelized grids derived from available structural data. The efficient implementation enables the development of novel modeling approaches like machine learning from dynamical information.

3. Chapter 3.1 prospectively evaluates our CNN models in the third D3R community benchmarking challenge, where our submission was among the best submitted for affinity ranking. It provides some evidence that allowing the CNN to contribute to pose sampling (rather than just rescoring poses generated by another method) may enhance its performance.
4. Chapter 3.2 provides evidence that dataset bias is likely to affect *any* dataset used for drug discovery that has not been explicitly debiased, and that bias learned from one dataset and one drug discovery task (e.g. binding affinity prediction) provides information that can be used to achieve significantly better than random performance on other datasets and tasks (e.g. virtual screening), while not providing any insight that would generalize to challenging real-world drug discovery tasks. It also demonstrates that while other ML scoring functions may have just learned to recapitulate their training dataset bias, our latest CNN scoring functions *have not*.
5. Chapter 4 implements a novel method for examining what a model has learned and evaluates an extension of that method for virtual screening.

1.2 Outline

In Chapter 2 we justify the development of a novel machine learning scoring function based on a prospective evaluation of the performance of techniques that reflected the state-of-the-art at virtual screening and binding pose prediction at the beginning of my thesis work. Then we discuss relevant details of implementing and testing our novel voxel-based CNNs and describe the API for a general-purpose library we developed to facilitate the community's evaluation and extension of our approach.

In Chapter 3 we subject the resulting CNN-based scoring functions to a similar prospective evaluation as performed at the beginning of Chapter 2, with promising results. This evaluation relied in part on poses that were refined using the CNNs themselves. We then perform a more rigorous assessment of the CNN's ability to generalize and transfer what it has learned to virtual screening, a task for which it was not trained, comparing this method with other machine learning-based approaches and carefully analyzing the results for information about what the CNNs have learned and how that relates to their training data. In particular we establish a performance baseline that thoroughly exploits known chemical biases in available data in an attempt to delineate the performance that might be expected if those biases reflected the total information a given machine learning model had extracted during training.

In Chapter 4 we implement and test a more direct approach to model explanation, using trained networks to optimize inputs with respect to either the full network or to individual nodes. This allows us to interrogate which features the network has actually learned to be optimal, and to potentially exploit that information in the context of virtual screening. We describe how this approach was implemented and preliminary results.

Chapter 5 consists of a final discussion and conclusions, as well as an overview of future work that could take “unfinished business” from this thesis as its starting point.

2.0 Grid-based CNNs: Why and How

2.1 A prospective evaluation of preexisting approaches

The first half of this chapter describes our initial prospective evaluation of training simple machine learning models to use 2D fingerprints and precomputed 3D structural descriptors for pose and affinity prediction. We used these models to submit predictions to the Drug Design Data Resource (D3R) Grand Challenge 2. The results demonstrated the potential of training neural networks using three-dimensional information, since they had a broader domain of applicability than the fingerprint-based models and performed comparably to the best fingerprint models at affinity ranking; still, they did not consistently outperform Vina, our empirical scoring function baseline.

Chapter 2.1 was reproduced with permission from Sunseri, Jocelyn, Matthew Ragoza, Jasmine Collins, and David Ryan Koes. "A D3R prospective evaluation of machine learning for protein-ligand scoring." Journal of computer-aided molecular design 30, no. 9 (2016): 761-771.

2.1.1 Background

A scoring function that accurately represents and predicts ligand-protein interactions is essential for molecular docking, energy minimization, and hit identification/lead optimization in structure-based drug discovery [43, 121, 27, 197, 97, 198, 32, 31, 173, 79]. The development of an accurate and reliable scoring function remains an unsolved problem [42, 156, 74, 148, 196, 79]. Ideally, given a protein-ligand structure, a scoring function would be able to correctly place the true, crystal pose of a ligand at a global minimum (pose prediction) and, if provided poses at this global minimum, correctly distinguish between active and inactive ligands (virtual screening performance) by producing scores equivalent to the binding free energy (binding affinity prediction).

Scoring function design philosophies generally span a continuum between force-field based scoring, empirical scoring, and knowledge-based scoring. Force-field based scoring [68, 205, 25, 32, 55, 19, 112, 92, 91] attempts to compute the physical interaction of the protein and small molecule and includes terms such as van der Waals and electrostatic interactions. These terms are typically parameterized from first principles. Empirical scoring functions [98, 53, 15, 195, 100, 53, 58, 186] include physically meaningful terms that may not directly map to physical forces, such as hydrophobic terms, and are parameterized to reproduce binding affinities or other data. Knowledge-based scoring [78, 125, 63, 210, 123, 10, 77] takes advantage of the growing amount of structural data to derive statistical potentials for ligand-protein interaction patterns.

Parametric machine learning methods, such as linear regression, are often used to parameterize empirical scoring functions. In contrast, non-parametric machine learning methods, such as neural networks [152, 107], provide greater flexibility and expressiveness as they learn both their model structure and parameters from data. Such methods have successfully been applied to scoring protein-ligand interactions [7, 93, 154, 47, 10, 35, 212, 93, 155, 48, 49, 40]. These scoring functions take as input a set of descriptors extracted from a protein-ligand complex. These descriptors are either terms common to empirical scoring [48], such as measures of electrostatic attraction, atom interaction counts [49], or more abstract interaction fingerprints [35]. A disadvantage of non-parametric methods is that their increased expressiveness increases the probability of overfitting the model to the data, in which case the scoring function will not generalize to protein targets or ligand chemotypes not in the training data. The risk of overfitting increases the importance of rigorous validation [101, 59], but the inherent increase in flexibility allows non-parametric methods to outperform more constrained methods when trained on the identical input set [108].

As our entry in the Drug Design Data Resource (D3R) blind challenge, we investigated a variety of machine learning techniques. We evaluated both structure-based classification models and ligand-based regression models. For our structure-based classification we explored using the DUD-E dataset [127] for training. In contrast to the CSAR dataset [173] we have previously used [98], DUD-E is much larger (more than 1 million ligands), but lacks crystal structures for its more than 22,000 active ligands. Our goal in entering the D3R evaluation was to prospectively assess the performance of using structure-based training

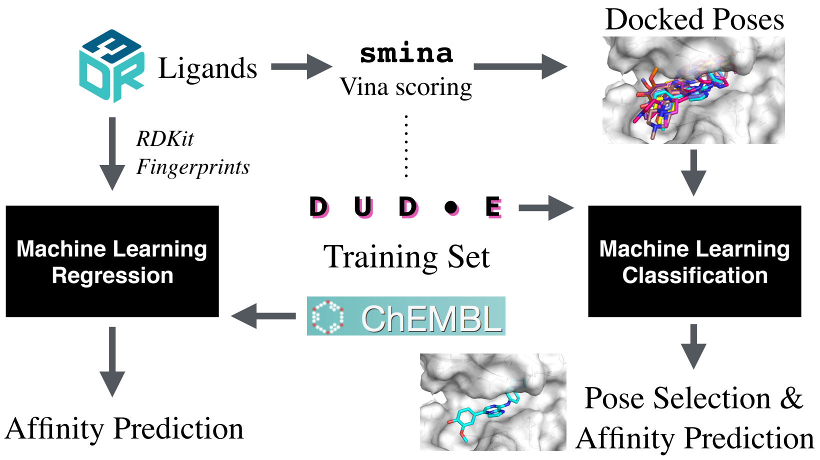


Figure 1: The overall approach for our D3R 2015 Grand Challenge submission. D3R ligands were ranked using a 2D QSAR approach trained using ChEMBL data (left side) or through a structure-based docking and scoring approach that used the DUD-E data set to train custom scoring functions for re-ranking poses docked using smina and the AutoDock Vina scoring function (right side).

with generated DUD-E poses with both parametric and non-parametric machine learning methods while also evaluating a purely ligand-based QSAR method.

2.1.2 Methods

Our overall approach is shown in Figure 1. We considered both a ligand-based Quantitative Structure Activity Relationship (QSAR) approach and a structure-based docking and scoring approach. For the ligand-based approach we train a regression model from binding affinity data using RDKit [149] and a variety of chemical fingerprints. For the structure-based approach we make extensive use of smina [98], a fork of AutoDock Vina with enhanced capabilities for custom scoring function development, and the AutoDock Vina [186] scoring function. We evaluated a unique approach where we train *classification* models on binary binding data and used these classification models to rank and select docked poses.

The 2015 D3R Grand Challenge consisted of both affinity prediction and pose prediction exercises for two blinded collections of compounds for two targets: Heat Shock Protein 90 (HSP90) and mitogen-activated protein kinase kinase kinase kinase (MAP4K4). 180 ligands were provided for HSP90 with IC50 activities ranging from 5nM to inactive and six crystal structures were left blinded as part of the competition. The MAP4K4 dataset consisted of 30 compounds, all with co-crystal structures, but only 18 of which had measured IC50 data. Consequently, the HSP90 target is most useful for assessing binding affinity prediction and virtual screening performance while the MAP4K4 target is most suited for pose prediction evaluation.

2.1.2.1 Ligand-Based Regression The goal of 2D QSAR modeling [33] is to generate a predictive model of a desired property, in our case binding activity, from a training set of molecules with known activity using descriptors generated from the 2D topology of the compounds. Using three different 2D fingerprint descriptors, we created three different models for HSP90 binding from the same training set. The code used to build our models is available under a permissive open source license at `https://github.com/dkoes/qsar-tools` and complete details of our approach are provided in the Supplementary Information.

Compounds with published activity were extracted from the ChEMBL bioactivity database [12]. Specifically, we collected active compounds from the CHEMBL3880 target (HSP90 alpha) that had IC50 values with an equality relation expressed in nM units and a pChEMBL greater than zero (this is a negative logarithm used to standardize across different activity measurements). The resulting 355 active compounds spanned a pChEMBL range from 4 (100μM) to 9 (1nM). These compounds were then stripped of any salts and a variety of descriptors were calculated.

We calculated Boolean fingerprint descriptors, which encode a molecule as a binary string where each bit position corresponds to the presence or absence of a specific pattern of atoms in the molecule. We evaluated three different fingerprints: default RDKit (2048 bits), unfolded path (variable bits), and circular ECFP6 (2048 bits).

The default RDKit fingerprint enumerates all paths (including branched paths) of a molecule with up to 7 bonds. These paths, including their atom and bond type information, are then doubly hashed to a bit position within a 2048 bit vector. The use of a constant fingerprint size means the fingerprint is general and can be broadly applied for similarity calculations, but introduces the likelihood of collisions where the same bit position corresponds to multiple distinct atom patterns.

For our unfolded path fingerprints, we enumerated all possible unbranched paths, including atom and bond type information, present in the molecules of the training set resulting in 6628 distinct atom patterns. Each path was then assigned a unique bit in a bit vector. In this case, every bit in the fingerprint unambiguously corresponds to a specific atom pattern. If new atom patterns are encountered when fingerprinting molecules not in the training set they are ignored.

Extended connectivity fingerprints [150] enumerate atom patterns that represent the neighborhood of each atom up to a circular diameter, in our case 6, of bond lengths. These descriptors are then folded into a 2048 bit fingerprint with RDKit.

We create predictive linear models from the training set and descriptors using the ElasticNet module of the popular scikit-learn [141] Python package. An elastic net model includes both an L1 and L2 regularization factor:

$$\underset{w}{min} \frac{1}{2n_{samples}}||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$$

where X are the input binary features, y are the labels (affinity values), w are the weights of the model, and α and ρ are parameters controlling the degree of regularization. Increased regularization drives weight values to zero, reducing the number of selected features. This reduces the amount of overfitting in the model at the cost of reduced expressiveness. In order to set these regularization parameters we apply an internal cross-validation to identify the best parameters for the training set. Using this approach we achieved internal cross-validation R^2 correlations of 0.52, 0.50, and 0.60 using the default RDKit fingerprints, unfolded path, and circular ECFP6 fingerprints respectively.

2.1.2.2 Structure-Based Classification For our structure-based workflow, which unlike the ligand-based regression also produces pose predictions, we use docked poses to train models to distinguish between binders and non-binders. These models are then used to re-rank and select docked poses of the D3R ligands.

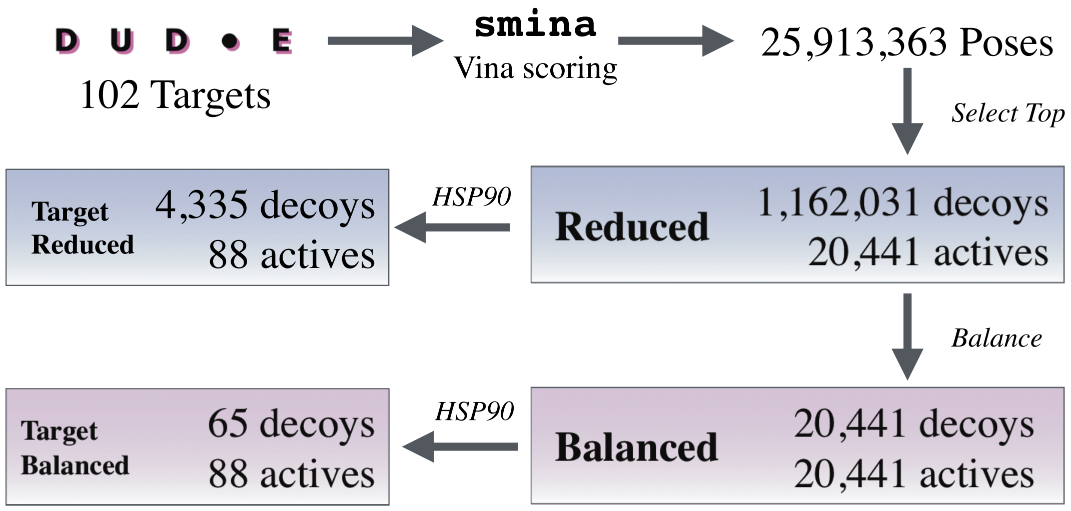


Figure 2: Our workflow for generating structural training sets from the DUD-E dataset.

The workflow for training set construction is shown in Figure 2. Somewhat unconventionally, we chose to train structure-based models using a dataset, the Enhanced Directory of Useful Decoys (DUD-E) [127], that lacks protein-ligand structures. The advantage of DUD-E is its large size: it consists of 102 targets, more than 20,000 active molecules, and more than a million decoy molecules. The disadvantage is that compounds are classified as active/decoy (no binding affinity information) and structures are not available. To address this, we train our models as binary classifiers on docked poses. The docked poses are generated using smina [98] using the AutoDock Vina scoring function. Input ligands are converted to a single 3D conformer using RDKit which is then docked as a flexible ligand (hence the need to only generate a single conformer as rotatable bonds are sampled during docking). We dock against the reference receptor provided with DUD-E in a box centered around the reference ligand with 8Å of padding and select the top ranked pose as the structure to use for that ligand. This results in a highly imbalanced and noisy training set that is dominated by decoys. To enhance the signal exhibited by the active set, we also create a balanced set with equal numbers of decoy and active compounds. Since DUD-E includes an HSP90 target, we also extract a target specific set from these two larger training sets.

Steric	gauss(o=0,w=0.5,c=8)
	gauss(o=3,w=2,c=8)
	gauss(o=1.5,w=0.3,c=8)
	gauss(o=2,w=0.9,c=8)
	gauss(o=1,w=0.9,c=8)
	gauss(o=1,w=0.5,c=8)
	gauss(o=1,w=0.3,c=8)
	gauss(o=1,w=0.7,c=8)
	gauss(o=2,w=0.5,c=8)
	gauss(o=2,w=0.7,c=8)
	gauss(o=3,w=0.9,c=8)
	repulsion(o=0,c=8)
Hydrophobic	hydrophobic(g=0.5,b=1.5,c=8)
	hydrophobic(g=0.5,b=1,c=8)
	hydrophobic(g=0.5,b=2,c=8)
	hydrophobic(g=0.5,b=3,c=8)
	non_hydrophobic(g=0.5,b=1.5,c=8)
van der Waals	vdw(i=4,j=8,s=0,≙100,c=8)
	vdw(i=6,j=12,s=1,≙100,c=8)
	e_vdw
Hydrogen Bond	non_dir_h_bond(g=-0.7,b=0,c=8)
	non_dir_h_bond(g=-0.7,b=0.2,c=8)
	non_dir_h_bond(g=-0.7,b=0.5,c=8)
	non_dir_h_bond(g=-1,b=0,c=8)
	non_dir_h_bond(g=-1,b=0.2,c=8)
	non_dir_h_bond(g=-1,b=0.5,c=8)
	non_dir_h_bond(g=-1.3,b=0,c=8)
	non_dir_h_bond(g=-1.3,b=0.2,c=8)
	non_dir_h_bond(g=-1.3,b=0.5,c=8)
	non_dir_anti_h_bond_quadratic(o=0,c=8)
	non_dir_anti_h_bond_quadratic(o=0.5,c=8)
	non_dir_anti_h_bond_quadratic(o=1,c=8)
	non_dir_h_bond_lj(o=-0.7,≙100,c=8)
	non_dir_h_bond_lj(o=-1,≙100,c=8)
	non_dir_h_bond_lj(o=-1.3,≙100,c=8)
	e_hb
	e_ligPen
Solvation	ad4_solvation(d-sigma=3.6,s/q=0.01097,c=8)
	ad4_solvation(d-sigma=3.6,s/q=0.01097,c=8)
	e_s1
	e_s2
	e_s3
	e_s4
	e_s5
Electrostatic	electrostatic(i=1,≙100,c=8)
	electrostatic(i=2,≙100,c=8)
	e_E0
	e_E1
Counts	num_tors_div
	num_heavy_atoms_div
	num_heavy_atoms
	num_tors_add
	num_tors_sqr
	num_tors_sqrt
	num_hydrophobic_atoms
	ligand_length
	numBonds
	bfO
	bfN
	myRotors

Figure 3: Structure-based descriptors used to train machine learning models. Italicized features are computed outside of smina and include solvent accessible surface area (SASA) atom type specific solvation terms (*es_1-es_6*) and buriedness terms (*bfO*, *bfN*).

We distill every protein-ligand pose in our training set into a numerical vector of interaction features computed using smina and custom code to produce the descriptors shown in Figure 3. The custom code is primarily used to calculate descriptors that include a solvent accessible surface area (SASA) term, since smina only computes pairwise interaction terms. An assortment of parameterized smina terms are used and include steric, hydrophobic, van der Waals, hydrogen bonding, solvation, electrostatic (partial charges computed using Open Babel [136]), and non-interaction count/summation terms. Finally, we also include the AutoDock Vina score for a total of 61 features.

For purposes of internal validation, we first evaluated our models using clustered cross-validation [101] where we partitioned the DUD-E training set at a target granularity into multiple folds. This provides a greater measure of generalizability since trained models are tested on entirely new targets. For training, as we are performing classification, we evaluated the ability of a model to properly rank compounds using the area under the curve (AUC) of the receiver operating characteristic (ROC) curve. A perfect ranking of ligands produces an AUC of 1.0 and a random ranking results in an AUC of 0.5. Models used for our D3R predictions were trained on the entire training set.

Using scikit-learn [141] with default parameters we evaluated both linear regression and logistic regression, which is more commonly used in classification tasks, and found they produced nearly identical results in our cross-validation analysis. They both achieved an average AUC of 0.77 in 10-fold cross-validation when trained on the balanced training set. Since linear models are faster to evaluate and train, have more interpretable coefficients, and produce a wider range of prediction values (unlike logistic regression which is capped between zero and one), we selected a linear regression model for our D3R submission.

As an additional model, we trained a neural net with a single hidden layer of 20 nodes and two output classes (active or decoy) using the Caffe deep learning framework [85]. The model used sigmoid activation in the hidden layer and a softmax function to normalize the output. It was trained by stochastic gradient descent with an inverse learning rate decay function ($\eta_i = 0.01, \gamma = 0.0004, power = 2$) and momentum ($\alpha = 0.9$) to minimize the multinomial logistic loss. Training occured for 10,000 iterations with a batch size of 20,000 examples. When trained on the unbalanced dataset, class weights were applied when computing the loss to balance the influence of the negative examples with the underrepresented positive examples. The 10-fold cross-validated AUC for the model was 0.74 using the balanced dataset and 0.73 on the unbalanced dataset.

2.1.2.3 Test Set The provided SMILES of the D3R ligands were converted into a single conformer with RDKit and then docked with smina [98]. The binding site was defined using the cognate ligand of the receptor. Unlike with the training set, we increased the amount of sampling performed during docking (`--exhaustiveness 50`) to increase the chance of identifying high-quality poses. For the HSP90 target we limited ourselves to the four receptors referenced by the D3R organizers: 2JJC, 2XDX, 4YKR, 4YKY. Since the presence of waters was explicitly called out by the organizers, we docked to variations of these structures with zero, one, or two waters within the binding site. For each receptor structure, we generated up to 9 distinct poses for a total of 21,893 poses.

For MAP4K4, we also limited ourselves to the two structures referenced by the organizers: 4OBO and 4U44. In this case, since the organizers explicitly called out the flexibility of the structure we ran a 100ns molecular dynamics simulation using Amber14 and and the amberff14sb force field with TIP3P water under neutral conditions. In order to prepare the structures for simulation, we modeled missing loops as needed with the FREAD loop modeling server [34] and PyMOL. A greedy top-down clustering algorithm was then used to select ten diverse, as measured by backbone RMSD, frames from the 100ns simulation. The distributions of sampled backbone RMSDs are shown in Figure S1. Compounds were docked to these ten structures and the original crystal and up to 9 distinct poses were generated for each receptor for a total of 5,329 poses.

Our linear regression and neural network models were applied to all generated poses and the best scoring poses for each ligand were submitted as our predicted poses and the score of the best scoring pose was used for our submitted affinity predictions.

2.1.3 Results

The MAP4K4 and HSP90 datasets each served as a specialized evaluation task based on the nature of the data available: MAP4K4 had 30 blinded crystal structures that served to test pose prediction performance, while HSP90 had blinded affinity data for 180 ligands that could be used to evaluate virtual screening methodologies. Both sets had a relative paucity of data available for the other task - affinity data for 18 ligands in the case of MAP4K4 and crystal poses for 6 ligands in the case of HSP90 - and we thus focus much of the analysis of our performance on each task on the dataset suited to that task.

There are several axes of analysis, each elucidating the utility of a particular method we used to train and select classification models as well as generate instances to test them. Broadly, there are differences in the type of classification model (linear regression, linear regression including an L1 regularization term, and a neural net), the dataset used to train the classifier (the balanced and reduced datasets, and the targeted datasets in the case of HSP90), and the methods used to generate an ensemble of receptor structures used to created poses for the test sets. Although we did not submit the predictions from our linear classifier with an L1 lasso to the challenge, we include the data from its predictions here for the purposes of evaluation.

2.1.3.1 Pose Prediction Given the 30 MAP4K4 crystal poses released at the close of Stage 1 of the challenge, we computed RMSDs of our predicted poses to the provided 4OBO aligned crystal poses. This information was then used to assess the performance of our training and testing methodologies across all the axes described above. In general we wanted to know whether it was more likely to observe low RMSD poses using a particular methodology, either in the top ranked pose for a given ligand or considering a subset of the top-ranked poses.

Since a scoring function's ability to rank low-RMSD poses is limited by our ability to sample low-RMSD poses, we first focus on our pose sampling in the test set. Figure 4 shows that half of the ligands have at least one pose under 2.0 RMSD with the 4OBO crystal structure, which outperforms all other receptor structures in generating low RMSD poses. One of the 4U44 ensemble receptors outperforms the 4U44 crystal structure at generating low RMSD poses, successfully yielding a pose under 2.0 RMSD 37% of the time, compared to 23% of the time with the crystal structure. However, that structure is the first frame from the molecular dynamics simulation, suggesting that the pre-simulation minimization of the crystal structure may have been sufficient to produce better poses with 4U44. Of the 23 ligands for which we successfully sampled a pose under 2.0 RMSD, docking to one of the crystal structures was sufficient to produce such a pose for all but one. However, considering the lowest RMSD pose available in the test set for each of the ligands reveals that 14 ligands exhibited their lowest RMSD pose when docked to a simulation derived receptor rather than a crystal receptor structure, suggesting there was some value in performing the ensemble docking.

Figure 5 shows the mean across all ligands of the RMSD of the best pose seen so far at a given rank for each of the methods used to score and then rank poses. It indicates that for the majority of the values shown, the scoring functions trained on the reduced datasets outperformed those that were trained on the balanced datasets. The linear regression scoring function trained with lasso was the method that performed best overall, returning a pose within 4.0 RMSD on average by the fourth ranked pose; however, all methods except Vina (included as a baseline) and the lasso method trained on the balanced dataset returned a pose within 4.0 RMSD on average within the top five ranked poses. On average, no methods returned a pose within 2.0 RMSD in the top 25 ranked poses, despite the fact that 23 ligands had such a pose in the set of poses we generated via docking for the test set.

Figure 6 demonstrates that if only poses generated from the 4OBO crystal structure had been scored, a greater number of poses within 4.0 RMSD would have appeared as the top ranked pose chosen by all of the scoring methods except Vina. The assessment of the poses generated by 4U44 is more equivocal; while the lasso- and neural net-based methods demonstrate improved sampling of low RMSD poses if they are restricted to poses generated using 4U44, only for the lasso method trained using the balanced dataset does the lower

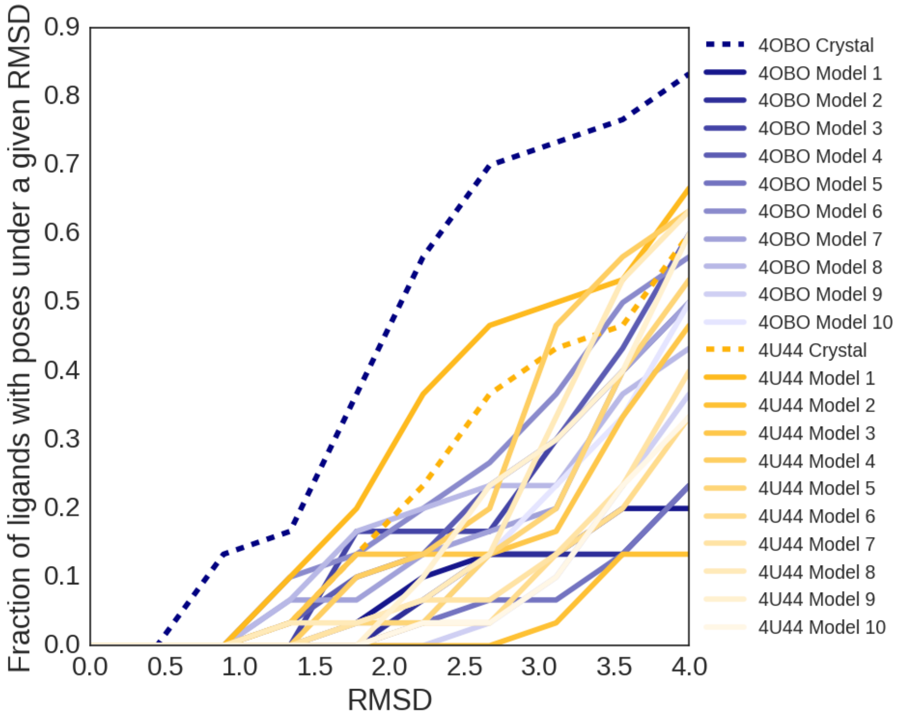


Figure 4: Fraction of ligands with poses under a given RMSD, colored by the receptor structure associated with the subset of poses used for the calculation. The starting PDB crystal structures are shown with dashed lines in the darkest colors, while the structures generated from molecular dynamics simulations are colored according to a gradient based on their frame number, representing their distance from the initial crystal structure used to start the simulation.

quartile improve by nearly 2.0 RMSD, with a weaker effect observed for the other methods. The medians of the linear scoring function rankings improve by around 2.0 RMSD by using the full set of poses generated via the complete receptor ensemble rather than the 4U44 crystal structure.

Figure 7 views the rank1 poses produced by each method based on the ligand with which they were associated. One of our methods (not including Vina) gave top rank to a

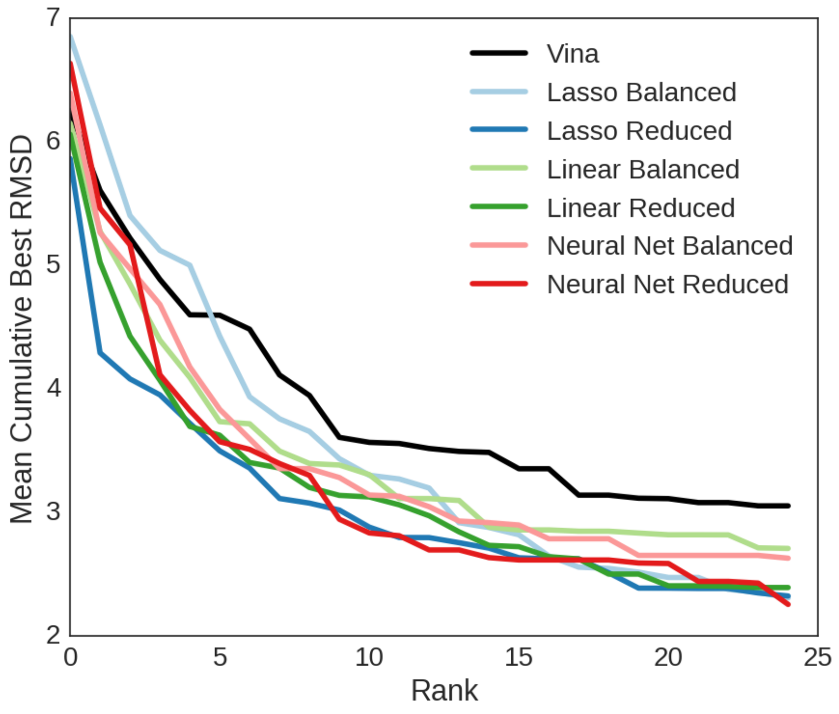


Figure 5: The average of the best RMSD observed for each ligand up to a given rank, compared across all the methods with Vina's performance used as a baseline.

pose within 4.0 RMSD for 13 of the ligands and within 2.0 RMSD for 6 of the ligands. Linear regression or linear regression with lasso, both trained using the reduced dataset, were the methods most successful at selecting low RMSD poses. Of the 17 ligands for which we failed to place a pose under 2.0 RMSD at rank 1, seven had no such pose in the dataset. The remaining ten had at least one pose under 2 RMSD in the dataset, but none of our methods correctly identified any such pose at rank 1. The ligand the methods had the most trouble with, despite the presence of a low (< 1Å) RMSD pose, was MAP07, which is shown in Figure 8. MAP07 has a cyclopropane group that is solvent exposed in the crystal, but all our methods (and Vina) prefer poses where this group is more buried.

A similar analysis was performed for the 6 ligands in the HSP90 dataset for which

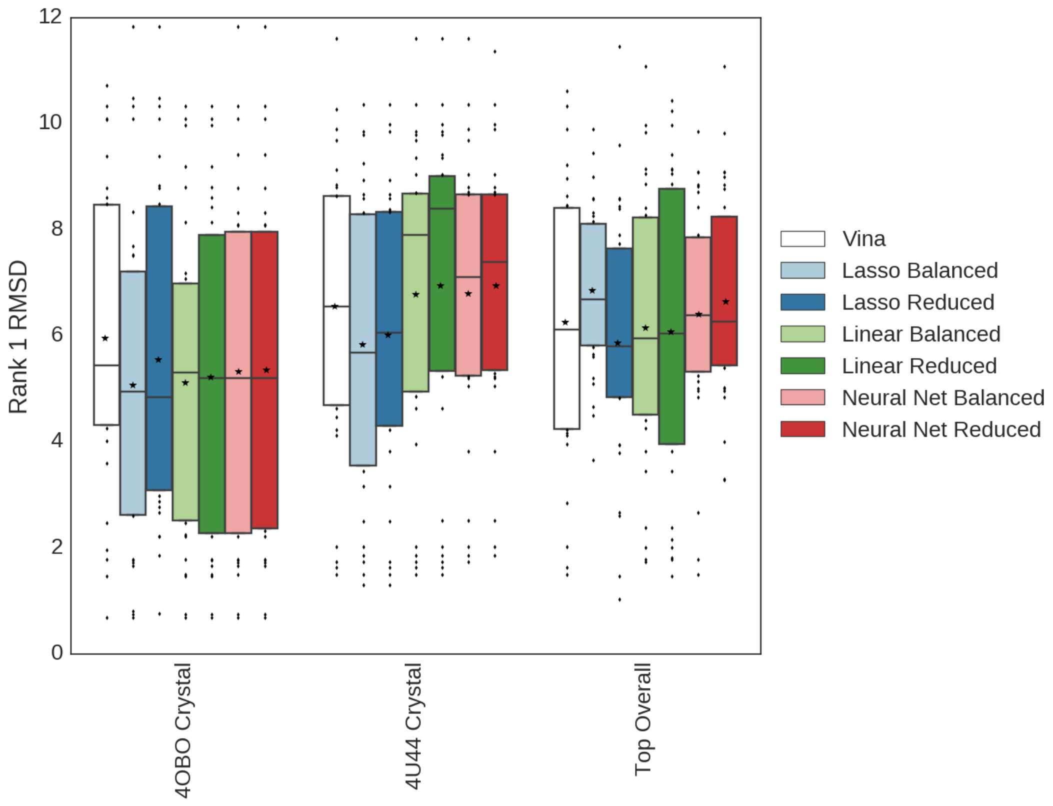


Figure 6: Assessment of the mean rank 1 pose RMSD across all 30 ligands, comparing the different classifiers and the methods used to train them, as well as the receptor structures used to generate poses. Boxes show quartiles, lines bisecting the boxes indicate the location of the median, while stars indicate the location of the mean.

crystal structures were made available at the end of the challenge. The linear regression model trained on the balanced dataset was the best performing method on that task, with an average rank 1 pose RMSD of 1.9. However, Vina was more successful at predicting the lowest RMSD poses found in the top 5 ranking, reaching 0.98 RMSD by rank 5. The other methods performed significantly worse than Vina at every rank, and there was no clear consensus regarding whether training with the balanced or the reduced datasets proved advantageous for this task. The targeted training set produced scoring functions that performed the worst overall, generating poses that were on average 1-2.5 RMSD worse than those generated by

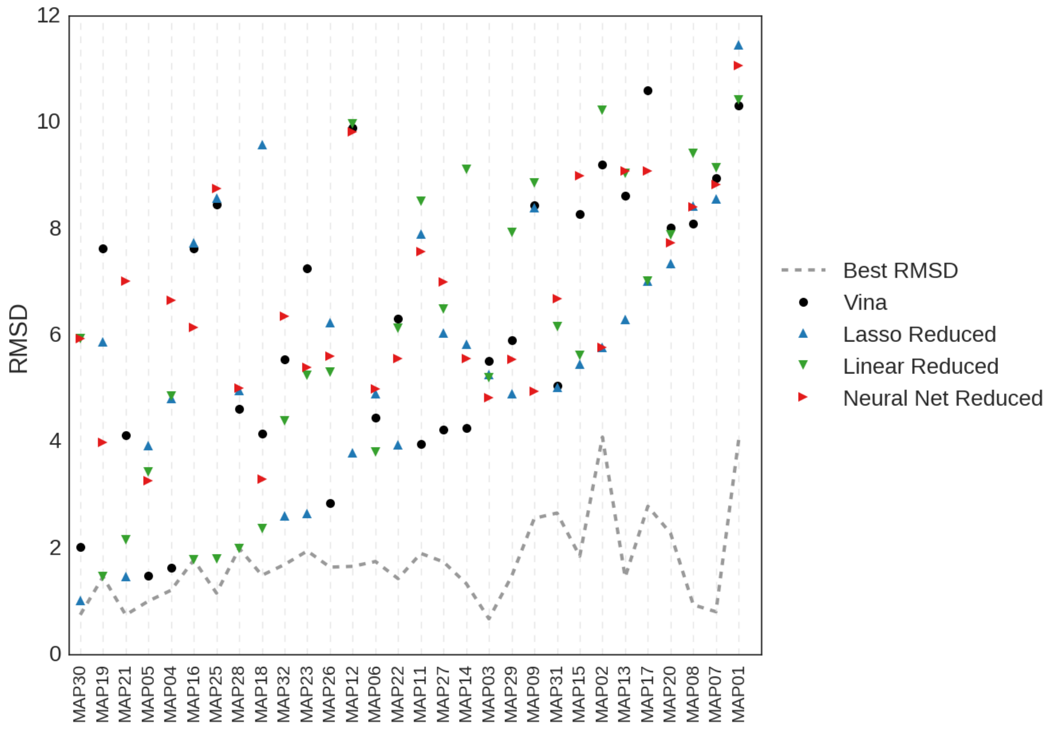


Figure 7: RMSD of the rank 1 pose each method selected for each ligand in the MAP4K4 test set. The dashed gray line shows the lowest RMSD that could have been obtained by selecting poses from the test set.

the non-targeted version of the scoring function. In terms of sampling, two of the receptors produced poses within 2.0 RMSD for five of the six ligands, ten receptors produced poses within 4.0 RMSD for all of the ligands, and all receptors produced poses within 4.0 RMSD for four of the six ligands. Of the seven receptors that produced poses within 2.0 RMSD for at least four ligands, five used crystal waters in docking and two did not; of the two receptors that produced poses within 2.0 RMSD for at least five ligands, one used crystal waters and the other did not. This suggests that including crystal waters may enhance sampling of low RMSD poses and, at a minimum, is not detrimental. However, the small size of the HSP90 test set prevents any definitive conclusion.

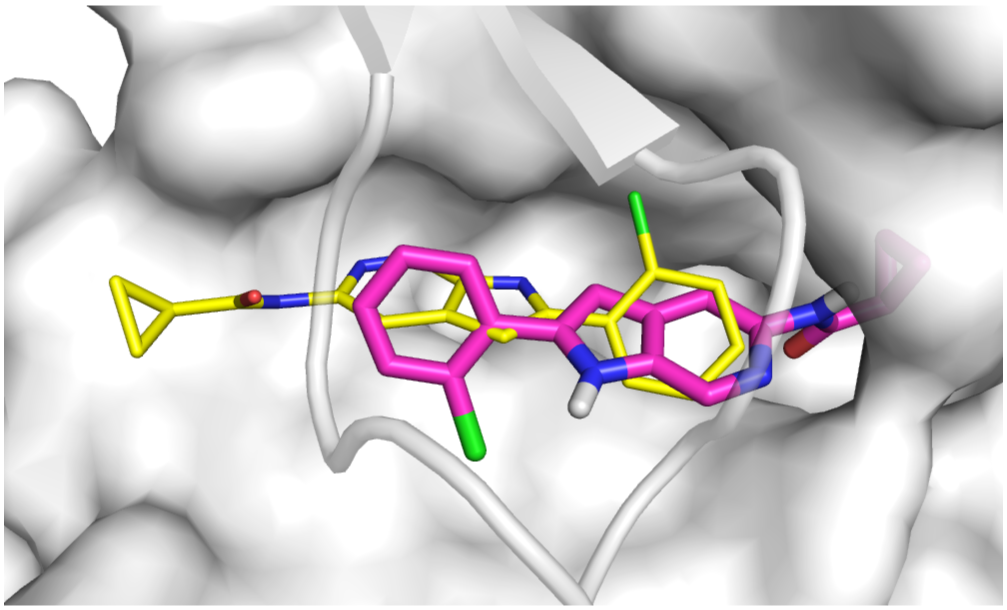


Figure 8: An example of a challenging ligand for pose selection. The crystal pose for MAP07 is shown as thin yellow sticks while a pose top-ranked by a linear method for this receptor is shown in magenta sticks. The surface of the top of the binding pocket is removed for clarity.

2.1.3.2 Virtual Screening Performance

As the HSP90 set of 180 ligands included inactive compounds, it provides a means to evaluate virtual screening performance, that is, how well the various scoring methods discriminate between binders and non-binders. The cutoff for activity was set at 50μM resulting in 136 active and 44 inactive compounds. The score of the top ranked pose selected by each scoring method was used to rank each ligand. The area under the curve (AUC) of the receiver operating characteristic (ROC) curve for the various methods is shown in Figure 9 with 95% confidence intervals, and the ROC curves for selected methods are shown in Figure 10. The best AUC of 0.65 was achieved by Vina, but the structure-based methods trained using the balanced set and the ECFP6 ligand-based method all performed similarly with AUCs of 0.63 or better.

Methods trained using the reduced set, in which decoy examples are not down-sampled to balance the effect of active and inactive compounds on training, fared more poorly. Methods trained on the HSP90 target-specific balanced set were worse than random. Of the ligand-

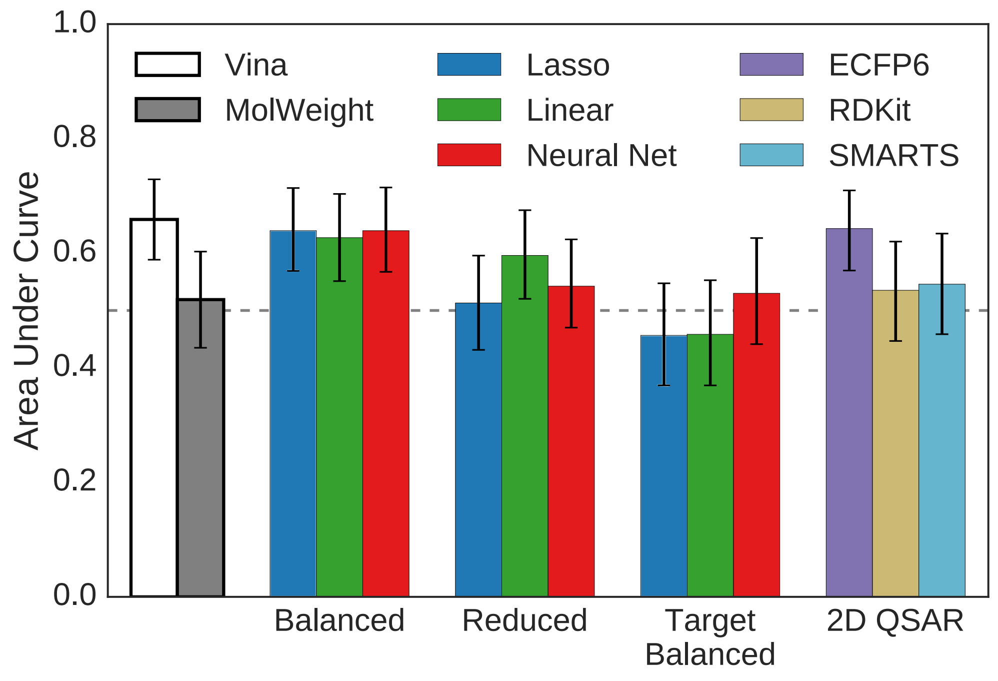


Figure 9: Area under the curve (AUC) of the ROC curves generated using various methods to rank HSP90 ligand poses. Error bars indicate the 95% confidence interval as determined by bootstrapping with replacement (1000 iterations). Compounds with a reported activity greater or equal to 50µM were considered inactive.

based methods, only ECFP6 fingerprints produced an AUC that was meaningfully above random.

Although Vina, ECFP6, and the balanced methods perform similarly, they still score ligands differently, as shown in Figure 11 which plots the ligand scores of the different methods with respect to each other. The structure-based methods are more correlated with one another than with the ligand-based method.

2.1.3.3 Affinity Prediction Both the HSP90 and MAP4K4 sets provide an opportunity to assess the ability of the methods to accurately predict the reported activity. Overall correlations between predicted and experimental activity for the HSP90 ligands are shown in Figure 12. As with the virtual screening results, Vina and the structure-based meth-

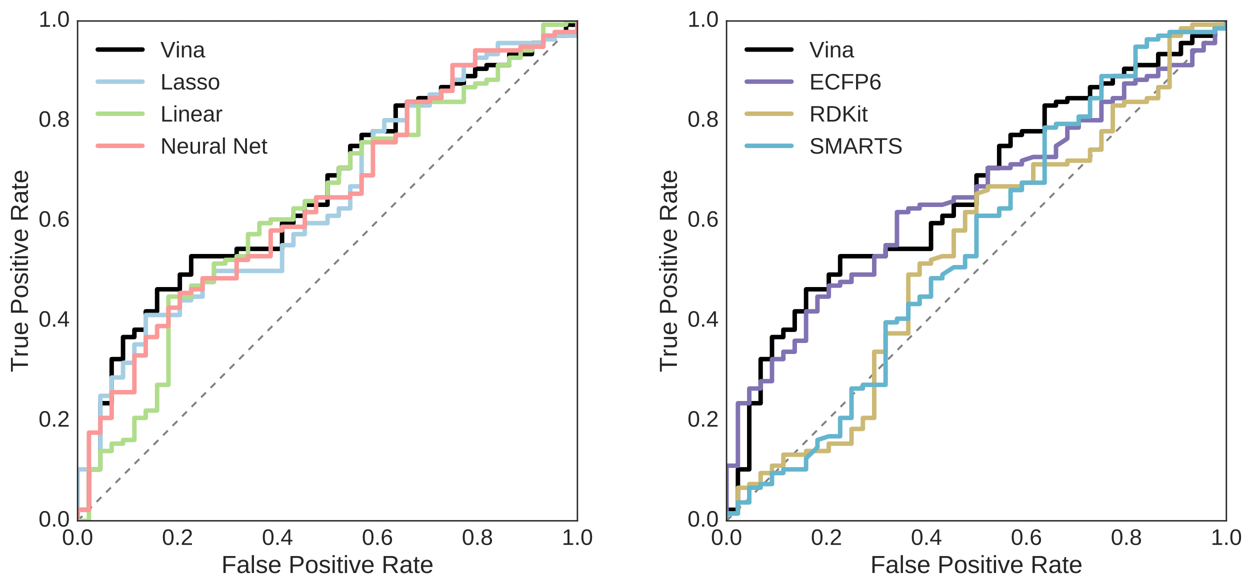


Figure 10: The ROC curves for HSP90 ligands generated using structure-based methods trained on a balanced training set (left) and curves of ligand-based methods (right). Compounds with a reported activity greater or equal to 50μM were considered inactive.

ods trained on the balanced set perform similarly and the methods trained on the HSP90 target-specific training set perform poorly. Methods trained on the reduced set have similar performance to Vina, with the neural net model achieving the highest Spearman correlation coefficient of 0.40. However, this is not a particularly high correlation and is not substantially more than the dataset's correlation with molecular weight (0.34). Both the ECFP6 and RDKit 2D QSAR regression models achieve statistically significant correlations, but do not outperform the structure-based methods.

The MAP4K4 dataset has substantially fewer compounds with reported activities (17 ligands) which, as indicated by the 95% confidence intervals in Figure 13, makes it difficult to meaningfully compare methods. However, since the MAP4K4 dataset has the advantage of providing crystal structures for all 17 ligands, we also evaluated affinity prediction performance when scoring the pose with the closest RMSD to the crystal ligand instead of the pose top-ranked by the scoring function. Interestingly, as shown in Figure 13, scoring this

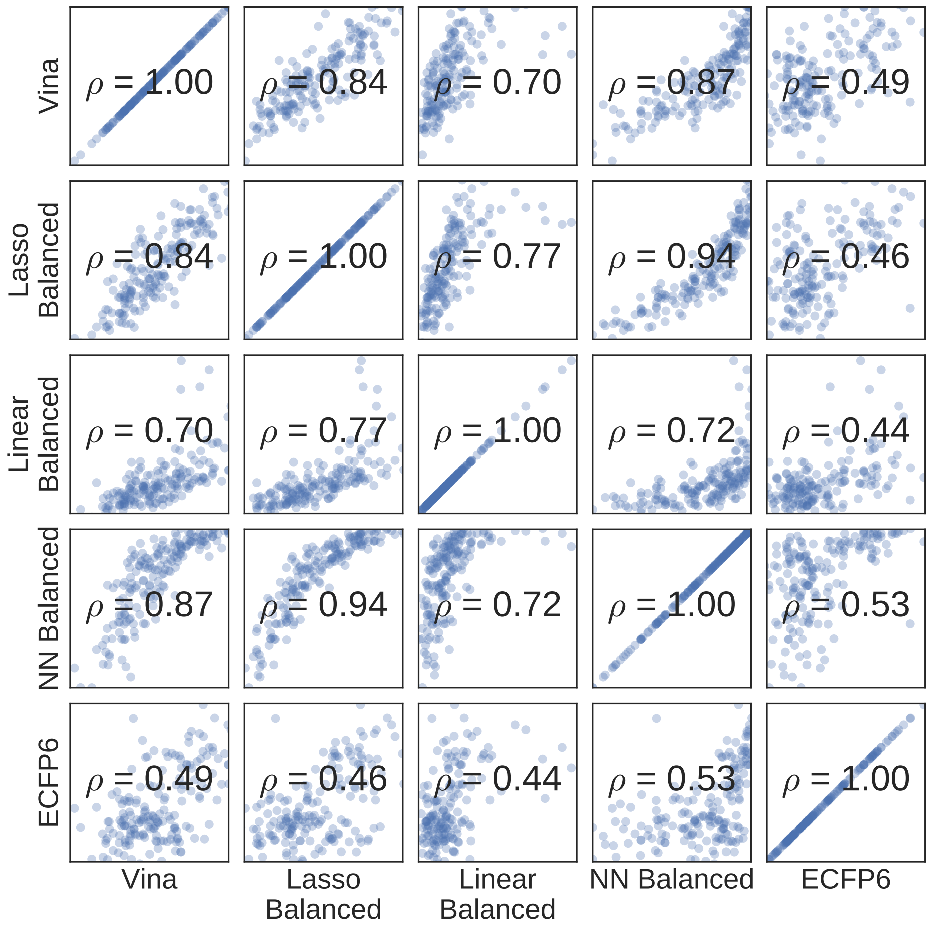


Figure 11: Spearman correlations of HSP90 ligand scores for selected methods. The logit function (an inverse sigmoid) was applied to the neural network score for visualization purposes.

superior pose did not result in improved correlations, statistically significant or otherwise.

2.1.4 Discussion

The 2015 D3R Grand Challenge provided an excellent opportunity to prospectively evaluate pose prediction and scoring methods. We evaluated both structure-based and ligand-based machine learning approaches. Somewhat surprisingly [182], the 3D structure-based methods outperformed the 2D methods for the one target, HSP90, where there was sufficient

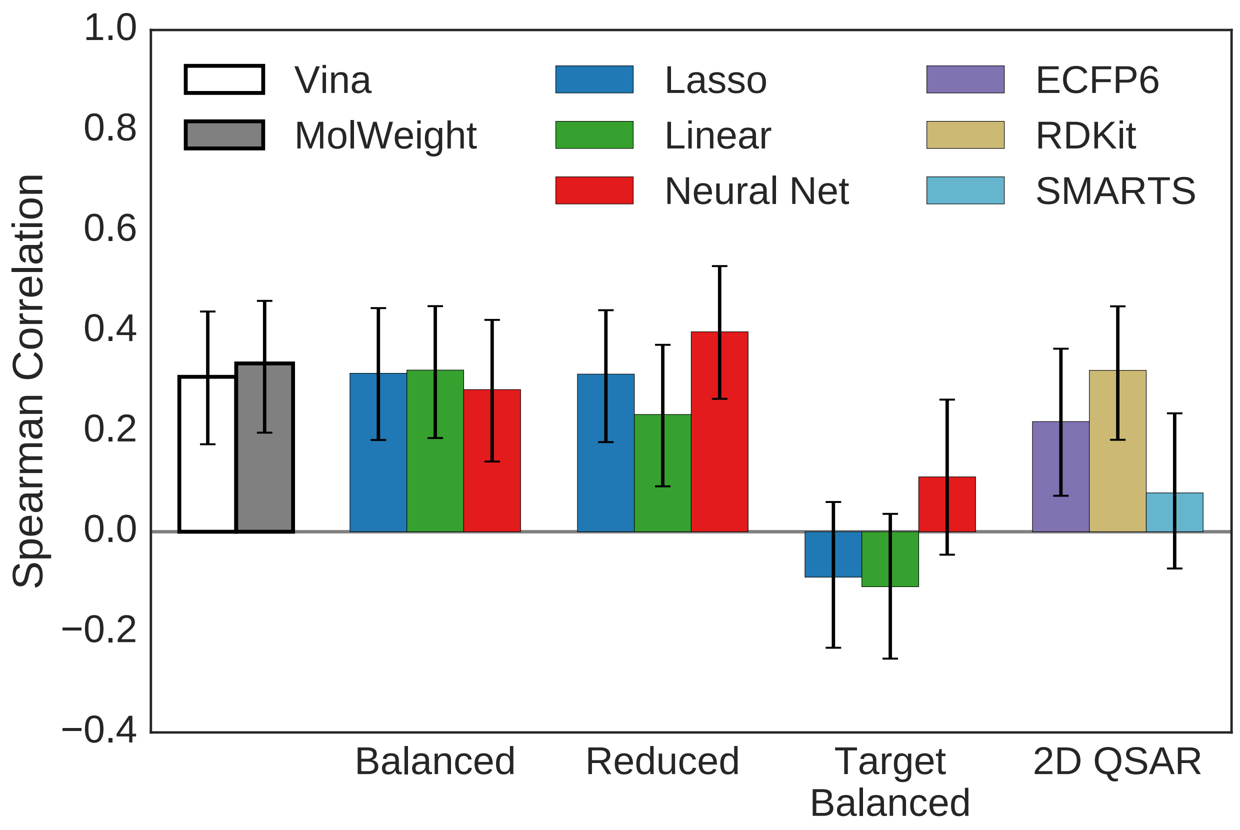


Figure 12: Spearman correlation coefficients using various methods to rank HSP90 ligands. Only ligands with reported IC50s below 50μM were considered. Error bars indicate the 95% confidence interval as determined by bootstrapping with replacement (1000 iterations).

data to construct QSAR models. Although it is possible that the use of more expressive features [116] or models [28] would improve the results, a more likely issue lies in the coverage of the training set with respect to the D3R ligands. Figure 14 show the HSP90 datasets plotted with respect to the first two principal components of the D3R ligands as computed using OpenBabel FP2 fingerprints. The three congeneric series of the D3R set are clearly distinguished as three separate clusters. The ChEMBL dataset used with the ligand-based methods fully covers one cluster but only partially covers the remaining two. In contrast, the DUD-E HSP90 set used for the target-specific structure-based scoring functions has little overlap with the D3R ligands. This property, combined with the set's small size (88 active

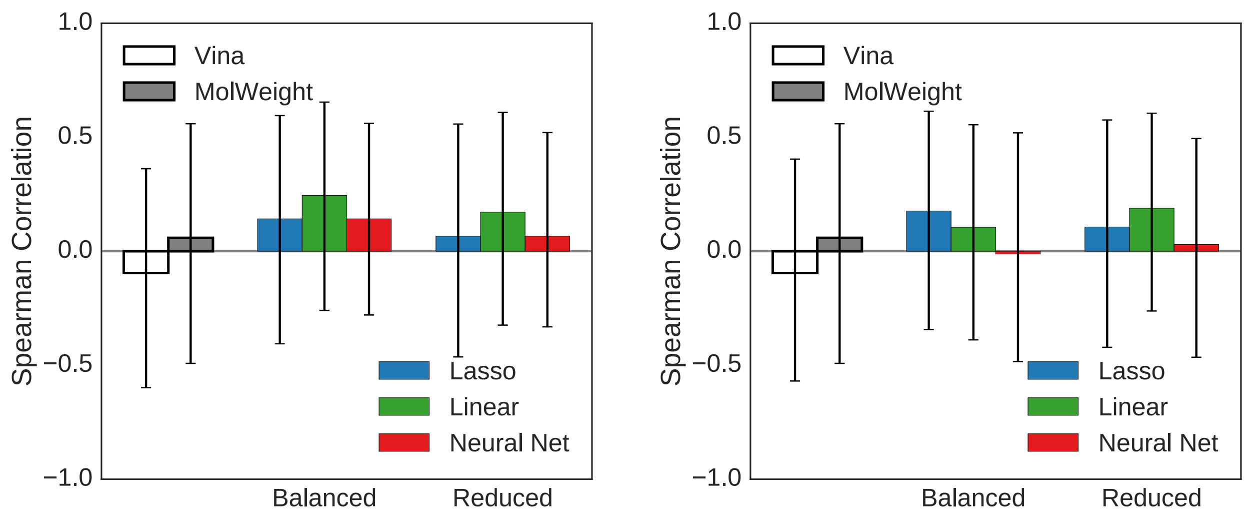


Figure 13: Spearman correlation coefficients using various methods to rank MAP4K4 ligands using the ligand pose top ranked by the given method (left) or the ligand pose with the smallest RMSD to the crystal structure (right). Error bars indicate the 95% confidence interval as determined by bootstrapping with replacement (1000 iterations).

ligands), was likely a major factor in the poor performance of these methods.

We used the D3R exercise to evaluate a variety of machine learning based scoring methods that were trained using a novel classification approach. Rather than fitting to affinity data or pose RMSDs, this approach seeks to leverage the large amount of high-throughput screening data available from a wide variety of sources. Framing the problem as a classification between binders and non-binders automatically normalizes for different assay outcomes. In order to utilize binding data in a structure-based approach, protein-ligand structures must be produced through docking. The end result is a large (the DUD-E set used here has more than one million ligands), but extremely noisy (due to docking inaccuracies) dataset. A key goal of this exercise was to evaluate the feasibility of such a training approach as well as compare different approaches to training set construction (i.e., balanced vs reduced).

Although the machine learning approaches did not outperform the AutoDock Vina scoring function, they did perform comparably at pose prediction, virtual screening, and affinity

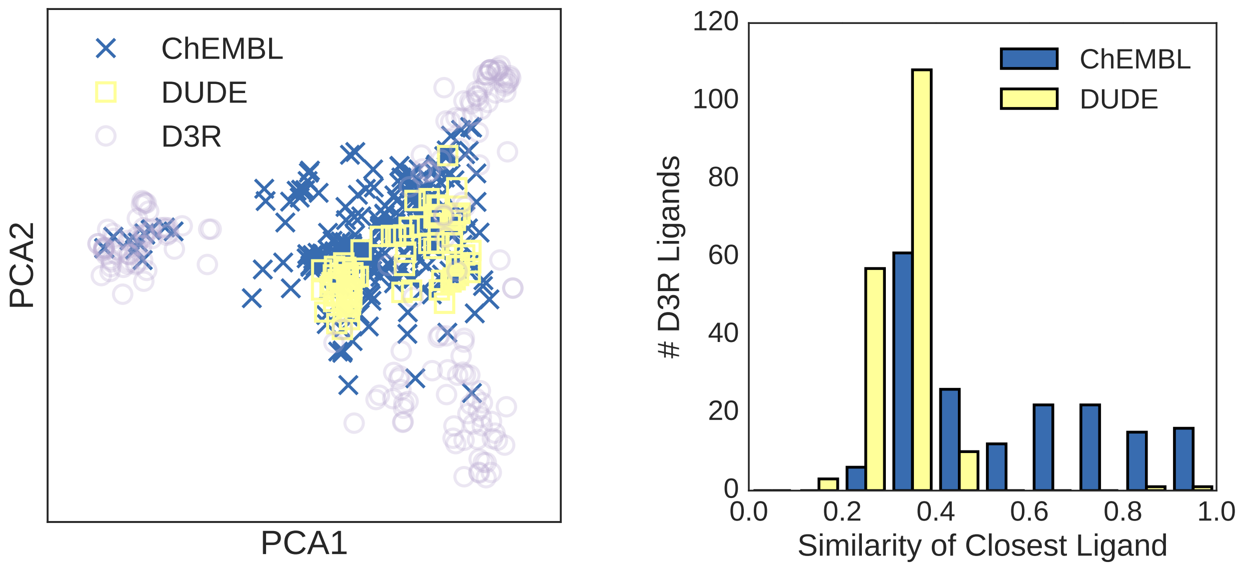


Figure 14: (Left) Structure-based (DUD-E) and ligand-based (ChEMBL) training sets projected on the first two principal components of the D3R ligand set. (Right) Histogram of the Tanimoto coefficient of D3R ligands with the most similar ligand in a given training set. Similarities are computed using OpenBabel FP2 fingerprints.

prediction. This may not seem surprising as the terms of the AutoDock Vina scoring function were included as training features. In fact, one of the features was the AutoDock Vina score itself, but omission of the Vina score from the training data produces essentially identical results (score predictions correlate with R>0.99). Nonetheless, we find it encouraging that two distinct modeling methods, linear regression and neural networks, can exploit a large, noisy dataset such as docked DUD-E poses, to achieve comparable results to a state-of-the art scoring function. The comparison between training set construction approaches was inconclusive with the reduced set producing somewhat better results for MAP4K4 pose prediction and the balanced set outperforming on HSP90 screening and affinity prediction. Larger, more accurate training data combined with more expressive structural input features or alternative machine learning approaches should further improve the usability and accuracy of scoring functions learned through classification of docked poses.

2.2 A standalone library for molecular gridding and some simple extensions

Motivated by the results in the previous section, our group pursued a novel 3D neural network approach based on convolutions mapped over a voxelized representation of ligands bound to proteins. We developed methods for partitioning and sampling input data, using data augmentation to overcome the coordinate frame-dependence of our input representation. This work is described in:

Ragoza, Matthew, Joshua Hochuli, Elisa Idrobo, Jocelyn Sunseri, and David Ryan Koes. 'Protein–ligand scoring with convolutional neural networks." Journal of chemical information and modeling 57, no. 4 (2017): 942-957.

Two of my specific contributions to that work included the algorithm we use to compute grids on-the-fly from molecules, and the use of independent test sets for evaluation to avoid problems associated with benchmarking dataset bias. The efficiency of our gridding approach helps us leverage the trained CNNs to optimize input molecules, a process during which the Cartesian and voxel representations are repeatedly interconverted. The approach we use is derived from similar approaches described in [13]. Briefly, we generate grids using a two-step parallel approach. GPU threads in the same block are assigned to the same spatial subgrid consisting of as many voxels as there are threads; they will compute the amount of atom density at a given location based on the atoms that overlap that location. The number of atoms overlapping a given subgrid is much smaller than the total number of atoms in the input, so if threads within the same region checked all atoms to determine whether they overlap their location, they would do a large number of redundant, mostly negative checks. Instead the threads in the subgrid first parallelize over the atoms to build up a boolean mask of atoms that overlap any voxel within the subgrid; then they perform a parallel exclusive scan using block-level shared memory and CUDA SIMD warp intrinsics to generate indices from the atom mask. Threads then write a final array of the small number of atom indices that could possibly overlap voxels in their spatial subgrid by parallelizing over the atom mask once again; this time if the mask value at an index is true, the thread reads the exclusive scan output at the same index - the exclusive scan value is the unique index in the *output* array where the atom index should be written. This permits concurrent accesses to the final array of atom indices. An example in two dimensions is shown in Figure 15, and the performance comparison between our approach and the simpler one based on simply parallellizing over voxels is shown in Figure 16.

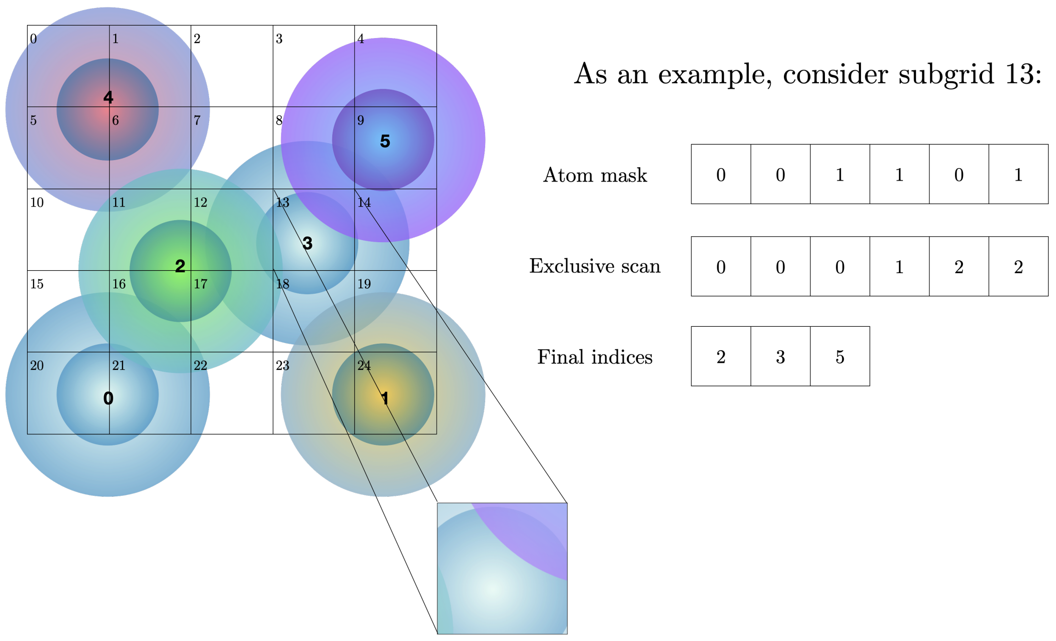


Figure 15: Graphical depiction of our approach to generating input grids from atoms with a simplified two-dimensional example.

I also implemented functionality to process spatial and temporal recurrences over inputs, which makes it possible to process sequences of molecular dynamics frames or arbitrarily-sized inputs. These and other capabilities were refactored into a standalone library, `libmolgrid`, written in C++ with Python bindings, described in the following section. We will revisit effective model evaluation and benchmarking dataset bias in chapter 3.

Chapter 2.2 was reproduced with permission from Sunseri, Jocelyn, and David R. Koes. "libmolgrid: Graphics Processing Unit Accelerated Molecular Gridding for Deep Learning Applications." Journal of Chemical Information and Modeling 60, no. 3 (2020): 1079-1084.

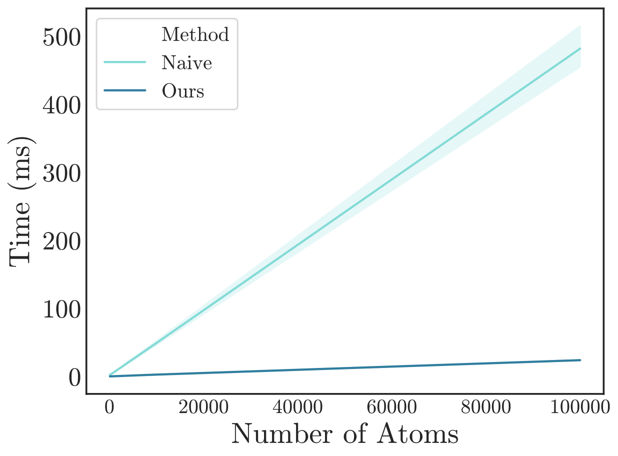


Figure 16: Speedup for a 2D test implementation of our two-step parallel gridding approach compared with the simpler approach of simply parallelizing over voxels. Numbers of atoms tested were $[100, 1000, 10000, 100000]$ and 25 trials with different random initializations of the input were performed.

2.2.1 Background

Deep learning has emerged as an important area of research in computational chemistry. It holds great promise for unprecedented improvements in predictive capabilities for such problems as virtual screening[93], binding affinity prediction[10, 212], pose prediction[7, 35], and lead optimization[52, 204, 211, 89]. The representation of input data can fundamentally limit or enhance the performance and applicability of machine learning algorithms[117, 51, 66]. Deep learning can derive class-defining features directly from training examples. Common input representations include molecular formats like SMILES and/or InChi strings[66, 201], molecular graphs[117, 188, 51, 95, 143, 56], and voxelized spatial grids[191, 145] representing the locations of atoms.

Compared with other representation schemes, spatial grids possess certain virtues including minimal overt featurization by the user (theoretically permitting greater model expressiveness) and full representation of three-dimensional spatial interactions in the input. For regular cubic grids, this comes at the cost of coordinate frame dependence, which can be ameliorated by data augmentation and can also be theoretically addressed with various types of inherently equivariant network architectures or by using other types of multidimensional grids. Spatial grids have been applied successfully to tasks relevant to computational chemistry like virtual screening[191, 145, 167], pharmacophore generation[168], molecular property prediction[94, 90], molecular classification[5, 94], protein binding site prediction[71, 88, 87], molecular autoencoding[105], and generative modeling.[18, 17, 183]

Chemical datasets have many physical and statistical properties that prove problematic for machine learning, and special care must be taken to manage them. Classes are typically highly imbalanced, with many more known inactive than active compounds for a given protein target; regression tasks may span many orders of magnitude, with nonuniform representation of the underlying chemical space at particular ranges of the regressor; and examples with matching class labels or regression target values may also be unequally sampled from other underlying classes (e.g. there may be significantly more binding affinity data available for specific proteins that have been the subject of greater investigation, such as the estrogen receptors, or for protein classes like kinases). By offloading data processing tasks required to manage these problems to an open source library specialized for chemical data, computational chemists can systematically obtain better results in a transparent manner.

Using multidimensional grids (e.g. voxels) to represent atomic locations (and potentially distributions) is computationally efficient - their generation is embarrassingly parallel and therefore readily amenable to modern GPU architectures - and preserves three dimensional spatial relationships present in the original input. Coordinate frame dependence can be removed or circumvented. However, commonly available molecular parsing and conversion libraries do not yet provide gridding functionality; nor do they implement the other tasks required to obtain good performance on typical chemical datasets, such as strategic resampling and data augmentation. Thus we abstracted the gridding and batch preparation functionality from our past work, `gnina`[145], into a library that can be used for general molecular modeling tasks but also interfaces naturally with popular Python deep learning libraries. Implemented in C++/CUDA with Python bindings, `libmolgrid` is a free and open source project intended to accelerate advances in molecular modeling via multidimensional spatial arrays.

2.2.2 Implementation

Key `libmolgrid` functionality is implemented in a modular fashion to ensure maximum versatility. Essential library features are abstracted into separate classes to facilitate use independently or in concert as required by a particular application.

2.2.2.1 Grids The fundamental object used to represent data in `libmolgrid` is a multidimensional array which the API generically refers to as a grid. Grids are typically used during training to represent voxelized input molecules or matrices of atom coordinates and types. They can be constructed in two flavors, `Grid`s and `ManagedGrid`s. `ManagedGrid`s manage their own underlying memory, while `Grid`s function as views over a preexisting memory buffer. `Grid`s and `ManagedGrid`s are convertible to NumPy arrays as well as Torch tensors. Additional exposition is available in Figure S59.

Because of automatic conversions designed for PyTorch interoperability, a user intending to leverage basic batch sampling, grid generating, and transformation capabilities provided by `libmolgrid` in tandem with PyTorch for neural network training can simply use Torch tensors directly, with little to no need for explicit invocation of or interaction with `libmolgrid` grids. Memory allocated on a GPU via a Torch tensor will remain there, with grids generated in-place. An example of this type of usage is shown in the first example in Listing 1.

A `Grid` may also be constructed explicitly from a Torch tensor, a NumPy array, or, if necessary, from a pointer to a memory buffer. Examples of constructing a `Grid` from a Torch tensor are shown in the second usage section in Listing 1. The third usage section shows provided functionality for copying NumPy array data to `ManagedGrid`s, while the fourth usage section shows functionality for constructing `Grid` views over NumPy array data buffers. In the fourth example, note that in recent NumPy versions the default floating-point data type is float64, so the user should take care to match the data type between arrays and `Grid`s.

```
# Usage 1: molgrid functions taking Grid objects can be passed Torch tensors directly,
# with conversions managed internally
tensor = torch.zeros(tensor_shape, dtype=torch.float32, device='cuda')
molgrid.gmaker.forward(batch, input_tensor)

# Usage 2: construct Grid as a view over a Torch tensor with provided helper function
tensor = torch.zeros((2,2), dtype=torch.float32, device='cuda')
grid = molgrid.tensor_as_grid(tensor) # dimensions and data location are inferred
# alternatively, construct Grid view over Torch tensor directly
grid = molgrid.Grid2fCUDA(tensor)

# Usage 3: copy ManagedGrid data to NumPy array
# first, construct a ManagedGrid
mgrid = molgrid.MGrid1f(batch_size)
# copy to GPU and do work on it there
mgrid.gpu()
# (do work)
# copy ManagedGrid data to a NumPy array with helper function;
# this copies data back to the CPU if necessary
array1 = mgrid.tonumpy()
# alternatively, construct NumPy array with a copy of ManagedGrid CPU data;
# must sync to CPU first
mgrid.cpu()
array2 = np.array(mgrid)

# Usage 4: construct Grid from NumPy array
array3 = np.zeros((2,2), dtype=np.float32) # must match source and destination dtypes
tensor = molgrid.Grid2f(array3)
```

Listing 1: Examples of `Grid` and `ManagedGrid` usage.

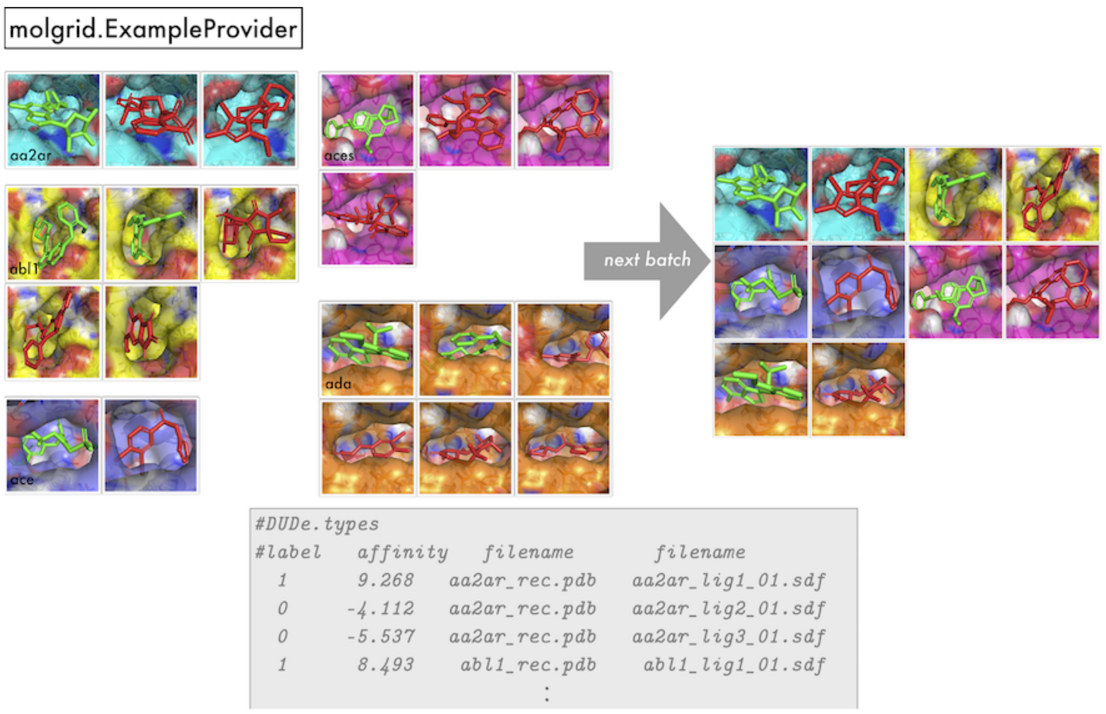
2.2.2.2 Atom Typing Several atom typing schemes are supported, featuring flexibility in the ways types are assigned and represented. Atoms may be typed according to XS atom typing, atomic element, or a user-provided callback function. Types may be represented by a single integer or a vector encoding. For a typical user, typing (with either index or vector types) can be performed automatically via an `ExampleProvider`.

2.2.2.3 Examples `Example`s consist of typed coordinates that will be analyzed together, along with their labels. An `Example` may consist of multiple `CoordinateSet`s (which may each utilize a different scheme for atom typing) and may be one of a sequence of `Example`s within a group. For example, a single `Example` may have a `CoordinateSet` for a receptor and another `CoordinateSet` for a ligand to be scored with that receptor, or perhaps multiple `CoordinateSet`s corresponding to multiple poses of a particular ligand. `Example`s may be part of a group that will be processed in sequence, for example as input to a recurrent network; in that case distinct groups are identified with a shared integer value, and a sequence continuation flag indicates whether a given `Example` is a continuation of a previously observed sequence or is initiating a new one.

2.2.2.4 ExampleProvider To obtain strategically sampled batches of data for training, a user can employ an `ExampleProvider`. The desired sampling options are specified to the `ExampleProvider` constructor, which can then be populated with one or more files specifying examples. Properly sampled `Example`s are obtained via `ExampleProvider::next` or `ExampleProvider::next_batch`. Figure 17 shows graphically how an `ExampleProvider` might obtain a batch of 10 shuffled, class-balanced, receptor-stratified `Example`s from a larger dataset, with accompanying code.

Currently, the simplest way to initialize a provider is to populate it with one or more files that specify metadata for `Example`s, with one `Example` per line. At a high level, that line will specify class and regression target values for the `Example`, any group identification associated with the `Example` (i.e. a shared integer label identifying `Example`s to be processed sequentially, as with temporal data provided as input to a recurrent network), and then one or more strings identifying filenames of molecules corresponding to that `Example`. The default line layout is `[(int)group][(float)label]*[molfile]+`. An example is shown in Figure 17. In the examples we provide with our project, these files have a `.types` suffix.

Listing S2 shows all the available options at the time of construction. These options are described in more detail in the Supporting Information and online documentation.

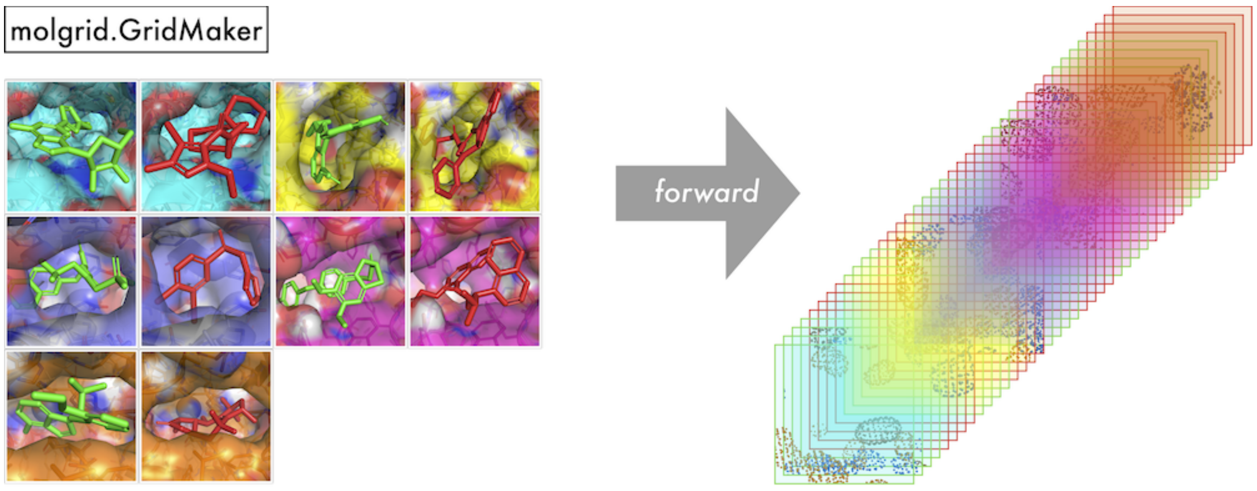


```
#DUDe.types
#label   affinity   filename        filename
  1       9.268    aa2ar_rec.pdb   aa2ar_lig1_01.sdf
  0      -4.112    aa2ar_rec.pdb   aa2ar_lig2_01.sdf
  0      -5.537    aa2ar_rec.pdb   aa2ar_lig3_01.sdf
  1       8.493     abl1_rec.pdb    abl1_lig1_01.sdf
                         :
```

```python
exprovider = molgrid.ExampleProvider(shuffle=True, balanced=True, stratify_receptor=True)
exprovider.populate('DUDe.types')
batch = exprovider.next_batch(10)
```

Figure 17: An illustration of `molgrid::ExampleProvider` usage, sampling a batch of 10 randomized, balanced, and receptor-stratified examples from a dataset.

2.2.2.5 GridMaker A `GridMaker` is used to generate a voxel grid from an `Example`, an `ExampleVec`, a `CoordinateSet`, or paired `Grids` of coordinates and types. `GridMaker` can operate directly on a user-provided Torch tensor or `Grid`, or it can return into a new NumPy array via `GridMaker::make_ndarray` or Torch tensor via `GridMaker::make_tensor`. `GridMaker` features GPU-optimized gridding that will be used if a compatible device is available. `GridMaker` options pertaining to the properties of the resulting grid are specified when the `GridMaker` is constructed, while the examples from which a grid will be generated and their instantiation properties (including any transformations) are specified by a particular invocation of `GridMaker::forward`. Specifically, Listing S3 shows the possible constructor arguments, which are described in more detail in the Supporting Information. Figure 18 shows an example of basic `GridMaker` usage, default-constructing a GridMaker and using it to populate a grid with densities for a batch of molecules. If desired, the values of random_translation and random_rotation can be set in the call to `gmaker::forward`, thereby applying random data augmentation to each example in the batch. If it is desirable to retain the applied transformation, then transformations can be created explicitly as shown in Figure 60. The `GridMaker` class also defines a `backward` function that computes atomic gradients, which can be used for tasks ranging from visualizing what a network has learned to using a trained network to optimize the coordinates and types of input molecules.



```python
gmaker = molgrid.GridMaker()
gmaker.forward(batch, input_tensor, random_translation=0.0, random_rotation=False)
```

Figure 18: An illustration of `molgrid::GridMaker` usage, generating a 4-dimensional grid from a batch of molecules, with data layout $NxCxLxWxH$.

2.2.2.6 Transformations Data augmentation in the form of random rotations and translations of input examples can be performed by passing the desired options to `GridMaker::forward` as described in the previous section. Specific translations and rotations can also be applied to arbitrary `Grids`, `CoordinateSets`, or `Examples` by using the `Transform` class directly. `Transforms` can store specific rotations, described by a `libmolgrid::Quaternion`; an origin around which to rotate, described by a `libmolgrid::float3`, which is also interconvertible with a Python tuple; and a specific translation, expressed in terms of Cartesian coordinates and also described by a `float3`. These prove useful for sophisticated networks such as the spatial transformer. Additional examples of `Transform` constructor invocation are shown in Listing S4, and information about `Transform::forward` is shown in Figure S60.

2.2.3 Results

We demonstrate model training with input tensors populated by `libmolgrid` and neural networks implemented using Caffe, PyTorch, and Keras with a Tensorflow backend (code available at `https://gnina.github.io/libmolgrid/tutorials.html`). Training loss performance is similar across all three frameworks, as shown in Figure S61. `libmolgrid` is fully functional with any of these popular libraries. Its overall speed and memory footprint varies significantly with the user's chosen library, however. As shown in Figure 19a and Figure 19b, the performance when using a GPU for gridding and neural network training is much faster when using Caffe and PyTorch than it is when using Tensorflow via Keras, with modest improvements in performance for Caffe and PyTorch when using the newer Titan V GPU rather than the older GTX Titan X. This is due to `libmolgrid`'s ability to directly access underlying data buffers when interoperating with Caffe and PyTorch, thus avoiding unnecessary data migration between the CPU and GPU; this is not currently possible with Tensorflow, and so passes through the network involve grids being generated on the GPU by `libmolgrid`, copied into a NumPy array on the CPU, and then copied back onto the GPU by Tensorflow when training begins. This results in a significant performance penalty, with memory transfers fundamentally limiting performance; future versions of `libmolgrid` will seek to mitigate this issue with Tensorflow 2.0. The discrepancy in memory utilization shown in Figure 19c is somewhat less dramatic, but similarly, memory utilization when doing neural network training with Tensorflow is less efficient than using the other two libraries.

As an example of a more specialized task that uses the backwards gradients computed

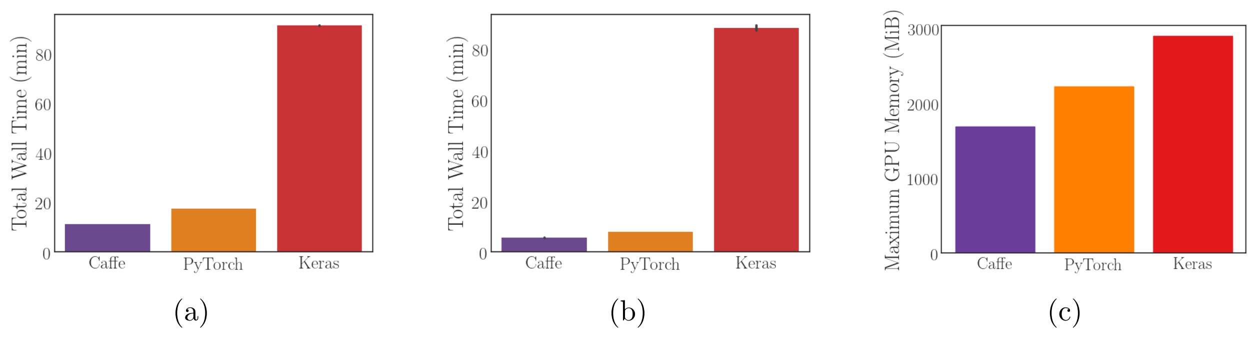
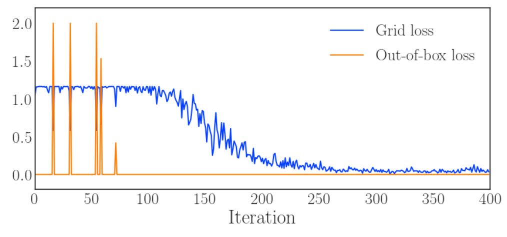
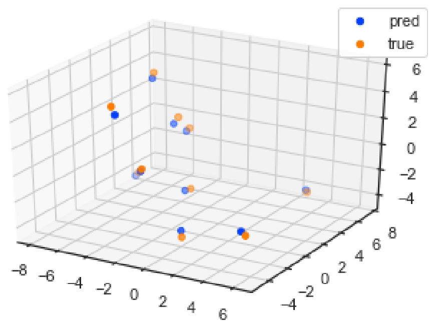


Figure 19: Performance information for using `libmolgrid` with each major supported neural network library. All error bars are 98% confidence intervals computed via bootstrap sampling of five independent runs. a Walltime for training the simple model shown training above using a GTX Titan X. b Walltime for training the same simple model using a Titan V. c Maximum GPU memory utilization while training.

by `GridMaker`, we demonstrate training a CNN to convert voxelized atomic densities to Cartesian coordinates. Each training example consists of a single atom, provided to the network as a voxelized grid for which the network will output Cartesian coordinates. The loss function is a simple mean squared error grid loss for coordinates that fall within the grid, and a hingelike loss for coordinates outside. As shown in Figure 20a, the model initially has difficulty learning because the atomic gradients only receive information from the parts of the grid that overlap an atom, but eventually converges to an accuracy significantly better than the grid resolution of 0.5Å. Example predictions are shown in Figure 20b. This task could be applicable to a generative modeling workflow, and also demonstrates `libmolgrid`'s versatility as a molecular modeling tool.



(a)



(b)

Figure 20: Cartesian coordinates from grid densities. (a) Loss per iteration for both the grid loss and out-of-box loss for training with naively initialized coordinates, showing `libmolgrid`'s utility for converting between voxelized grids and Cartesian coordinates. (b) Sampled coordinate predictions compared with the true coordinates, demonstrating a root mean squared accuracy of 0.09Å.

3.0 Evaluating the CNNs Prospectively, Including Pose Refinement and Transfer Learning

This chapter revisits the problem of proper model evaluation for machine learning from molecular data. It begins as the last chapter began, with a community benchmarking challenge that afforded an opportunity to perform a blinded assessment of our trained 3D CNN models. This time our models performed close to the top among submissions to D3R Grand Challenge 3 for affinity ranking, and several of our top-performing predictions were derived from poses generated by a novel modification to our Monte Carlo sampling routine that used an approximation of the Autodock Vina scoring function for initial pose sampling and the CNN for the final minimization of those poses. Our pose prediction performance was poor, due in large part to the large pocket and Vina's preferential sampling of poses on the opposite side of the pocket from where the challenge ligands bind, but there were some positive indications that pose refinement with the CNNs could be useful, particularly for virtual screening.

Chapter 3.1 was reproduced with permission from Sunseri, Jocelyn, Jonathan E. King, Paul G. Francoeur, and David Ryan Koes. "Convolutional neural network scoring and minimization in the D3R 2017 community challenge." Journal of computer-aided molecular design 33, no. 1 (2019): 19-34.

3.1 Applying CNNs prospectively, with rescoring and pose sampling

3.1.1 Background

Predicting whether a given small molecule binds strongly to a protein target of interest and explaining the strength of that interaction are topics of major importance in computational drug discovery [194, 36, 20, 142]. Developing new, more accurate methods for performing these tasks holds significant promise in combating the blight of human disease [170], but these methods must be tested on *de novo* case studies or blinded challenges to ensure that performance expectations are not inflated by inadvertent tuning to preexisting datasets for which the answers were publicly known when the predictions were made [83, 16]. The annual Drug Design Data Resource (D3R) blind challenge provides just such an opportunity to evaluate new methods behind a veil of ignorance, assessing how the gamut of strategies currently under development in the community perform on a series of novel tests [61, 60].

Typical problems to be solved in a drug discovery pipeline include predicting absolute binding affinities [90, 122, 4], accurately ranking compounds in order of binding strength [60, 175, 96, 6], predicting the probable molecular configuration during binding [194, 36], and performing each of these tasks under various conditions of dataset construction with relevance to drug design [61, 60, 22, 172, 23], e.g. congeneric compound series, wild type and point-mutated forms of a target protein, and predicting target specificity for compounds that bind to a set of related proteins. By designing challenges that are diverse in terms of both their chemical content and predictive classes, computational models can be comprehensively assessed in terms of their accuracy, ability to generalize, and (if applicable) their transfer learning capacity. Specific strengths and weaknesses of a particular model compared to others can be identified, and interrogating the failures of particular approaches is particularly valuable as we continue to pursue methodological advances.

Methods of estimating the relative strength of binding broadly range from physics-based to statistical in their approaches. Physics-based methods [68, 205, 25, 32, 55, 19, 112, 92, 91] typically rely on force fields parameterized from first principles and experimental data and may compute binding free energies directly using methods that are theoretically exact. In practice their accuracy is limited by both the adequacy of their configurational sampling and the accuracy of their force field parametrization; the former is generally limited by time considerations, as are the methods chosen to compute binding free energies. Empirical scoring functions [98, 53, 15, 195, 100, 53, 58, 186] use terms that represent features or interactions known to be relevant to molecular binding, and they may be parametrized (e.g. using parametric machine learning methods) to recapitulate experimental data such as binding affinities. Knowledge-based methods [78, 125, 63, 210, 123, 10, 77] are statistical potentials that favor contacts that appear with high frequency in the datasets from which they are computed.

Nonparametric machine learning methods, such as neural networks, learn both their parameters and model structure from data [152, 107]. As a result, they are less constrained by the frontiers of our knowledge during their construction - that is, they are not limited to the set of structures we can imagine imposing on them, or the set of features we can imagine providing them as input. They may take as descriptors the types of inputs found in widespread use among empirical scoring methods, including measures of electrostatic attraction or interaction fingerprints [48, 49, 35, 7, 93, 10, 212], but they may also be trained using an approach that avoids overt featurization and instead provides minimally processed experimental structural data as input to the network [90, 65, 191, 51, 158]. That has been our approach in our recent work developing grid-based convolutional neural networks (CNNs), which are remarkably successful at image classification [103, 179, 70], trained to perform various tasks relevant to protein-ligand scoring and pose prediction [146, 147, 73].

We used the 2017 D3R Grand Challenge 3 (GC3) as an opportunity to evaluate the performance of our default CNN-based scoring model (the version used for the challenge was commit b3fa6ae) in comparison with other state-of-the-art methods, including Autodock Vina [186, 185], a conventional empirical scoring function. Our CNN-based scoring models are implemented as part of the **gnina** molecular docking program, which is available under an open source license at `https://github.com/gnina`.

3.1.2 Methods

Our general workflow is shown in Figure 21. We used a structure-based docking and scoring approach, with pose sampling fundamentally based on the Autodock Vina scoring function as implemented in smina [98], a fork of the original project with increased support for minimization and custom scoring function development. We used our CNN scoring function to both further refine and simply rescore the poses generated by docking with smina, using the CNN affinity prediction and pose score as the basis for distinct submissions. This yielded a minimum of four unique CNN-based submissions for each subchallenge. We compare to smina's performance to test whether the CNN model is capable of improving on the accuracy

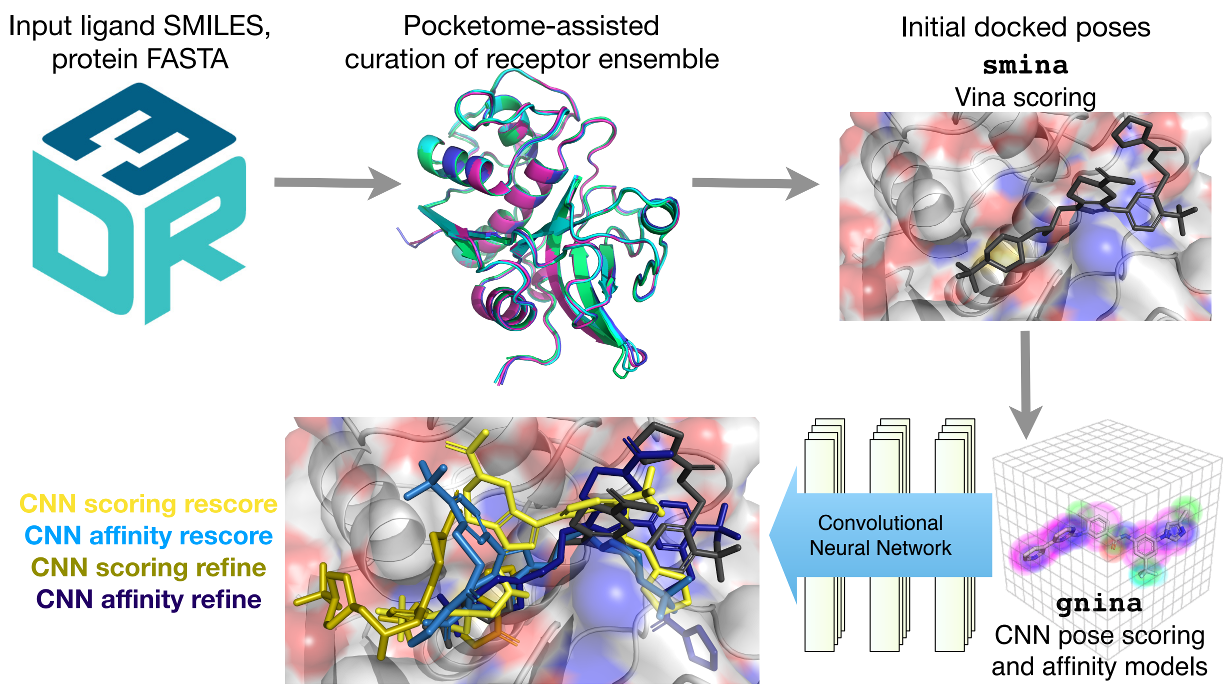

Input ligand SMILES, protein FASTA
Pocketome-assisted curation of receptor ensemble
Initial docked poses
smina
Vina scoring
gnina
CNN pose scoring and affinity models
Convolutional Neural Network
CNN scoring rescore
CNN affinity rescore
CNN scoring refine
CNN affinity refine

Figure 21: Workflow used to produce gnina convolutional neural network-based predictions for binding poses and binding affinity rankings.

of an existing scoring model, and independently evaluate the performance of the affinity and scoring outputs, as well as the CNN's ability to score putative binding modes and sample those modes itself.

D3R Grand Challenge 3 consisted of five subchallenges, the first of which consisted of three phases. Only the multiphase subchallenge 1 involved a pose prediction component, while all subchallenges involved predicting affinities and/or affinity rankings. Subchallenge 1, for which the target was Cathepsin S, involved both cross-docking (stage 1A) and redocking (stage 1B) tasks for 24 ligands for which ligand-protein co-crystal structures were available but unreleased until after stage 1B, and predicting affinity rankings for 136 compounds that were a superset of those 24 both before (stage 1A) and after (stage 2) unblinding of the co-crystal structures. The remaining four subchallenges all involved kinases. The stated aim of subchallenge 2 was to test compound selectivity prediction; accordingly, it featured the kinases VEGFR2, JAK2, and p38α and 54 compounds for which K_d values were available for all three of these proteins. Subchallenges 3 and 4 were both designed to test accuracy at predicting large changes in binding affinity due to small changes in compound structure; in subchallenge 3 the target was again JAK2 and it involved 17 congeneric compounds, while in subchallenge 4 the target was TIE2 and it involved 18 congeneric compounds. Subchallenge 5 was designed to test accuracy at predicting the effect of target protein mutations on compound binding affinity, and its target was the wild type and five mutants of the target ABL1, with only two compounds. Subchallenges 2-4 did not specify the phosphorylation state of the target proteins, while subchallenge 5 noted that all proteins were unphosphorylated; we simply used the phosphorylation state of the reference receptors chosen from the PDB. Subchallenge 3 noted that chiral compounds were measured as a racemic mixture, and so for this subchallenge we docked all enantiomers of each compound.

3.1.2.1 CNN Training Our architecture, input format, and training approach have been described previously [146, 147, 73]. Briefly, we designed a four-dimensional grid-based input representation consisting of a vector of spatially distributed atom densities for each supported atom type, where atom types fundamentally distinguish between protein and ligand atoms, different elements, and the protonation states of those atoms. The density of a particular atom within its relevant atom channel is represented as a piecewise continuous function $g(d, r)$, where d is the distance from the atom center and r is the van der Waals radius:

$$g(d,r) = \begin{cases} e^{-\frac{2d^2}{r^2}} & 0 \leq d < r \\ \frac{4}{e^2r^2}d^2 - \frac{12}{e^2r}d + \frac{9}{e^2} & r \leq d < 1.5r \\ 0 & d \geq 1.5r \end{cases} \tag{3.1}$$

The CNN maps its input to an output value that is either a probability distribution over class labels (i.e. whether or not a given input is a binding pose) or a real-valued affinity prediction. The architecture used during D3R GC3 can be found in Figure 22.

The CNN was trained using poses generated by redocking the 2016 PDBbind refined set [114] using the Autodock Vina scoring function as implemented in smina. Poses within

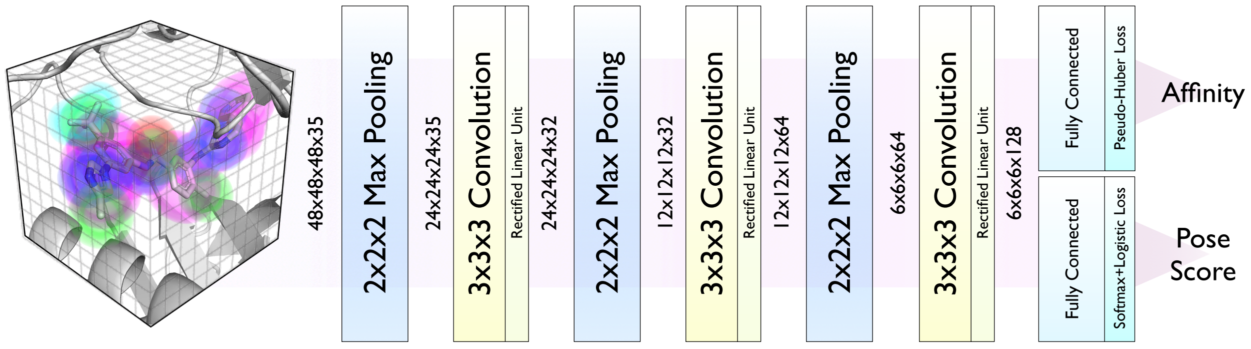


Figure 22: Architecture of the neural network used to rescore and refine poses. The input is a voxelized grid of Gaussian atom type densities.

2Å RMSD are labeled as actives for the binary classification output and given the binding affinity as the target value for the regression output, while all other poses are labeled as inactives for the purposes of binary classification and are penalized (via a hinge loss) only if the predicted affinity is too high. In order to increase the number of active examples in the training set, these docked poses were supplemented with crystal poses minimized using the Autodock Vina scoring function. The training set was then further expanded by performing three rounds of iterative training during which a model was trained, used to refine the docked poses, and then the poses resulting from that process labeled based on the crystal structure and added to the training set for the next round. Using this training set of 250,000 poses, the final model was trained for 150,000 iterations with a batch size of 50 using our customized version of the GPU-optimized Caffe [86] deep learning framework. Each batch was balanced to contain an equal number of positive and negative examples (low and high RMSD poses) as well as stratified by receptor so that every receptor target was uniformly sampled, regardless of the number of docked structures. At each iteration, a random rotation and translation was applied to every input complex in order to prevent the network from learning coordinate-frame dependent features.

3.1.2.2 Pose Generation The basic information provided for each subchallenge consisted of SMILES for the relevant compounds and FASTA sequences for the relevant proteins. RDKit [149] was used to generate a three-dimensional conformer based on the provided SMILES for each compound; only one conformer was required (or one conformer per enantiomer in subchallenge 3) because the conformational space of the ligand was subsequently explored during docking. Each protein was used to query Pocketome [104] for relevant PDB accession IDs. All available holo structures were aligned and visually inspected to manually select a conformationally diverse subset of reference structures for docking. Table 1 shows the PDB accession IDs for the reference structures that were chosen. The IDs associated with ABL1 include 1FPU, 1OPJ, and 2G1T (used as references for the wild type protein and also point-mutated in PyMOL [39] as references for the F317I, F317L, and Q252H mutations); 2G2F, 2G2H, and 2G2I (references for the H396P mutation); and 2V7A (reference for the T315I mutation). The generated conformers for each compound were then docked into the corresponding reference receptor ensemble using Vina; an additional set of docked poses was generated by performing the final minimization of the poses sampled by Vina during the Monte Carlo routine with the CNN pose scoring layer ("CNN refinement") - this is a hybrid technique where the fast Vina scoring is used for the Metropolis criterion during Monte Carlo sampling and the slower CNN scoring is only used to minimize ligand poses selected by the sampling. The Vina docked poses were then rescored using both the CNN scoring and affinity layers, and the CNN refined poses were also rescored using the CNN affinity layer. Ranking the poses by score thereby produces a maximum of five predictions per subchallenge, although CatS phase1B (the redocking subchallenge) featured ten submissions due to redocking either with crystal waters present or absent. The top five poses for each method were submitted for the pose prediction tasks, while scoring tasks utilized the top-scoring pose for each compound to make the scoring prediction and compound ranking. Vina's predictions were only submitted for CatS; for the other targets only the CNN-based predictions were officially submitted, though we show the results of Vina scoring for comparison in the ensuing analysis. Furthermore, we also show our results for ABL1 (subchallenge 5) using the same methods as were used in the rest of the challenge, although the required calculations were completed after the end of the challenge period.

Target	reference PDB IDs	PDBbind/ similarity	mean/max Tanimoto
CatS	2F1G, 2HXZ 2G7Y,3KWN 2HHN	2HHN/0.991	0.209/0.256
JAK2(SC2)	2W1I, 3E62 3JY9, 3UGC 4AGC, 5I4N 5UT2, 5UT5 5UT6	4JIA/0.980	0.291/0.437
VEGFR2	1VR2, 1YWN 2OH4, 2P2H 2P2I, 2GU5 3B8R, 3VNT	4ASE/0.657	0.214/0.414
p38α	1M7Q, 1OVE 1W82, 1W83 1WBS, 2GHL 2ZB1, 3ITZ 3L8S, 3NNU	1YQJ/0.978	0.278/0.488
JAK2(SC3)	2W1I, 3E62 3JY9, 3UGC 4AGC, 5I4N 5UT2, 5UT5 5UT6	4JIA/0.980	0.357/0.413
TIE2	2OO8, 2OSC 2P4I, 2WQB 3L8P, 4X3J	4V01/0.482	0.239/0.363
ABL1	1FPU, 1OPJ 2G1T,2G2F 2G2H,2G2I 2V7A	3K5V/0.979	0.302/0.332

Table 1: Targets that appeared in at least one of the subchallenges of D3R Grand Challenge 3, with the PDB IDs used as references for docking; the most similar target in the PDBbind 2016 refined set, used to train the CNN, with its similarity to the provided FASTA sequence for the D3R target; mean and maximum Tanimoto coefficient of the crystal ligand associated with the PDBbind refined set target to the compounds in the challenge.

3.1.3 Results

General information about the targets used in D3R GC3 is found in Table 1, including the aforementioned PDB IDs used to produce binding poses as well as basic measures of the similarity of the challenge targets and compounds to their most comparable targets and compounds used in the training set. In particular, the third column shows the PDB accession ID of the target in the training set that has highest sequence similarity to each GC3 target, and the fourth column shows the mean and maximum Tanimoto coefficient for that target's co-crystal ligand to the accompanying GC3 target's compounds. Tanimoto coefficients were calculated using OpenBabel [1] with FP2 fingerprints. Notably, while the most similar target to TIE2 does not have high global sequence similarity compared with the other D3R targets and their most similar training set target, it is FGFR1, whose catalytic domain is known to be highly similar to TIE2[163]. From these data we can conclude that while our training data included at least one target that was highly similar to each of the GC3 targets, the associated poses used for training did not include compounds that were particularly similar to the GC3 compounds; therefore GC3 may serve as a fair test of the CNN's generalization ability.

3.1.3.1 Pose Prediction

The GC3 pose prediction task was limited to a 24 compound subset of the Cathepsin S subchallenge. Participants were asked to provide predicted binding poses without (stage 1A) and with (stage 1B) knowledge of the cognate receptor structure for each compound. Up to five poses could be submitted. To maintain an automated and general approach for the submission, we produced poses by docking into a diverse receptor ensemble, using the entire binding site as the search space. Figure 23 shows the RMSD of the best pose submitted per compound for each scoring method, grouped by the method and showing the RMSD distribution for each challenge stage. The following statistics are computed based on the best-submitted pose per compound. The method associated with the lowest mean RMSD among all our submissions in stage 1A was using the CNN pose scoring model to rescore Vina-generated docked poses ("CNN Scoring Rescore"); its mean RMSD was 8.35Å and its rank among all submissions based on the mean RMSD was 31/44. The method associated with the lowest median RMSD among all our submissions in stage 1A was using the CNN pose scoring model to refine poses sampled by Vina during the Monte Carlo search ("CNN Scoring Refine"); its median RMSD was 8.16Å and its rank among all submissions based on the median RMSD was 31/44. The method associated with the lowest mean and median RMSD among all our submissions in stage 1B was using the CNN pose scoring model to rescore Vina-generated docked poses; its mean RMSD was 9.70Å and its rank among all submissions based on the mean RMSD was 23/47. Its median RMSD was 7.33Å and its rank among all submissions based on the median RMSD was 13/47.

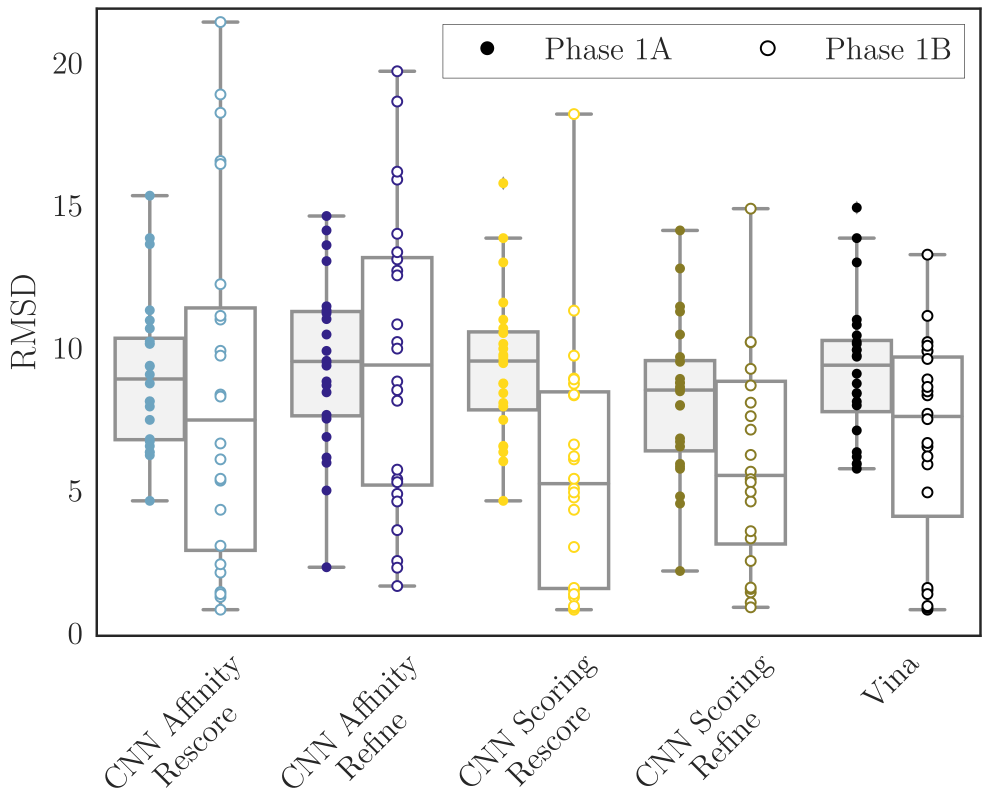


Figure 23: Per-compound best RMSDs for each method's submissions in stage 1A (cross-docking, left) and stage 1B (redocking, right). Pose prediction submissions consisted of the top 5 poses according to each scoring method.

Our CatS pose prediction performance was generally poor, even when redocking. A pose within 2.5Å RMSD was sampled for only a third of the test compounds. Nine poses were sampled per ligand, per reference receptor, resulting in 45 poses per ligand in stage 1A and 18 poses per ligand in stage 1B. Redocking in particular was characterized by high variance in the best-predicted RMSDs across the set of test compounds. Docking into the large CatS binding site using our scoring methods yielded predictions that were distributed throughout the search space, but in reality binding appears to be localized to a specific region.

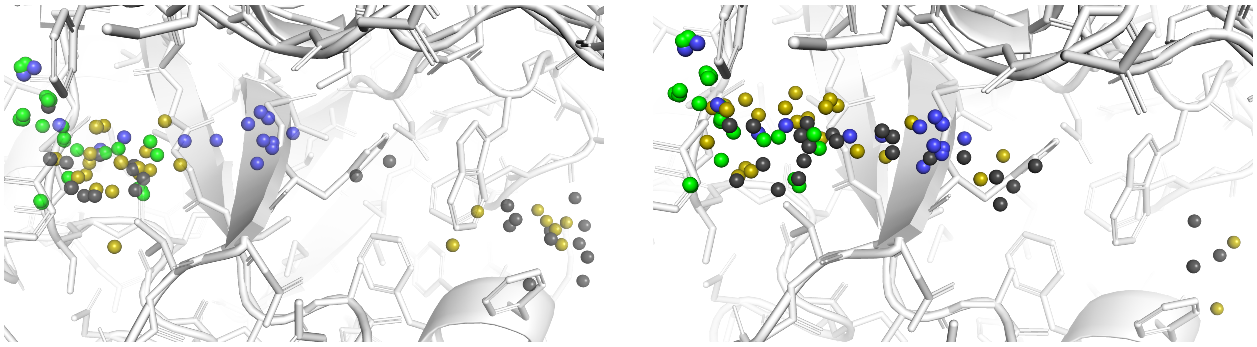


Figure 24: Center of mass locations for the unblinded crystal poses of the GC3 CatS co-crystal ligands (green), the co-crystal ligands of the reference PDB structures used during phase 1 docking (blue), and the highest ranked docked poses for each compound generated by redocking (the stage 1B task) with Vina (gray), and CNN refinement (gold). The left subfigure shows the results from docking with crystal waters present, while the right subfigure shows the results from docking without them.

Figure 24 shows the center of mass locations for available reference structures and for our top-ranked predictions. The GC3 compounds are densely clustered in one region of the pocket, while the available experimental data from the PDB support a somewhat larger binding region (and potentially more diverse binding modes). However, both Vina and the CNN produced many high-ranking poses that appeared in a different region of the pocket altogether. In particular, when docking with water using Vina the average distance to the closest center of mass in the set of D3R ligands is 8.11Å, and the average distance to the closest center of mass in the set of reference receptor ligands is 6.00Å; when using the CNN for the final refinement the average distance to the closest center of mass in the set of D3R ligands is 5.00Å, and the average distance to the closest center of mass in the set of reference receptor ligands is 3.97Å. When docking without water using Vina the average distance to the closest center of mass in the set of D3R ligands is 4.01Å, and the average distance to the closest center of mass in the set of reference receptor ligands is 2.48Å; when using the CNN for the final refinement the average distance to the closest center of mass in the set of D3R ligands is 3.18Å, and the average distance to the closest center of mass in the set of reference receptor ligands is 2.22Å. Thus both methods produced on average more poses in the region of the pocket where the GC3 ligands actually bind when docking without crystal waters, but the CNN was better in both cases at generating as top-ranked poses those that were closer to the general region of the pocket where the crystal ligands appear.

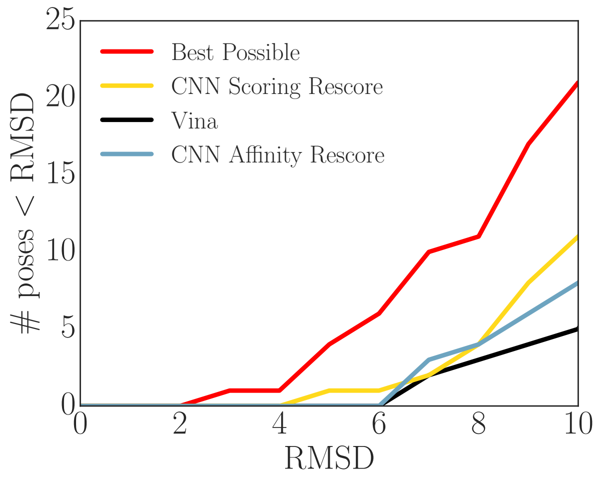
While docking without crystal waters present resulted in more poses in the general region of the pocket where the GC3 ligands bind, when low RMSD poses (here defined as those within 2.5Å) were sampled, they were most often produced by docking and refinement that utilized the crystal waters. Table 2 shows all 28 of the low RMSD poses sampled by any docking method for both stage 1A (the cross-docking task) and stage 1B (the redocking task). Rows are grouped by compound ID and sorted internally by the pose RMSD to the crystal pose. Values that are not relevant in a particular column are indicated with N/A; specifically, no waters were used during cross-docking and therefore the solvent category is N/A for poses sampled during that task, and the Vina score is N/A for poses generated by the CNN scoring model refinement method.

Significant categorical features are highlighted, including: cross-docking versus redocking; poses produced by full Vina docking versus Vina Monte Carlo sampling followed by refinement with the CNN scoring model; crystal waters used or removed; and rank among all of that compound's poses scored by a particular method, with any rank within the top five highlighted. The CNN scoring model was used to both rescore Vina's poses and produce its own refined poses, and the CNN affinity model was used to rescore both Vina's docked poses and the CNN refined poses. Consequently, these methods have two associated sets of rankings for each compound, which correspond to separate submissions to the challenge; they may therefore have up to two poses at any given rank in the table. Additionally, poses generated by docking with and without waters are grouped together to produce the ranking shown in the table; the effect of solvent will be explored in greater detail in section 3.1.3.1.

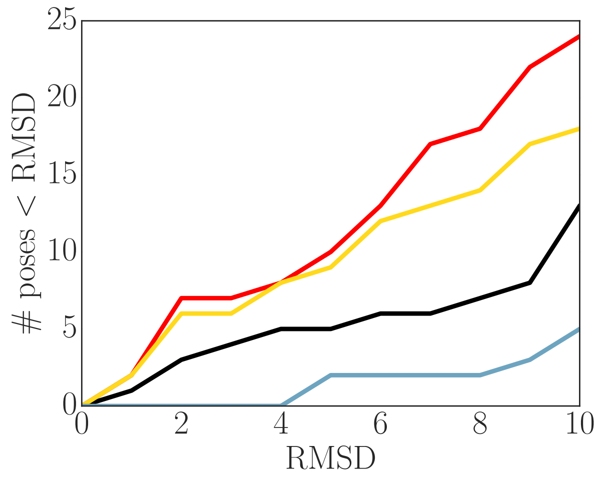
Only 8/24 compounds had any pose within 2.5Å RMSD of the crystal pose; only 3 had a low-RMSD pose sampled during the cross-docking task, while 7 had a low-RMSD pose sampled during redocking. It is notable that all five low-RMSD poses generated during the cross-docking task were produced by using the CNN for final refinement, though these poses were ranked in the top five only twice, once by the CNN scoring model (which sampled them) and once by the CNN affinity model (which re-scored them). CNN refinement outperformed Vina for only one compound during the redocking task (CatS 24), though it produced nearly as many low-RMSD poses (11 poses to Vina's 12). Though the CNN's sampling was guided at a coarse-grained level by Vina, which was used during Monte Carlo sampling, it is worth noting that in most cases its refinement did not move "good" poses in such a way that they were no longer "good" according to our threshold, and that in a few cases the CNN appears to have succeeded in moving a pose closer to the crystal pose than Vina did, as evidenced by succeeding in sampling a good pose during stage 1A, by improving on the best Vina pose RMSD, or by producing more low-RMSD poses than Vina did. Examples of cases where the CNN improved on a Vina pose (as shown in Table 2) include CatS 5, CatS 10, CatS 15, CatS 17, CatS 20, and CatS 24. A notable exception is CatS 11; only Vina sampled a low RMSD pose for that compound.

Compound ID	cross-docking/ redocking	RMSD (Å)	generation method	solvent	Vina rank	CNN scoring rank	CNN affinity rank
CatS 5	re	1.463	Vina	water	4	1	2
CatS 5	re	1.745	CNN refine	water	N/A	7	4
CatS 5	cross	1.743	CNN refine	N/A	N/A	15	18
CatS 5	cross	1.976	CNN refine	N/A	N/A	24	26
CatS 5	re	2.257	Vina	water	11	2	3
CatS 5	re	2.498	CNN refine	water	N/A	6	5
CatS 10	cross	2.272	CNN refine	N/A	N/A	2	11
CatS 11	re	1.362	Vina	water	11	1	4
CatS 15	re	0.915	Vina	water	1	1	2
CatS 15	re	1.166	CNN refine	water	N/A	4	8
CatS 15	cross	1.436	CNN refine	N/A	N/A	26	15
CatS 15	cross	2.404	CNN refine	N/A	N/A	13	3
CatS 16	re	0.989	Vina	water	2	1	6
CatS 16	re	1.522	CNN refine	water	N/A	1	6
CatS 16	re	1.535	Vina	water	10	2	5
CatS 16	re	1.768	Vina	water	14	5	12
CatS 16	re	2.072	CNN refine	water	N/A	7	12
CatS 16	re	2.254	Vina	water	1	3	6
CatS 16	re	2.388	CNN refine	water	N/A	3	3
CatS 17	re	1.599	Vina	water	1	2	10
CatS 17	re	1.643	CNN refine	water	N/A	3	15
CatS 17	re	1.762	CNN refine	water	N/A	1	14
CatS 20	re	1.679	Vina	water	2	1	13
CatS 20	re	1.691	CNN refine	water	N/A	2	15
CatS 20	re	2.005	CNN refine	no water	N/A	14	10
CatS 20	re	2.217	Vina	no water	5	8	4
CatS 24	re	0.998	CNN refine	water	N/A	2	10
CatS 24	re	1.043	Vina	water	1	1	8

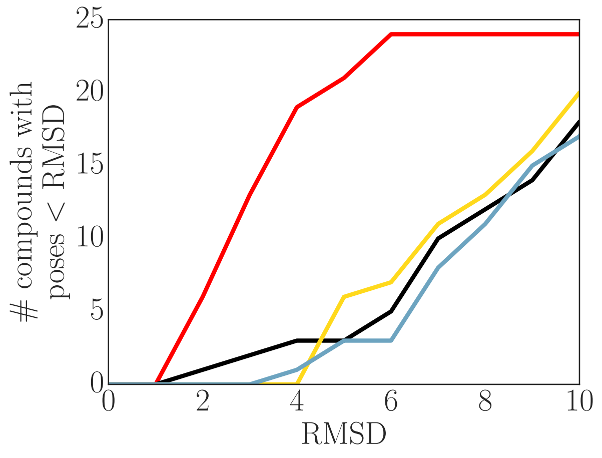
Table 2: Poses generated for CatS compounds that were within 2.5Å RMSD of the crystal pose. Colors differentiate cross-docking (brown) from redocking (black); poses produced by Vina alone (black) or with CNN refinement (purple); with solvent (blue) or without (pink); and poses that were ranked in the top 5 (green) or outside of the top 5 (black). Entries are marked as N/A if they do not apply for a specific row (solvent columns for cross-docked poses and Vina's rank for CNN-generated poses) and they are colored peach to reduce their visual impact.



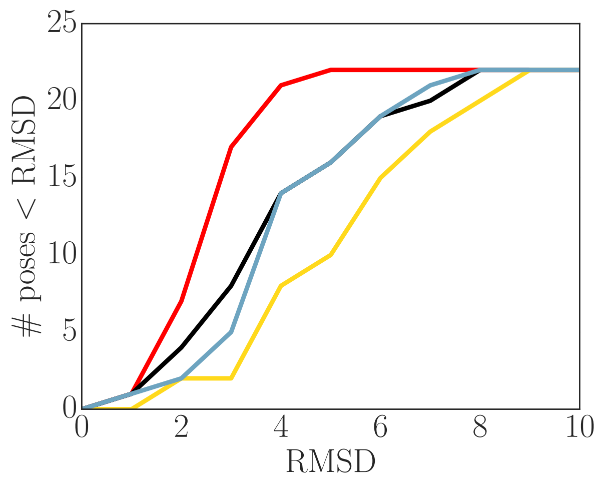
(a) Stage 1A



(b) Stage 1B



(c) Automated scaffold alignment



(d) Hand-selected scaffold alignment

Figure 25: Number of poses within a given RMSD that can be found as a top-ranked pose using Vina or re-scoring Vina-generated poses with the CNN scoring or affinity models for CatS stage 1A (a), stage 1B (combining poses sampled with and without solvent) (b), and two ligand similarity-based methods ((c) and (d)). The performance that would be attained if the sampled pose with the lowest RMSD were selected as the top pose is shown in red.

Vina and the CNN combined only sampled a low-RMSD pose for a third of the CatS compounds. When Vina sampled a low-RMSD pose during the challenge, the CNN scoring model was more likely than either Vina or the CNN affinity model to identify it as the top-ranked prediction for the associated compound. Fig 25 shows the best possible performance given the poses sampled with Vina (red lines), and then shows how significantly each scoring model deviated from that performance in its selection of top-ranked poses.

The CNN scoring model outperforms the other models when using the receptor ensemble approach utilized during the challenge as well as when performing redocking. The CNN's improved performance on stage 1A, in Figure 25a, is marginal; the CNN scoring model appears to be the only method to feature a top-ranked pose within 5Å RMSD when choosing among the poses sampled by Vina. Figure 25b, showing stage 1B performance, is more unequivocal, with the CNN scoring model nearly matching the best possible performance for low RMSD poses while the other methods fail to identify several available low-RMSD poses.

Figures 25c and 25d show new analyses performed after the subchallenge ended; they utilize preexisting experimental data to guide pose prediction. The available PDB structures of CatS were queried to identify the crystal ligand with the highest Tanimoto coefficient with each GC3 compound, and the GC3 compound was then aligned by scaffold to that crystal ligand using up to 100 conformers. In (c) the scaffold was chosen by generating a Murcko decomposition of each query and reference compound and the maximum common substructure of these were aligned; in (d) the scaffolds were chosen by visual inspection. The aligned poses were then minimized and rescored according to the same procedure used to sample and rescore poses for the original submissions, and the compounds were also cross-docked into a box defined by the binding pose of the chosen reference. This method is less general than docking agnostically into regions of the pocket, since it relies on information about similar ligands being available, but it produces significantly improved pose prediction performance in this case. Using this procedure for sampling, Vina outperforms the CNN at identifying available low-RMSD poses, though all methods fail to approach the "best available" performance.

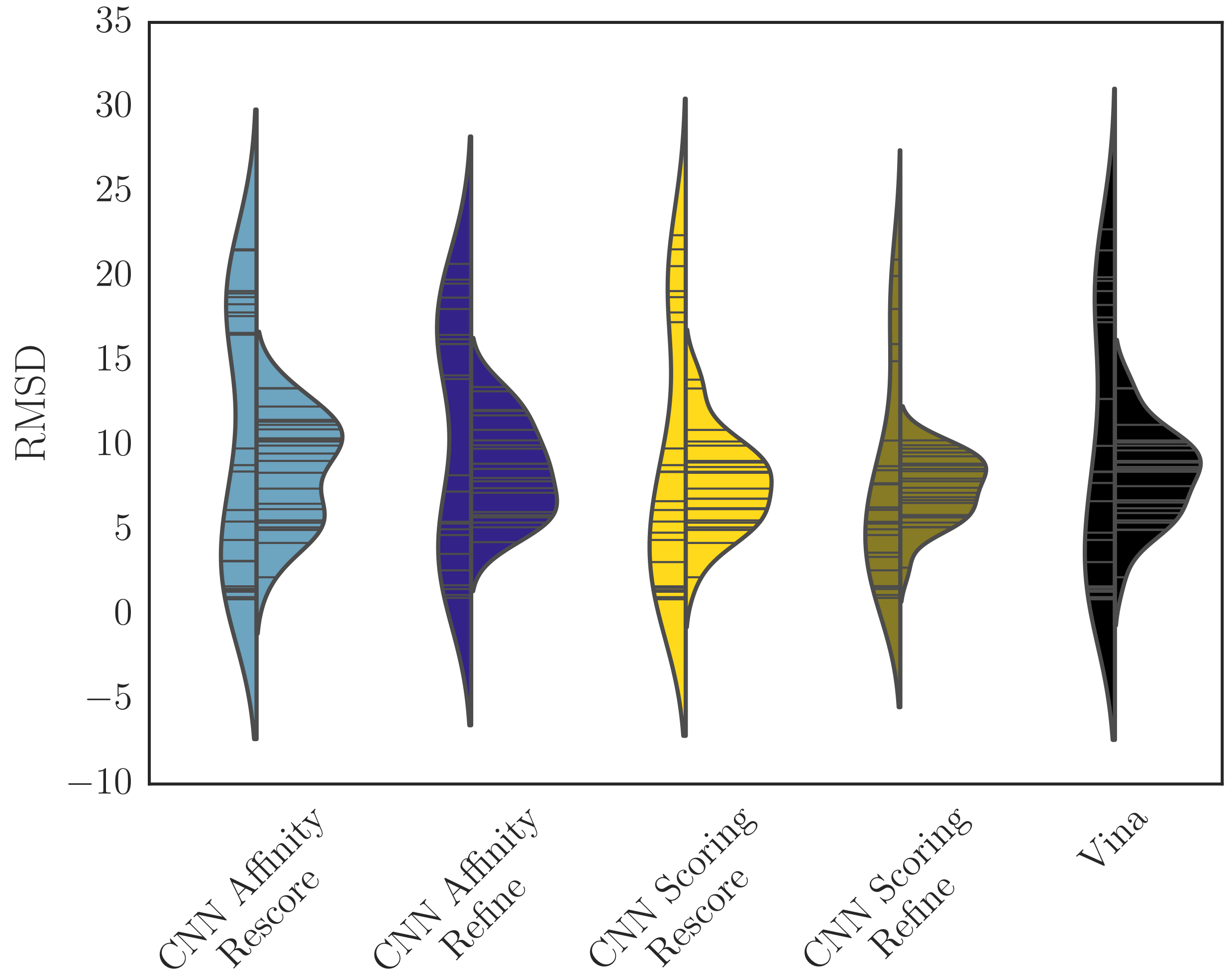
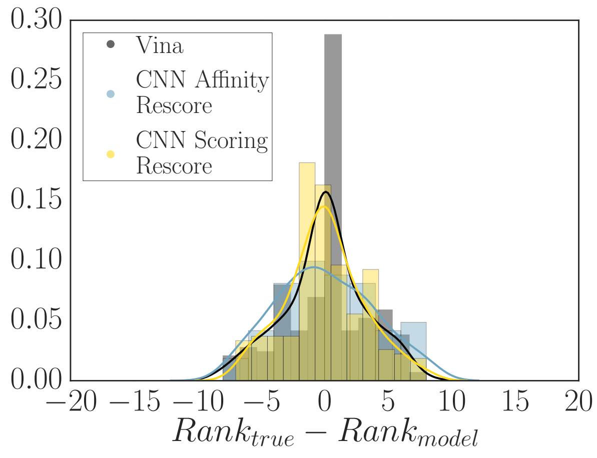


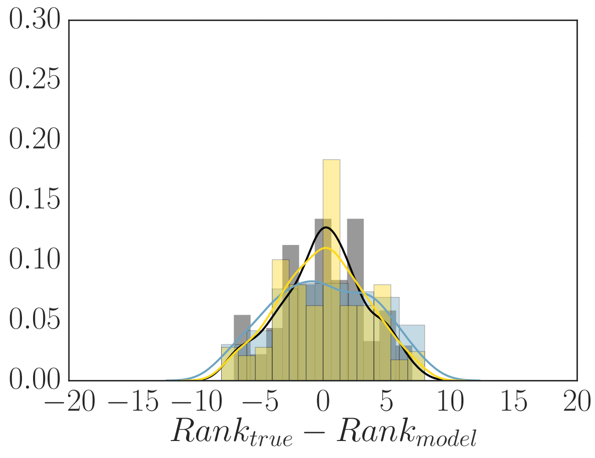
Figure 26: Per-compound best RMSDs for each method's submissions in stage 1B, split between solvent (left) and no solvent (right).

Redocking in stage 1B, for which crystal waters were available, affords an opportunity to examine whether Vina and the CNN scoring models differ in their abilities to correctly rank poses generated with and without crystal waters. Figure 26 shows the RMSD of the best pose submitted per-compound using each method. Including solvent increases the variance in the best predicted RMSDs, producing an apparently bimodal distribution with peaks at both lower and higher RMSDs than the medians of the distributions without solvent. The method that used the CNN for both refinement and the final ranking ("CNN Scoring Refine") may slightly improve on Vina's performance by both reducing the density at high RMSD when sampling with solvent and by shifting the median toward a slightly lower RMSD when sampling without solvent.

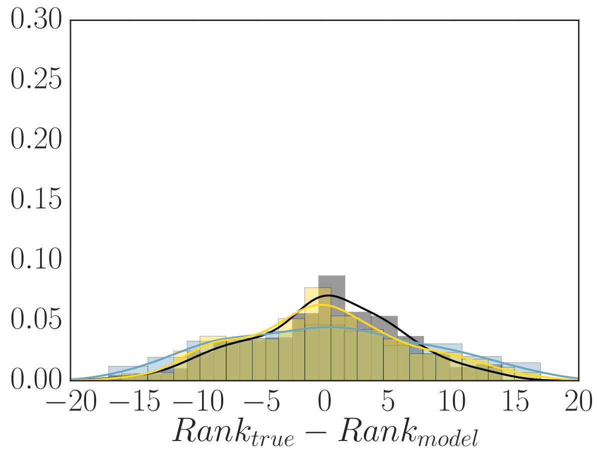
Figure 27 considers how many times a given method provided a pose ranking that deviated from that pose's true ranking by specific amounts. A perfect classifier would have its entire density at 0, and greater spread corresponds to a less accurate ranking; compared to a correlation metric, this analysis gives information about where ranking deviations occur. All methods have lower standard deviation of their ranking error when ranking poses sampled with solvent than those sampled without it. They also have kurtosis closest to 0 when sampling with solvent - Vina and CNN scoring both have kurtosis around 0 in that case, compared with kurtosis of -0.393 and -0.492 respectively when sampling without solvent, and -0.163 and -0.370 respectively when ranking all poses. CNN scoring is less skewed when ranking poses sampled without solvent than Vina is (-0.026 versus -0.132), which corresponds to making fewer errors in misclassifying high RMSD poses as low RMSD poses. Vina's skew is consistently negative, with its most negative skew when ranking poses sampled with solvent, while CNN scoring has positive skew when ranking poses sampled with solvent. The CNN affinity model has consistently more negative kurtosis and higher standard deviation than the other two methods, which accords with its generally worse performance at pose prediction.



(a) docking with water



(b) docking without water



(c) ranking with poses combined

Figure 27: Deviation of pose rankings from their true ranking, for docking with (a) and without (b) water. A classifier is more accurate if it is more strongly peaked around the center; a perfect predictor would have all of its density at 0. Combining poses generated with and without solvent produces (c).

Since so few low-RMSD poses were sampled, it merits investigating whether our poor CatS pose prediction performance was primarily a sampling problem (potentially due to too large a search space) or whether our scoring methods generally failed to score poses near the crystal pose well when performing sampling in the CatS binding site. To do this, we both re-scored the crystal poses using all three scoring methods and also minimized those poses using Vina and the CNN scoring model, then re-scored the minimized poses with the CNN scoring and affinity models as appropriate.

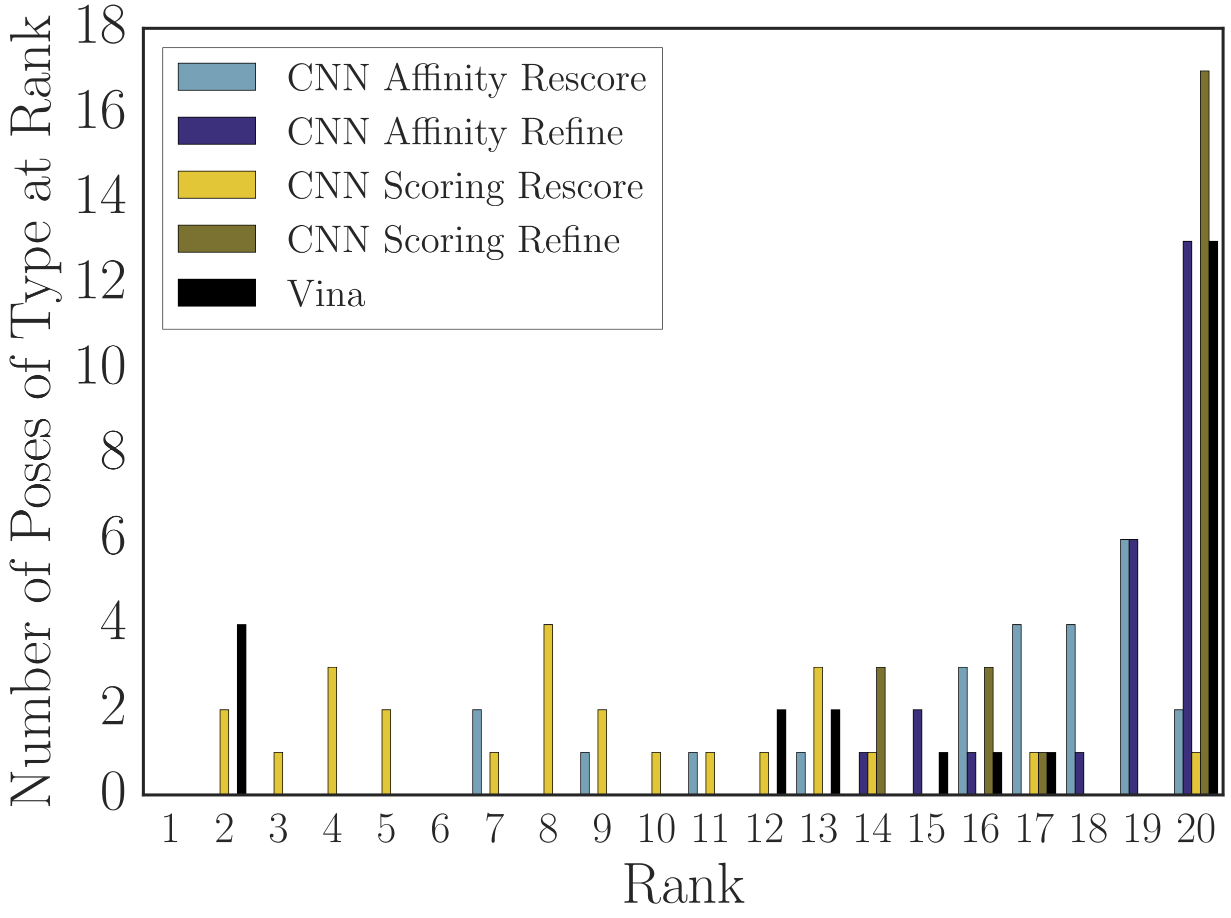


Figure 28: Performance of each method at ranking the re-scored crystal poses among all other poses generated during stage 1B and the minimized crystal pose.

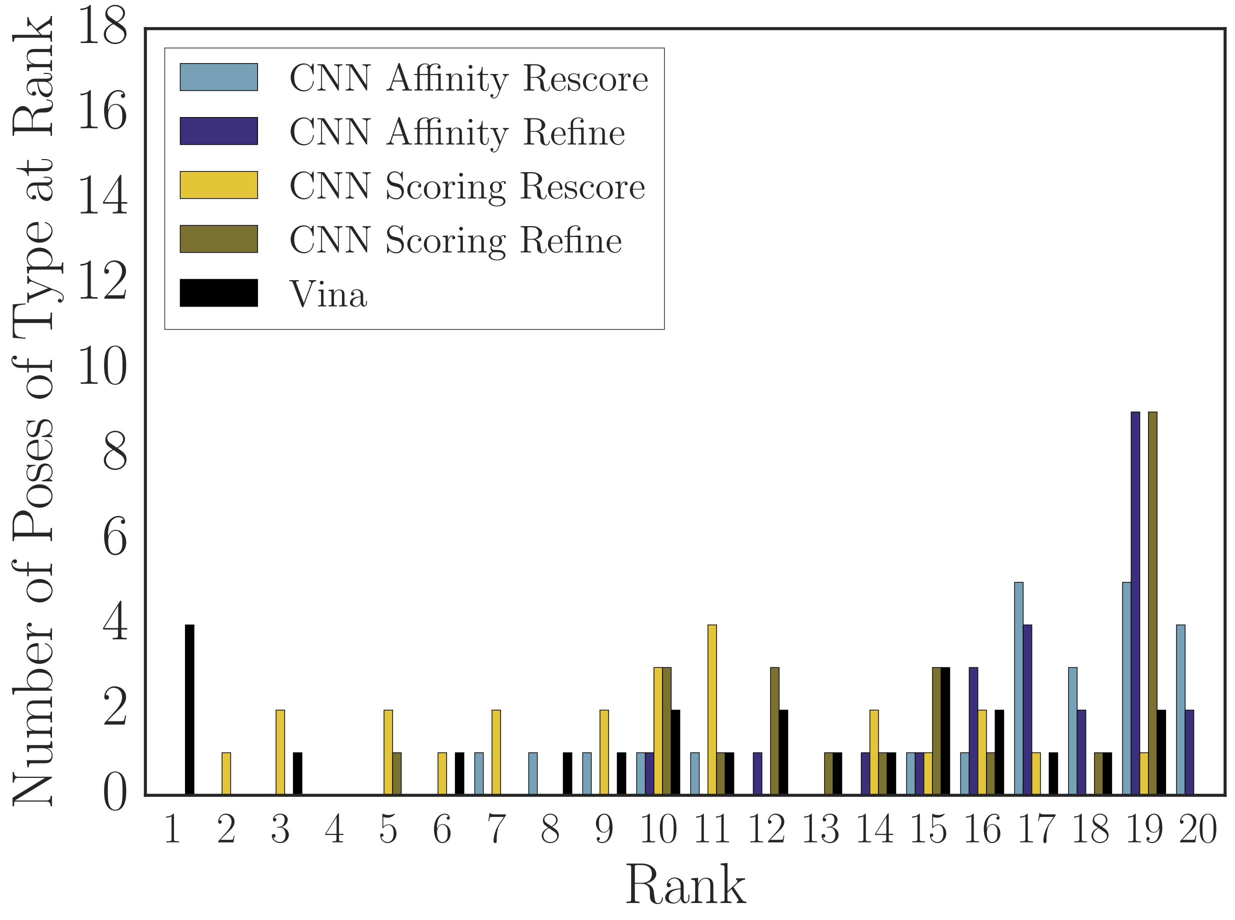


Figure 29: Performance of each method at ranking the minimized crystal pose among all other poses generated during stage 1B and re-scored crystal pose.

As one measure of the accuracy of a scoring method, we can use the re-scored crystal and minimized crystal poses to determine at which rank they appear when ranking them among the poses sampled during the challenge. Figures 28 (crystal poses) and 29 (minimized crystal poses) show the results of performing that ranking. Vina and the two CNN refinement-based methods rank the crystal poses for over half of the compounds at the lowest position in the ranking (rank 20), while the CNN affinity model rescore of the crystal pose ranked with its rescore of Vina-sampled poses ("CNN Affinity Rescore") places the crystal poses for over half the compounds in the last three positions of the ranking. In contrast, the CNN scoring model rescore of Vina-sampled poses ranks the crystal poses for 8 compounds in its top 5 poses, and it accounts for over half of the crystal poses by rank 8. Only Vina and the CNN scoring model rank any crystal poses in their top 5 for any of the compounds.

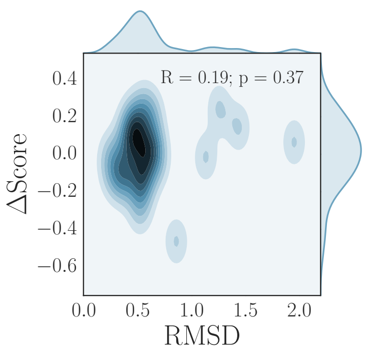
Methods were somewhat more likely to place crystal poses minimized with respect to their own scoring function at a high rank than the crystal poses themselves. The CNN refinement method with the scoring model ranking has one such pose in its top 5, and all methods feature at least one minimized crystal pose in their top 10. Vina has more mimimized crystal than crystal poses in its top 5, and their average rank is higher; in contrast the CNN scoring model applied to rescoring Vina's poses ("CNN Scoring Rescore") has fewer of Vina's minimized crystal poses in its top 5 than it had crystal poses, but it ranks half of the compound's minimized crystal poses in the top 10 compared with Vina's ranking of half within the top 12.

These figures and the associated underlying data suggest that the CNN scoring model has a slight preference for the true crystal poses over Vina's minimized crystal poses. Specifically, 8 crystal poses appeared in its top 5 while 5 Vina-minimized crystal poses appeared in its top 5. 10 of those poses were crystal/minimized crystal pairs, with 3 crystals appearing at a higher rank than their minimized partner and 2 minimized poses appearing above their crystal partner; when a minimized pose appeared above the crystal, the average deviation in their ranks was 1, but when a crystal was ranked higher, the average deviation in their ranks was 2.3. The remaining 3 poses were crystal poses for which the corresponding minimized crystal pose appeared outside of the top 5. In contrast, Vina ranks 4 minimized crystals at rank 1, followed by their corresponding crystal poses at rank 2, and then a lone minimized crystal pose at rank 3. However, since CNN refinement generally produced poses even further away from the crystal pose, and the CNN scoring refinement pose generation method mostly ranked those poses over the crystal or minimized crystal poses, it is not the case that the CNN scoring model generally has a global minimum closer to the CatS crystal pose than Vina does; all that appears to be true is that the crystal pose is typically closer to a CNN scoring model minimum or saddle than a Vina-produced pose is. Furthermore, across all models it is true that the crystal pose or the nearest local minimum according to either Vina or the CNN pose scoring model generally do not coincide with the global minimum.

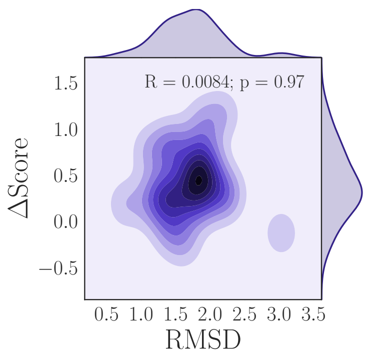
Figure 30 takes a closer look at a projection of the landscape of the three scoring models in the region around the crystal poses for the CatS pose prediction compounds (the CNN scoring model output, which is a probability, has had the logit transform applied). The left column shows the RMSD of the poses produced by minimizing crystal poses with Vina and the change in score associated with the CNN affinity model (30a), CNN scoring model (30c), and Vina (30e). The right column shows the RMSD of the poses produced by minimizing crystal poses with the CNN scoring model and the change in score associated with the CNN affinity model (30b) and the CNN scoring model (30d).

One pattern that emerges is that minimizing with the CNN scoring model (middle right) tends to produce poses that are further from the crystal than Vina does; it also produces a larger range of changes in score, with a distribution that is potentially bimodal, including examples for which it performed comparatively large rearrangements of the input to produce correspondingly large changes in the final score. This does not appear to happen when performing minimization with Vina, suggesting that the CNN scoring model has a smoother landscape, at least around these minima, since it moves a larger distance before converging; alternatively Vina may on average have minima nearer to the crystal pose than the CNN scoring model does, or a combination of both factors may be relevant. Additionally, it is evident that the CNN scoring model is not correlated with Vina, nor is it correlated with the CNN affinity model; the only scoring model relationship that shows any correlation is the CNN affinity model with Vina.

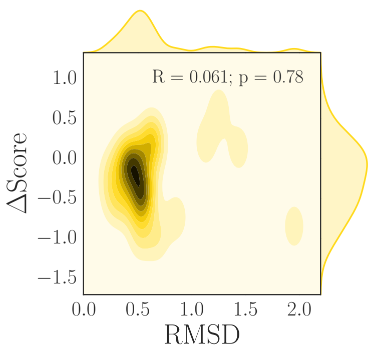
Figure 30f shows a related analysis for crystal pose minimization; challenge stages 1 and 2 involved submitting affinity predictions that were based on poses generated as described. For stage 1 the analysis above demonstrates that these poses were typically far from the true poses, while the process of minimizing the crystal poses produced low-RMSD poses that coincided with a scoring model local minimum for nearly every compound (one compound was minimized to a configuration that was slightly outside of our definition of “low-RMSD” but is still much closer to its crystal pose than the stage 1 poses were). Notably, our submission for stage 1 produced the top-ranked correlation for predicting the affinity rankings of the cocrystal ligand CatS subset among all GC3 submissions, and our overall affinity rankings were also reasonably well-correlated during stage 1 (Table 3). However, in stage 2, with unblinded cocrystal structures available, our affinity prediction performance for CatS actually worsened. Additionally, the method that produced good correlations when predicting affinities for the cocrystal ligand subset was the CNN affinity model, which we do not train to predict poses. Thus we have some reason to suspect that our CNN affinity rankings for CatS are pose insensitive, or at least that whatever aspect of the poses that is useful for predicting affinities is not related to the experimental validity of those poses. Figure 30f shows that while Vina's affinity prediction correlation significantly improves when using just the minimized crystal poses compared with the random poses from stage 1, and the CNN scoring refinement method improves to a lesser extent, the CNN scoring method that simply rescores Vina's poses has virtually identical correlation in these two cases (i.e. the poses do not matter) and the two CNN affinity-based methods actually have worse performance when using the *correct* poses.



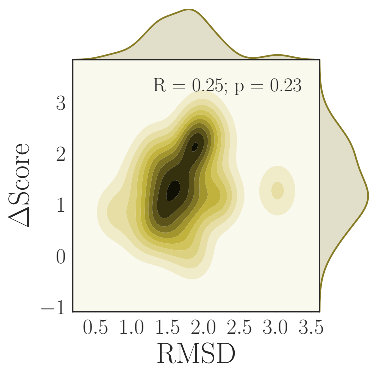
(a) CNN affinity, rescore of Vina



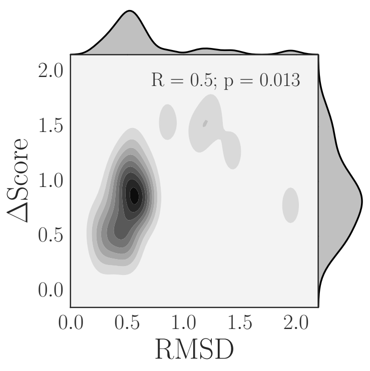
(b) CNN affinity, rescore of CNN score



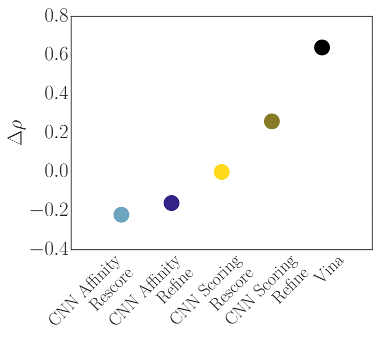
(c) CNN scoring, rescore of Vina



(d) CNN scoring



(e) Vina



(f) Correlation change, stage 1 to minimized crystal

Figure 30: (a-e) Change in score vs. change in RMSD for crystal pose minimization, showing differences and correlation (or lack of correlation) in the functional landscape between the different scoring methods. The logit of the CNN score was used to compute ΔScore on the relevant plots. (f) Change in Spearman ρ when scoring with the high RMSD stage 1 CatS poses versus scoring with the minimized crystal poses.

3.1.3.2 Affinity rankings Next we consider our performance at producing affinity rankings. Table 3 shows the ranks and Spearman ρ correlations of our best performing CNN models, as well as whether they outperformed Vina according to this metric. We find that the CNN models, particularly CNN scoring, generally outperform Vina at producing scores that correlate with compounds' true affinity. Additionally, the correlations associated with both JAK2 subchallenges, as well as the TIE2 subchallenge, are relatively strong, and the overall rank of our best submissions for those subchallenges are competitive with others who participated in GC3, as shown in the table's rank column. It is notable that the best-performing method for two of those three correlated sets of predictions is the CNN affinity model. In contrast, our performance on target p38α in the original challenge was extremely poor.

Table 4 shows the ranks and Matthews correlation coefficients in a manner similar to the previous table; this statistic represents performance at binary classification of actives. This analysis suggests that the CNN affinity model has an advantage at active/inactive discrimination when compared with both the CNN scoring model and Vina.

Target	**Rank**	ρ	**Method**	**Vina**
CatS (1a)	6/53	**0.37**	CNN scoring refine	0.19
JAK2 (SC2)	1/27	**0.74**	CNN scoring refine	0.05
VEGFR2	14/33	0.39	CNN scoring refine	**0.51**
p38α	7/29	**0.04**	CNN scoring refine	-0.34
JAK2 (SC3)	2/18	**0.75**	CNN affinity refine	-0.33
TIE2	3/18	**0.67**	CNN affinity rescore	0.14
ABL1	N/A	0.56	CNN affinity rescore/refine (tie)	**0.72**

Table 3: The rank of our top-performing CNN method among all submissions to D3R GC3 affinity ranking tasks, along with the Spearman ρ associated with that method, and the Spearman ρ associated with Vina's predictions. The higher correlation between the best-performing CNN method and Vina's is bolded. The ABL1 results were not submitted during the challenge period and all official submissions were partials, so we do not show a rank here.

Target	**Rank**	**MCC**	**Method**	**Vina**
JAK2 (SC2)	3/27	**0.44**	CNN affinity refine	0.07
VEGFR2	1/33	**0.53**	CNN scoring rescore	0.34
p38α	9/29	**0.21**	CNN affinity refine	0.15
JAK2 (SC3)	2/18	**0.23**	CNN affinity refine	-0.55
TIE2	1/17 (tie)	**0.78**	CNN affinity rescore	0.55
ABL1	N/A	0.56	CNN affinity rescore/refine (tie)	**1.00**

Table 4: The rank of our top-performing CNN method among all submissions to D3R GC3 affinity ranking tasks based on active/inactive discrimination, along with the Matthews correlation coefficient associated with that method, and the Matthews correlation coefficient associated with Vina's predictions. The higher correlation between the best-performing CNN method and Vina's is bolded. The ABL1 results were not submitted during the challenge period and all official submissions were partials, so we do not show a rank here.

Figure 31 shows the correlations associated with all the methods we used to generate affinity rankings, as well as the correlation that can be obtained by simply using the compounds' molecular weight (the molecular weight ranking is misleading for target ABL1, since there are only two compounds that have differing affinities for ABL1 mutants, and the molecular weight gives a high correlation here but completely fails at the prediction task for which the challenge was designed).

There is no one method that performs well across all targets. The CNN scoring model is the top-performing method on one of the JAK2 subchallenges, while the CNN affinity model is the top-performing method on the other, and for each method it is the case that in the subchallenge for which they are not the top-ranked method, they actually perform very poorly. From this figure, it is not clear whether the CNN affinity or CNN scoring model is better-suited to performing the affinity ranking task; each performed well on half of the targets shown here, and the targets on which one method performed well preserve the pattern seen with JAK2 - one CNN model having highly correlated scores for a target is mutually exclusive with respect to the other CNN model.

Similarly, Figure 32 shows the Matthews correlation coefficients associated with all the methods used to generate affinity rankings for each target for which this analysis is appropriate, as well as the correlation obtained by using molecular weight. It again suggests that the CNN affinity model has an advantage at active/inactive classification over the other methods, where it is the best-performing method on four of the targets and comparable to the best-performing method on another; here its *consistent* good performance is unique among the discrimination methods used.

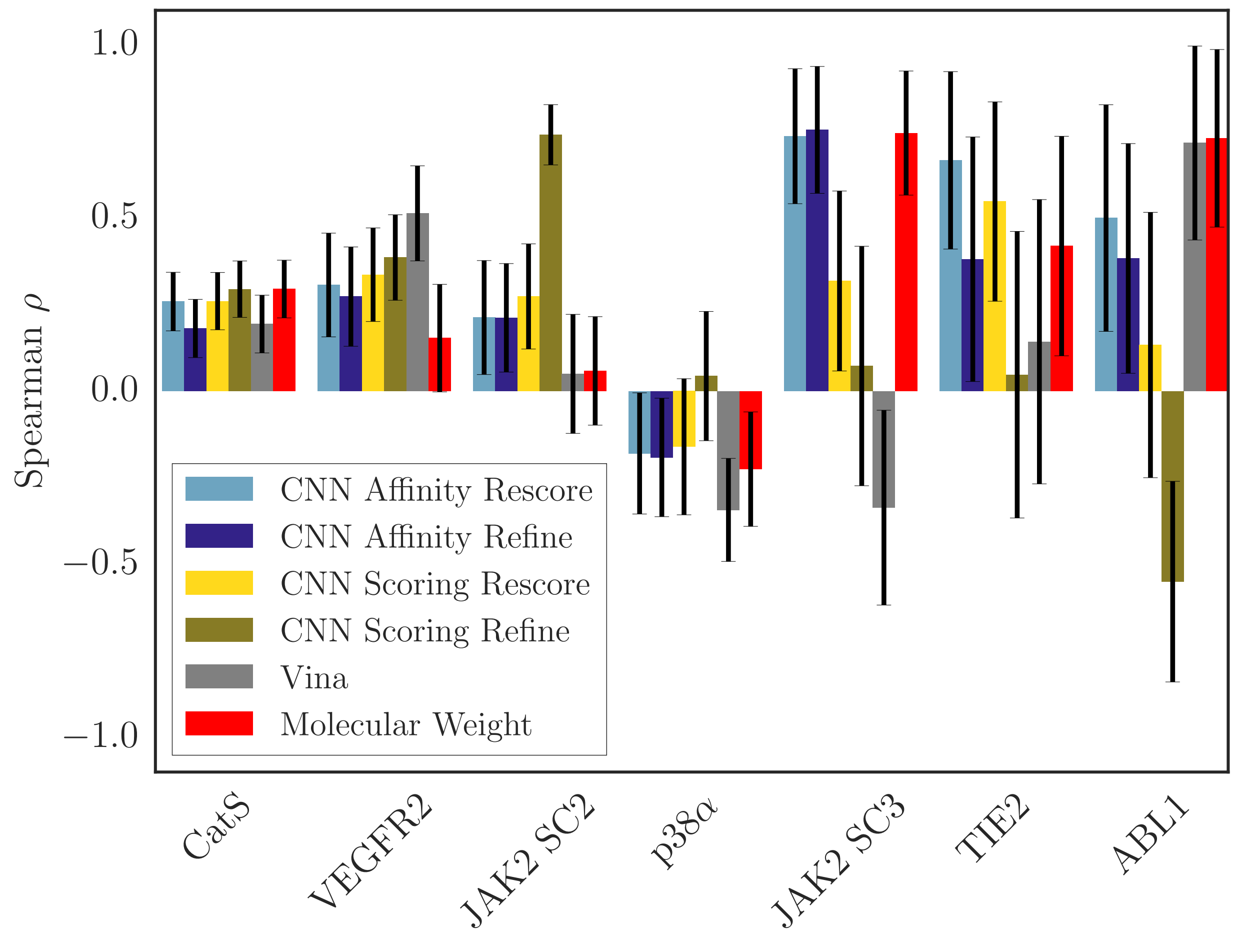


Figure 31: Spearman ρ of each scoring method with the associated experimental data for each target; compounds with $K_d \geq 10\mu M$ have been omitted. Black lines indicate error bars computed by bootstrapping the correlation for 10,000 iterations by resampling data points with replacement. For the bootstrapped correlations, the experimental data was perturbed with randomly generated Gaussian noise $\epsilon \sim \mathcal{N}(0, RT \ln(I_{err}))$, where I_{err} was taken to be 2.5.

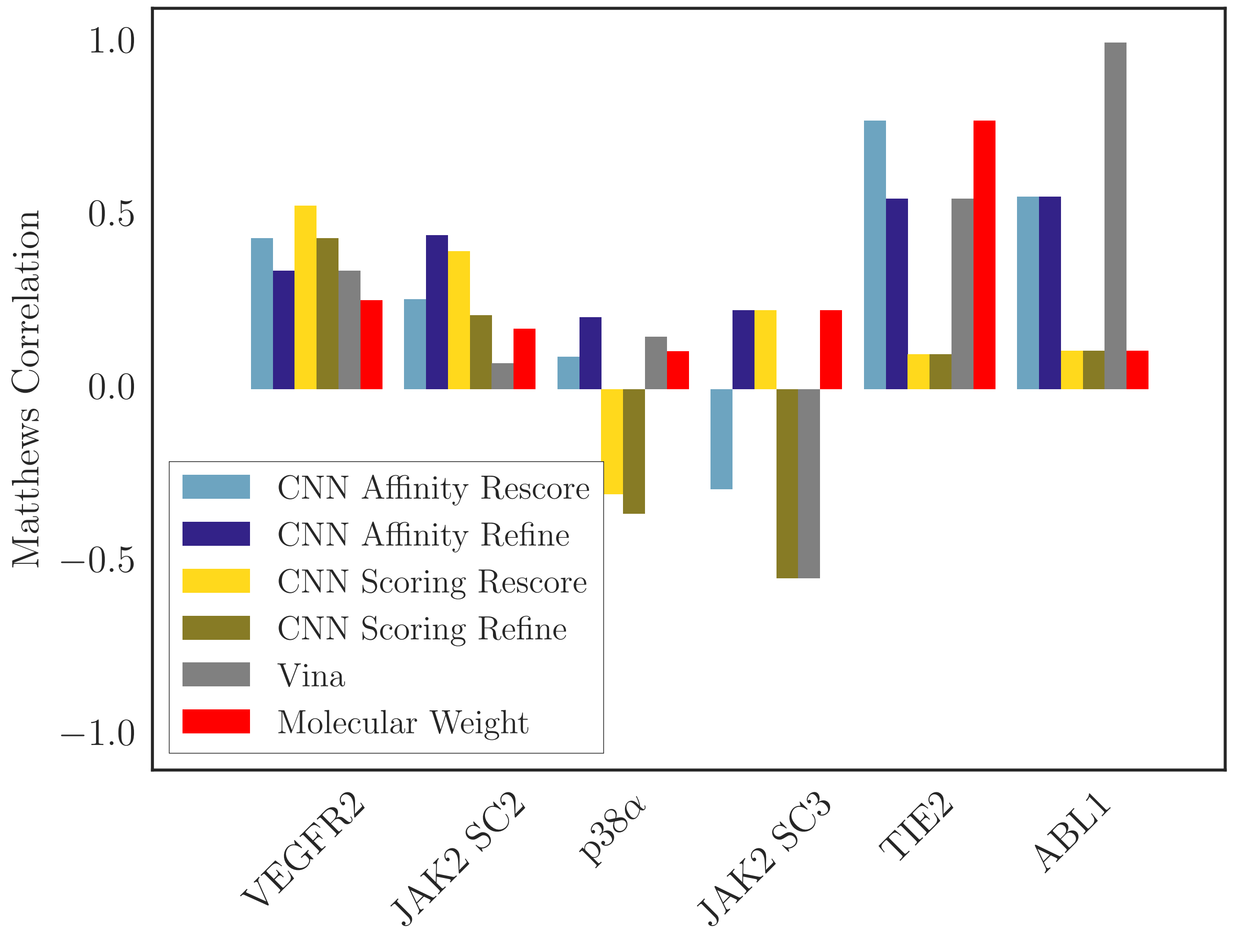


Figure 32: Matthews correlation coefficient of each scoring method with the associated experimental data for each target. The K_i compounds with $K_d \geq 10\mu M$ were taken as inactive, and the corresponding K_i bottom-ranked compounds for each prediction method were also taken as inactive. The remaining compounds in both cases were taken as active, and the resulting set of true/predicted pairs were used to compute the correlation. Error bars are not shown as this procedure is not amenable to bootstrapped error estimates.

As an alternative view of the CNN ranking performance, Figure 33 shows the per-target ROC plots for the six subchallenges that had compounds with affinities both above and below $10\mu M$, with affinities at or above that point being considered inactive and affinities below that point considered active. Figure 34 instead shows boxplots of the AUCs derived from the ROC plots in Figure 33. In this binding discrimination task, the CNN affinity model is clearly better than the CNN scoring model, and it is generally better than Vina as well, especially when the poses it is scoring were produced via refinement with the CNN scoring model.

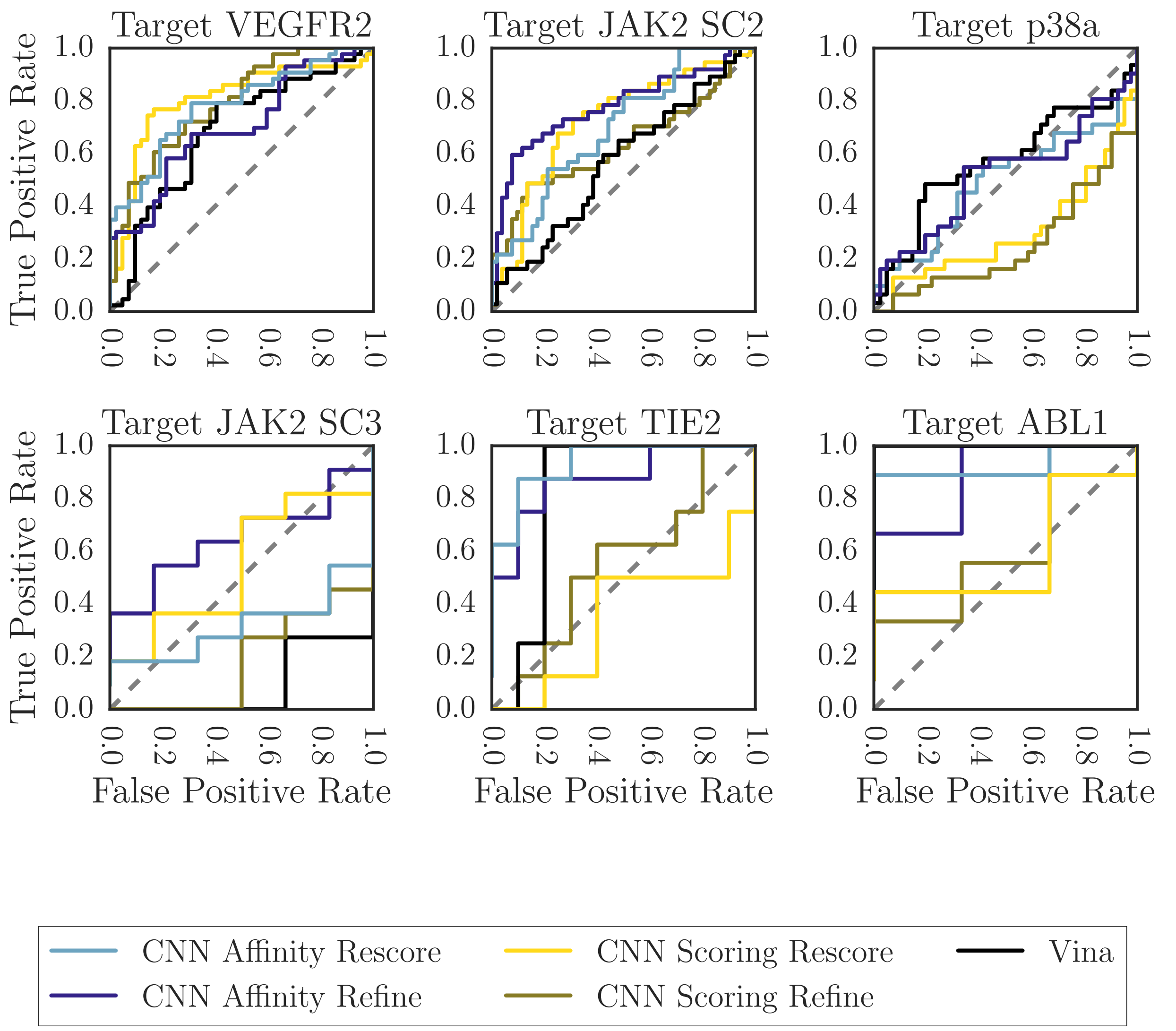


Figure 33: ROC curves for all GC3 targets for which there were compounds with $K_d \geq 10\mu M$ and compounds with $K_d < 10\mu M$; the former were considered inactive and the latter active for the purposes of this figure.

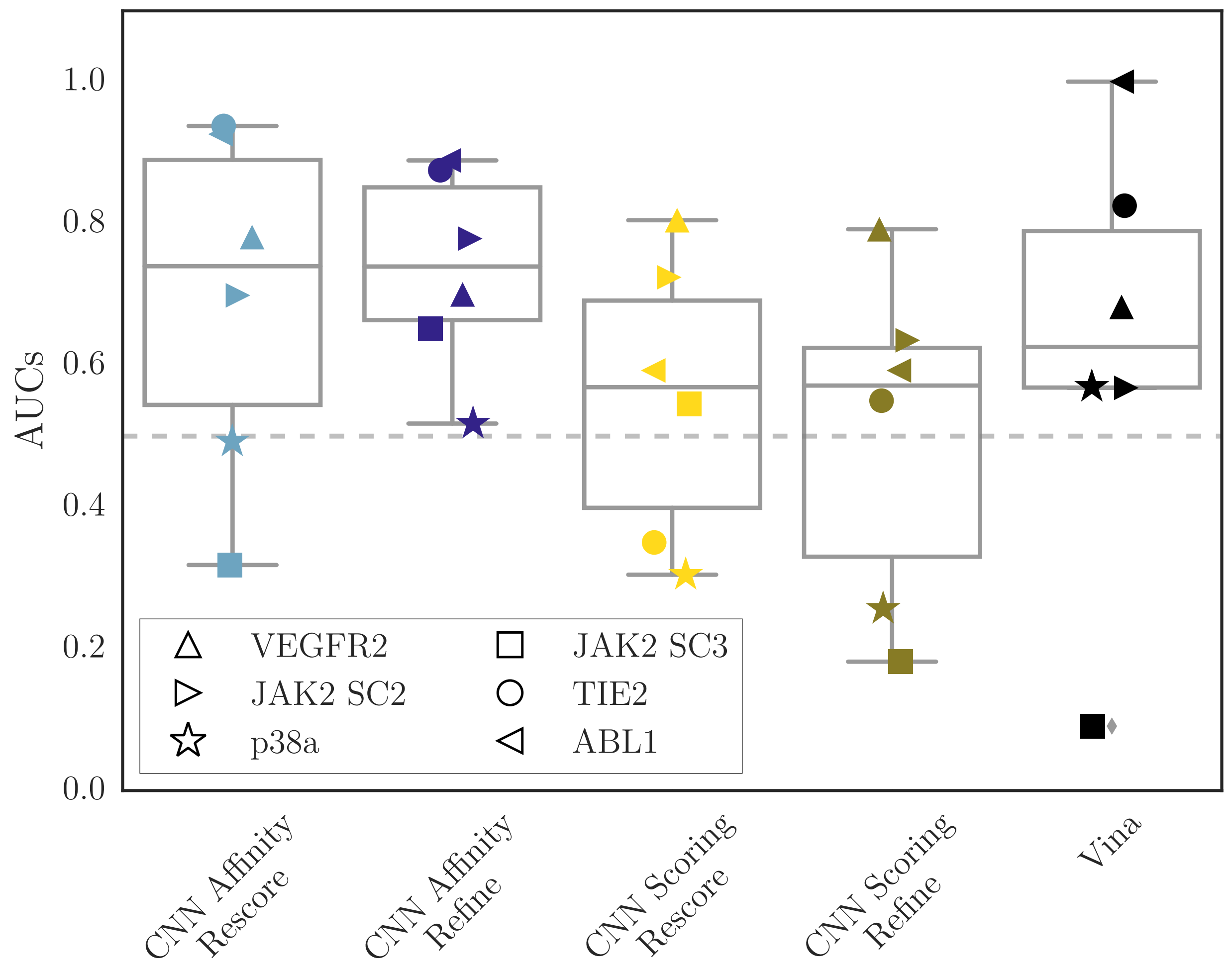
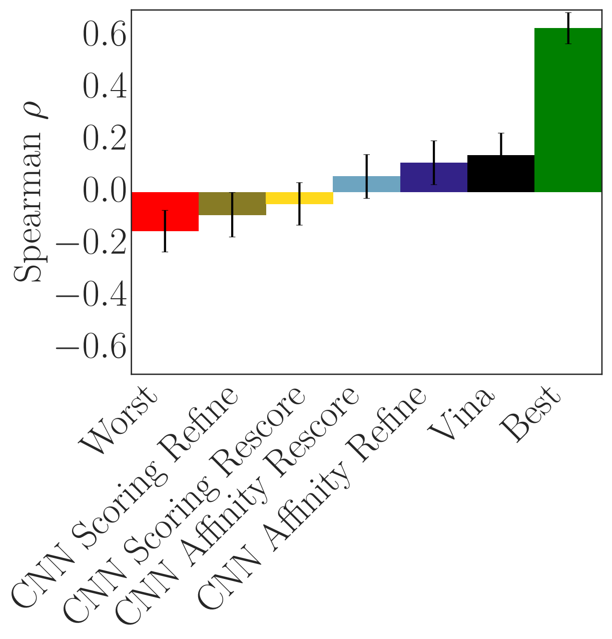
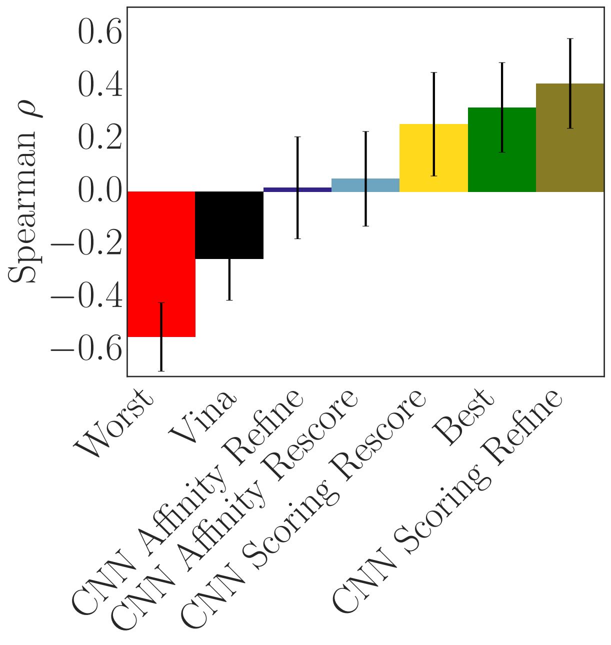


Figure 34: Boxplots of AUCs across all methods and GC3 targets for which there were compounds with $K_d \geq 10\mu M$ and compounds with $K_d < 10\mu M$; the former were considered inactive and the latter active for the purposes of this figure.

Since we did not use ligand similarity between prediction compounds and available reference structures to identify possible binding modes during the challenge, we considered whether using this information would have been beneficial when producing affinity rankings (particularly on the targets for which we performed poorly). To that end, we utilized the approach described in section 3.1.3.1 to perform alignment, minimization, and docking based on the available reference structure whose crystal ligand had highest similarity to each query compound.



(a) CatS



(b) p38a

Figure 35: Performing the scoring and affinity ranking process again using ligand structural information for (a) CatS and (b) p38α. The best and worst overall submissions from the original challenge are shown with the new results. Using this process, we had worse performance on predicting CatS affinity rankings but much better performance on p38α.

The results of this approach on pose prediction were shown in Figure 25c and Figure 25d, while its effect on the correlation of the final scores (when taking the top-ranked prediction for each compound as its predicted affinity) is shown for CatS in Figure 35a (which shows the correlation produced by the same method used to generate 25d). In general, when using our current scoring functions for CatS, methods that produce better poses also produce worse correlation for the predicted affinities - for example, our original stage 1A Spearman ρ was 0.37, while the ligand-similarity based method shown in 25d and 35a had a best-case Spearman ρ of 0.14.

We also performed this analysis to generate new predictions for p38α using ligand similarity. The best correlation produced by that analysis is shown in Figure 35b. This method produced good correlation using the CNN scoring model, particularly when the scoring model was used to sample poses itself in the final refinement. Notably, these results were dependent on the method used to identify similar ligands as well as the alignment method; for CatS, Tanimoto distance computed with OpenBabel FP2 fingerprints and a scaffold alignment (using the hand-selected scaffold) produced the best results for pose prediction (though no similarity-based method we tried produced correlation that matched our original stage 1 submission), while for p38α the best result was produced using a Tanimoto distance computed with RDKit Daylight-like fingerprints and O3A shape alignment.

3.2 Transfer learning and a foray into model explanation

While D3R Grand Challenge 3 was a prospective evaluation, it was small and consisted of at least one affinity ranking task that was perfectly correlated with molecular weight. In theory a virtual screening benchmark could provide a more expansive opportunity to evaluate our latest CNN models, with two caveats: (1) we do not train our newest models with any inactive *compounds*, only inactive *poses*, which some[202] have asserted to be inadequate for good virtual screening performance (though we suspect based on the previous section that knowledge transfer from one task to the other should occur), and (2) in the last year several papers have reported[165][30][192] that existing virtual screening benchmarks are highly biased due to historical artifacts in compound identification and synthesis. The latter problem results in datasets that reward memorization of these historically biased features, which when used to train statistical models produces scoring functions that work well on many datasets compiled from the literature but fail to learn features that meaningfully represent the interaction between a protein and ligand. They therefore fail to generalize to examples that don't conform to the bias, which primarily exists due to human-guided compound search and optimization. One goal of the following section is to investigate how well our models generalize to virtual screening, particularly in comparison with other existing scoring functions (both machine learning and empirical). In light of the discussion about model evaluation and dataset bias, we also propose a novel baseline based on fitting models to our training datasets using the known biases as features as part of a larger goal of attempting to explain the performance of our models.

3.2.1 Introduction

Virtual screening poses this problem: given a target molecule and a set of compounds, rank the compounds so that all those that are active relative to the target are ranked ahead of those that are inactive. An *in vitro* screen is the source of ground truth for this binding classification problem, but there are at least four significant limitations associated with performing them: time and cost limit the number of screens that can be run; only compounds that physically exist can be screened this way; the screening process is not always accurate; and *in vitro* activity against a given target is necessary but not sufficient for identifying useful drugs (perhaps this is a separate problem from virtual or *in vitro* screening, but from a practical standpoint it would be desirable to exclude compounds with problematic properties from the beginning of a drug discovery campaign, and in theory a virtual screening method could penalize such compounds in a ranking). Thus virtual screening has attracted significant interest as a way of overcoming these limitations to identify strong drug candidates at reduced cost - in theory. In practice those benefits have yet to be realized.

Virtual screening methods can be broadly classified as ligand-based or structure-based. Ligand-based methods rely on information about known active compounds and base their predictions on the similarity between compounds in the screening database and these known actives. No 3D structures are required, but at least one known active is. There are many possible similarity metrics, but regardless of which is used, identifying truly novel actives with this approach is unlikely. In contrast, structure-based approaches derive from a model of the interaction between a protein and ligand, facilitating identification of truly novel interactions betwen the two. A “scoring function” maps the input structure representing the relative location and orientation of the pair of molecules to a score representing the strength of their interaction. Several different approaches have been applied to scoring function development, yielding four major classes. Force fields, empirical scoring functions, and knowledge-based functions (also referred to as statistical potentials) are known collectively as “classical” scoring functions, distinguishing them from the newer machine learning scoring functions. Briefly, force fields rely on physics-based terms mostly representing electrostatic interactions; empirical scoring functions may include counts of specific features as well as physics-inspired pairwise potentials; and knowledge-based statistical potentials calculate close contacts between molecules in structural databases and fit potentials that favor structures that resemble that existing data. In comparison, modern machine learning scoring functions tend to impose fewer restrictions on the final functional form and often attempt to learn the relevant features from data (for example, they may consist of a neural network that processes the structural input directly).

Because structure-based approaches rely on a representation of the binding mode defined between the protein and ligand structures, the first step in using them is often generating one or more plausible binding modes. A typical approach is to start from a protein structure and use a scoring function to identify favorably scored conformations of all compounds of interest (i.e. “docking”) within a search space defined on the surface of the protein. That scoring function may differ from the scoring function that will be used to generate the final compound ranking for the virtual screen; a persistent problem in this domain has been difficulty in simultaneously optimizing scoring functions for accurate binding pose scoring and accurate compound ranking. This “pose prediction” task should be fundamental to structure-based approaches to virtual screening, since these approaches aim to use the physical interactions underlying binding to guide scoring; if those interactions are not represented accurately by a pose used for scoring, the scoring method is likely to primarily derive virtual screening accuracy from the ligand alone - i.e. it would in theory offer minimal advantage over purely ligand-based approaches. In practice it has been found that for existing scoring functions, accurate input poses are not essential for good performance at binding affinity prediction[109].

Well-designed benchmarks can be constructed to *require* more than simple descriptors derived solely from the ligand to achieve good virtual screening performance; poorly designed benchmarks are particularly susceptible to delivering “state-of-the-art” performance when used to train and evaluate machine learning scoring functions merely because they can be perfectly classified using descriptors so simple that classical scoring functions would never be so naive as to use them as the sole basis of a scoring model. Such biased benchmarks may have limited utility for evaluating an existing scoring function - good performance on the benchmark could derive from either a uselessly simple or a sophisticated model, and the dataset's bias means that if the goal of the benchmark is to predict a model's ability to perform well on an unknown dataset, the benchmark may only provide information about the model's prospective performance on another dataset with the same bias. These benchmarks may be of limited utility for training machine learning scoring functions that generalize to real-world tasks, since training on them may merely produce a model that recapitulates their biases. Thus while these biased benchmarks could have served as acceptable assessments of classical scoring functions, where the explicit design choices made by human researchers eschewed the achievement of perfect performance via exploitation of dataset bias, fitting modern machine learning scoring functions to them risks creating models that have been "taught to the test" and cannot be expected to generalize beyond it.

Once problems with an existing dataset are identified, the challenge of constructing an improved alternative remains; this problem, combined with the need to compare new scoring functions with existing published results for older scoring functions (which may have exclusively had access to benchmarks now deemed problematic), ensures the continued relevance of now disfavored benchmarks. Such is the case with DUD[76], DUD-E[128], and MUV[151], three virtual screening benchmarks that have been widely used to assess scoring functions in the literature.

More recent literature[165, 192, 30] has demonstrated that both MUV and DUD-E are biased and are likely to be unsuitable for training or even validating machine learning scoring functions. Sieg et al[165] found that for DUD, DUD-E, and MUV, better-than-random (and in the case of DUD and DUD-E, perfect) AUCs could be obtained merely by fitting cross-validated models on exactly the simple chemical descriptors that the dataset developers had attempted to control for during dataset construction. For DUD-E, synergistic effects were associated with using multiple descriptors together; the authors note that this probably derives from the construction process, which matches each feature separately in its one-dimensional feature space, unlike MUV, which considers distances within the multidimensional feature space. Accordingly, the authors find that MUV does not afford synergistic performance when including additional features. Wallach et al[192] explain that MUV considered only the difference between active-active and active-inactive distances, omitting a comparison of inactive-inactive and inactive-active distances; since class labels for machine learning models are arbitrary, the MUV approach may produce datasets where "actives" are not clumped but the "inactives" are, and a machine learning scoring function can in principle learn from the intraclass similarity of either class. Further, as Sieg et al[165] point out, the MUV dataset was constructed for ligand-based similarity search, and therefore it is likely to be inappropriate for benchmarking machine learning methods due to inherent analogue bias. Finally, Chen et al[30] note that there is high similarity among inactives across targets in DUD-E, biasing that benchmark even futher.

In their paper describing the limitations of only considering distances relative to actives in the MUV dataset construction approach, Wallach et al[192] propose Asymmetric Validation Embedding (AVE), an improved measure of bias that considers clumping among inactives and between examples from the same class used in the training and validation sets. They do not construct a new dataset using AVE, however; rather, Tran-Nguyen[184] first reported a novel dataset, LIT-PCBA, that used AVE for unbiasing and was explicitly designed for training and validation of machine learning scoring functions. It consists of 15 target sets, with a total of 10033 actives and 2798737 inactives (though some of those are duplicated across multiple targets) after initial filtering of 9780 actives and 407839 inactives after constructing AVE-unbiased training and validation splits. 13 of these targets have more than one PDB template provided. All compounds are taken from assay data and therefore all inactives are confirmed rather than assumed. The authors also confirmed that the included actives were not too biased toward high affinity compounds (i.e. the actives have typical potencies found in HTS decks) and that they were diverse when compared with other actives included for a given target. For all included targets, an EF1% > 2 was achievable by at least one of a fingerprint-based, shape-based, or structure-based approach prior to AVE unbiasing.

3.2.2 Methods

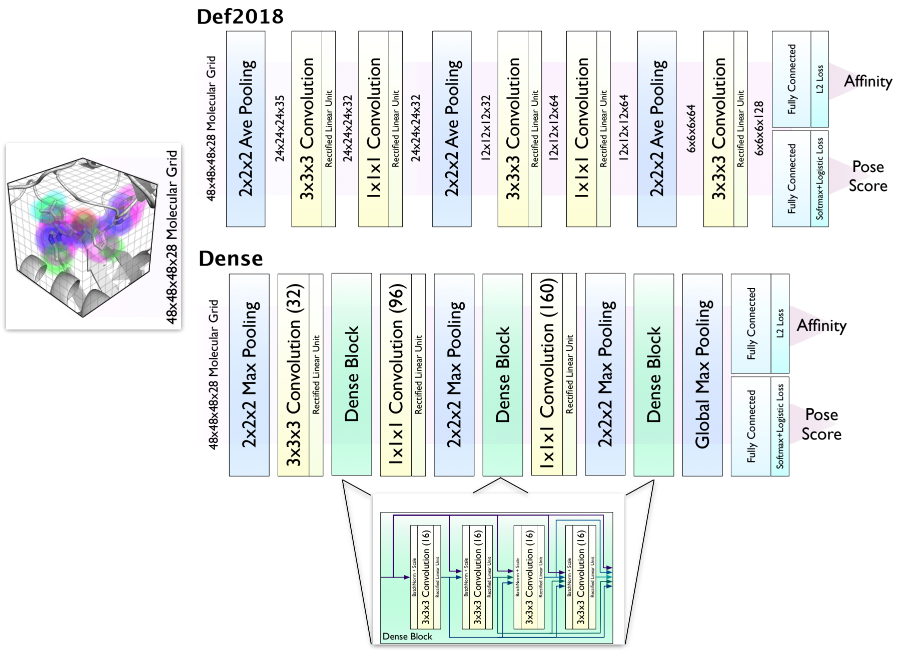


Figure 36: Voxelized grid-based CNN architectures evaluated in this work.

Our approach to using machine learning for molecular modeling is based on using 3D grids derived from voxelizing a fixed-size box centered on a protein binding site[147, 176]. We have experimented with pairing this input representation with several different model architectures[147, 176, 72, 57] and have also evaluated several types of training data in terms of their performance for pose and binding affinity prediction. We previously[147, 178, 176] tested the use of CNNs for virtual screening, but always in the context of simpler model architectures and training sets based on virtual screening benchmarks. Here we use our latest model ensembles[57], based on two architectures as shown in Figure 36; the Def2018 architecture was trained on two sets of training data with one consisting of redocked poses from the 2016 PDBbind General set and the other consisting of cross-docked poses generated based on Pocketome v17.12. The Dense architecture (based on [75]) was only trained on the cross-docked poses from Pocketome. The Def2018 architecture is a Bayesian optimized version of our earlier architectures, while the Dense architecture includes densely connected blocks that are intended to combine features at multiple levels of abstraction when making a prediction. All models were trained with five random seeds, yielding an ensemble for each model (for more details of the training procedure, see [57]). These ensembles allow us to compare the effects of architecture and training data on virtual screening performance, as well as approximate the uncertainty in our predictions.

Note that *none* of these models were trained to perform virtual screening; their outputs do not classify an input as “active” or “inactive” directly, nor were they in general provided distinctly “active“ or “inactive” *compounds* as input examples (i.e. they were not trained on any virtual screening datasets). They were simultaneously trained to predict whether a given input is a binding mode based on its distance from an available crystal structure and, if so, what its affinity would be; if a pose is not a binding mode, the predicted affinity is instead optimized to be lower (in pKa units) than the binding affinity. While accurate binding affinity prediction might theoretically be expected to correlate with virtual screening performance, in practice the available data distributions for these two tasks are very different. This has led to a bifurcation in the machine learning scoring function/binding affinity prediction literature, with a given model typically developed for and assessed on one task or the other despite the theoretical transferability of knowledge between tasks.

We use DUD-E and the more recently published LIT-PCBA dataset to assess virtual screening performance. DUD-E is primarily used to facilitate comparisons with published work. LIT-PCBA is appealing due to its principled construction; even though we do not use the training and validation splits and instead assess our performance on the full dataset, it still features diverse actives, more typical (lower) potency actives, and topological similarity between actives and inactives. Neither dataset was used for training, primarily because in our past work[147] we found that when training without known poses (i.e. purely or mostly with assay data and computer-generated poses), the learned models were effectively ligand-based. DUD-E’s known bias also suggests that it is unsuitable for model fitting, but that does not *necessarily* imply that it is useless for evaluating a model fit on other data. Further, our past work suggests there is utility in comparing model performance when testing on an independent dataset versus performing cross-validation, since improved performance at classification on a dataset when training on a subset of it could be due to dataset-wide bias artificially enhancing performance (as appears to be the case with DUD-E, where similarity among inactives between targets constitutes test set leakage[30]).

Since most of the poses we use for training and scoring are generated with the smina[99] fork of Autodock Vina[187], we take Vina as our conventional scoring function baseline on whose performance we hope to improve. We also compare with Vinardo[144], a modified version of the Vina scoring function that aims to improve performance at pose prediction. However, machine learning scoring functions should always be compared to other machine learning scoring functions, ideally trained on the same dataset; thus we also include virtual screening results from two versions of RFScore (RFScore-VS[202], which was trained on DUD-E, and RFScore-4[109], which was trained on the 2014 PDBBind refined set). These two random forest based scoring functions are an interesting contrast to our approach: RFScore-4 has similar training data to ours but is a different type of statistical model that was fit to predict binding affinity with a different training strategy and distinct features, while RFScore-VS was trained specifically for virtual screening.

We used docked poses we had previously generated for DUD-E, obtained with the default smina arguments `--seed 0 --autobox_add 4 --num_modes 9` and a box defined by the crystal ligand associated with the DUD-E reference receptor. For rescoring poses sampled with another function, we have since found it advantageous to generate a larger set of more diverse poses (to give the alternative function opportunities to correct inaccuracies in the sampling function)[176]; thus for the LIT-PCBA evaluation we used `--seed 0 --autobox_add 16 --num_modes 20`. We used our CNN models, RFScore-VS, and RFScore-4 to rescore these poses generated with the Vina scoring function, and we also compared these with the performance of the Vinardo scoring function, which sampled a distinct set of poses. A method's maximum predicted score for a (target, compound) pair was taken as its prediction except where noted otherwise, and the prediction per pose for each CNN model was the mean computed over the five-model ensemble's predictions except where noted otherwise.

DUD-E	MUV
molecular weight	
number of hydrogen bond acceptors	number of hydrogen bond acceptors
number of hydrogen bond donors	number of hydrogen bond donors
number of rotatable bonds	
logP	logP
net charge	
	number of all atoms number
	number of heavy atoms
	number of boron atoms
	number of bromine atoms
	number of carbon atoms
	number of chlorine atoms
	number of fluorine atoms
	number of iodine atoms
	number of nitrogen atoms
	number of oxygen atoms
	number of phosphorus atoms
	number of sulfur atoms
	number of chiral centers
	number of ring systems
6 features	17 features

Table 5: Descriptors used in the construction of DUD-E and MUV

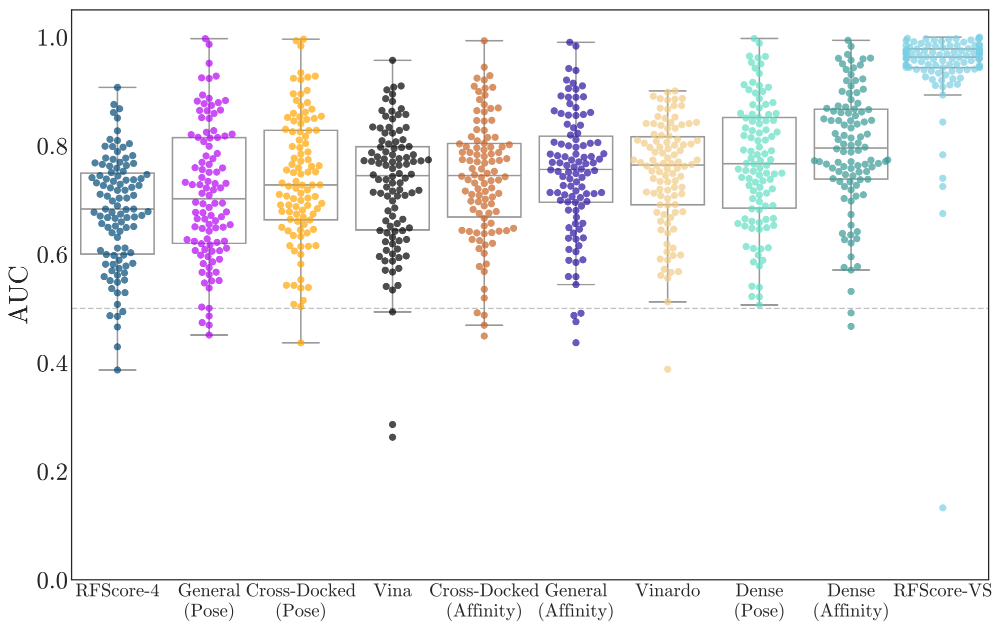
Finally, we also establish baseline performance using a variety of statistical models fit to our training datasets with the simple chemical descriptors used in the construction of DUD-E and MUV as their input features. These descriptors are shown in Table 5.

3.2.3 Results

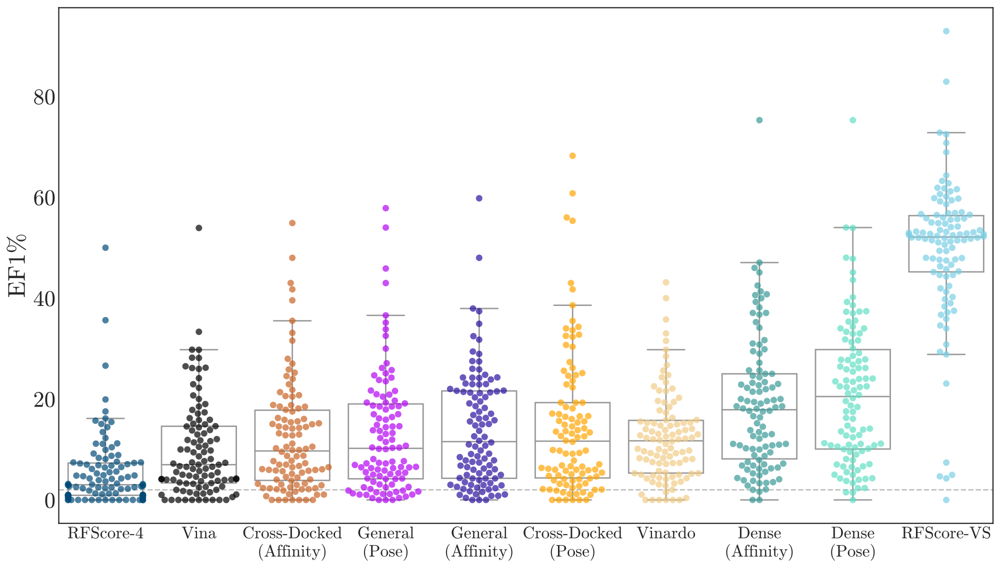
First we will summarize virtual screening performance of our convolutional neural networks, initially comparing with Vina, Vinardo, RFScore-4, and RFScore-VS. We also assess pose prediction performance on the reference receptors provided with LIT-PCBA, which in 13 out of 15 cases involve multiple templates and therefore constitute cross-docking tasks. Next we attempt to explain aspects of the observed performance, in particular taking inspiration from [165] and establishing a baseline ML model fit to the "simple" descriptors of our training sets (Table 5) with which we can compare our performance. Finally we investigate the utility of our CNN model ensembles for either enhancing or explaining virtual screening performance.

3.2.3.1 Virtual Screening Performance Virtual screening performance on DUD-E is shown in Figure 37 and on LIT-PCBA in Figure 38. We provide AUCs for comparison with other literature, but a metric intended to assess early enrichment is a better evaluation of virtual screening performance; we use EF1% since the LIT-PCBA paper reports that value for three baseline methods: fingerprints (1D), shape overlap (2D), and Surflex-Dock (SD). Each target that was included in the final LIT-PCBA benchmark could reach at least EF1%=2 by at least one of those three methods prior to AVE unbiasing; the methods that achieved that minimum performance level are labeled per-target in the legends of the LIT-PCBA figures for comparison with our results. Later we also use the normalized EF1% (NEF) as in [113], where the EF1% for a given target is normalized by the maximum possible EF1% for that target. Notably, LIT-PCBA performance for all of the methods tested differs from the performance achieved by Surflex-Dock in the LIT-PCBA paper. This could be due in part to sampling differences between Surflex-Dock and Vina/Vinardo, but it could also be evidence of shape or 2D descriptors being learned by the ML models. Notably the CNN pose models achieved an EF1%$\geq$6 for ADRB2, and several CNN models (pose and affinity) achieved an EF1%$\approx$4 for PPARG while neither Vina nor the RF-Score models did, and these were both targets for which all models tested in the LIT-PCBA paper achieved an EF1%>2.

For all models, average performance according to either metric is much better on DUD-E than on LIT-PCBA. In the case of RFScore-VS, the performance discrepancy between the two benchmarks suggests that its cross-validation performance on DUD-E was not an accurate representation of its generalization ability, perhaps due to the data biases discussed previously. RFScore-4 has virtual screening performance comparable to other methods tested (particularly Vina), despite not being trained with inactive examples, which have previously been suggested to be essential[202] for good virtual screening performance. Among the CNN models, the Dense models generally perform best (see Table 6 for 95% confidence intervals and p-values for the difference in EF1% between the Dense affinity model and the non-CNN models on each LIT-PCBA target). They were trained on the same data as the Cross-Docked models, and while their greater number of parameters makes them more susceptible to overfitting, if that occurred, we would expect them to have worse performance on out-of-sample tests (later we will examine how far “out-of-sample” these tests really are).

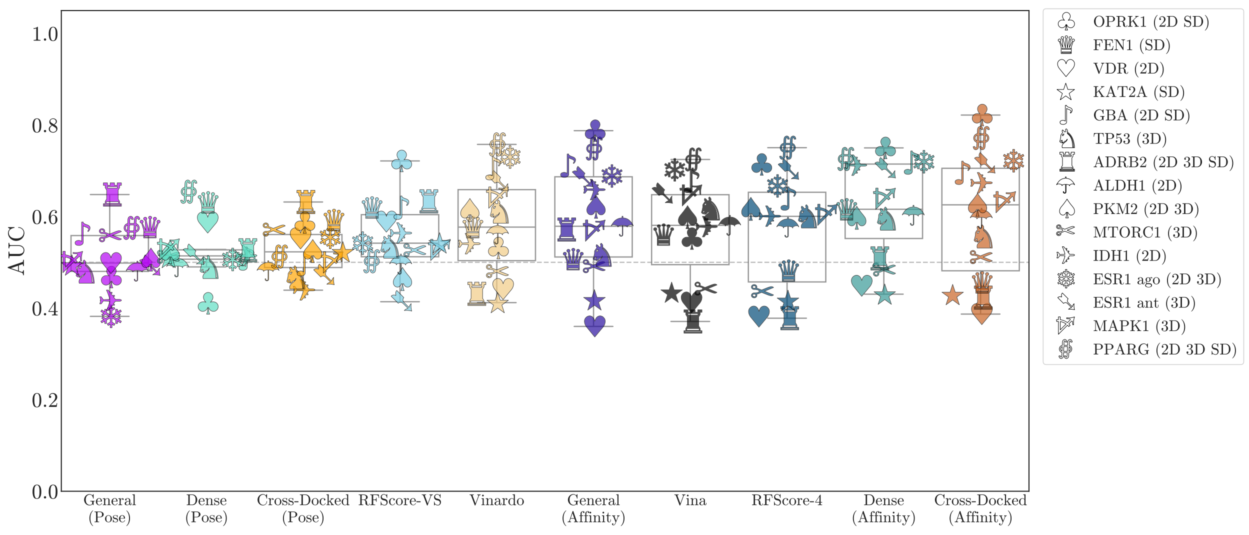


(a)



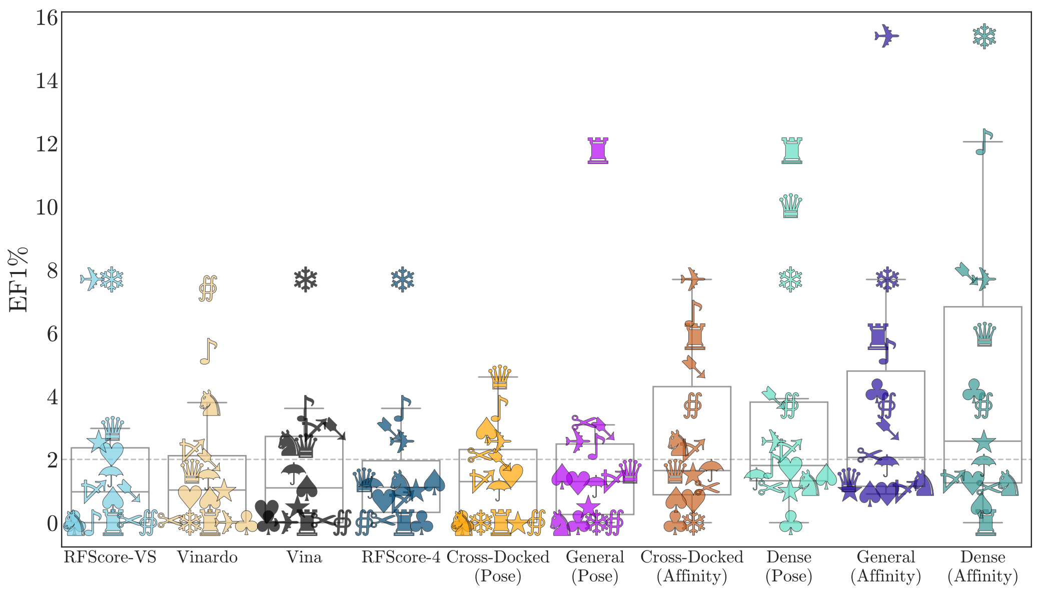
(b)

Figure 37: Assessment of virtual screening performance on DUD-E, using (a) AUC and (b) EF1% as the performance metrics. Note that while EF1% = 1 is equivalent to random performance, we have annotated EF1% = 2 since that was the minimum enrichment required for inclusion in LIT-PCBA[184].



AUC
1.0
0.8
0.6
0.4
0.2
0.0
General (Pose)
Dense (Pose)
Cross-Docked (Pose)
RFScore-VS
Vinardo
General (Affinity)
Vina
RFScore-4
Dense (Affinity)
Cross-Docked (Affinity)
OPRK1 (2D SD)
FEN1 (SD)
VDR (2D)
KAT2A (SD)
GBA (2D SD)
TP53 (3D)
ADRB2 (2D 3D SD)
ALDH1 (2D)
PKM2 (2D 3D)
MTORC1 (3D)
IDH1 (2D)
ESR1 ago (2D 3D)
ESR1 ant (3D)
MAPK1 (3D)
PPARG (2D 3D SD)

(a)

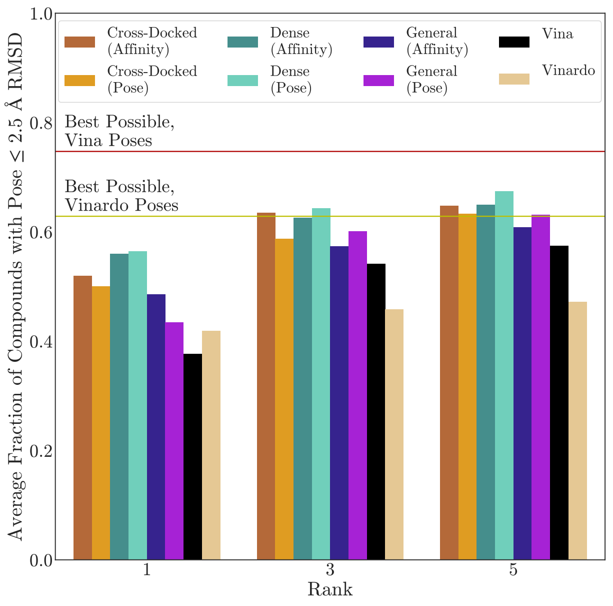


EF1%
16
14
12
10
8
6
4
2
0
RFScore-VS
Vinardo
Vina
RFScore-4
Cross-Docked (Pose)
General (Pose)
Cross-Docked (Affinity)
Dense (Pose)
General (Affinity)
Dense (Affinity)

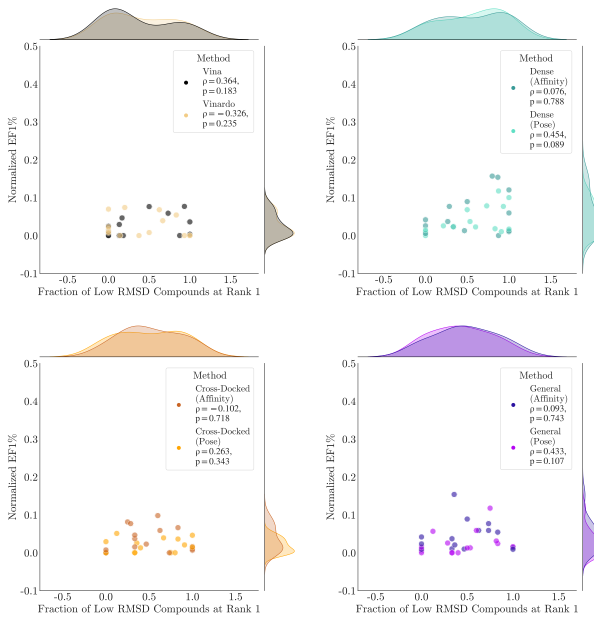
(b)

Figure 38: Assessment of virtual screening performance on LIT-PCBA, using a AUC and (b) EF1% as the performance metrics. Targets are assigned a unique symbol; the legend notes which method achieved an EF1%> 2 in[184], where '1D' signifies fingerprints, '2D' signifies shape, and '3D' signifies docking with Surflex-Dock. Targets in the legend are sorted in order of increasing number of PDB reference templates provided (the legend of Figure 40 shows exactly how many templates were provided per target). Note that while EF1% = 1 is equivalent to random performance, we have annotated EF1% = 2 since that was the minimum enrichment required for inclusion in LIT-PCBA[184].

3.2.3.2 Pose Prediction Performance Next we examine the CNN ensemble's pose prediction performance on the templates provided with LIT-PCBA. When more than one template was provided, we redocked and cross-docked each crystal ligand into each available receptor structure and used each scoring function to rank all available poses. The CNN models were used to rescore Vina-generated poses, and all these were compared with Vinardo, which was derived from Vina but intended to improve its pose prediction performance. Such an improvement did not manifest on this benchmark, as shown by the average fraction of compounds with a "good" (≤ 2.5Å RMSD) pose sampled at ranks 1, 3, and 5 in Figure 39a and the actual fraction of compounds with a good pose sampled per-target in Figure 40. The CNN models improve on Vina's pose ranking, whether using the output from the pose or affinity layer.



(a)



(b)

Figure 39: (a) Percent of a target's compounds with a good pose at ranks 1, 3, and 5, averaged across all fifteen targets in LIT-PCBA. Labeled horizontal lines show the best performance possible with the poses sampled by Vina (red) and Vinardo (chartreuse). (b) Correlation between normalized EF1% for the virtual screening task and fraction of compounds with a good pose ranked first in the pose prediction task.

There may be a slight positive correlation between virtual screening and pose prediction performance on LIT-PCBA for the CNN pose models and Vina, as shown in Figure 39b. The CNN affinity models have Spearman ρ values close to 0, though the Dense affinity model data may suggest a relationship (NEF is higher when the fraction of low RMSD compounds is higher). In general the evidence in support of a relationship for any model is not particularly strong. Different compounds are used for the virtual screening and pose prediction tasks (since the available data is disjoint with respect to the compounds), so this is not evidence of whether accurate structural interactions are the basis of the relevant CNN virtual screening scores. Finally, Figure 39b uses the normalized EF1% as its virtual screening metric, and demonstrates that despite the high EF1% values for some targets, enrichment is universally much lower than its maximum possible value.

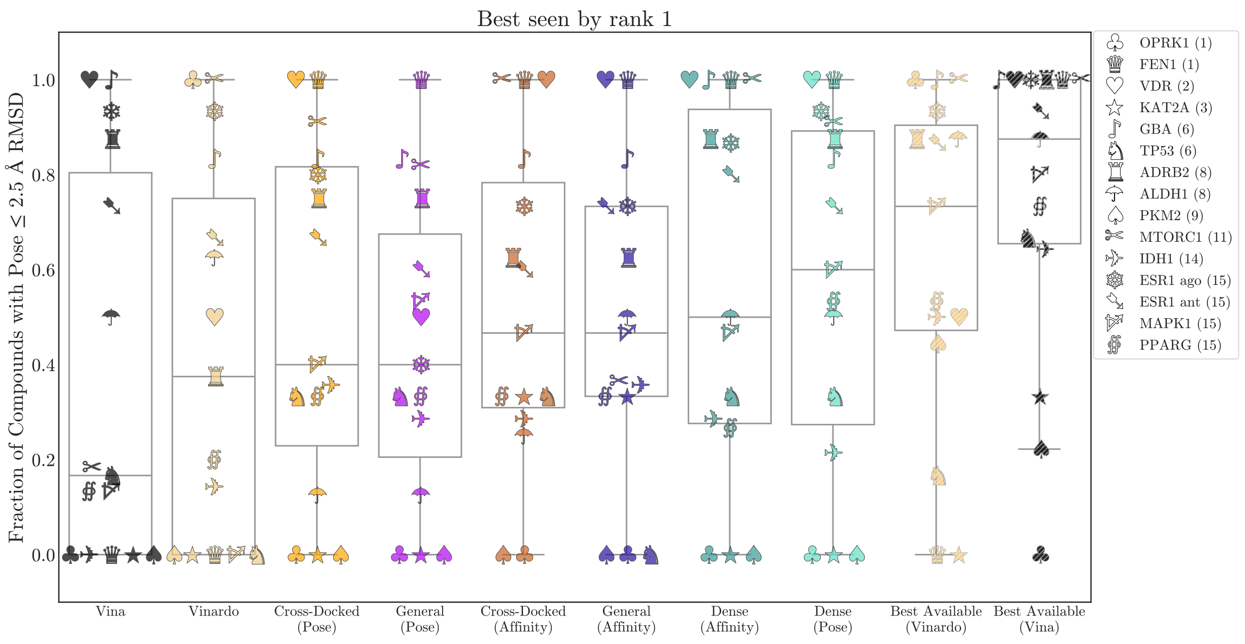


Figure 40: Boxplots showing the fraction of compounds, per-target, that have a pose ≤ 2.5Å RMSD from the provided crystal pose ranked first by each method. The two plots with hatched markers show the fraction of compounds for which such a pose was *sampled*.

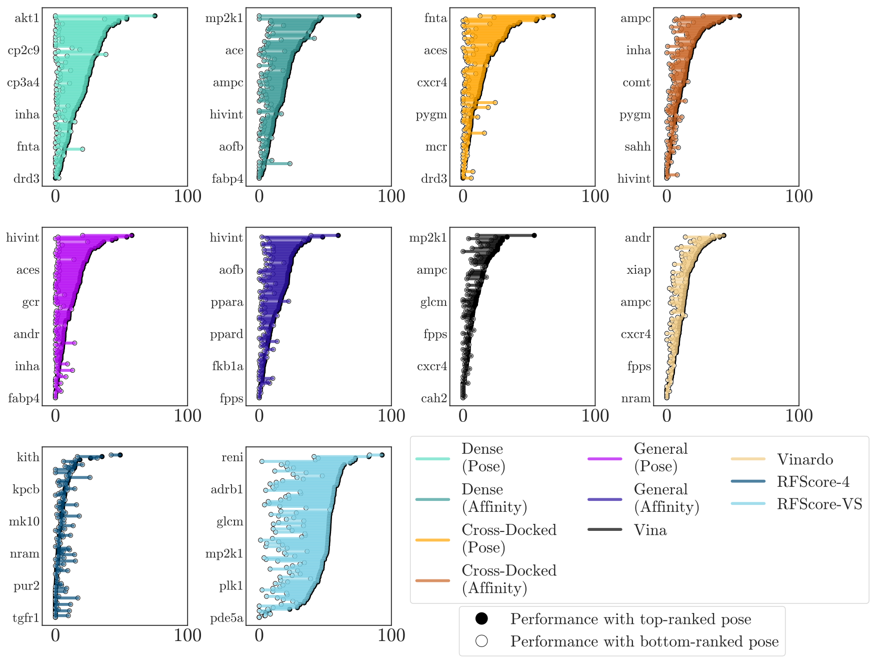
3.2.3.3 Explaining Performance The CNN predictions (particularly the affinity values) appear to be useful for virtual screening. On LIT-PCBA, which was designed to more closely resemble true HTS experiments, they outperform the other methods tested, with the Dense affinity score performing best. Table 6 provides 95% confidence intervals and p-values computed for the difference ($Dense_{Affinity}$ − Non-CNN Model) for each LIT-PCBA target and shows that it is probably statistically better than the other methods on five targets: ALDH1 (better than RFScore-4), ESR1-ant (better than RFScore-4, RFScore-VS, Vinardo, and Vina), FEN1 (better than RFScore-4, Vinardo, and Vina), GBA (better than RFScore-4, RFScore-VS, Vinardo, and Vina), and VDR (better than Vina) though in some cases the effect size is small; conversely, no methods appear to be statistically significantly better than the Dense affinity model for any target in LIT-PCBA. The Dense models were also trained with more data than any other model except the CrossDocked models. In general we would like to explain the mechanisms underlying good and bad performance; we would especially like to examine whether our predictions are pose sensitive, whether trivial descriptors are the primary basis of model performance, and whether performance is predictable based on similarity to training data.

First, we check whether our virtual screening predictions are pose sensitive by comparing EF1% when basing a compound's prediction on its highest- versus its lowest-ranked pose. Figure 41 shows this assessment for DUD-E (a) and LIT-PCBA (b). All methods exhibit some pose sensitivity, with the top-ranked pose generally exhibiting better performance, but there are also many cases where non-random performance is achievable with even the lowest-ranked pose, and every model also has at least one task for which choosing the lowest-ranked pose outperforms the highest-ranked one. This suggests that pose information is being used but (1) it is not always correct and (2) it is not the sole basis of the prediction.

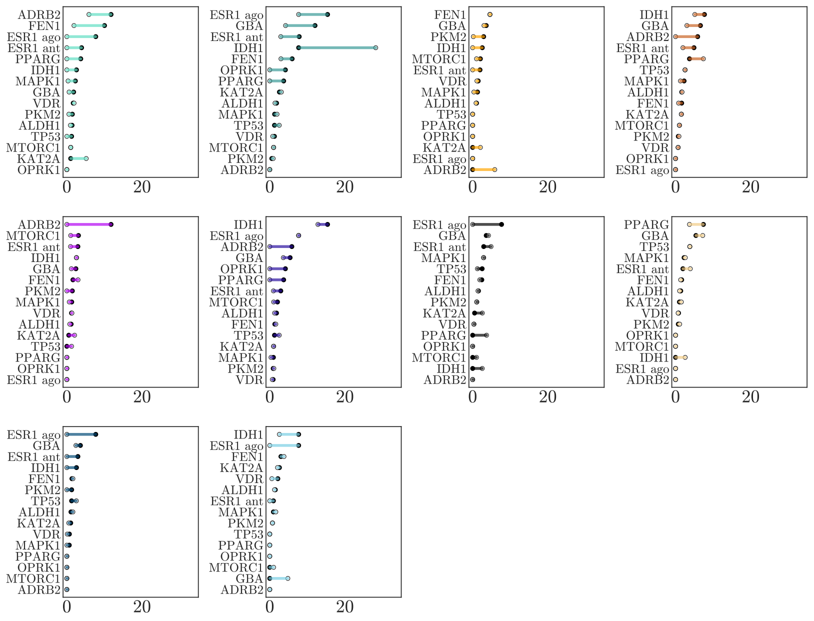
Next we investigated the set of "simple chemical descriptors" that are known to afford perfect performance on DUD-E when used to fit models. Since none of our CNN models were fit to DUD-E (nor indeed to any virtual screening dataset), we might hope to have avoided fitting models that derive their performance from these descriptors. However, these descriptors are useful because of historical bias in the underlying datasets from which most benchmarks are drawn, so it is entirely possible for models fit to other datasets to have a bias with respect to these descriptors. In [57], motivated by this consideration, we assessed similar "Simple Descriptor" models for performance at binding affinity prediction on PDBbind and Pocketome test sets and found them to have better-than-random and, in some cases. competitive to state-of-the-art performance. Therefore it seems necessary to compare other model performance to the baseline established by one or more reasonably trained models that use these simple descriptors as a baseline.

To that end, we establish a performance baseline by fitting a variety of linear and nonlinear regression models (Lasso, K-nearest neighbors, Decision Tree, Random Forest, Gradient Boosted Tree, and Support Vector regressors) available through `sklearn`[140] to the ligand and affinity data associated with each of our training sets, with either the descriptors used in the construction of DUD-E, the descriptors used in the construction of MUV (see Table 5 for both), or ECFP4 fingerprints as implemented in OpenBabel[137] as features. Hyperparameter optimization was performed for all models via cross-validation on the PDBbind-Refined 2016 set. We then evaluate how these "simple descriptor" models perform at virtual screening on each of our test sets.

In Figures 42-44, we take the maximum performance per-target across any of the simple descriptor models and compare it with the "advanced" ML model with the same underlying training data. The lighter line shows the maximum EF1% achieved on a target with the simple descriptor models, while the darker line shows the difference between the "advanced" ML model EF1% and the simple descriptor EF1%. The lines are sorted by most to least improvement, so more weight at the top of the hourglass indicates better performance. Figures 62-64 show actual EF1% values per training dataset, with the simple descriptor models broken down into the best performance achievable for each set of descriptors. Increasing training dataset size yields CNN models with performance gains relative to simple descriptor models on both datasets, but no model achieves better performance than the best available simple descriptor model for a majority of targets in LIT-PCBA.

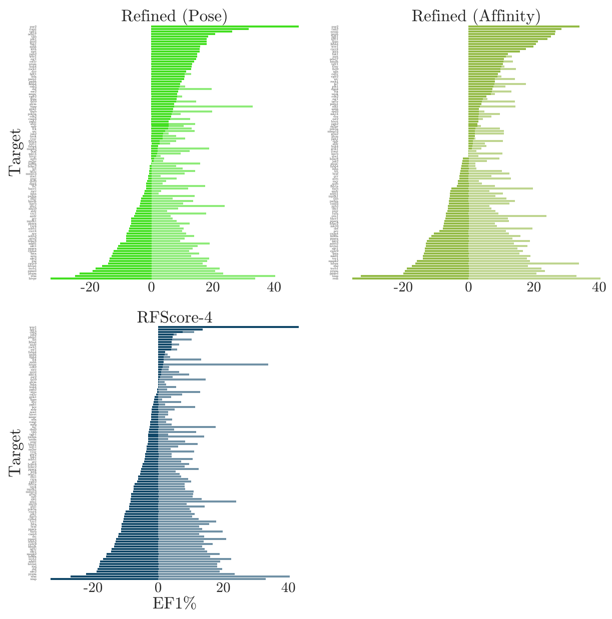


(a)

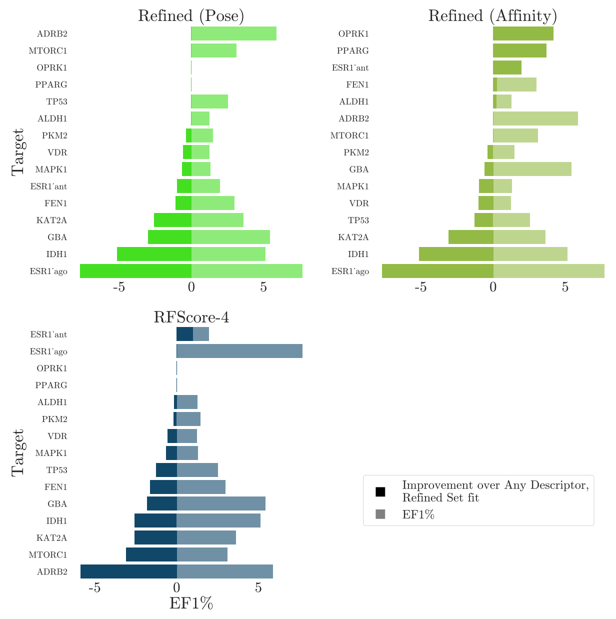


(b)

Figure 41: Change in EF1% per target for the evaluated methods when basing the score on the top-ranked versus the bottom-ranked pose for each example. Results are shown for DUD-E (a) and LIT-PCBA (b).



(a)



(b)

Figure 42: Change in EF1% per target for CNN and RFScore models fit to the PDBbind Refined Set relative to the best performance achievable with a basic model fit to simple descriptors. Results are shown for DUD-E (a) and LIT-PCBA (b). Note that RFScore-4 was fit to PDBBind-Refined version 2014, but our CNN model was fit to PDBbind-Refined version 2016.

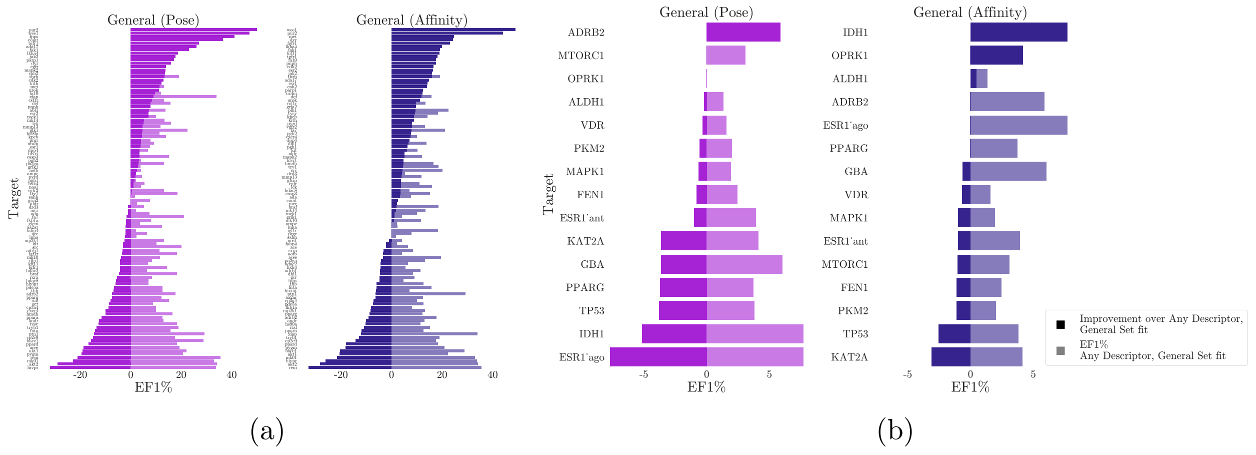


Figure 43: Change in EF1% per target for CNN models fit to the PDBbind General Set relative to the best performance achievable with a basic model fit to simple descriptors. Results are shown for DUD-E (a) and LIT-PCBA (b).

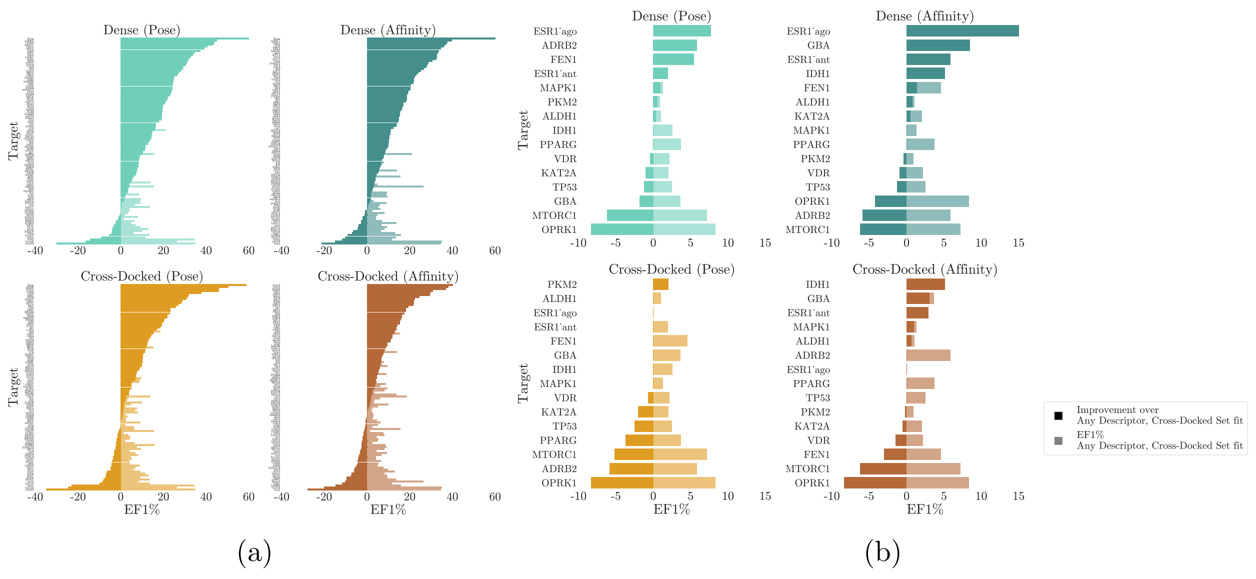


Figure 44: Change in EF1% per target for CNN models fit to the Cross-Docked Set relative to the best performance achievable with a basic model fit to simple descriptors. Results are shown for DUD-E (a) and LIT-PCBA (b).

The CNN models demonstrate improved benchmark performance with more training data, which is a trend observed in the machine learning literature generally but has often *not* been observed in the drug discovery literature, particularly for binding affinity prediction[90, 64]. It has been reported that models fit to the PDBbind-Refined set (∼4000 complexes) do not gain and sometimes lose performance when fit to the PDBbind-General set (∼10,000 complexes). We have observed the performance improvement in our CNN models for the binding affinity prediction task[57], and our best performing models were fit to our curated cross-docked dataset (∼20,000 complexes). We show the performance trends for our CNN models and the simple descriptor models, broken down by descriptor type, in Figure 45. Notably the models that were fit to simple descriptors exhibit no significant gain in performance (and sometimes a loss) as the training dataset size increases, presumably because the information content of the dataset with respect to those descriptors is identical. The models fit to fingerprints exhibit generally worse performance than the models fit to the descriptors that underlie the bias in the dataset. Performing better than a fingerprint model may indicate that a more advanced model has simply learned the bias of the dataset in simple descriptor space and that becomes more likely if increasing training data volume does not yield a more accurate model.

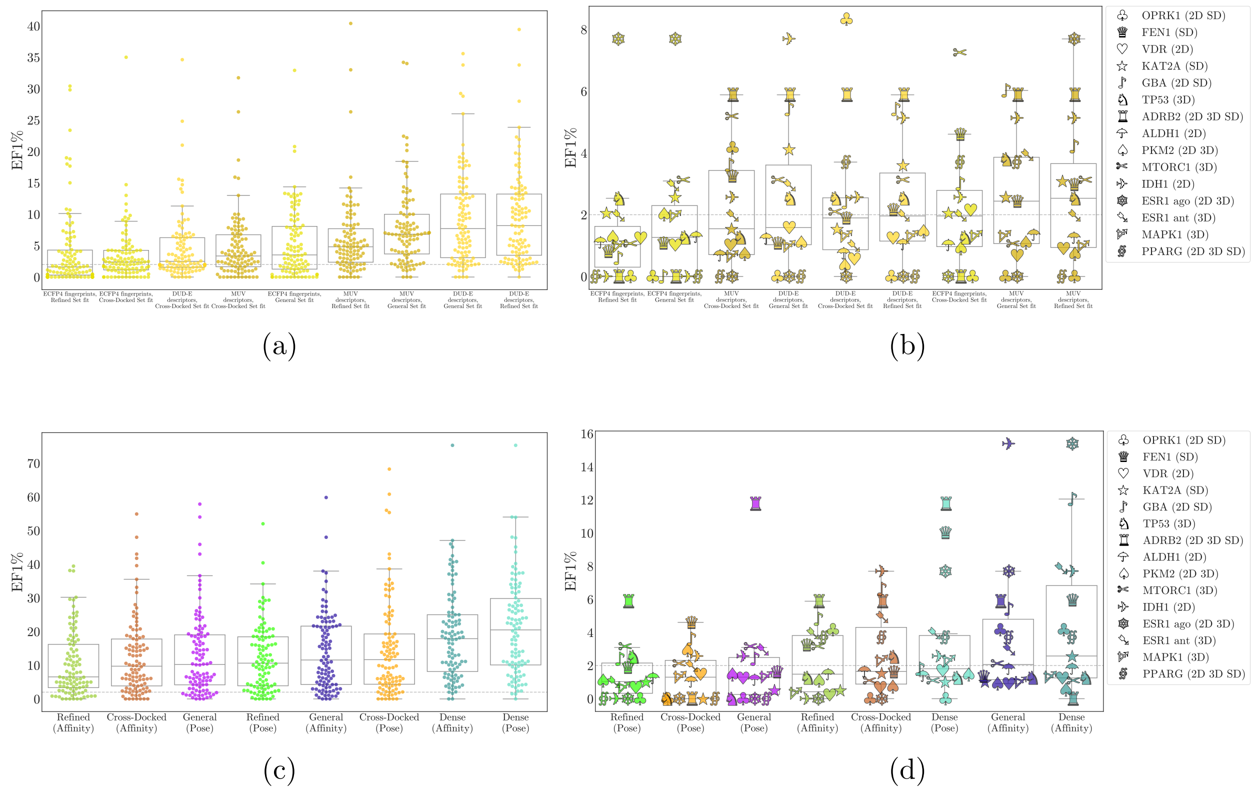
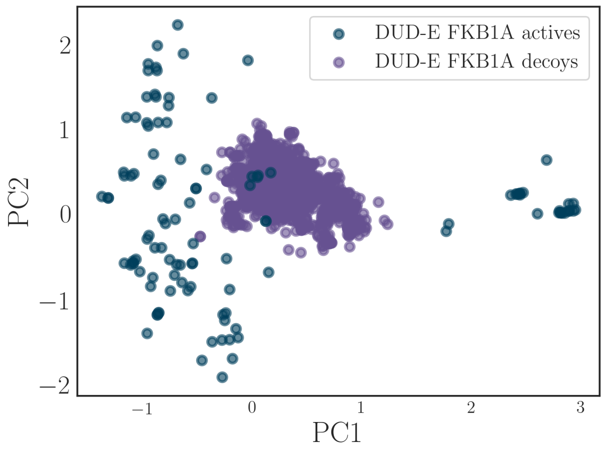
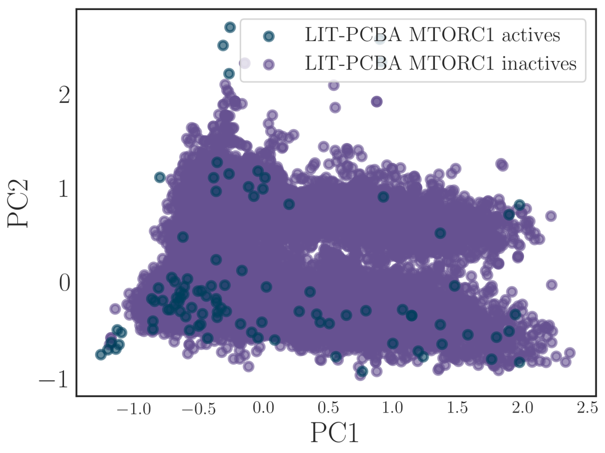


Figure 45: Comparison of the change in model performance for simple descriptor models and CNN models as they are provided with more training data.

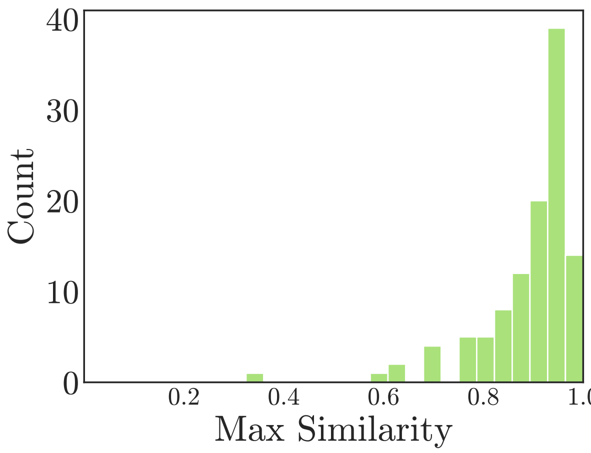
Next we consider similarity between training and test datasets. To motivate our approach, we first contrast the two test sets. LIT-PCBA contains six targets that have greater than 50% sequence similarity to targets in DUD-E (one of those is ESR1, which constitutes two distinct tasks — antagonist vs agonist identification — in LIT-PCBA). Of these, the CNNs have comparable performance on four targets, but there appears to be a large difference between the virtual screening performance on the remaining two: adrb2/ADRB2 and fkb1a/MTORC1 (identifiers are arranged DUD-E/LIT-PCBA). The explanation for the performance discrepancy must lie with the screening compounds. Indeed, as shown in Figures 46-47, the DUD-E tasks exhibit greater separation between actives and inactives as well as higher similarity between the available actives. Notably the LIT-PCBA authors were concerned that MTORC1 contained actives that were too similar to one another, which was anomalous when compared with the other targets included in the benchmark.



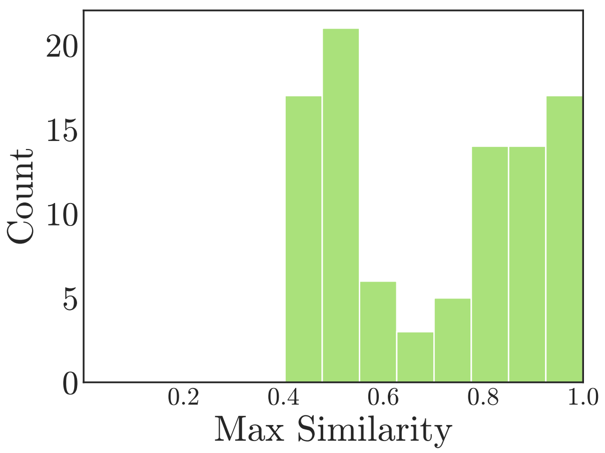
(a)



(b)

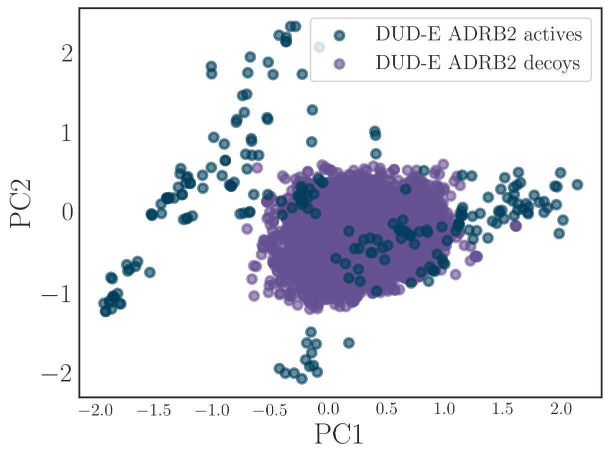


(c)

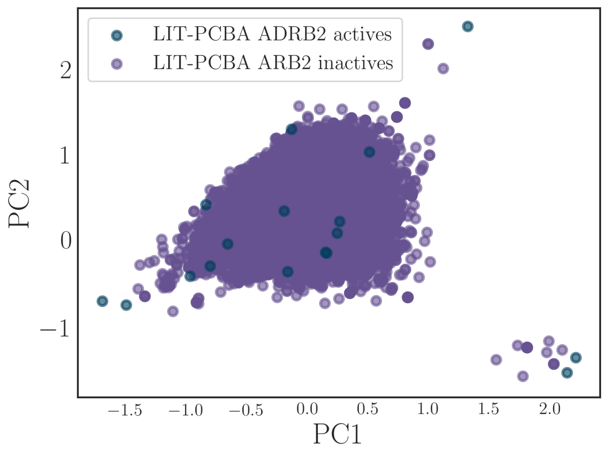


(d)

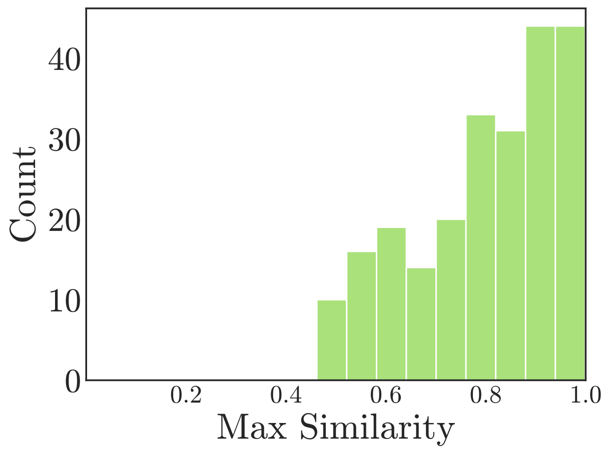
Figure 46: Comparison between DUD-E target fkb1a and LIT-PCBA target MTORC1, showing the separation between actives and inactives after projecting onto the first two principal components from ECFP4 fingerprints derived from the actives for DUD-E (a) and LIT-PCBA (b) and the distribution of maximum Tanimoto similarity between the provided actives for DUD-E (c) and LIT-PCBA (d).



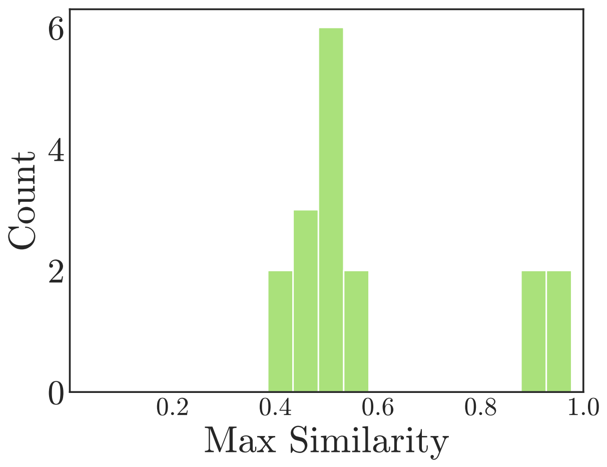
(a)



(b)



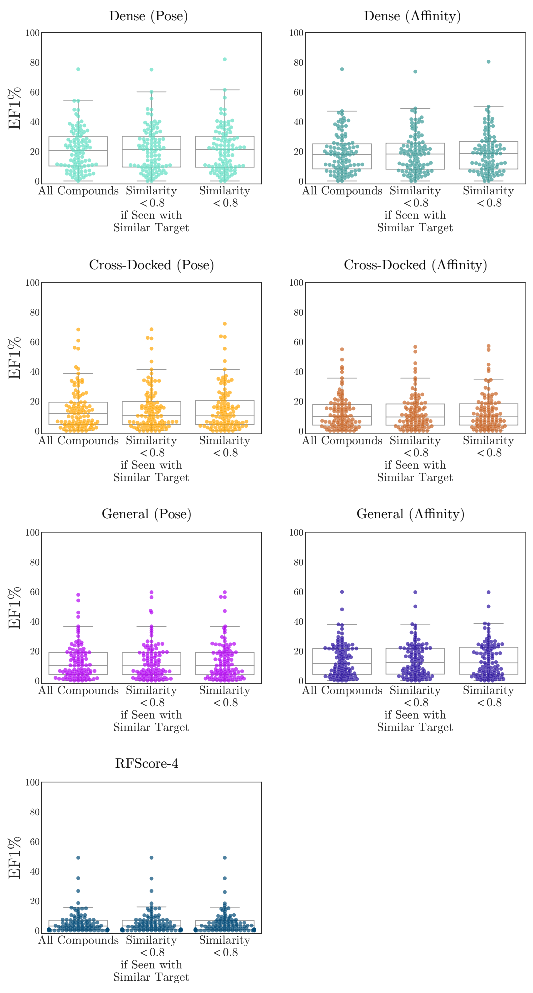
(c)



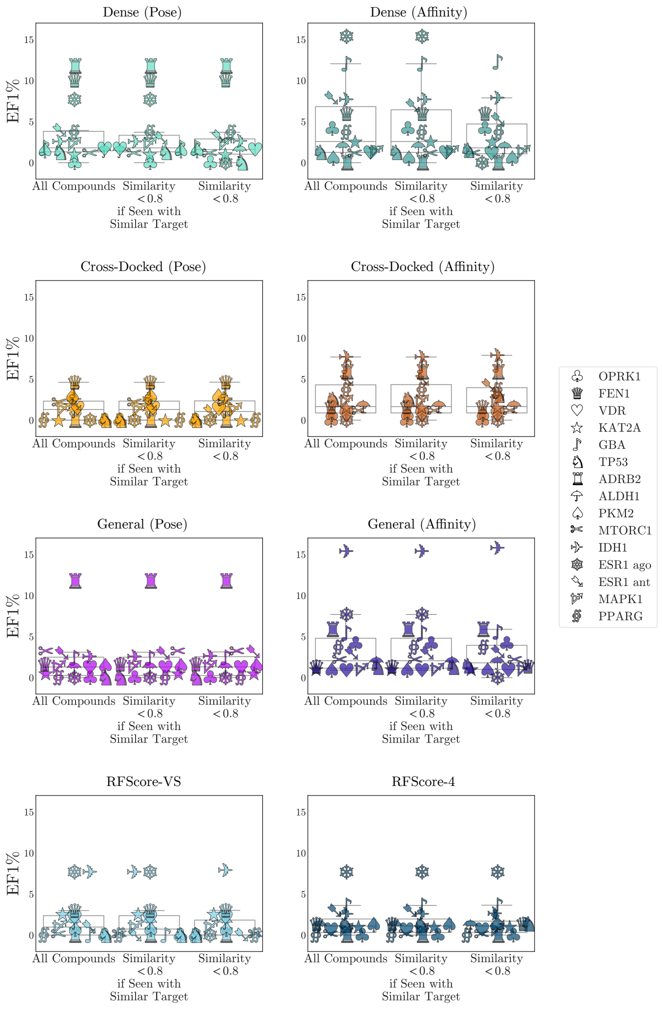
(d)

Figure 47: Comparison between DUD-E target adrb2 and LIT-PCBA target ADRB2, showing the separation between actives and inactives after projecting onto the first two principal components from ECFP4 fingerprints derived from the actives for DUD-E (a) and LIT-PCBA (b) and the distribution of maximum Tanimoto similarity between the provided actives for DUD-E (c) and LIT-PCBA (d).

Motivated by this evidence of compound similarity as a possible explanation for differences in performance, we examine how much of our performance derives from identifying active compounds that were seen during training. To do this, we compare EF1% (Figure 48) and the number of actives in the top 100 compounds (Figure 65) with all actives present versus after dropping actives based on their similarity to compounds in the training set. We use a maximum Tanimoto similarity threshold of 0.8, and apply it in two ways: to compounds that were observed during training in complex with a protein with at least 50% sequence identity to the one with which the compound was observed at test time; and over all compounds, agnostic to the protein with which they were in complex during training. We make a distinction here because if an example was observed with a similar protein, it is possible that relevant features of the interaction were learned, making their subsequent identification more promising from a generalization standpoint than if the compound was seen with a very different protein during training - in the latter case it is less likely that interactions relevant to binding were identified, and more likely that the compound itself was merely memorized. Both statistics suggest that training set compound similarity is not an explanatory factor related to performance except for the two estrogen receptor tasks in LIT-PCBA, especially the agonist, and the compounds that were identified that bore similarity to the training set were not observed with a similar protein.



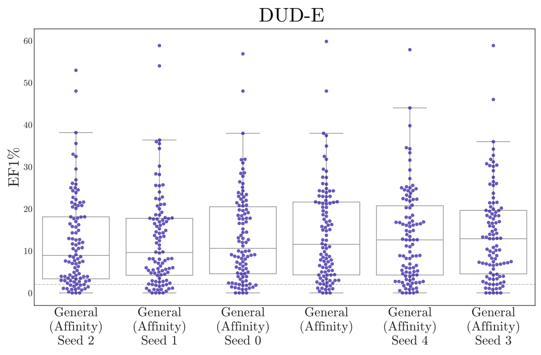
(a)



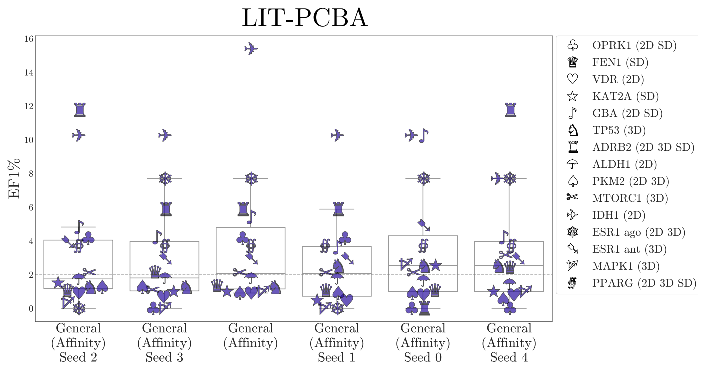
(b)

Figure 48: Comparison of EF1% for DUD-E (a) and LIT-PCBA (b) with all compounds included versus removing compounds based on their similarity to compounds seen in the training set. The threshold for compound similarity was a Tanimoto coefficient of 0.8. We applied that threshold to compounds that were provided as training examples as part of a complex with a protein that had at least 50% sequence identity with the test set target being evaluated, as well as across all compounds seen during training.

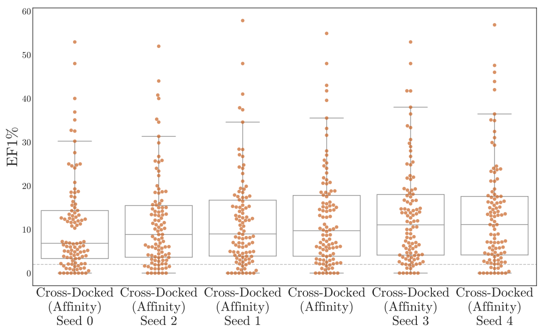
3.2.3.4 CNN Model Ensembles Finally, we assess benefits afforded by the ensemble of models fit for each combination of training dataset and CNN model architectures. Ensembles of models are useful for stabilizing predictions and providing a means to estimate uncertainty. First we note that the ensemble may not have better performance than individuals within the ensemble have on a given benchmark, but the best performing replicate will generally depend on the benchmark itself. A goal of the ensemble is to have more *consistent* performance across benchmarks, reducing the risk of catastrophic failure due to replicate bias. Furthermore, selecting replicates to include based on a dataset is a form of fitting to that dataset, so it should be treated as carefully as any other way fitting is performed (i.e. not something done on a test dataset with unblinded performance). See Figure 49 for a comparison of the ensemble and individual replicate performance for each CNN model considered on the two test datasets. In general different replicates achieve the best overall performance for the two datasets (see e.g. the General Set model, for which seed 3 achieved the best performance on DUD-E but second worst performance on LIT-PCBA; and the Cross-Docked model, for which seed 4 achieved the best performance on DUD-E but the worst performance on LIT-PCBA), but by averaging over the ensemble we achieve performance that is more stable (maintaining nearly the same rank for different datasets), that is never the *worst* compared with the individual replicates, and in the case of the best performing model (the Dense Affinity model), the ensemble mean perfoms better than any individual replicate.



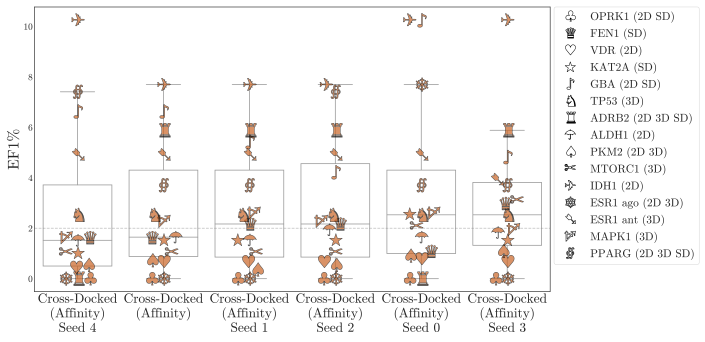
(a)



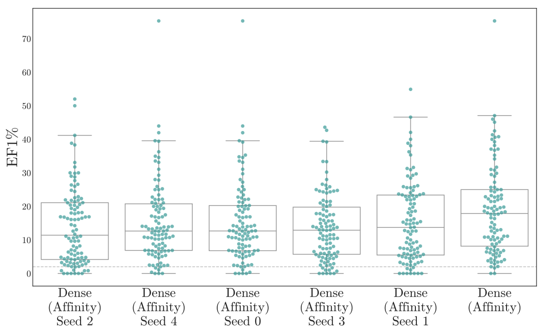
(b)



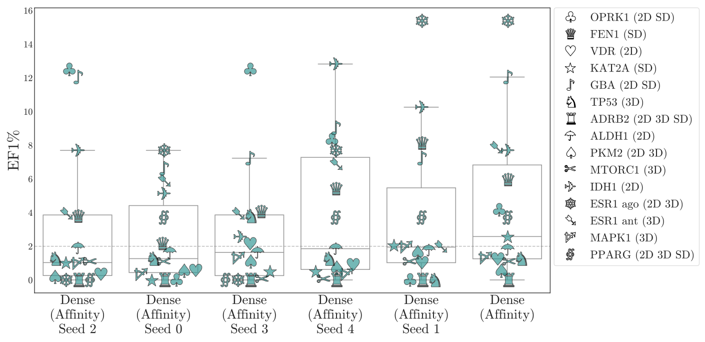
(c)



(d)



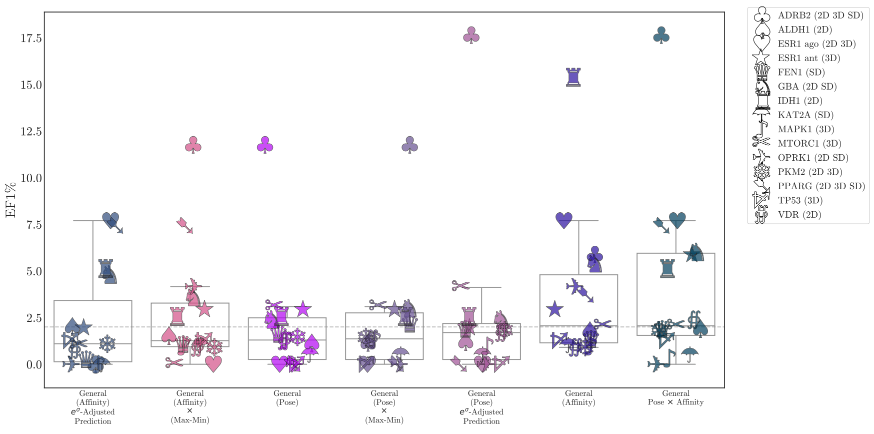
(e)



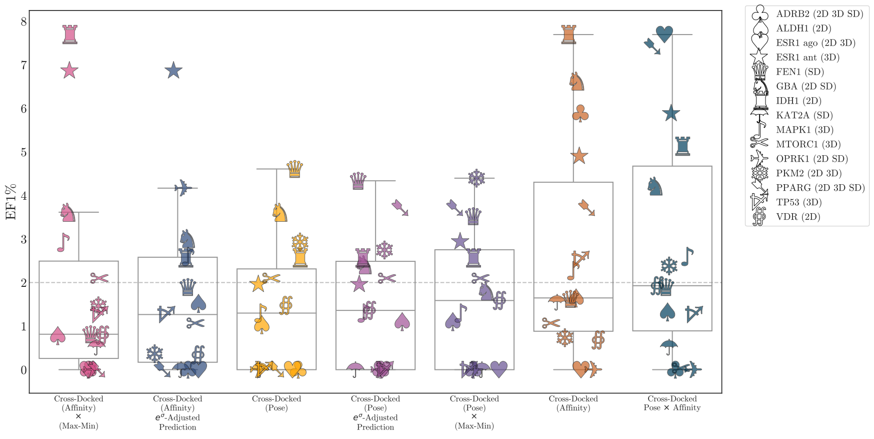
(f)

Figure 49: Performance of ensemble mean and individual replicates within the ensemble for the CNN for DUD-E and LIT-PCBA

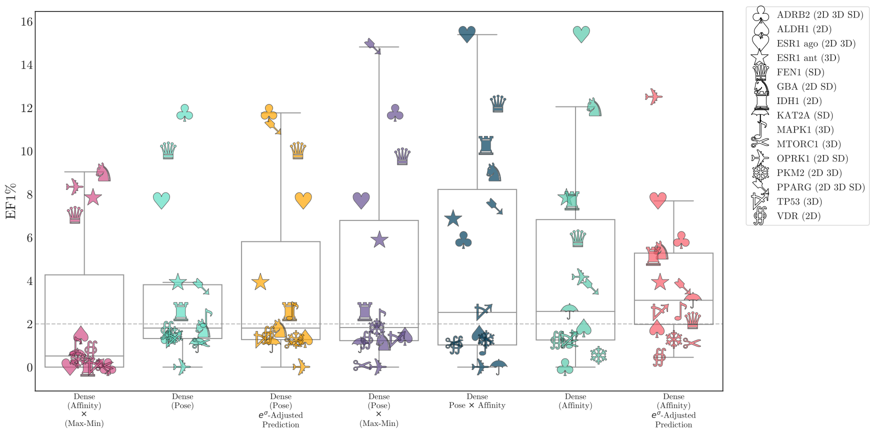
Next we consider the ensemble variance and the role of pose information when making a prediction. We have not yet performed the calibration necessary to convert the ensemble variance to a confidence interval via conformal prediction[133, 37], but we can still attempt to use the variance to prioritize predictions that are more precise and assess whether it is beneficial. We do this by dividing ensemble predictions by e^{σ}, such that low variance predictions are basically unaffected while high variance predictions are reduced. Additionally, we see evidence of predictions being based on a combination of pose and non-pose information, and it is more likely that the non-pose information derives from bias; we would therefore like to test whether performance improves if its influence is reduced. We test two ways of doing this. One is within-model, by inferring that the gap in score between the maximum and minimum pose for a compound is evidence of whether a good pose was indeed found; most poses should not be close to a binding mode, so the bottom-ranked poses for a compound should not (if the method is pose-sensitive) accurately capture the protein-ligand interactions associated with binding. If there is no gap in score between the top and bottom pose, that likely means no good pose was found, or the score is derived from information that is not related to the pose. Therefore we use the gap between the maximum and minimum score to recalibrate the prediction. The other method we use is to combine the pose and affinity outputs to produce a single prediction, with the expectation that the pose output may be more pose sensitive and combining it with the affinity output will increase the overall pose sensitivity (of course, it may also compound the effect of any biases they share). This was partially motivated by Figure 67, where the CNN pose models show strong separation between the classes in DUD-E. This does not appear to hold for LIT-PCBA, however (Figure 68). The results for LIT-PCBA are shown in Figure 50 and for DUD-E in Figure 66. Calibrating the scores to improve performance appears promising, but additional understanding of why (and therefore when) they work is necessary before recommending them for use (for example, they could work by increasing the influence of features that undergird dataset bias, in which case they might be useful for model explanation but not for use in virtual screening).



(a)



(b)



(c)

Figure 50: Effect of various score calibration approaches on LIT-PCBA EF1%.

3.2.4 Conclusion

Dataset bias is a serious obstacle to applying data-driven approaches to solve problems in drug discovery. Unless care has been taken to assess the bias of a dataset and un-bias accordingly, machine learning models fit to that dataset will learn that bias. Since many of the existing biases are historical, it is entirely possible to subsequently evaluate performance on a test dataset that shares similar biases and inaccurately report improvements in generalization when in fact the resulting model is worse at generalizing than the conventional scoring functions that predate it. We are still developing appropriate datasets and evaluation methods to ensure that we can effectively leverage data without fitting to the artifactual patterns it contains.

Here we have demonstrated that machine learning models fit for binding affinity prediction and pose selection can be used for virtual screening. We have also demonstrated that models fit using simple chemical descriptors on available binding affinity datasets can achieve results comparable to “advanced” machine learning models fit to the same dataset. These simple descriptor models do not benefit from increases in training set size over a small threshold, and they outperform the topological fingerprint models commonly used as a baseline for advanced model comparison in the literature.

We demonstrate that our CNN models exhibit pose sensitivity and our DenseNet-based models in particular achieve performance that is better than other non-CNN models with which they were compared on the LIT-PCBA benchmark. This benchmark may be better suited to model evaluation than preexisting virtual screening benchmarks (e.g. DUD-E), although we did not benefit from the AVE unbiasing they used to construct training and validation splits because we used it as an independent test set rather than for training. Improvements in the performance of these models as training dataset size increases is a promising indicator that their performance does not derive exclusively from simple descriptors and that they will continue to improve over time as more data becomes available.

A significant direction that was not explored in this work was the use of the trained CNNs for pose sampling as well as scoring. This was used in [178] and it appeared to improve CNN virtual screening performance. Evaluating different models using the LIT-PCBA training and validation splits should also be performed, as well as fine-tuning existing models on LIT-PCBA. Identifying the problems with existing virtual screening benchmarks will enable researchers to improve on them and ensure that models are extracting features that enhance generalization performance. A recent study of 14 machine learning scoring functions for virtual screening found that none of them outperformed classical scoring functions, except for RFScore-VS, which only outperformed classical scoring functions on DUD-E (the dataset to which it was fit)[162]. This is a sobering result, but we believe it derives from problems with training data rather than with the data-driven approach itself.

4.0 CNN Input Optimization for Screening and Model Explanation

The work in this final chapter is still preliminary. The basic approach is implemented, but will require more effort to obtain good results. We will describe what is currently implemented and the assessments we have performed so far, with directions for future work.

4.1 Background

Once a neural network is trained, its weights parametrize a mapping from a set of input features to a prediction. If its predictions are accurate, then it has learned informative predictors from the input domain that correspond to class-defining features (when performing classification) or positive or negative contributions to a real-valued prediction (when performing regression). In the case of convolutional neural networks, these patterns are encoded in the weights of its convolutional filters. This information stored in the network can be used to understand what a network has learned and explain its predictions, and it can also be used to optimize input examples[138].

Model explanation is an important problem when working with neural networks, especially when overt human-directed featurization is limited. With the networks given free rein to identify arbitrary patterns from the data, they may identify features beyond what we know to be useful for a chosen problem - a powerful trait, with the potential for disappointing results if the networks regurgitate the biases we ourselves have introduced into the data rather than inferring scientific laws or meaningful relationships between discrete input regions. As discussed in the last chapter, machine learning for molecular modelling can sometimes make accurate predictions on preliminary assessments, even significantly outperforming conventional approaches, only to significantly underperform later because of exactly this phenomenon. Thus we are motivated to explain what a network has learned in general, as well as the justification for particular predictions, to reduce the risk of deploying models that don't generalize to real-world data and exhibit poor prospective predictive accuracy.

There are multiple widely-adopted options for visualizing what a network has learned, several of which were explored by members of our group in the past[72]. Those we have previously explored include direct filter visualization, which is usually only readily interpretable for the first layer; gradient visualization[166], where loss gradients are backpropagated onto the input and they or their magnitude are visualized; layer-wise relevance propagation[9, 106], where the network's output is propagated back onto the input, attempting to explain each part of the input's contribution to the final prediction; and iterative masking of the input[180], which involves repeatedly removing parts of the input and then rescoring it, attributing the score difference to the part of the input that was removed. Other options include partitioning a corpus of input data based on their ability to induce a large activation of a network's internal filters, taking the (usually limited) attributes shared by a collection of examples that maximally activate a particular filter as evidence of the feature that filter has learned to detect[206]; or using deconvolutions to highlight which input regions contributed to a filter's activation[208]. Finally, there is the option we have begun investigating here: using one or more filters to *synthesize* an input that maximally activates them[206, 138, 119, 129, 131, 199, 54, 179].

An approach based on synthesizing inputs via optimization has several advantages. It can directly show us patterns that maximally activate a filter, rather than merely identifying examples or patterns that *correlate* with activation. It can be performed with respect to various parts of the network - with respect to a single filter, or jointly for multiple filters, an entire channel or layer, or even the full network (by starting at the final layer before the output layer). Regarding the network filters as basis vectors defining the network's "activation space", we can consider the results of optimizing along the various directions in activation space defined by combinations of filters; we can also define an optimization objective that interpolates between two filters, similar to latent space interpolation used for generative approaches[138]. We can also start from a real example and allow the network to optimize it, perhaps conditioning on part of the input, which will be kept constant. For example, we might start from a protein binding site and use one or more of the output layers to optimize the ligand density while keeping the protein density constant. The resulting ligand density is likely to be nonphysical, but it represents locations, relative to the binding site, where the network associates a higher predicted score with the presence of specific ligand atom types. This is quite similar to a pharmacophore, representing molecular features that characterize a strong interaction between a ligand and the protein of interest, thus I will call the resulting optimized ligand density a "pseudo-pharmacophore". This project therefore has a dual purpose of implementing a new approach for explaining what our networks have learned during training, as well as a means of generating pseudo-pharmacophores that could in theory be used for tasks like virtual screening.

4.2 Methods

The basic approach here is quite simple, with two complications: the literature suggests that, at least for images, uninformative high frequency information dominates the optimized input unless appropriate priors are used[138, 119, 130, 129]; and to use optimized grids (the result of optimizing ligand density grids with respect to a trained network) for virtual screening, we need to consider how best to compute distances between grids.

4.2.1 Approach

For node visualization and other model explanation tactics, we start from a randomly initialized input. For virtual screening, the receptor channels are initialized from the target and the ligand channels are empty; only the ligand channels are updated later. We do a forward pass from the input x to neuron i to compute the necessary activation $a_i(x)$, then do a backward pass onto the input to obtain $\frac{\partial a_i(x)}{\partial x}$. Then we apply the standard gradient ascent update

$$x = x + \alpha \cdot \frac{\partial a_i(x)}{\partial x} \tag{4.1}$$

repeatedly until convergence. We test performing optimization with respect to the network as a whole by computing the loss as a function of the initial output and the target class (active) and/or affinity (an arbitrarily large $pK \in \mathbb{R}$)) and performing gradient descent; however, at least for the softmax output, the literature suggests[138, 181] that optimizing the input through the softmax is problematic, possibly because it operates by reducing the likelihood of the alternative class rather than maximizing the likelihood of the target class. Therefore we also evaluate the alternative approach of maximizing the pre-softmax logits for the classification output.

Unfortunately, employing this approach without any modification generates adversarial examples[118], distinguished by the prominence of high frequency patterns that in the best case have semantic meaning for humans as texture[62]. Typically they have no distinguishable visual differences from the starting example to a human observer and are considered a classic failure mode and therefore attack vector for neural networks[3, 21]. One means of establishing resistance to these attacks is by generating them as counterexamples during training, e.g. via projected gradient descent[118, 67, 160, 80], which performs an initial optimization step to generate maximally misclassified examples within some constraint set such as the L_∞ ball around the starting example. This is followed by standard training, where these examples are provided as input to the network, paired with the labels associated with the original example. We explored adversarial training as well, but found it provided little benefit for virtual screening performance. Instead, priors must be included to regularize optimization to generate interpretable visualizations.

4.2.2 Appropriate priors

The primary aim of the imposed priors is to reduce high frequency signals that would otherwise characterize the optimized input, leading to rapid convergence to examples of little explanatory or practical value. The image classification literature often describes them as "imagelike" or "natural image" priors since they are often motivated based on the goal of achieving generated images with spatial statistics that resemble real-world images. There are three general classes of these priors[138]: those based on penalizing high frequency signals (by enforcing correlation of neighboring pixels, applying Gaussian or bilateral filters, or using gradient preconditioning); those based on enforcing robustness to small, local transformations like scaling, jitter, and rotations; and those based on learning a prior via a GAN, autoencoder, or Gaussian Mixture Model. The first two are based on heuristics about what the inputs should look like, while the third type learns a model of the data and attempts to enforce that. We only employ the first type in the current work; while implementing them during optimization is in progress, we currently only evaluate using thresholding on the final grids.

4.2.3 Distance metrics for screening

Since the current study was performed as a proof-of-concept, we only consider the case of using optimized ligand grids generated from a binding site to screen poses of screening compounds docked into the same binding site. This avoids the problem of needing to align or orient the grids to maximize their overlap, which would need to be addressed in the more general case of conformer screening. To rank docked ligands, we score them based on their similarity to the network-optimized pseudo-pharmacophore; we have implemented five methods for computing similarity. These included the standard L1 and L2 distances; the Hadamard (elementwise) product of the pair of grids; a method that thresholded low magnitude densities to 0 and then provided a constant reward when the grids exhibited concordance regarding the location of positive density and a constant negative penalty when the screening grid had density in a location where the optimized grid had negative density; and the Greenkhorn distance, an approximate earth mover's distance. For the Greenkhorn distance, "color" histograms (where the color channels correspond to distinct atom types as usual) were computed as signatures that were provided as input to the transport calculation. Only the first four methods are evaluated in the current results. All methods have parallel CPU (via OpenMP[38]) and GPU (via CUDA[135]) implementations.

Both input optimization and adversarial training were implemented as custom solvers in `Caffe` and are distributed as part of `gnina` at `www.github.com/gnina/gnina`, with combined input optimization and virtual screening functionality provided in `gninadream`, a standalone program.

4.3 Results

Activation maximization for the purpose of examining what a network has learned and optimization conditioned on an input binding site to generate pseudo-pharmacophores for virtual screening are described below.

4.3.1 Activation Maximization

The result of optimizing randomly initialized input with respect to layers of each of our three latest CNN models (Dense, Cross-Docked, and General, described in Chapter 3.2) offers insight into the basis of the differences in their performance when applied to essential drug discovery tasks. For illustrative purposes, visualization of inputs optimized with respect to the affinity output layer is shown in Figure 51 and with respect to select filters in the final convolutional layer is shown in Figure 52.

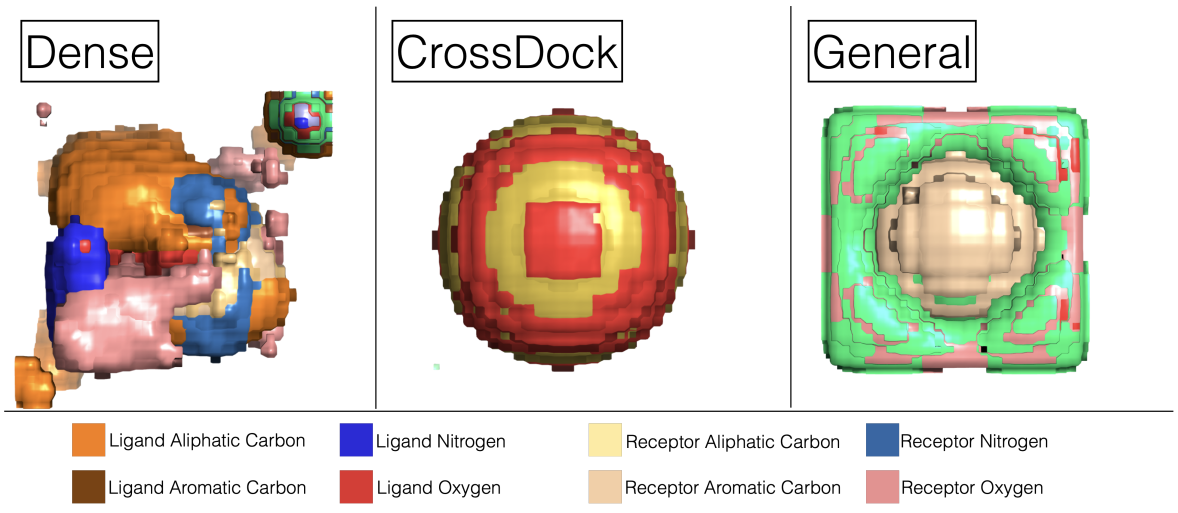


Figure 51: Visualization of inputs that maximally activate the affinity output of our latest models. These inputs were generated from randomly initialized density by performing gradient ascent with respect to the affinity output layer for each of our latest trained CNN models.

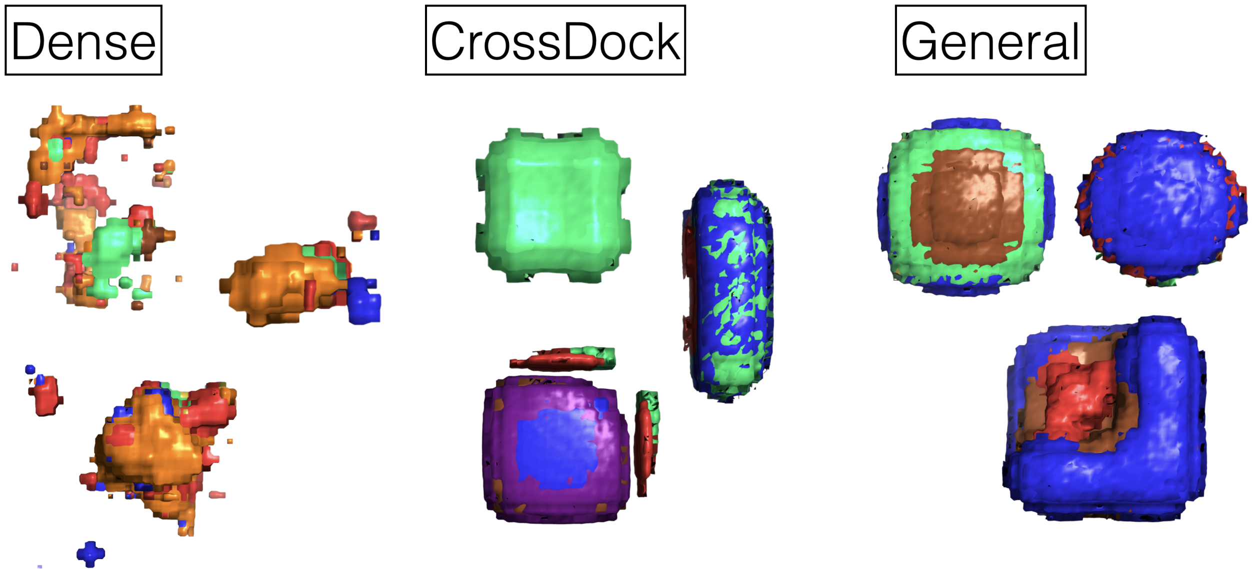


Figure 52: Visualization of inputs that maximally activate selected filters from the final convolutional layer of our latest models. These inputs were generated from randomly initialized density by performing gradient ascent with respect to the center of the feature map for a selected filter.

Broadly, both the affinity output- and final convolutional filter-optimized inputs suggest that more sophisticated features have been learned by the Dense model compared with the other two. The Cross-Docked and General models seem to have learned to broadly detect the presence of specific atom types, to provide substantial rewards in proportion to amounts of input density, they exhibit an obvious spatial relationship in which ligand density primarily appears in the center of the grid and receptor density on the exterior, and they also exhibit radial symmetry of the final features. In comparison, the Dense nets seem to have learned more "natural" representations of optimal receptor-ligand relationships, with the final convolutional filter depicting more specific input features with multiple atom types in close proximity. To draw stronger conclusions about these features, more analysis will be required.

4.3.1.1 Virtual screening First we evaluate the efficacy of the grid distance functions for virtual screening by using the DUD-E reference crystal ligand for each target as the screening grid and measuring the grid distance between that and the benchmark compounds to rank them for screening. The result is shown in Figure 53. It appears that the various distances, especially the L1 distance (commonly used for comparing images) are reasonably effective at ranking the benchmark compounds using the crystal ligand.

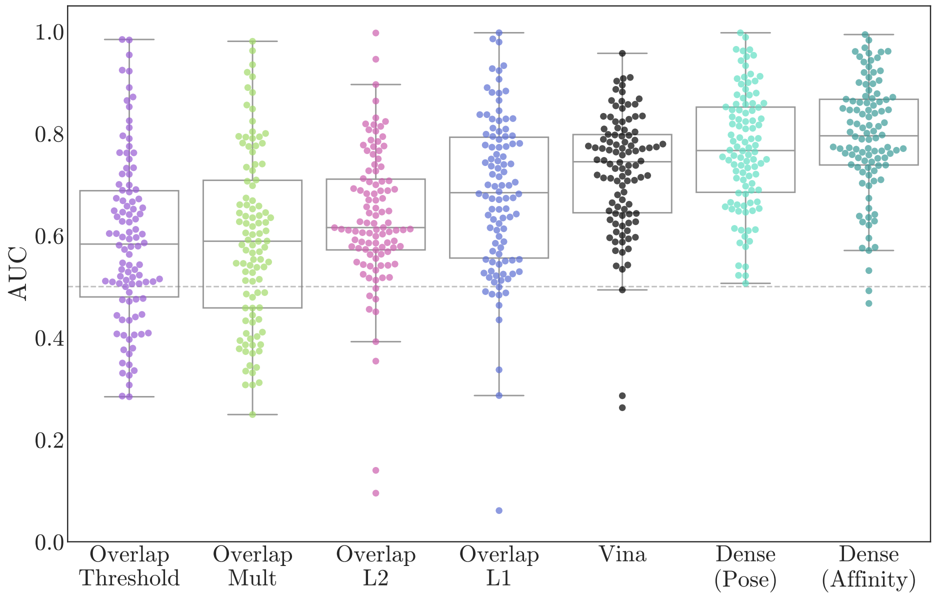


Figure 53: Evaluating distance function performance based on using a provided crystal ligand to screen for compounds, compared with the Dense CNN score and affinity models as well as Vina as baselines.

In Figure 54 we look at using a generated pseudo-pharmacophore to screen, compared with a centroid baseline and rescoring docked poses with the same model used to generate the density. The Cross-Docked model and threshold distance generally have the best performance for screening with the model-optimized input densities, so they are shown here. Note that generating the pseudo-pharmacophore and screening with it via a grid distance function is around 10 times faster than rescoring compound poses with the same model when there are a large (>1000) number of conformers to be screened. Unfortunately there is a substantial loss of accuracy, though the method still performs better than merely screening based on centroid distances between the crystal ligand and docked poses of the screening ligands.

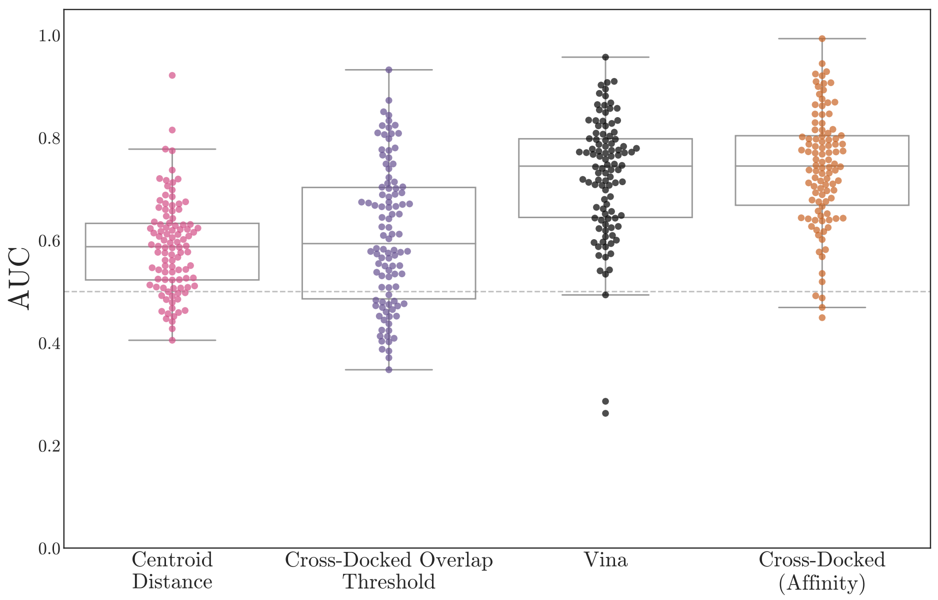


Figure 54: Virtual screening performance comparison for optimized input versus rescoring poses with the same model, using the centroid distance between test compounds and the crystal ligand and Vina as baseline methods. This shows the boxplots of AUCs per-target for DUD-E

Figure 55 shows the correlation between the AUCs per DUD-E target for pseudo-pharmacophore screening for each of our latest CNN models and centroid screening. The Dense model has no correlation in performance with centroid screening, while the other two models have weak correlation. We can see for all three models that the pseudo-pharmacophore screening has many targets for which good performance is achieved when centroid screening is basically random; this suggests that more information is contained in the pseudo-pharmacophores than just the location of the center of the binding site.

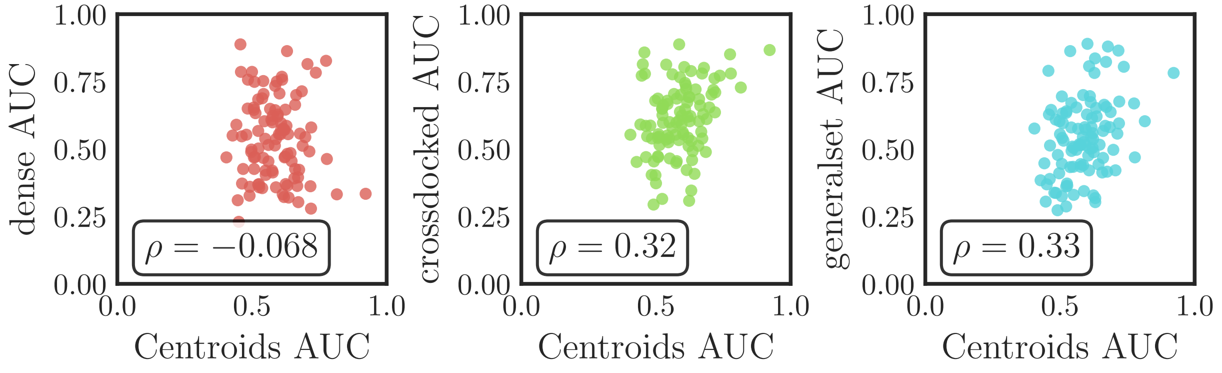


Figure 55: Correlation between performing naïve unregularized input optimization and screening based on centroid distance alone for DUD-E, based on AUCs computed per-target

Figure 56 shows the change in AUC per-target for using the pseudo-pharmacophores generated by a given CNN model versus rescoring poses with the model directly, as well as the actual AUC for ranking with the pseudo-pharmacophore. For all models, pseudo-pharmacophore screening is worse than rescoring for most but not all DUD-E targets, suggesting that the naïve approach is not a good substitute for rescoring with the underlying model, despite its modest performance speedup. This approach is not universally worse, however, and could be improved by incorporating methods discussed earlier such as the "natural example" priors.

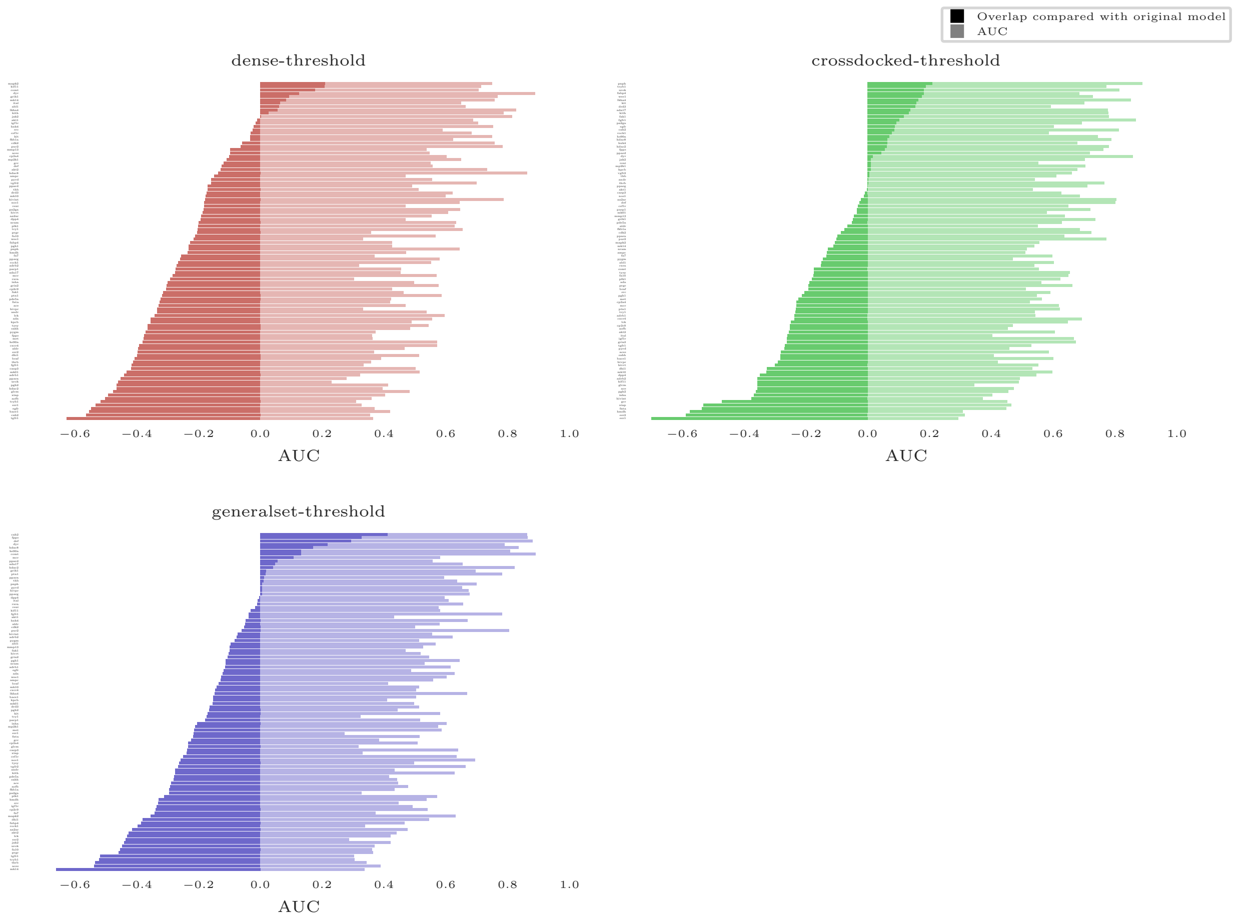


Figure 56: The difference in AUC between generating optimized input and screening with the "threshold" distance for a given model versus rescoring poses with the model directly, as well as the actual AUC for screening with the overlap method.

An adversarial solver was implemented in Caffe to support training an adversarially-resistant model via Projected Gradient Descent. The original goal was to prevent the model from converging quickly during optimization in cases where very little density had been generated, based on the hope that adversarially resistant models would generate more density overall. Screening performance for DUD-E did not improve with the adversarially trained models, and the overall amount of density in pseudo-pharmacophores did not increase. However, the density may be more localized to the binding site rather than the optimization box boundaries when optimizing with an adversarially trained model (Figure 57), and it could be worth exploring performance differences between models trained with this type of counterexample during training.

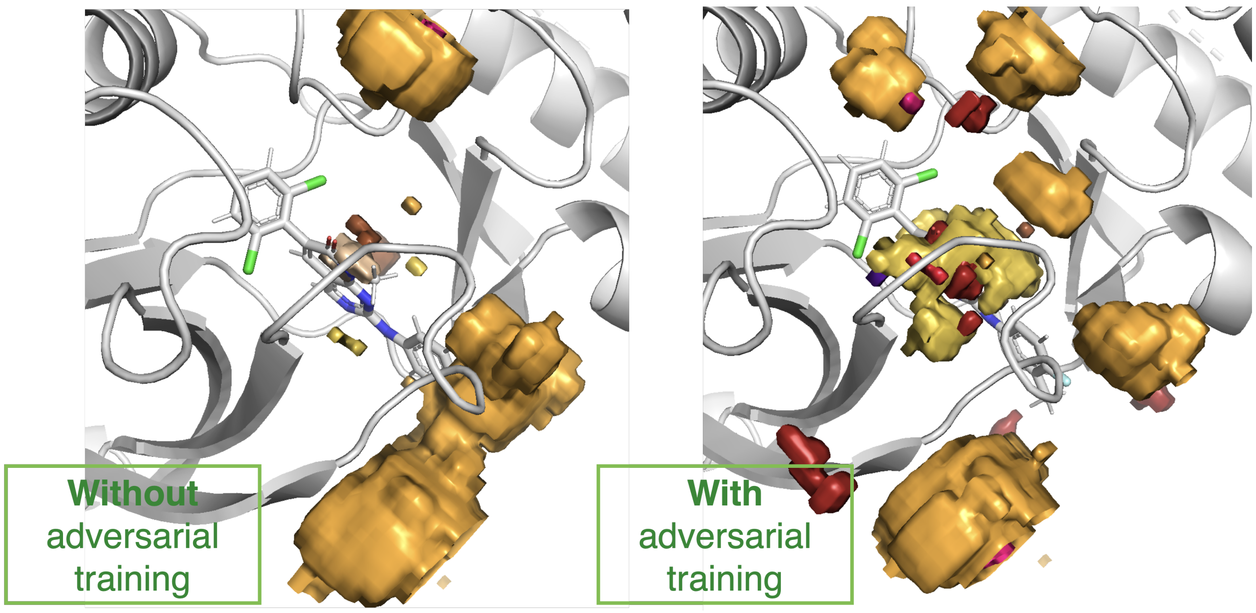


Figure 57: Optimized input for the DUD-E target ABL1 from models trained with and without adversarial perturbations have different distributions of density.

4.4 Conclusions

This project is only partially complete, but the work performed to date suggests that even for the most naïve approach to using a trained network to optimize inputs it is possible to obtain better than random virtual screening performance from the resulting pseudo-pharmacophore, and that performance is not merely derived from centering the grid on the binding site. Futhermore, while our CNN models are vulnerable to adversarially perturbed inputs (a trait likely common to all neural networks[181]), this can be addressed by using Projected Gradient Descent during training to augment input data and the resulting models may have distinct properties worth investigating. Unfortunately the basic approach assessed to date provides faster virtual screening than a conventional conformer rescoring approach (assuming a typical number of compounds to be screened), but that is accompanied by an unacceptable reduction in accuracy. Improving the priors applied during training and the distance functions used to compare the screening and test compound grids may result in a beneficial improvement in both speed and accuracy, as well as the utility for model explanation afforded by neuron-specific input optimization. Currently the activation maximization approach for evaluating what a model has learned provides suggestive insight into the basis of the improved performance of the Dense model compared with our other latest models, but more analysis is required to explain what it has actually learned and whether *chemical* insight can be gained from the features on which its predictions are based.

5.0 Conclusions

The work in this thesis has focused on developing, evaluating, and extending a novel machine learning approach to drug discovery. This approach uses 3D voxels with separate channels for each recognized atom type to featurize input structural data, and over the last few years we have experimented with various training strategies (including distinct tasks and datasets), different ways of using trained CNNs to extend our molecular docking program `gnina`, and more rigorous ways to assess and explain model performance.

In Chapter 2 we focused on motivating a novel approach to molecular modeling for drug discovery by prospectively evaluating selected linear and nonlinear methods fit to molecular fingerprints and precomputed 3D molecular descriptors. While those efforts were not entirely unsuccessful, the methods we used initially could clearly be improved on. This led our group to develop 3D voxel-based CNNs, a project to which I contributed, especially via the algorithm used to generate grids and advocating for the use of independent benchmarks for model evaluation, which I ultimately constructed. Our gridding approach and a desire to support alternative gridding strategies and modern neural network libraries led to the development of `libmolgrid`, which makes our approach to input featurization, transformation, and batch construction available to Python users working with PyTorch or Keras.

In Chapter 3 we revisited the problem of model evaluation. We began by using our novel CNN scoring functions to make submissions to a blinded community benchmark, as in Chapter 2 - but this time we performed close to the top in the affinity prediction tasks. We also tested allowing the CNN to do limited pose sampling by optimizing poses sampled by an approximate version of the Autodock Vina scoring function during docking, with some promising results. However, that challenge was small and somewhat limited, leaving us with lingering questions about how well our models generalize and what they have actually learned. This led us to evaluate their virtual screening performance, a task for which they were not trained. While our initial assessment suggested they performed quite well, the expected utility of many existing virtual screening benchmarks has recently been undermined by reports of significant bias. This literature suggests it is possible to trivially partition actives from inactives in these benchmarks using simple chemical descriptors. Motivated by this problem, we developed a "simple descriptor" baseline with which other models can be compared after fitting to the same training set, and we attempted to understand what underlies our CNN model performance at a high level.

Finally, in Chapter 4 we developed a new approach to understanding the features a model has learned to recognize. This approach can also be used to generate a pseudo-pharmacophore representing what the network has learned to bind to an input binding site. The method is based on using a trained network to perform optimization on its input, either with respect to the last layer before the output (to generate an overall optimal ligand density) or with respect to the network's internal nodes for the purpose of interrogating the features the network has learned. We describe the implementation of this method, relate it to adversarial examples (training with adversarially perturbed examples was also implemented), and perform a preliminary evaluation of using the generated pseudo-pharmacophores for virtual screening.

There are several projects that I started but left unfinished during my thesis work that someone else could resume for theirs. Two of these involved generating a fairly substantial amount of molecular dynamics trajectories, which are currently being used by two other members of our group for their own projects. One of these projects was intended to evaluate the effect of different ligand parametrization methods on the accuracy of free energy calculations; for it, I generated the required simulations for the MMGBSA and MMPBSA three-trajectory method (i.e. receptor-only, ligand-only, and the complex, all in explicit solvent) for all reference complexes in DUD-E with binding affinity data available, with up to three different charge models. I also performed MMGBSA and MMPBSA free energy calculations. The other was intended to evaluate using a Long Short-Term Memory (LSTM) network to map a full molecular dynamics trajectory to the associated binding affinity for the complex, attempting to use attention over the course of the trajectory to identify residues critical to binding. For that project, I implemented a method for specifying simulation frames in `gnina` (later moved to `libmolgrid`) and generated simulations for over half of PDBBind-Refined 2017 (again, I generated the simulations that would be required for a three-trajectory free energy calculation). I also did a preliminary analysis of fitting an LSTM to trajectory data to predict binding affinity and it seemed promising. For these projects, I wrote scripts that automate generating ligand parameters with Antechamber[193], setting up a complex with multiple structures for simulation, and performing initial minimization and pre-production MDs with Amber[24]; these scripts are now used by our entire group.

I was also interested in exploring alternative approaches to voxelizing and traversing structural input data. The only of these that I actually implemented was decomposing inputs into smaller subgrids that are traversed sequentially. This makes the representation size extensive, in theory, although the preliminary evaluation I performed suggested that accuracy degraded when generalizing to input grids that were larger than those on which a network was trained. A reason this size extensiveness could be useful is that instead of needing to choose a large enough training grid size to reasonably contain anything we might like to score later, we might be able to train on a smaller-sized grid and still generalize to larger examples if necessary. Attention could again be leveraged to allow the network to direct its exploration of the input, which could improve the performance of this approach (by avoiding the need to score the full input) and providing some model explanation insight by evaluating what the network is attending to when making predictions.

Finally, more work could be done for the project described in Chapter 4. The virtual screening capabilities could be combined with the outputs of generative adversarial networks (GANs) or variational autoencoders (VAEs), to test whether their outputs are useful for virtual screening. A GAN or VAE might also be used to generate a more effective “natural” prior during the backward pass of input optimization for the approach I explored, evaluating whether that improves the quality of filter activation visualization or pseudo-pharmacophore generation.

Appendix A Supplemental Information for Chapter 2.1

A.1 Ligand-Based Regression Protocol

The CHEMBL3880 HSP90-alpha target dataset was downloaded from `https://www.ebi.ac.uk/chembl/`. The data was then processed to extract compounds with valid IC50 values using Python:

```
import pandas as pd
hsp = pd.read_csv('bioactivity -15_21_16_49.txt',sep='\t')
smi = hsp[(hsp.STANDARD_TYPE == 'IC50') & (hsp.RELATION == '=') &
      (hsp.STANDARD_UNITS == 'nM') & (hsp.PCHEMBL_VALUE > 0)]
      .loc[:,['CANONICAL_SMILES','PCHEMBL_VALUE']]
smi.to_csv('hsp90.smi',sep='\t',index=False,header=False)
```

Salts were then removed from `hsp90.smi` by extracting only the largest connected component. Scripts from `https://github.com/dkoes/qsar-tools` were then used to create models and make predictions.

A.1.1 RDKit

```
outputfingerprints.py hsp90.smi -o hsp90_rdkit_fp.gz --rdkit
trainlinearmodel.py -o hsp90_rdkit.model hsp90_rdkit_fp.gz --maxiter 10000 --elastic
outputfingerprints.py --rdkit hsp90_test.smi -o hsp90_test_rdkit_fp.gz
applylinearmodel.py hsp90_rdkit.model hsp90_test_rdkit_fp.gz > rdkit.out
join hsp90_test.smi rdkit.out | awk '{print $2,$3}' > LigandScores-1.csv
```

A.1.2 SMARTS

```
createsmartsdescriptors.py hsp90.smi -o hsp90.smarts
outputfingerprints.py hsp90.smi -o hsp90_smarts_fp.gz --smarts
```

```
                      --smartsfile hsp90.smarts
trainlinearmodel.py -o hsp90_smarts.model hsp90_smarts_fp.gz --elastic
outputfingerprints.py hsp90_test.smi --smartsfile hsp90.smarts -o hsp90_test_fp.gz
                                     --smarts
applylinearmodel.py hsp90_smarts.model hsp90_test_fp.gz > smarts.out
join hsp90_test.smi smarts.out | awk '{print $2,$3}' > LigandScores-2.csv
```

A.1.3 ECFP6

```
outputfingerprints.py hsp90.smi --ecfp6 -o hsp90_ecfp6_fp.gz
trainlinearmodel.py -o hsp90_ecfp6.model hsp90_ecfp6_fp.gz --elastic
outputfingerprints.py --ecfp6 hsp90_test.smi -o hsp90_test_ecfp6_fp.gz
applylinearmodel.py hsp90_ecfp6.model hsp90_test_ecfp6_fp.gz > ecfp.out
join hsp90_test.smi ecfp.out | awk '{print $2,$3}' > LigandScores-3.csv
```

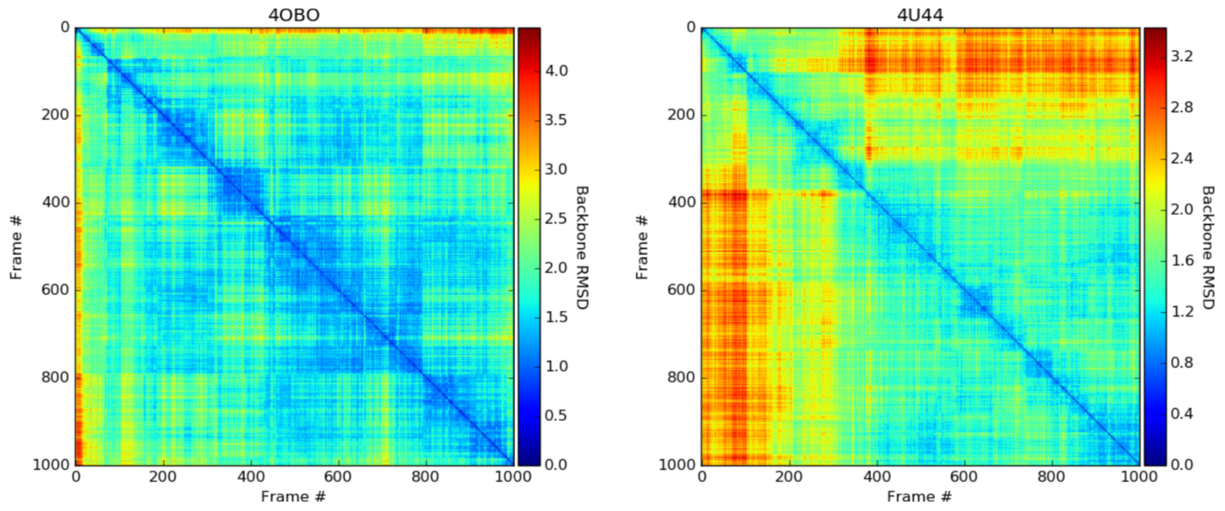
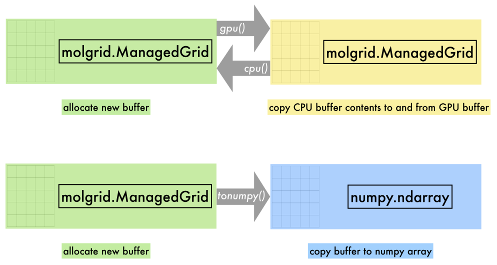


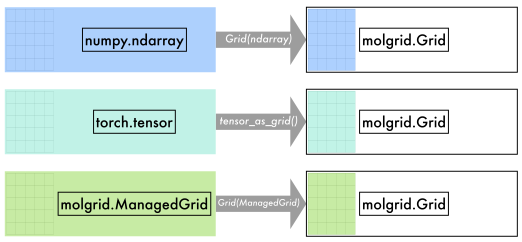
Figure 58: Heatmaps of backbone RMSDs between frames within 100ns trajectories of MAP4K4 molecular dynamics simulations.

Appendix B Supplemental Information for Chapter 2.2

B.1 Grids



(a)



(b)

Figure 59: `ManagedGrid`s manage their own memory buffer, which can migrate data between the CPU and GPU and copy data to a NumPy array as shown in (a). `Grid`s are a view over a memory buffer owned by another object; they may be constructed from a Torch tensor, a `ManagedGrid`, or an arbitrary data buffer with a Python-exposed pointer, including a NumPy array as shown in (b).

Figure 59 illustrates the behavior of `ManagedGrid`s (59a) and `Grid`s (59b). `ManagedGrid`s can migrate data between devices, and they create a copy when converting to or from other objects that have their own memory. `Grid`s do not own memory, instead serving as a view over the memory associated with another object that does; they do not create a copy of the buffer, rather they interact with the original buffer directly, and they cannot migrate it between devices.

The explicit specialization of a grid exposed in the Python `molgrid` API has a naming convention that specifies its dimensionality, underlying data type, and in the case of `Grid`s, the device where its memory buffer is located. The structure of the naming convention is `[GridClass][NumDims][DataType]["CUDA" if GridClass=="Grid" and DataLoc`

```
exprovider = molgrid.ExampleProvider(shuffle=False, balanced=False,
stratify_receptor=False, labelpos=0, stratify_pos=1, stratify_abs=True, stratify_min=0,
stratify_max=0, stratify_step=0, group_batch_size=1, max_group_size=0,
cache_structs=True, add_hydrogens=True, duplicate_first=False, num_copies=1,
make_vector_types=False, data_root="", recmolcache="", ligmolcache="")
```

Listing 2: Available arguments to `ExampleProvider` constructor, along with their default values.

`== "GPU"]`. Since `ManagedGrid`s can migrate their data from host to device, their names do not depend on any particular data location. For example, a 1-dimensional `ManagedGrid` of type float is an `MGrid1f`, a 3-dimensional `Grid` of type float is a `Grid3f`, and a 5-dimensional `Grid` of type double that is a view over device data is a `Grid5dCUDA`.

B.2 ExampleProvider

Listing 2 shows the options that can be set via the `ExampleProvider` constructor. Randomization is enabled with the `shuffle` option; oversampling of underrepresented classes to provide equal representation from all available classes categorized by the `Example` label is enabled with `balanced`; resampling based on a specific molecule associated with an `Example` (determined by the first filename encountered on a given metadata line) comes from `stratify_receptor` (as the name suggests, this is often used to sample equally from `Example`s associated with different receptors); `labelpos` specifies the location of the binary classification label on each line of the metadata file, in terms of an index starting from 0 that numbers the entries on a line; `stratify_pos` similarly specifies the location of a regression target value that will be used to stratify `Example`s for resampling (for example a binding affinity); `stratify_abs` indicates that stratification of `Example`s based on a regression value will use the absolute value, which is useful when a negative value has a special meaning such as with a hinge loss; and `stratify_min`, `stratify_max`, and `stratify_step` are used to define the bins for numerical stratification of `Examples`.

Additional options provide customization for interpreting examples and optimizations for data I/O. When using a recurrent network for processing a sequence of data, such as the case of training with molecular dynamics frames, `group_batch_size` specifies the number of frames to propagate gradients through for truncated backpropagation through time and `max_group_size` indicates the total number of `Examples` associated with the largest `Example` group (e.g. the maximum number of frames). `add_hydrogens` will result in protonation of parsed molecules with OpenBabel. `duplicate_first` will clone the first `CoordinateSet` in an `Example` to be separately paired with each of the subsequent `CoordinateSets` in that `Example` (e.g., a single receptor structure is replicated to match different ligand poses). `num_copies` emits the same example multiple times (this allows the same structure to be presented to the neural network using multiple transformations in a single batch). `make_vector_types` will represent types as a one-hot vector rather than a single index. `cache_structs` will keep coordinates in memory to reduce training time. `data_root` allows the user to specify a shared parent directory for molecular data files, which then allows the metadata file to specify the filenames as relative paths. Finally, `recmolcache` and `ligmolcache` are binary files that store an efficient representation of all receptor and ligand files to be used for training, with each structure stored only once. These are created using the `create_caches2.py` script from `https://github.com/gnina/scripts`. Caches combine many small files into one memory mapped file resulting in a substantial I/O performance improvement and reduction in memory usage during training.

B.3 GridMaker

Listing 3 shows the available arguments to the `GridMaker` constructor. `GridMaker` options include the grid resolution; dimension along each side of the cube; whether to constrain atom density values to be a binary indicator of overlapping an atom, rather than the default of a Gaussian to a multiple of the atomic radius (call this *grm*) and then decaying to 0 quadratically at $\frac{1+2grm^2}{2grm}$; whether to index the atomic radius array by type id (for vector

```
gmaker = molgrid.GridMaker(resolution=0.5, dimension=23.5, binary=False,
radius_type_indexed=False, radius_scale=1.0, gaussian_radius_multiple=1.0)
```

Listing 3: Available arguments to the `GridMaker` constructor, along with their default values.

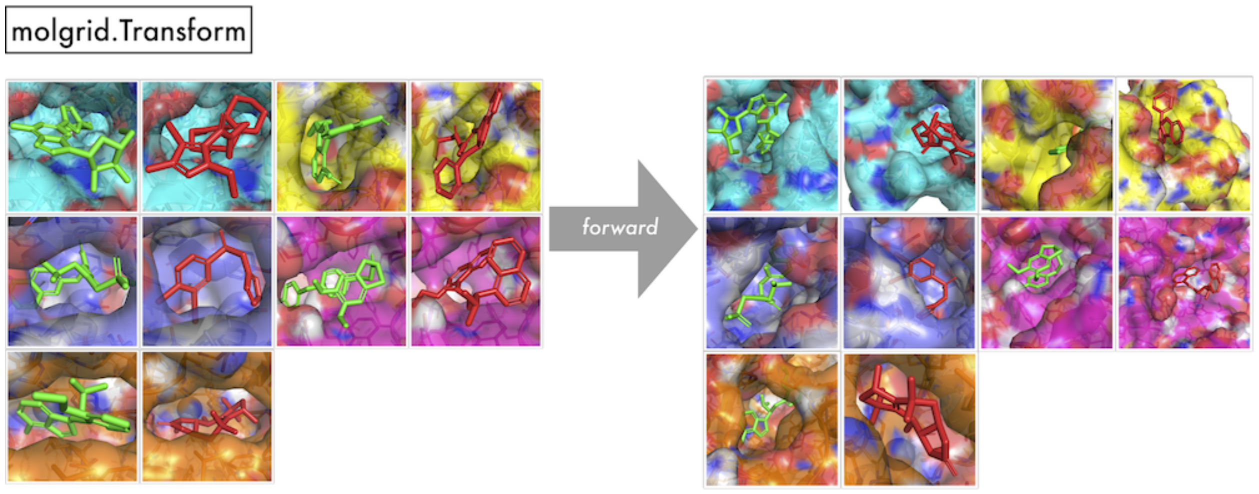
types); a real-valued pre-multiplier on atomic radii, which can be used to change the size of atoms; and, if using real-valued atomic densities (rather than the alternative binary densities), the multiple of the atomic radius to which the Gaussian component of the density extends.

B.4 Transform

```
# Usage 1: specify a center, maximum distance for random translation,
# and whether to randomly rotate
transform1 = molgrid.Transform(center=molgrid.float3(0.0,0.0,0.0), random_translate=0.0,
random_rotation=False)

qt = molgrid.Quaternion(1.0, 0.0, 0.0, 0.0)
center = molgrid.float3(0.0, 0.0, 0.0)
translate = molgrid.float3(0.0, 0.0, 0.0)
# Usage 2: specify a particular rotation, to be performed around the molecule's center
transform2 = molgrid.Transform(qt)
# Usage 3: specify a particular rotation and the center around which it will be performed
transform3 = molgrid.Transform(qt, center)
# Usage 4: specify a particular rotation and center, along with a specific translation
transform4 = molgrid.Transform(qt, center, translate)
```

Listing 4: Available `Transform` constructors.



```python
for data in batch:
    t = molgrid.Transform(center=(0,0,0), random_translate=2.0, random_rotation=True)
    t.forward(data, transformed_data, dotranslate=True)
    # do something with transformed_data
```

Figure 60: An illustration of `molgrid::Transform` usage, applying a distinct random rotation and translation to each of ten input examples. These transformations can also be applied separately to individual coordinate sets. Transformations to grids being generated via a `molgrid::GridMaker` can be generated automatically by specifying `random_rotation=True` or `random_translation=True` when calling `Gridmaker::Forward`.

In order to provide some more detail about specialized use of `molgrid::Transform`, Figure 60 shows the behavior of `Transform::forward`, taking an input `Example` and returning a transformed version of that `Example` in `transformed_example`. Usage examples for the `Transformer` constructors are shown in Listing 4.

B.5 Results

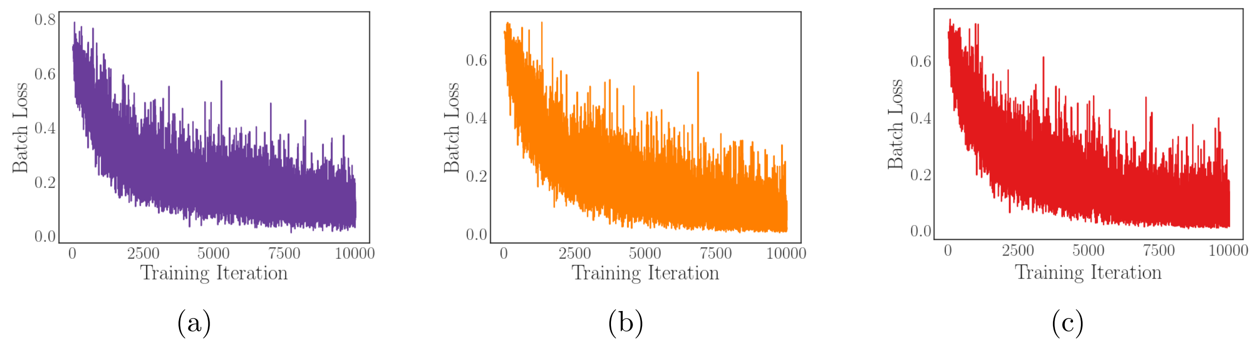


Figure 61: Loss per iteration while training a simple model, with input gridding and transformations performed on-the-fly with `libmolgrid` and neural network implementation performed with (a) Caffe, (b) PyTorch, and (c) Keras with a Tensorflow backend.

Figure 61 shows successful training of a basic feed-forward network on a toy dataset using each of these three deep learning frameworks to perform binary classification of active versus inactive binding modes. Timing calculations for the main text performance figures were performed using GNU `time`, while memory utilization was obtained with `nvidia-smi -q -i 1 -d MEMORY -l 1`. The Caffe data was obtained using `caffe train` with the model at `https://github.com/gnina/models/blob/master/affinity/affinity.model` with the affinity layers removed; the PyTorch data was obtained using `https://gnina.github.io/libmolgrid/tutorials/train_basic_CNN_with_PyTorch.html`, run for 10,000 iterations; and the Keras data was obtained using `https://gnina.github.io/libmolgrid/tutorials/train_basic_CNN_with_Tensorflow.html`, run for 10,000 iterations. The metadata file for training is at `https://github.com/gnina/libmolgrid/blob/master/test/data/small.types`, using structures found at `https://github.com/gnina/libmolgrid/tree/master/test/data/structs`. The Cartesian reduction example can be found at `https://gnina.github.io/libmolgrid/tutorials/train_simple_cartesian_reduction.html`.

Appendix C Supplementary Information for Section 3.2

C.1 Confidence Intervals for EF1% Differences Between Methods

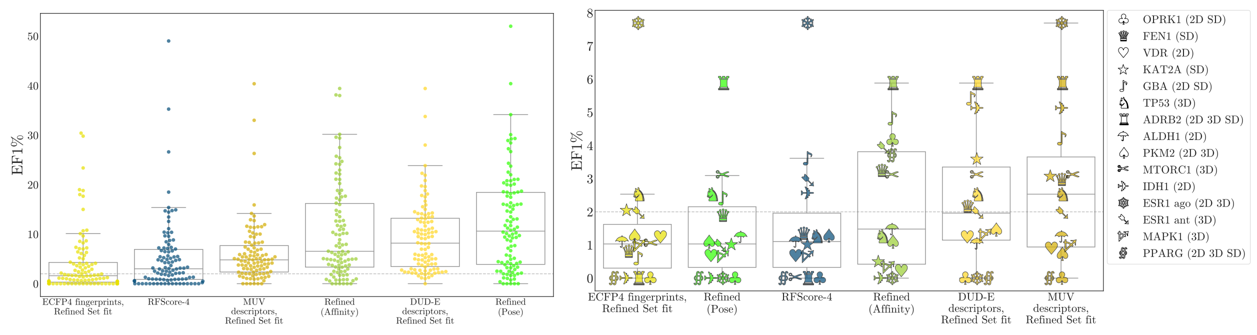
There has been some debate about how to statistically assess claims made about machine learning model performance for drug discovery tasks like virtual screening. The norm has been, as in much of the other recent machine learning literature that reports model performance on community benchmarks, to avoid statistically evaluating them at all. More recently some have used nonparametric tests like the Wilcoxon on a given benchmark as a whole[2] to compute a p-value indicating the probability of whether a given pair of methods could have had the observed or more extreme performance on the benchmark if they were not actually performing differently. This is potentially problematic for a virtual screening benchmark since the number of unique observations is equal to the number of targets, and there are often fewer than the recommended number for using the Wilcoxon to make accurate calculations; furthermore, a practitioner choosing between scoring functions probably cares about whether a given method is likely to better than another *on the target they are working on*, and it may be the case that the method appeared better on the benchmark as a whole but exhibited worse performance on a similar target to the one of interest. A reasonable approach when assessing models fit to a given dataset is to perform 5×2 cross validation[44], which consists of five rounds of two-fold cross validation, but in the case of an independent test benchmark we would like to avoid fitting to the test data. Instead we chose to use bootstrapping to compute 95% confidence intervals on the difference in EF1% values between methods per target[132], and permutation tests on the active ranks[209] to calculate p-values indicating whether the methods could have had the observed or more extreme differences in performance and still be expected to perform the same prospectively on a screen for that target. Unfortunately this means many comparisons between methods were made, and that data are difficult to summarize concisely. We have included the comparisons made with respect to the Dense affinity model (Table 6 for LIT-PCBA only, since it consists of real data from high throughput screens and therefore the assumption that the data distribution reflects real-world data distributions seems more likely to hold for LIT-PCBA than for DUD-E.

Target	RF-Score-4	RF-Score-VS	Vinardo	Vina
ADRB2	0.000, [0.000, 0.000], p=1.000	0.000 [0.000, 0.000], p=1.000	0.000 [0.000, 0.000], p=1.000	0.000 [0.000, 0.000], p=1.000
ALDH1	**0.698 [0.363, 1.088], p=0.000**	0.335 [-0.084, 0.740], p=0.127	0.433 [0.056, 0.851], p=0.044	0.265 [-0.140, 0.656], p=0.244
ESR1 ago	7.692 [-15.385, 30.769], p=1.000	7.692 [-15.385, 30.769], p=1.000	15.385 [-7.692, 30.769], p=0.484	7.692 [-15.385, 30.769], p=1.000
ESR1 ant	**4.902 [0.980, 12.745], p=0.208**	**6.863 [1.961, 12.745], p=0.007**	**5.882 [0.980, 12.745], p=0.033**	**4.902 [0.980, 11.765], p=0.212**
FEN1	**4.607 [1.897, 7.046], p=0.001**	2.981 [-0.271, 5.691], p=0.071	**4.336 [1.355, 7.046], p=0.002**	**3.523 [0.542, 6.233], p=0.025**
GBA	**8.434 [3.012, 13.855], p=0.007**	**12.048 [6.627, 16.867], p=0.000**	**6.627 [0.602, 12.651], p=0.053**	**8.434 [3.012, 13.855], p=0.005**
IDH1	5.128 [-5.128, 12.821], p=0.613	0.000 [-12.821, 12.821], p=1.000	7.692 [0.000, 15.385], p=0.241	7.692 [-2.564, 15.385], p=0.243
KAT2A	1.546 [-1.031, 4.124], p=0.446	0.000 [-3.093, 3.093] p=1.000	1.546 [-1.031, 4.124], p=0.443	2.062 [-0.515, 4.124], p=0.213
MAPK1	0.649 [-0.974, 2.273], p=0.694	0.325 [-1.299, 1.948], p=1.000	-0.974 [-2.922, 1.299], p=0.538	-1.623 [-3.896, 0.649], p=0.259
MTORC1	1.031 [-1.031, 2.062], p=1.000	1.031 [-1.031, 2.062], p=1.000	1.031 [-1.031, 2.062], p=1.000	1.031 [-1.031, 3.093], p=1.000
OPRK1	4.167 [-8.333, 8.333], p=1.000	4.167, [-4.167, 8.333], p=1.000	4.167 [-4.167, 8.333], p=1.000	4.167 [-8.333, 8.333], p=1.000
PKM2	-0.733 [-1.832, 0.366], p=0.342	-0.183 [-1.099, 0.733], p=1.000	-0.183 [-1.099, 0.916], p=1.000	-0.549 [-1.648, 0.549], p=0.502
PPARG	3.704 [-3.704, 7.407], p=1.000	3.704 [-3.704, 7.407], p=1.000	-3.704 [-14.815, 7.407], p=1.000	3.704 [-3.704, 7.407], p=1.000
TP53	0.000 [-5.063, 2.532], p=1.000	1.266 [-2.532, 3.797], p=1.000	-2.532 [-7.595, 2.532], p=0.621	-1.266 [-6.329, 2.532], p=1.000
VDR	0.567 [-0.340, 1.587], p=0.334	-0.907 [-2.041, 0.227], p=0.202	0.454 [-0.567, 1.361], p=0.492	**0.907 [0.113, 1.701], p=0.059**

Table 6: Actual difference, 95% confidence intervals, and p-values for difference in EF1% between the Dense Affinity model and non-CNN methods, evaluated for the targets in LIT-PCBA. Cells where the confidence interval does not include 0 are bolded.

C.2 Simple Descriptor Model Performance

Actual virtual screening performance (measured by EF1%) is shown for models fit to the PDBbind-Refined set (Figure 62), the PDBbind-General set (Figure 63), and the CrossDock set (Figure 64). Simple descriptor model performance is broken down by input feature type, with the performance shown the best performance achieved for the target for any of the models fit to that set of features.



(a) (b)

Figure 62: Actual EF1% values for models fit to the PDBbind Refined Set. Simple descriptor models are broken down by descriptor type. Results are shown for DUD-E (a) and LIT-PCBA (b).

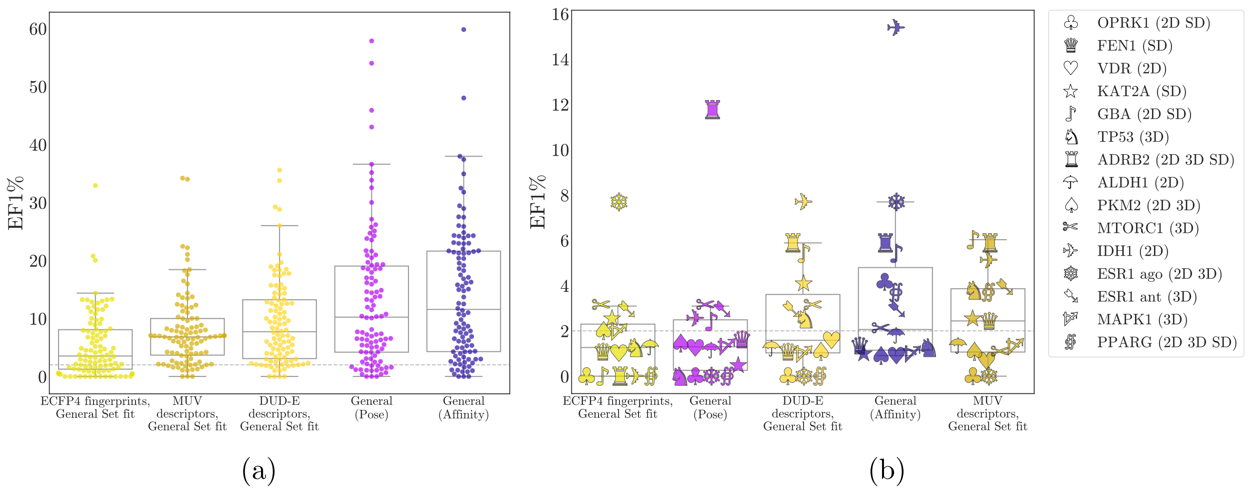


Figure 63: Actual EF1% values for models fit to the PDBbind General Set. Simple descriptor models are broken down by descriptor type. Results are shown for DUD-E (a) and LIT-PCBA (b).

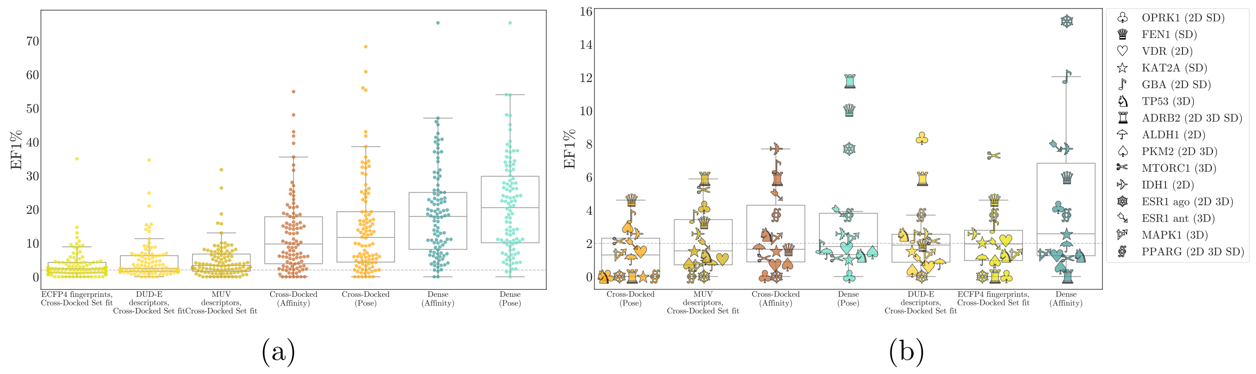
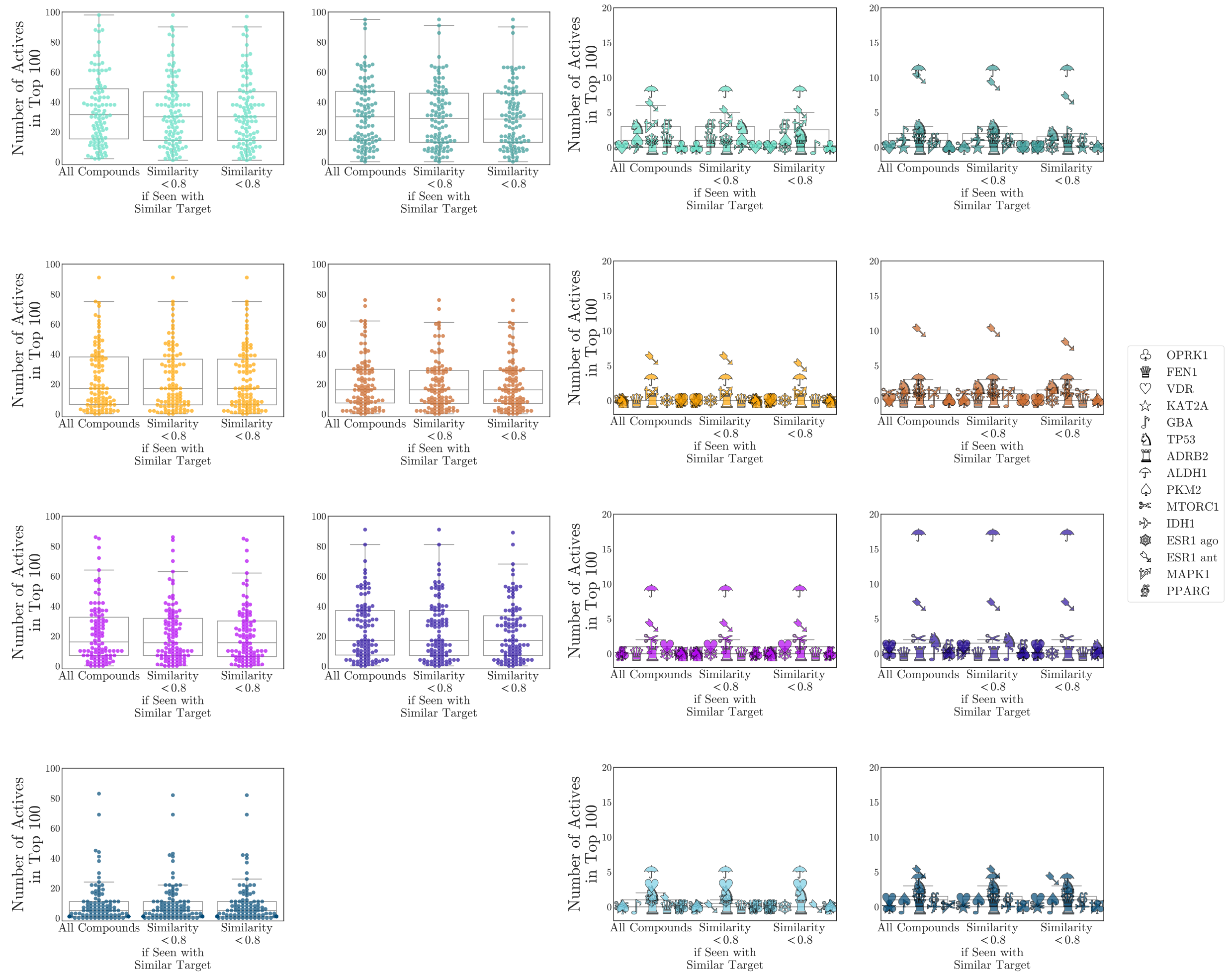


Figure 64: Actual EF1% values for models fit to the Cross-Docked set[57]. Simple descriptor models are broken down by descriptor type. Results are shown for DUD-E (a) and LIT-PCBA (b).

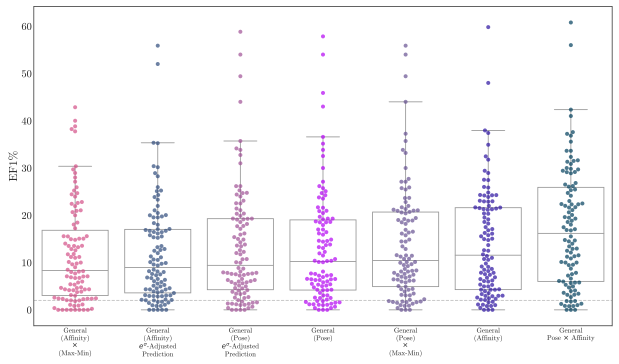
C.3 The effect of training set similarity on performance



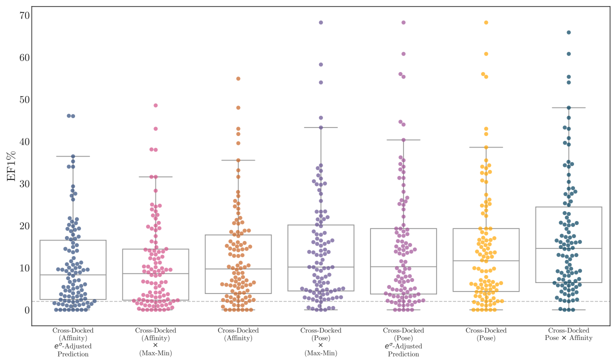
(a) (b)

Figure 65: Comparison of EF1% for DUD-E (a) and LIT-PCBA (b) with all compounds included versus removing compounds based on similarity to training set. The threshold for compound similarity was a Tanimoto coefficient of 0.8. We applied that threshold to compounds that were provided as training examples as part of a complex with a protein that had at least 50% sequence identity with the test set target being evaluated, as well as across all compounds seen during training.

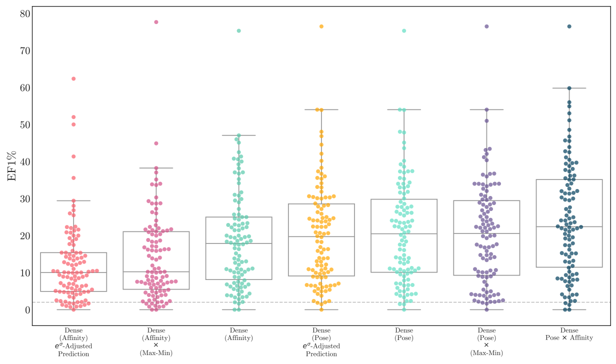
C.4 Reweighting scores to calibrate performance



(a)



(b)



(c)

Figure 66: Effect of various score calibration approaches on DUD-E EF1%.

C.5 Score Distributions Per Class

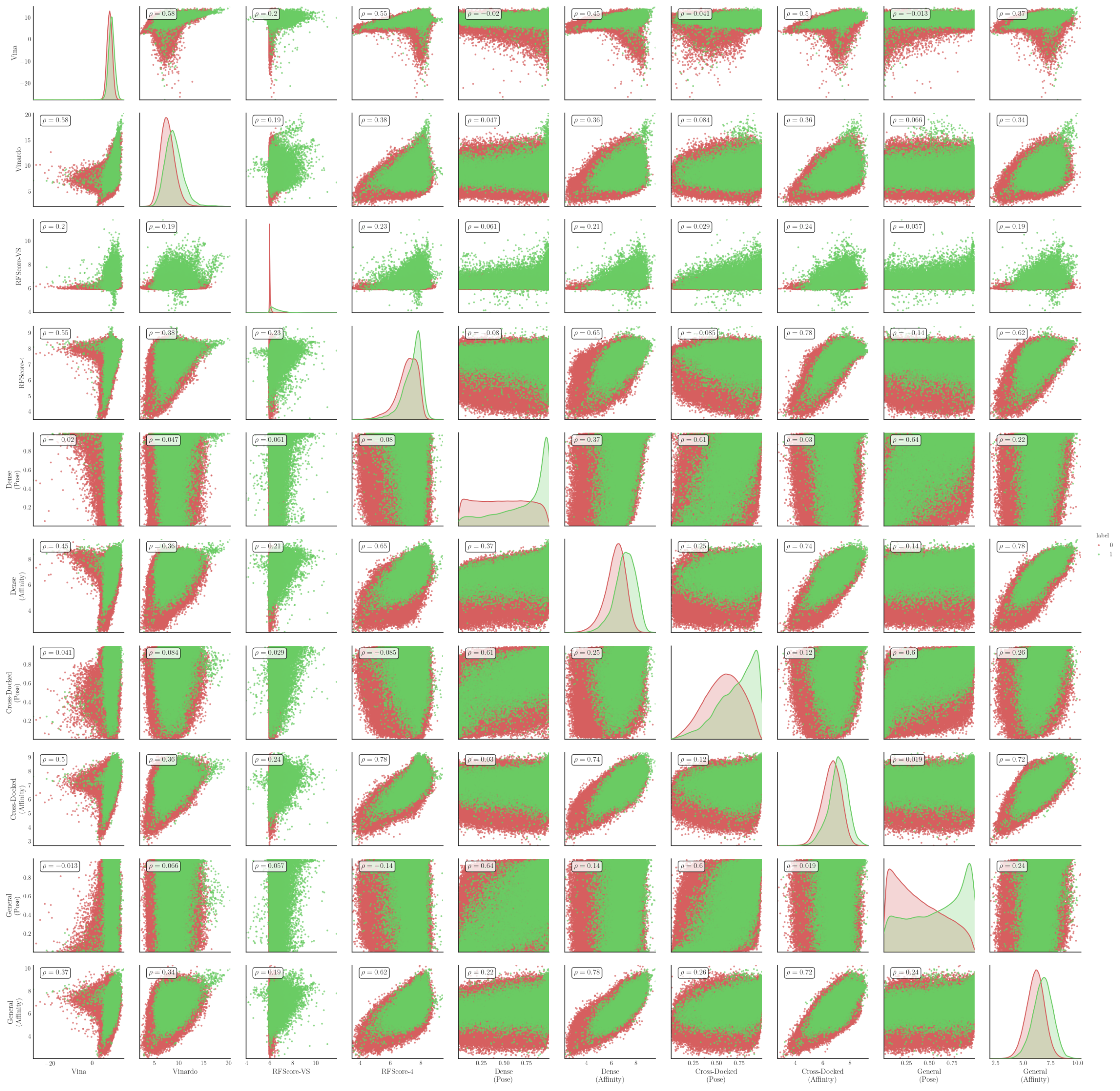


Figure 67: Score distributions and correlations for the different methods tested for DUD-E, separated by class.

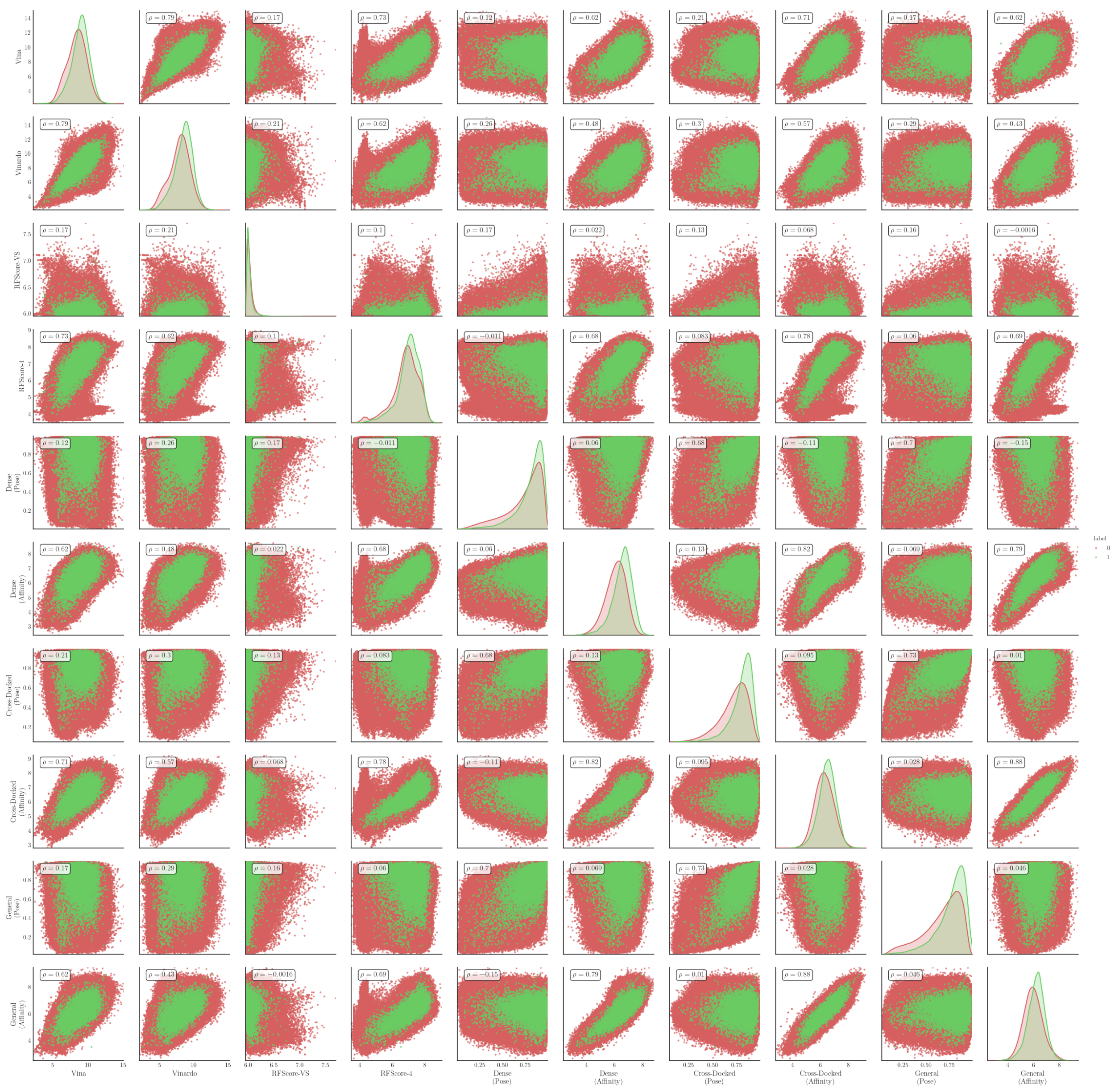


Figure 68: Score distributions and correlations for the different methods tested for LIT-PCBA, separated by class.

Bibliography

[1] The Open Babel Package. version 2.3.

[2] Yusuf Adeshina, Eric Deeds, and John Karanicolas. Machine learning classification can reduce false positives in structure-based virtual screening. *bioRxiv*, 2020.

[3] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[4] Matteo Aldeghi, Alexander Heifetz, Michael J Bodkin, Stefan Knapp, and Philip C Biggin. Accurate calculation of the absolute free energy of binding for drug molecules. *Chemical science*, 7(1):207–218, 2016.

[5] Afshine Amidi, Shervine Amidi, Dimitrios Vlachakis, Vasileios Megalooikonomou, Nikos Paragios, and Evangelia I Zacharaki. Enzynet: enzyme classification using 3d convolutional neural networks on spatial representation. *PeerJ*, 6:e4750, 2018.

[6] Hossam M Ashtawy and Nihar R Mahapatra. A comparative assessment of ranking accuracies of conventional and machine-learning-based scoring functions for protein-ligand binding affinity prediction. *IEEE/ACM Transactions on computational biology and bioinformatics*, 9(5):1301–1313, 2012.

[7] Hossam M. Ashtawy and Nihar R. Mahapatra. Machine-learning scoring functions for identifying native poses of ligands docked to known and novel proteins. *BMC Bioinformatics*, 16(6):1–17, 2015. [PubMed:25916860] [PubMed Central:PMC4416170] [doi:10.1186/1471-2105-16-S6-S3].

[8] Todor Kirilov Avramov, Dan Vyenielo, Josue Gomez-Blanco, Swathi Adinarayanan, Javier Vargas, and Dong Si. Deep learning for validating and estimating resolution of cryo-electron microscopy density maps. *Molecules*, 24(6):1181, 2019.

[9] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, 07 2015.

[10] P. J. Ballester and J. B. O. Mitchell. A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking. *Bioinformatics*, 26(9):1169, 2010. [PubMed:20236947] [PubMed Central:PMC3524828] [doi:10.1093/bioinformatics/btq112].

[11] Pedro J. Ballester and John B. O. Mitchell. A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking. *Bioinformatics*, 26(9):1169–1175, 2010.

[12] A. Patrícia Bento, Anna Gaulton, Anne Hersey, Louisa J. Bellis, Jon Chambers, Mark Davies, Felix A. Krüger, Yvonne Light, Lora Mak, Shaun McGlinchey, Michal Nowotka, George Papadatos, Rita Santos, and John P. Overington. The ChEMBL bioactivity database: an update. *Nucleic Acids Research*, 42(D1):D1083–D1090, nov 2013.

[13] Guy E Blelloch. Pre x sums and their applications. Technical report, Citeseer, 1990.

[14] Regine S Bohacek, Colin McMartin, and Wayne C Guida. The art and practice of structure-based drug design: a molecular modeling perspective. *Medicinal research reviews*, 16(1):3–50, 1996.

[15] H. J. Böhm. The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure. *J. Comput.-Aided Mol. Des.*, 8(3):243–256, 1994. [PubMed:7964925].

[16] Paul C Boutros, Adam A Margolin, Joshua M Stuart, Andrea Califano, and Gustavo Stolovitzky. Toward better benchmarking: challenge-based methods assessment in cancer genomics. *Genome biology*, 15(9):462, 2014.

[17] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[18] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[19] B. R. Brooks, R. E. Bruccoleri, and B. D. Olafson. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.*, 4(2):187–217, 1983.

[20] Rodolpho C Braga, Vinicius M Alves, Arthur C Silva, Marilia N Nascimento, Flavia C Silva, Luciano M Liao, and Carolina H Andrade. Virtual screening strategies in medicinal chemistry: the state of the art and current challenges. *Current topics in medicinal chemistry*, 14(16):1899–1912, 2014.

[21] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[22] Heather A Carlson. Lessons learned over four benchmark exercises from the community structure–activity resource, 2016.

[23] Heather A Carlson, Richard D Smith, Kelly L Damm-Ganamet, Jeanne A Stuckey, Aqeel Ahmed, Maire A Convery, Donald O Somers, Michael Kranz, Patricia A Elkins, Guanglei Cui, et al. Csar 2014: a benchmark exercise using unpublished data from pharma. *Journal of chemical information and modeling*, 56(6):1063–1077, 2016.

[24] D. A. Case, V. Babin, J. T. Berryman, R. M. Betz, Q. Cai, D. S. Cerutti, T. E. Cheatham, T. A. Darden, R. E. Duke, H. Gohlke, Andreas W Götz, S. Gusarov, N. Homeyer, P. Janoski, J. Kaus, I. Kolossvary, A. Kovalenko, T. S. Lee, S. LeGrand, T. Luchko, R. Luo, B. Madej, K. M. Merz, F. Paesani, D. R. Roe, A. Roitberg, C. Sagui, R. Salomon-Ferrer, G. Seabra, C. Simmerling, W. Smith, J. Swails, R. C. Walker, J. Wang, R. M. Wolf, X. Wu, and P. A. Kollman. AMBER 14.

[25] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods. The Amber biomolecular simulation programs. *J. Comput. Chem.*, 26(16):1668–1688, 2005. [PubMed:16200636] [PubMed Central:PMC1989667] [doi:10.1002/jcc.20290].

[26] Ludovic Chaput, Juan Martinez-Sanz, Nicolas Saettel, and Liliane Mouawad. Benchmark of four popular virtual screening programs: construction of the active/decoy dataset remains a major determinant of measured performance. *Journal of cheminformatics*, 8(1):1–17, 2016.

[27] P. S. Charifson, J. J. Corkery, M. A. Murcko, and W. P. Walters. Consensus scoring: A method for obtaining improved hit rates from docking databases of three-dimensional structures into proteins. *J Med Chem*, 42(25):5100–9, 1999. [PubMed:10602695].

[28] Beining Chen, Robert F. Harrison, George Papadatos, Peter Willett, David J. Wood, Xiao Qing Lewell, Paulette Greenidge, and Nikolaus Stiefl. Evaluation of machine-

learning methods for ligand-based virtual screening. *Journal of Computer-Aided Molecular Design*, 21(1-3):53–62, jan 2007.

[29] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 2019.

[30] Lieyang Chen, Anthony Cruz, Steven Ramsey, Callum J Dickson, Jose S Duca, Viktor Hornak, David R Koes, and Tom Kurtzman. Hidden bias in the dud-e dataset leads to misleading performance of deep learning in structure-based virtual screening. *PloS one*, 14(8):e0220113, 2019.

[31] T. Cheng, Q. Li, Z. Zhou, Y. Wang, and S. H. Bryant. Structure-based virtual screening for drug discovery: a problem-centric review. *AAPS J*, 14(1):133–41, 2012. [PubMed:22281989] [PubMed Central:PMC3282008] [doi:10.1208/s12248-012-9322-0].

[32] T. Cheng, X. Li, Y. Li, Z. Liu, and R. Wang. Comparative assessment of scoring functions on a diverse test set. *J Chem Inf Model*, 49(4):1079–93, 2009. [PubMed:19358517] [doi:10.1021/ci9000053].

[33] Artem Cherkasov, Eugene N Muratov, Denis Fourches, Alexandre Varnek, Igor I Baskin, Mark Cronin, John Dearden, Paola Gramatica, Yvonne C Martin, Roberto Todeschini, et al. Qsar modeling: where have you been? where are you going to? *Journal of medicinal chemistry*, 57(12):4977–5010, 2014. [PubMed:24351051] [PubMed Central:PMC4074254] [doi:10.1021/jm4004285].

[34] Yoonjoo Choi and Charlotte M. Deane. FREAD revisited: Accurate loop structure prediction using a database search algorithm. *Proteins*, pages NA–NA, 2009.

[35] Vladimir Chupakhin, Gilles Marcou, Igor Baskin, Alexandre Varnek, and Didier Rognan. Predicting ligand binding modes from neural networks trained on protein–ligand interaction fingerprints. *Journal of chemical information and modeling*, 53(4):763–772, 2013. [PubMed:23480697] [doi:10.1021/ci300200r].

[36] Lucy J Colwell. Statistical and machine learning approaches to predicting protein-ligand interactions. *Current opinion in structural biology*, 49:123–128, 2018.

[37] Isidro Cortés-Ciriano and Andreas Bender. Deep confidence: a computationally efficient framework for calculating reliable prediction errors for deep neural networks. *Journal of chemical information and modeling*, 59(3):1269–1281, 2018.

[38] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[39] W.L. DeLano and Schrödinger, LLC. The PyMOL molecular graphics system, version 1.8. November 2015.

[40] Wei Deng, Curt Breneman, and Mark J. Embrechts. Predicting protein-ligand binding affinities using novel geometrical descriptors and machine-learning methods. *Journal of Chemical Information and Computer Sciences*, 44(2):699–703, 2004. [PubMed:15032552] [doi:10.1021/ci034246+].

[41] Zhan Deng, Claudio Chuaqui, and Juswinder Singh. Structural interaction fingerprint (sift): a novel method for analyzing three-dimensional protein- ligand binding interactions. *Journal of medicinal chemistry*, 47(2):337–344, 2004.

[42] R. L. DesJarlais, R. P. Sheridan, G. L. Seibel, J. S. Dixon, I. D. Kuntz, and R. Venkataraghavan. Using shape complementarity as an initial screen in designing ligands for a receptor binding site of known three-dimensional structure. *J Med Chem*, 31(4):722–9, 1988. [PubMed:3127588].

[43] R. S. DeWitte and E. I. Shakhnovich. SMoG: de Novo design method based on simple, fast, and accurate free energy estimates .1. Methodology and supporting evidence. *J. Am. Chem. Soc.*, 118(47):11733–11744, 1996.

[44] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[45] Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. Innovation in the pharmaceutical industry: new estimates of r&d costs. *Journal of health economics*, 47:20–33, 2016.

[46] Malgorzata N Drwal and Renate Griffith. Combination of ligand-and structure-based methods in virtual screening. *Drug Discovery Today: Technologies*, 10(3):e395–e401, 2013.

[47] Jacob D. Durrant and Rommie E. Amaro. Machine-learning techniques applied to antibacterial drug discovery. *Chemical Biology & Drug Design*, 85(1):14–21, 2015. [PubMed:25521642] [PubMed Central:PMC4273861] [doi:10.1111/cbdd.12423].

[48] Jacob D Durrant and J Andrew McCammon. Nnscore: A neural-network-based scoring function for the characterization of protein- ligand complexes. *Journal of chemical information and modeling*, 50(10):1865–1871, 2010. [PubMed:20845954] [PubMed Central:PMC2964041] [doi:10.1021/ci100244v].

[49] Jacob D Durrant and J Andrew McCammon. Nnscore 2.0: a neural-network receptor–ligand scoring function. *Journal of chemical information and modeling*, 51(11):2897–2903, 2011. [PubMed:22017367] [PubMed Central:PMC3225089] [doi:10.1021/ci2003889].

[50] Jacob D. Durrant and J. Andrew McCammon. Nnscore 2.0: A neural-network receptor-ligand scoring function. *J. Chem. Inf. Model.*, 51(11):2897–2903, 2011.

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

[52] Sean Ekins, Joel S Freundlich, Judith V Hobrath, E Lucile White, and Robert C Reynolds. Combining computational methods for hit to lead optimization in mycobacterium tuberculosis drug discovery. *Pharmaceutical research*, 31(2):414–435, 2014.

[53] M. D. Eldridge, C. W. Murray, T. R. Auton, G. V. Paolini, and R. P. Mee. Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. *J Comput Aided Mol Des*, 11(5):425–45, 1997. [PubMed:9385547].

[54] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[55] T. J. Ewing, S. Makino, A. G. Skillman, and I. D. Kuntz. DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases. *J Comput Aided Mol Des*, 15(5):411–28, 2001. [PubMed:11394736].

[56] Evan N Feinberg, Debnil Sur, Zhenqin Wu, Brooke E Husic, Huanghao Mai, Yang Li, Saisai Sun, Jianyi Yang, Bharath Ramsundar, and Vijay S Pande. Potentialnet for molecular property prediction. *ACS central science*, 4(11):1520–1530, 2018.

[57] Paul G Francoeur, Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B Iovanisci, Ian Snyder, and David R Koes. Three-dimensional convolutional neural net-

works and a cross-docked data set for structure-based drug design. *Journal of Chemical Information and Modeling*, 60(9):4200–4215, 2020.

[58] R. A. Friesner, J. L. Banks, R. B. Murphy, T. A. Halgren, J. J. Klicic, D. T. Mainz, M. P. Repasky, E. H. Knoll, M. Shelley, J. K. Perry, D. E. Shaw, P. Francis, and P. S. Shenkin. Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy. *J Med Chem*, 47(7):1739–49, 2004. [PubMed:15027865] [doi:10.1021/jm0306430].

[59] Joffrey Gabel, Jérémy Desaphy, and Didier Rognan. Beware of machine learning-based scoring functions? on the danger of developing black boxes. *Journal of chemical information and modeling*, 54(10):2807–2815, 2014. [PubMed:25207678] [doi:10.1021/ci500406k].

[60] Zied Gaieb, Shuai Liu, Symon Gathiaka, Michael Chiu, Huanwang Yang, Chenghua Shao, Victoria A Feher, W Patrick Walters, Bernd Kuhn, Markus G Rudolph, et al. D3r grand challenge 2: blind prediction of protein–ligand poses, affinity rankings, and relative binding free energies. *Journal of computer-aided molecular design*, 32(1):1–20, 2018.

[61] Symon Gathiaka, Shuai Liu, Michael Chiu, Huanwang Yang, Jeanne A Stuckey, You Na Kang, Jim Delproposto, Ginger Kubish, James B Dunbar, Heather A Carlson, et al. D3r grand challenge 2015: evaluation of protein–ligand pose and affinity predictions. *Journal of computer-aided molecular design*, 30(9):651–668, 2016.

[62] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[63] H. Gohlke, M. Hendlich, and G. Klebe. Knowledge-based scoring function to predict protein-ligand interactions. *J. Mol. Biol.*, 295(2):337–356, 2000.

[64] Joseph Gomes, Bharath Ramsundar, Evan N Feinberg, and Vijay S Pande. Atomic convolutional networks for predicting protein-ligand binding affinity. *arXiv preprint arXiv:1703.10603*, 2017.

[65] Joseph Gomes, Bharath Ramsundar, Evan N Feinberg, and Vijay S Pande. Atomic convolutional networks for predicting protein-ligand binding affinity. *arXiv preprint arXiv:1703.10603*, 2017.

[66] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[67] Harshit Gupta, Kyong Hwan Jin, Ha Q Nguyen, Michael T McCann, and Michael Unser. Cnn-based projected gradient descent for consistent ct image reconstruction. *IEEE transactions on medical imaging*, 37(6):1440–1453, 2018.

[68] Edward Harder, Wolfgang Damm, Jon Maple, Chuanjie Wu, Mark Reboul, Jin Yu Xiang, Lingle Wang, Dmitry Lupyan, Markus K. Dahlgren, Jennifer L. Knight, Joseph W. Kaus, David S. Cerutti, Goran Krilov, William L. Jorgensen, Robert Abel, and Richard A. Friesner. OPLS3: A force field providing broad coverage of drug-like small molecules and proteins. *J. Chem. Theory Comput.*, 12(1):281–296, jan 2016.

[69] Md Mahmudulla Hassan, Daniel Castaneda Mogollon, Olac Fuentes, and Suman Sirimulla. Dlscore: A deep learning model for predicting protein-ligand binding affinities. *ChemRxiv*, 2018.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. [arXiv:1512.03385.

[71] Manfred Hendlich, Friedrich Rippmann, and Gerhard Barnickel. Ligsite: automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of Molecular Graphics and Modelling*, 15(6):359–363, 1997.

[72] Joshua Hochuli, Alec Helbling, Tamar Skaist, Matthew Ragoza, and David Ryan Koes. Visualizing convolutional neural network protein-ligand scoring. *Journal of Molecular Graphics and Modelling*, 2018.

[73] Joshua Hochuli, Alec Helbling, Tamar Skaist, Matthew Ragoza, and David Ryan Koes. Visualizing convolutional neural network protein-ligand scoring. *arXiv preprint arXiv:1803.02398*, 2018.

[74] Jui-Hua Hsieh, Shuangye Yin, Shubin Liu, Alexander Sedykh, Nikolay V Dokholyan, and Alexander Tropsha. Combined application of cheminformatics- and physical force field-based scoring functions improves binding affinity prediction for CSAR data sets. *J Chem Inf Model*, 51(9):2027–35, September 2011. [PubMed:21780807] [PubMed Central:PMC3183266] [doi:10.1021/ci200146e].

[75] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[76] Niu Huang, Brian K Shoichet, and John J Irwin. Benchmarking sets for molecular docking. *Journal of medicinal chemistry*, 49(23):6789–6801, 2006.

[77] S. Y. Huang and X. Zou. An iterative knowledge-based scoring function to predict protein-ligand interactions: II. Validation of the scoring function. *J. Comput. Chem.*, 27(15):1876–1882, 2006. [PubMed:16983671] [doi:10.1002/jcc.20505].

[78] S. Y. Huang and X. Zou. Mean-Force Scoring Functions for Protein-Ligand Binding. *Annu. Rep. Comp. Chem.*, 6:280–296, 2010.

[79] Sheng-You Huang and Xiaoqin Zou. Scoring and lessons learned with the CSAR benchmark using an improved iterative knowledge-based scoring function. *J Chem Inf Model*, 51(9):2097–106, September 2011. [PubMed:21830787] [PubMed Central:PMC3190652] [doi:10.1021/ci2000727].

[80] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018.

[81] Fergus Imrie, Anthony R. Bradley, Mihaela van der Schaar, and Charlotte M. Deane. 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.*, 58(11):2319–2330, 2018.

[82] John J Irwin. Community benchmarks for virtual screening. *Journal of computer-aided molecular design*, 22(3-4):193–199, 2008.

[83] Johanna M Jansen, Rommie E Amaro, Wendy Cornell, Y Jane Tseng, and W Patrick Walters. Computational chemistry and drug discovery: a call to action. *Future medicinal chemistry*, 4(15):1893–1896, 2012.

[84] Julia B Jasper, Lina Humbeck, Tobias Brinkjost, and Oliver Koch. A novel interaction fingerprint derived from per atom score contributions: exhaustive evaluation of interaction fingerprint performance in docking based virtual screening. *Journal of cheminformatics*, 10(1):1–13, 2018.

[85] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[86] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[87] Mingjian Jiang, Zhen Li, Yujie Bian, and Zhiqiang Wei. A novel protein descriptor for the prediction of drug binding sites. *BMC bioinformatics*, 20(1):1–13, 2019.

[88] J Jiménez, Stefan Doerr, G Martínez-Rosell, AS Rose, and Gianni De Fabritiis. Deepsite: protein-binding site predictor using 3d-convolutional neural networks. *Bioinformatics*, 33(19):3036–3042, 2017.

[89] José Jiménez-Luna, Laura Pérez-Benito, Gerard Martínez-Rosell, Simone Sciabola, Rubben Torella, Gary Tresadern, and Gianni De Fabritiis. Deltadelta neural networks for lead optimization of small molecule potency. *Chemical Science*, 2019.

[90] José Jiménez Luna, Miha Skalic, Gerard Martinez-Rosell, and Gianni De Fabritiis. K deep: Protein-ligand absolute binding affinity prediction via 3d-convolutional neural networks. *Journal of chemical information and modeling*, 2018.

[91] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking. *J Mol Biol*, 267(3):727–48, 1997. [PubMed:9126849] [doi:10.1006/jmbi.1996.0897].

[92] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Am. Chem. Soc.*, 118(45):11225–11236, 1996.

[93] Robert N. Jorissen and Michael K. Gilson. Virtual screening of molecular databases using a support vector machine. *Journal of Chemical Information and Modeling*, 45(3):549–561, 2005. [PubMed:15921445] [doi:10.1021/ci049641u].

[94] Seiji Kajita, Nobuko Ohba, Ryosuke Jinnouchi, and Ryoji Asahi. A universal 3d voxel descriptor for solid-state material informatics with deep convolutional neural networks. *Scientific reports*, 7(1):16991, 2017.

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

[96] Ryangguk Kim and Jeffrey Skolnick. Assessment of programs for ligand binding affinity prediction. *Journal of computational chemistry*, 29(8):1316–1331, 2008.

[97] D. B. Kitchen, H. Decornez, J. R. Furr, and J. Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat Rev Drug Discov*, 3(11):935–49, 2004. [PubMed:15520816] [doi:10.1038/nrd1549].

[98] David Ryan Koes, Matthew P. Baumgartner, and Carlos J. Camacho. Lessons Learned in Empirical Scoring with smina from the CSAR 2011 Benchmarking Exercise. *Journal of Chemical Information and Modeling*, 2013. [PubMed:23379370] [PubMed Central:PMC3726561] [doi:10.1021/ci300604z].

[99] David Ryan Koes, Matthew P Baumgartner, and Carlos J Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of chemical information and modeling*, 53(8):1893–1904, 2013.

[100] O. Korb, T. Stützle, and T. E. Exner. Empirical scoring functions for advanced protein-ligand docking with PLANTS. *J. Chem. Inf. Model.*, 49(1):84–96, 2009. [PubMed:19125657] [doi:10.1021/ci800298z].

[101] Christian Kramer and Peter Gedeck. Leave-Cluster-Out Cross-Validation Is Appropriate for Scoring Functions Derived from Diverse Protein Data Sets. *J. Chem. Inf. Model.*, 50(11):1961–1969, 2010. [doi:10.1021/ci100264e].

[102] Mario Krenn, Florian Häse, A Nigam, Pascal Friederich, and Alán Aspuru-Guzik. Selfies: a robust representation of semantically constrained graphs with an example application in chemistry. *arXiv preprint arXiv:1905.13741*, 2019.

[103] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[104] Irina Kufareva, Andrey V Ilatovskiy, and Ruben Abagyan. Pocketome: an encyclopedia of small-molecule binding sites in 4d. *Nucleic acids research*, 40(D1):D535–D540, 2011.

[105] Denis Kuzminykh, Daniil Polykovskiy, Artur Kadurin, Alexander Zhebrak, Ivan Baskov, Sergey Nikolenko, Rim Shayakhmetov, and Alex Zhavoronkov. 3d molecular representations based on the wave transform for convolutional neural networks. *Molecular pharmaceutics*, 15(10):4378–4385, 2018.

[106] Sebastian Lapuschkin, Alexander Binder, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. The layer-wise relevance propagation toolbox for artificial neural networks. *Journal of Machine Learning Research*, 17(114):1–5, 2016.

[107] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[108] Hongjian Li, Kwong-Sak Leung, Man-Hon Wong, and Pedro J Ballester. The importance of the regression model in the structure-based prediction of protein-ligand binding. In *Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 219–230. Springer, 2014.

[109] Hongjian Li, Kwong-Sak Leung, Man-Hon Wong, and Pedro J Ballester. Correcting the impact of docking pose generation error on binding affinity prediction. *BMC bioinformatics*, 17(11):308, 2016.

[110] Hongjian Li, Kam-Heung Sze, Gang Lu, and Pedro J. Ballester. Machine-learning scoring functions for structure-based drug lead optimization. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, n/a(n/a):e1465.

[111] Jin Li, Ailing Fu, and Le Zhang. An overview of scoring functions used for protein–ligand interactions in molecular docking. *Interdisciplinary Sciences: Computational Life Sciences*, pages 1–9, 2019.

[112] E. Lindahl, B. Hess, and D. Van Der Spoel. GROMACS 3.0: a package for molecular simulation and trajectory analysis. *J. Mol. Model.*, 7(8):306–317, 2001.

[113] Shengchao Liu, Moayad Alnammi, Spencer S Ericksen, Andrew F Voter, Gene E Ananiev, James L Keck, F Michael Hoffmann, Scott A Wildman, and Anthony Gitter. Practical model selection for prospective virtual screening. *Journal of chemical information and modeling*, 59(1):282–293, 2018.

[114] Zhihai Liu, Minyi Su, Li Han, Jie Liu, Qifan Yang, Yan Li, and Renxiao Wang. Forging the basis for developing protein–ligand interaction scoring functions. *Accounts of Chemical Research*, 50(2):302–309, 2017. PMID: 28182403.

[115] Angela Lopez-del Rio, Alfons Nonell-Canals, David Vidal, and Alexandre Perera-Lluna. Evaluation of cross-validation strategies in sequence-based binding prediction using deep learning. *Journal of chemical information and modeling*, 59(4):1645–1657, 2019.

[116] Alessandro Lusci, Gianluca Pollastri, and Pierre Baldi. Deep architectures and deep learning in chemoinformatics: The prediction of aqueous solubility for drug-like molecules. *Journal of Chemical Information and Modeling*, 53(7):1563–1575, jul 2013.

[117] Alessandro Lusci, Gianluca Pollastri, and Pierre Baldi. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of chemical information and modeling*, 53(7):1563–1575, 2013.

[118] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[119] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 5188–5196. IEEE, 2015.

[120] Gilles Marcou and Didier Rognan. Optimizing fragment and scaffold docking by use of molecular interaction fingerprints. *Journal of chemical information and modeling*, 47(1):195–207, 2007.

[121] C. McInnes. Virtual screening strategies in drug discovery. *Curr Opin Chem Biol*, 11(5):494–502, 2007. [PubMed:17936059] [doi:10.1016/j.cbpa.2007.08.033].

[122] David L Mobley, Alan P Graves, John D Chodera, Andrea C McReynolds, Brian K Shoichet, and Ken A Dill. Predicting absolute ligand binding free energies to a simple model site. *Journal of molecular biology*, 371(4):1118–1134, 2007.

[123] W. T. Mooij and M. L. Verdonk. General and targeted statistical potentials for protein-ligand interactions. *Proteins*, 61(2):272–87, 2005. [PubMed:16106379] [doi:10.1002/prot.20588].

[124] John Moult. A decade of casp: progress, bottlenecks and prognosis in protein structure prediction. *Current opinion in structural biology*, 15(3):285–289, 2005.

[125] I. Muegge and Y. C. Martin. A general and fast scoring function for protein-ligand interactions: a simplified potential approach. *J Med Chem*, 42(5):791–804, 1999. [PubMed:10072678] [doi:10.1021/jm980536j].

[126] Asher Mullard. Biotech r&d spend jumps by more than 15%, 2016.

[127] M. M. Mysinger, M. Carchia, J. J. Irwin, and B. K. Shoichet. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *J Med Chem*, 55(14):6582–94, 2012. [PubMed:22716043] [PubMed Central:PMC3405771] [doi:10.1021/jm300687e].

[128] Michael M Mysinger, Michael Carchia, John J Irwin, and Brian K Shoichet. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, 55(14):6582–6594, 2012.

[129] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4467–4477, 2017.

[130] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 427–436. IEEE, 2015.

[131] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

[132] Anthony Nicholls. Confidence limits, error bars and method comparison in molecular modeling. part 1: the calculation of confidence intervals. *Journal of computer-aided molecular design*, 28(9):887–918, 2014.

[133] Ulf Norinder, Lars Carlsson, Scott Boyer, and Martin Eklund. Introducing conformal prediction in predictive modeling. a transparent and flexible alternative to applicability domain determination. *Journal of chemical information and modeling*, 54(6):1596–1603, 2014.

[134] Fedor N Novikov, Alexey A Zeifman, Oleg V Stroganov, Viktor S Stroylov, Val Kulkov, and Ghermes G Chilov. Csar scoring challenge reveals the need for new concepts

in estimating protein–ligand binding affinity. *Journal of chemical information and modeling*, 51(9):2090–2096, 2011.

[135] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[136] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison. Open Babel: An open chemical toolbox. *Journal of Cheminformatics*, 3:33, 2011. [PubMed:21982300] [PubMed Central:PMC3198950] [doi:10.1186/1758-2946-3-33].

[137] Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 3(1):33, 2011.

[138] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[139] Hakime Ozturk, Arzucan Ozgur, and Elif Ozkirimli. Deepdta: deep drug-target binding affinity prediction. *Bioinformatics*, 34:821–829, 2018.

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

[141] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[142] Javier Pérez-Sianes, Horacio Pérez-Sánchez, and Fernando Díaz. Virtual screening: a challenge for deep learning. In *10th International Conference on Practical Applications of Computational Biology & Bioinformatics*, pages 13–22. Springer, 2016.

[143] Trang Pham, Truyen Tran, and Svetha Venkatesh. Graph memory networks for molecular activity prediction. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 639–644. IEEE, 2018.

[144] Rodrigo Quiroga and Marcos A Villarreal. Vinardo: A scoring function based on autodock vina improves scoring, docking, and virtual screening. *PloS one*, 11(5):e0155183, 2016.

[145] Matthew Ragoza, Joshua Hochuli, Elisa Idrobo, Jocelyn Sunseri, and David Ryan Koes. Protein–ligand scoring with convolutional neural networks. *Journal of chemical information and modeling*, 57(4):942–957, 2017.

[146] Matthew Ragoza, Joshua Hochuli, Elisa Idrobo, Jocelyn Sunseri, and David Ryan Koes. Protein–ligand scoring with convolutional neural networks. *Journal of chemical information and modeling*, 57(4):942–957, 2017.

[147] Matthew Ragoza, Lillian Turner, and David Ryan Koes. Ligand pose optimization with atomic grid-based convolutional neural networks. *arXiv preprint arXiv:1710.07400*, 2017.

[148] Matthias Rarey, Bernd Kramer, Thomas Lengauer, and Gerhard Klebe. A Fast Flexible Docking Method using an Incremental Construction Algorithm. *J. Mol. Biol.*, 261(3):470–489, Aug 1996. [PubMed:8780787] [doi:10.1006/jmbi.1996.0477].

[149] RDKit: Open-source cheminformatics. http://www.rdkit.org. (accessed September 4, 2015).

[150] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, may 2010.

[151] Sebastian G Rohrer and Knut Baumann. Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data. *Journal of chemical information and modeling*, 49(2):169–184, 2009.

[152] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[153] Kevin Ryczko, David A Strubbe, and Isaac Tamblyn. Deep learning and density-functional theory. *Physical Review A*, 100(2):022512, 2019.

[154] T. Sato, T. Honma, and S. Yokoyama. Combining machine learning and pharmacophore-based interaction fingerprint for in silico screening. *J. Chem. Inf. Model.*, 50(1):170–185, 2009. [PubMed:20038188] [doi:10.1021/ci900382e].

[155] Leander Schietgat, Thomas Fannes, and Jan Ramon. Predicting protein function and protein-ligand interaction with the 3d neighborhood kernel. In *Discovery Science*, pages 221–235. Springer, 2015.

[156] G. Schneider. Virtual screening: an endless staircase? *Nature Reviews Drug Discovery*, 9(4):273–276, 2010. [PubMed:20357802] [doi:10.1038/nrd3139].

[157] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pages 991–1001, 2017.

[158] Kristof T Schütt, Pieter-Jan Kindermans, Huziel E Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Moleculenet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv preprint arXiv:1706.08566*, 2017.

[159] Chao Shang, Qinqing Liu, Ko-Shin Chen, Jiangwen Sun, Jin Lu, Jinfeng Yi, and Jinbo Bi. Edge attention-based multi-relational graph convolutional networks. *arXiv*, pages arXiv–1802, 2018.

[160] Yash Sharma and Pin-Yu Chen. Attacking the madry defense model with l_1-based adversarial examples. *arXiv preprint arXiv:1710.10733*, 2017.

[161] Chao Shen, Junjie Ding, Zhe Wang, Dongsheng Cao, Xiaoqin Ding, and Tingjun Hou. From machine learning to deep learning: Advances in scoring functions for protein–ligand docking. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, 10(1):e1429, 2020.

[162] Chao Shen, Ye Hu, Zhe Wang, Xujun Zhang, Jinping Pang, Gaoang Wang, Haiyang Zhong, Lei Xu, Dongsheng Cao, and Tingjun Hou. Beware of the generic machine learning-based scoring functions in structure-based virtual screening. *Briefings in Bioinformatics*, 2020.

[163] Lisa M Shewchuk, Anne M Hassell, Byron Ellis, WD Holmes, Roderick Davis, Earnest L Horne, Sue H Kadwell, David D McKee, and John T Moore. Structure of the tie2 rtk domain: self-inhibition by the nucleotide binding loop, activation loop, and c-terminal tail. *Structure*, 8(11):1105–1113, 2000.

[164] Dong Si, Spencer A Moritz, Jonas Pfab, Jie Hou, Renzhi Cao, Liguo Wang, Tianqi Wu, and Jianlin Cheng. Deep learning to predict protein backbone structure from high-resolution cryo-em density maps. *Scientific Reports*, 10(1):1–22, 2020.

[165] Jochen Sieg, Florian Flachsenberg, and Matthias Rarey. In need of bias control: evaluating chemical data for machine learning in structure-based virtual screening. *Journal of chemical information and modeling*, 59(3):947–961, 2019.

[166] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[167] Miha Skalic, Gerard Martínez-Rosell, José Jiménez, Gianni De Fabritiis, and Alfonso Valencia. Playmolecule bindscope: Large scale cnn-based virtual screening on the web. *Bioinformatics*, 2018.

[168] Miha Skalic, Alejandro Varela-Rial, José Jiménez, Gerard Martínez-Rosell, and Gianni De Fabritiis. Ligvoxel: Inpainting binding pockets using 3d-convolutional neural networks. *Bioinformatics (Oxford, England)*, 2018.

[169] Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. Computational methods in drug discovery. *Pharmacological reviews*, 66(1):334–395, 2014.

[170] Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. Computational methods in drug discovery. *Pharmacological reviews*, 66(1):334–395, 2014.

[171] Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017.

[172] Richard D Smith, Kelly L Damm-Ganamet, James B Dunbar Jr, Aqeel Ahmed, Krishnapriya Chinnaswamy, James E Delproposto, Ginger M Kubish, Christine E Tinberg, Sagar D Khare, Jiayi Dou, et al. Csar benchmark exercise 2013: evaluation of results from a combined computational protein design, docking, and scoring/ranking challenge. *Journal of chemical information and modeling*, 56(6):1022–1031, 2015.

[173] Richard D. Smith, James B. Dunbar, Peter Man-Un Ung, Emilio X. Esposito, Chao-Yie Yang, Shaomeng Wang, and Heather A. Carlson. CSAR Benchmark Exercise of 2010: Combined Evaluation Across All Submitted Scoring Functions. *J*

Chem Inf Model, 51(9):2115–2131, Aug 2011. [PubMed:21809884] [PubMed Central:PMC3186041] [doi:10.1021/ci200269q].

[174] Marta M Stepniewska-Dziubinska, Piotr Zielenkiewicz, and Pawel Siedlecki. Development and evaluation of a deep learning model for protein-ligand binding affinity prediction. *Bioinformatics*, 34(21):3666–3674, 2018.

[175] Eva Stjernschantz and Chris Oostenbrink. Improved ligand-protein binding affinity predictions using multiple binding modes. *Biophysical journal*, 98(11):2682–2691, 2010.

[176] Jocelyn Sunseri, Jonathan E King, Paul G Francoeur, and David Ryan Koes. Convolutional neural network scoring and minimization in the d3r 2017 community challenge. *Journal of computer-aided molecular design*, pages 1–16, 2018.

[177] Jocelyn Sunseri, Jonathan E King, Paul G Francoeur, and David Ryan Koes. Convolutional neural network scoring and minimization in the d3r 2017 community challenge. *Journal of computer-aided molecular design*, 33(1):19–34, 2019.

[178] Jocelyn Sunseri, Matthew Ragoza, Jasmine Collins, and David Ryan Koes. A d3r prospective evaluation of machine learning for protein-ligand scoring. *Journal of computer-aided molecular design*, 30(9):761–771, 2016.

[179] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[180] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013.

[181] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[182] Lu Tan, Hanna Geppert, Mihiret T. Sisay, Michael Gütschow, and Jürgen Bajorath. Integrating structure- and ligand-based virtual screening: Comparison of individual, parallel, and fused molecular docking and similarity search calculations on multiple targets. *ChemMedChem*, 3(10):1566–1571, oct 2008.

[183] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[184] Viet-Khoa Tran-Nguyen, Célien Jacquemard, and Didier Rognan. Lit-pcba: An unbiased data set for machine learning and virtual screening. *Journal of Chemical Information and Modeling*, 2020.

[185] O. Trott and A. J. Olson. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J Comput Chem*, 31(2):455–61, 2010. [PubMed:19499576] [PubMed Central:PMC3041641] [doi:10.1002/jcc.21334].

[186] Oleg Trott and Arthur J. Olson. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 9999(9999):NA, 2009. [PubMed:19499576] [PubMed Central:PMC3041641] [doi:10.1002/jcc.21334].

[187] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

[188] Gregor Urban, Niranjan Subrahmanya, and Pierre Baldi. Inner and outer recursive neural networks for chemoinformatics applications. *Journal of chemical information and modeling*, 58(2):207–211, 2018.

[189] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[190] O Anatole Von Lilienfeld, Raghunathan Ramakrishnan, Matthias Rupp, and Aaron Knoll. Fourier series of atomic radial distribution functions: A molecular fingerprint for machine learning models of quantum chemical properties. *International Journal of Quantum Chemistry*, 115(16):1084–1093, 2015.

[191] Izhar Wallach, Michael Dzamba, and Abraham Heifets. Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*, 2015.

[192] Izhar Wallach and Abraham Heifets. Most ligand-based classification benchmarks reward memorization rather than generalization. *Journal of chemical information and modeling*, 58(5):916–932, 2018.

[193] J Wang, W Wang, PA Kollman, and DA Case. Automatic atom type and bond type perception in molecular mechanical calculations. *J Molec Graph Model*, 25:247–260, 2006. [PubMed:16458552] [doi:10.1016/j.jmgm.2005.12.005].

[194] Jui-Chih Wang and Jung-Hsin Lin. Scoring functions for prediction of protein-ligand interactions. *Current pharmaceutical design*, 19(12):2174–2182, 2013.

[195] R. Wang, L. Lai, and S. Wang. Further development and validation of empirical scoring functions for structure-based binding affinity prediction. *J. Comput.-Aided Mol. Des.*, 16(1):11–26, 2002. [PubMed:12197663].

[196] R. Wang, L. Liu, L. Lai, and Y. Tang. SCORE: A New Empirical Method for Estimating the Binding Affinity of a Protein-Ligand Complex. *J. Mol. Model*, 4:379–394, 1998.

[197] R. Wang, Y. Lu, and S. Wang. Comparative evaluation of 11 scoring functions for molecular docking. *J Med Chem*, 46(12):2287–303, 2003. [PubMed:12773034] [doi:10.1021/jm0203783].

[198] G. L. Warren, C. W. Andrews, A. M. Capelli, B. Clarke, J. LaLonde, M. H. Lambert, M. Lindvall, N. Nevins, S. F. Semus, S. Senger, G. Tedesco, I. D. Wall, J. M. Woolven, C. E. Peishoff, and M. S. Head. A critical assessment of docking programs and scoring functions. *J Med Chem*, 49(20):5912–31, 2006. [PubMed:17004707] [doi:10.1021/jm050362n].

[199] Donglai Wei, Bolei Zhou, Antonio Torrabla, and William Freeman. Understanding intra-class knowledge inside cnn. *arXiv preprint arXiv:1507.02379*, 2015.

[200] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[201] Robin Winter, Floriane Montanari, Frank Noé, and Djork-Arné Clevert. Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chemical science*, 10(6):1692–1701, 2019.

[202] Maciej Wójcikowski, Pedro J Ballester, and Pawel Siedlecki. Performance of machine-learning scoring functions in structure-based virtual screening. *Scientific Reports*, 7:46710, 2017.

[203] Marciej Wojcikowski, Michal Kikielka, Marta M Stepniwska-Dziubinska, and Pawel Siedlecki. Development of a protein-ligand extended connectivity (plec) fingerprint and its application for binding affinity predictions. *Bioinformatics*, bty757, 2018.

[204] Nobuaki Yasuo, Keisuke Watanabe, Hideto Hara, Kentaro Rikimaru, and Masakazu Sekijima. Predicting strategies for lead optimization via learning to rank. *IPSJ Transactions on Bioinformatics*, 11:41–47, 2018.

[205] Shuangye Yin, Lada Biedermannova, Jiri Vondrasek, and Nikolay V. Dokholyan. MedusaScore: An accurate force field-based scoring function for virtual drug screening. *Journal of Chemical Information and Modeling*, 48(8):1656–1662, aug 2008.

[206] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

[207] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.

[208] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[209] Wei Zhao, Kirk E Hevener, Stephen W White, Richard E Lee, and James M Boyett. A statistical framework to evaluate virtual screening. *BMC bioinformatics*, 10(1):225, 2009.

[210] Hongyi Zhou and Jeffrey Skolnick. GOAP: a generalized orientation-dependent, all-atom statistical potential for protein structure prediction. *Biophys. J.*, 101(8):2043–52, October 2011. [PubMed:22004759] [PubMed Central:PMC3192975] [doi:10.1016/j.bpj.2011.09.012].

[211] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.

[212] David Zilian and Christoph A Sotriffer. Sfcscore rf: a random forest-based scoring function for improved affinity prediction of protein–ligand complexes. *Journal of chemical information and modeling*, 53(8):1923–1933, 2013. [PubMed:23705795] [doi:10.1021/ci400120b].