Title:

Improving the Accuracy of Protein-Ligand Binding Affinity Prediction by Deep Learning Models: Benchmark and Model

Authors:

Mohammad A. Rezaei[1,2 #], Yanjun Li[3 #], Xiaolin Li[3,*], and Chenglong Li[1,*]

[1] Department of Chemistry, University of Florida, Gainesville, FL, USA.

[2] Department of Medicinal Chemistry, Center for Natural Products, Drug Discovery and Development (CNPD3), University of Florida, Gainesville, FL, USA.

[3] Large-scale Intelligent Systems Laboratory, NSF Center for Big Learning, University of Florida, Gainesville, FL, USA.

[#] These authors contributed equally.

[*] Corresponding authors: lic@cop.ufl.edu, andyli@ece.ufl.edu

Keywords:

Binding affinity, deep learning, PDBbind, Binding MOAD.

ABSTRACT

***Introduction*:** The ability to discriminate among ligands binding to the same protein target in terms of their relative binding affinity lies at the heart of structure-based drug design. Any improvement in the accuracy and reliability of binding affinity prediction methods decreases the discrepancy between experimental and computational results.

***Objectives*:** The primary objectives were to find the most relevant features affecting binding affinity prediction, least use of manual feature engineering, and improving the reliability of binding affinity prediction using efficient deep learning models by tuning the model hyperparameters.

***Methods*:** The binding site of target proteins was represented as a grid box around their bound ligand. Both binary and distance-dependent occupancies were examined for how an atom affects its neighbor voxels in this grid. A combination of different features including ANOLEA, ligand elements, and Arpeggio atom types were used to represent the input. An efficient convolutional neural network (CNN) architecture, DeepAtom, was developed, trained and tested on the PDBbind v2016 dataset. Additionally an extended benchmark dataset was compiled to train and evaluate the models.

***Results*:** The best DeepAtom model showed an improved accuracy in the binding affinity prediction on PDBbind *core* subset (Pearson's R=0.83) and is better than the recent state-of-the-art models in this field. In addition when the DeepAtom model was trained on our proposed benchmark dataset, it yields higher correlation compared to the baseline which confirms the value of our model.

***Conclusions*:** The promising results for the predicted binding affinities is expected to pave the way for embedding deep learning models in virtual screening and rational drug design fields.

1. INTRODUCTION

Molecular recognition is key in interactions between biological molecules involved in the cellular processes. Proteins are the machinery that accomplish diverse molecular and cellular functions. Modulating the function of proteins by small molecules has been an active research area with applications in drug design and development. To quantify the binding of a ligand to its target protein, a commonly used measure is their binding affinity which describes how strong the ligand binds to its biological counterpart. This binding strength can be measured experimentally by Surface Plasmon Resonance (SPR), Isothermal Titration Calorimetry (ITC), and Fluorescence Polarization (FP) methods [Willander2009; Du2016].

Computational methods for calculation of binding affinity range from rough estimates as in molecular docking, to more rigorous force fields in molecular dynamics (MD) simulations and Quantum Mechanical (QM) calculations [Huey2007; Gilson2007; deAzevedo2008; Kim2008; Huang2010; Wang2013; Montalvo-Acosta2016; Hernández-Rodríguez2016; Ryde2016]. According to Liu and Wang, the scoring functions for protein-ligand interactions can be broadly put into four categories: **a.** *physics-based*, a.k.a. force field-based methods which rely on physical meaning; **b.** *empirical*, a.k.a. regression-based methods which aim to find the best weights for the features likely to contribute to binding energy; **c.** *knowledge-based*, a.k.a. potential of mean force-based methods which are of statistical nature and assume statistical preference leads to (un)favorable interactions; **d.** *descriptor-based*, a.k.a. machine learning-based methods which are based on a large pool of descriptors and a machine learning algorithm decides which features to keep [Liu2015].

In recent years, there has been a growing interest in the application of machine learning methods to predict the protein-ligand binding affinity [Ballester2010; Ballester2014; Li2014a; Ashtawy2015a; Ashtawy2015b; Khamis2015; Ain2015; Wójcikowski2017; Heck2017; Li2017; Li2018]. Apart from technological advances in hardware and software industry that has made the use of machine learning

algorithms feasible and widespread, one key justification here is that they reduce the bias in feature engineering. The conventional scoring functions developed for molecular docking and molecular dynamics simulations include terms completely based on the domain knowledge of their developers. Though such scientific insight is invaluable, it may be potentially biased especially in feature engineering, i.e. the terms to include in the force field and their parametric additive functional form [Li2014a]. This feature engineering tends to make the force fields more complex in order to reproduce experimental results more accurately. To showcase this complexity, the SFCscore developed by the Scoring Function Consortium (SFC) includes 66 tailored features mostly being of physicochemical nature [Sotriffer2008]. On the contrary, this feature engineering can be highly reduced in nonparametric machine learning methods where the choice of features and their weights is delegated to the algorithm, with no *a priori* assumption about the form of functional [Ain2015]. The features then can be as simple as the count of protein-ligand atom pairs for different atom types at different distance ranges [Ballester2010; Ballester2014; Ain2015; Wójcikowski2017].

Convolutional Neural Networks (CNNs) are a machine learning technique gaining recent popularity, although their roots can be traced back to Neocognitron in 1980 [Fukushima1980], a neural network model suggested for translation-invariant pattern recognition, i.e. being able to recognize features regardless of their position in the input. In contrast to a fully connected neural network where the nodes in each layer are connected to all nodes in the previous layer, a convolutional layer in a CNN is composed of nodes each connected to a typically small region in its input, called its *receptive field*. This enables the model to learn local features. In order to offer translation-invariance, the nodes should also share their weights. Together, use of receptive fields and weight sharing allows a convolutional layer recognize local patterns mostly independent of where they are located in the input. The second essential type of layer often used right after a convolutional layer is a *subsampling* layer. Through downsampling its input, this layer generates a lower resolution input of smaller size, so it diminishes the effect of small changes in the input on the network's output.

An important feature of CNNs is channels. In image processing, different combinations of the three colors red, green, and blue (RGB) are used to generate all other colors. Similar to RGB channels, using channels in a CNN allows the network to have multiple different views of a single input. This channel information will later be combined to provide a more complete view of the input. In practice, atom types and atom properties have been used to define different channels in a molecule. Ragoza and coworkers used *smina* atom types as channels in their CNN model; it included 34 distinct atom types with 16 protein types and 18 ligand types, where discrimination was made between ligand and protein types [Koes2013; Ragoza2017]. Jiménez *et al.* defined rules for atom properties based on AutoDock 4 atom types. As channels, they included hydrophobic, aromatic, hydrogen-bond acceptor, hydrogen-bond donor, positive ionizable, negative ionizable, metal, and excluded volume properties [Jiménez2017]. The present research aims to improve the binding affinity prediction by tuning several aspects of the modeling, from representation, to model architecture, and compilation of a more inclusive dataset used for training and evaluation of the models.

A comparative analysis in 2017 reported that several cheminformatics and machine learning methods can achieve experimental accuracy in binding affinity prediction, i.e. mean absolute error (MAE) less than 1 kcal/mol [Gomes2017]. This is a promising result compared to the 2-3 kcal/mol standard error in binding affinity prediction reported for AutoDock 4 as one of the most popular docking packages [Morris2009]. It is expected that machine learning methods will improve the accuracy even more, considering that they may mitigate the bias caused by functional form assumption. If they reach that level of accuracy and reliability, they can be a valuable tool chained to virtual screening methods and boost the hit enrichment, leading to better computational drug design efforts.

2. METHODS

The workflow for prediction of binding affinity needs to preprocess the input data and train a network on it. Details of the data and model components are discussed separately below.

2.1. Data

Datasets: The PDBbind dataset as well as Binding MOAD database are arguably the largest collections of protein-ligand binding affinity data; they also include experimental data about other biological entities such as nucleic acids [Wang2005; Liu2017; Hu2005; Ahmed2015]. The PDBbind data consists of three subsets, namely *general*, *refined*, and *core*. The *general* subset includes any experimentally resolved complex structure with reported binding affinity; as a result, it contains complexes with various levels of quality, e.g. high and low resolution structures, as well as binding affinity values in terms of K_d, K_a, K_i, and IC_{50}. Applying several filtration steps, the developers had reduced this dataset to *refined* subset; it contains protein-ligand structures at higher resolution and excludes any complexes with IC_{50} affinity data only. To benchmark different scoring functions for protein-ligand binding affinity prediction, the PDBbind *refined* subset has turned into the standard training dataset. Finally based on 90% sequence similarity threshold, the complexes in *refined* subset have been clustered into 58 protein families, with five representative complexes each to better span the binding affinity range. The resulting *core* subset is often used either as test or validation dataset. In the PDBbind v2016 dataset, the non-overlapping portions of the three subsets *general*, *refined*, and *core* contain 9,228, 3,767, and 290 complexes, respectively [Liu2017].

In addition to using the PDBbind v2016 dataset, we compiled an extended benchmark dataset. Two complementary resources were used in compilation of our benchmark dataset, namely PDBbind v2016 and v2018 [Liu2017] and Binding MOAD [Hu2005; Ahmed2015]. All complexes which belong to any subset of PDBbind v2016 or v2018 were included if their binding affinity data was reported as

K_d, K_a, or K_i. We excluded complexes with IC_{50} data only. In the case that the binding affinity of a complex was reported as K_d or K_a in one dataset and as K_i in the other, we kept the more reliable K_d / K_a data. Complexes with structural or binding affinity data issues were filtered out; for instance, the two complexes with PDB IDs 2W9I and 4KG1 were obsolete and superseded by newer complexes, so they were removed from the benchmark dataset. The two complexes 1CAM and 1D2V existed in the index file for general subset, but their structures were absent from the dataset; the corresponding structures were generated in those cases. For Binding MOAD dataset, several filters were applied. If a complex included a peptide or ion as its ligand, it was not considered. Similar to PDBbind filtering, only K_d, K_a, and K_i were accepted for binding affinity data. These steps resulted in 10,673 complexes. We will refer it as the benchmark 2018 dataset in the remainder of the text. We provide the PDB codes and pK values for complexes in this extended dataset (Supplementary Materials) as Comma-Separated Values (CSV) format.

Data Types: For the sake of binding affinity prediction, not all data are of the same reliability. The dissociation constant K_d and the association constant K_a directly relate to the protein-ligand binding energy and are universal binding measures. On the other hand, the inhibition constant K_i is a thermodynamic constant defined for enzymes only, and is a less reliable binding affinity measure compared to K_d / K_a. Among the four types of binding affinity quantities, the half maximal inhibitory concentration IC_{50} has the lowest reliability to be converted into binding affinity energy; this can be attributed to the dependence of IC_{50} on the concentrations of both protein and ligand. The general equation for IC_{50} is defined as:

$$IC_{50} = \frac{[E]_0}{2} + K_i^{(app)}$$

where $[E]_0$ represents the initial enzyme concentration and $K_i^{(app)}$ denotes the apparent K_i. The inhibition mechanism for an enzyme can be competitive, noncompetitive, uncompetitive, or mixed-type, which is represented by different terms in the apparent K_i value. Indeed, $K_i^{(app)}$ often depends on the ligand

concentration unless it follows noncompetitive mechanism [Cha1975]. Therefore the mapping from IC_{50} to binding energy by a computational model requires not only the structures of protein and ligand, but also the knowledge of inhibition mechanism. Additionally, K_d and K_i are equilibrium constants which may be compared if derived from multiple binding assays; however the dependence of IC_{50} values on experimental settings discourages its comparison across different assays [Li2014b]. This is why the training data for machine learning algorithms often excludes complexes with merely IC_{50} data.

Data Splitting: The data used to train and test a model determines whether different models can be compared. This is called "splitting" the dataset into train and test subsets. Unfortunately there is no consistency in the literature on how to split the datasets. This makes the comparison of models irrelevant, for example when the training set of two models is different, even when the models are tested on the same dataset. Random splitting of the datasets is commonly used, which also leads to variances in the output. Either the names of samples used in training should be reported or the splitting source code along with the random seed should be available; otherwise we should take into account a margin for differences in the model performance due to the variances in the training data. To extensively evaluate the performance of our DeepAtom model, we follow three different splitting schemes: a) following the splitting used by Jiménez *et al.* [Jiménez2018]; b) splitting the PDBbind v2016 dataset into training, validation, and test subsets; c) compiling a larger benchmark dataset and splitting it into training, validation, and test subsets. These schemes are discussed in more detail below.

We initially followed the splitting suggested by Jiménez *et al.* [Jiménez2018] using the PDBbind 2016 *refined* subset (excluding *core* subset) as the training set and the *core* subset to test the model performance. The main issue with their splitting is that it does not follow the standard scheme which divides the available data into training, validation, and test data; this recommended practice trains a model on training data only, while its hyperparameters are optimized based on the validation data. Different reasonable values are tried for each hyperparameter; later optimization steps will fix the hyperparameter at the value which yields highest performance for the model on the validation data.

Once the hyperparameters are all tuned, the best model is then evaluated on unseen test data. This best practice is not followed in the K_{DEEP} paper; their model is trained and tested on PDBbind v2016 refined and core subsets, respectively, without a validation set. This is followed by evaluating their K_{DEEP} model on additional test datasets [Jiménez2018]. We believe using this splitting effectively turns the core subset into the validation set, and the additional test sets are the true indicators of K_{DEEP}'s performance compared to the other models, where RF-Score model shows superior performance compared to K_{DEEP} model in all four additional test sets, and is either the best model or among the top two compared to the other models [Jiménez2018]. Therefore in our second data splitting we first combined the *core* and *refined* subsets of the PDBbind v2016 datasets, and then divided it into train-validation-test sets using 80-10-10 random splitting. Finally the same splitting scheme was followed for our compiled benchmark 2018 dataset to obtain train, valid, and test sets. Our baseline in the second and third splittings was RF-Score, because of its superior performance and its being open source.

Data Representation: Two major approaches have been reported when dealing with molecules as 3D input to a model: atom-centered representation and grid-based representation. The former represents the molecule as atoms whose neighbor lists are compiled based on distance and atom type of the neighbors for each atom [Schietgat2015; Gomes2017]. The latter generates a grid either around the whole molecule or the area we are interested in, e.g. the binding site of an enzyme. The contribution of atoms to each neighbor voxel is then calculated [Ragoza2017; Jiménez2017; Torng2017]. The data conventionally used in describing the structural data as well as the interactions between a protein and a small molecule ligand is in the tabular form. Often, there is no or small relations between the adjacent columns in this input data. This limits the applicability of advanced algorithms on such trivial data. As an example, spatial data is lost when an image is converted from a two dimensional matrix into a one-dimensional vector. This is the case for a fully-connected artificial neural network (ANN). To harness the power of deep learning models more efficiently, “representation” of the interacting molecules is the key. This representation of a molecule needs to encode the presence of neighboring atoms around a

reference atom. Such an encoding is often distance-dependent, although binary encoding has also been suggested [Ragoza2017].

Atom Types: In addition to the two atom types used by [Ragoza2017] and [Jiménez2017], two other atom types schemes have been defined but not yet used in deep learning models; they are ANOLEA [Melo1997; Melo1998] and Arpeggio [Jubb2017] atom types. For the standard amino acids, ANOLEA defines 40 different heavy atom types based on bond connectivity, chemical nature, and location level of the atom, i.e. whether it belongs to side-chain or backbone of the residue. This typing scheme was suggested to be used in structure prediction and fold recognition [Melo1997]. Arpeggio scheme used SMARTS-based atom types similar to the properties used by Jiménez *et al.* based on AutoDock 4 atom types [Jiménez2017]. This scheme is built on CREDO atom types and improves some of its type definitions [Schreyer2009; Schreyer2013]. The 11 features generated by Arpeggio include H-bond donor/acceptor, weak H-bond donor/acceptor, halogen bond, positive/negative ionizable, hydrophobic, carbonyl oxygen/carbon, and aromatic [Jubb2017]. Figure 2 depicts how features are extracted from a protein-ligand complex in a grid-based representation.

Before generating the channels, all datasets were preprocessed to ensure fewer problems in later steps, as well as to improve the data reliability. This included checking for atoms in the binding site which did not belong to the standard atom types used. Also alternate location “A” was picked in the case an atom occupied multiple alternate locations in the PDB file.

2.2. Model

Model Architecture: To give a better overview, some of the architectural ideas are briefly discussed here. The model benefits from residual modules. These units include direct links between input and output of residual modules as bypass, so that the unit learns features with reference to its input, rather than learning an unreferenced mapping. This architectural idea has proved to improve the model reliability; it alleviates the overfitting problem as well as the “vanishing gradients” issue where

the back-propagated error signal is too small when it reaches the initial layers so it cannot train them efficiently [He2015]. Batch normalization was applied to the output of residual modules to stabilize the model; this transformation normalizes each mini-batch by its mean and variance [Ioffe2015]. The model also makes use of two filter types: 1. *Depthwise filters* which still have a receptive field greater than 1x1, but only apply at individual channels or depths; 2. *Pointwise filters* which are the opposite of depthwise filters, i.e. they have 1x1 receptive field spanning multiple channels. Together these two types of filters replace the standard convolutional filters in each block, and are called depthwise separable convolutions [Howard2017]. Because the pointwise 1x1 filters are a bottleneck, they are replaced by *pointwise group* convolution filters followed by a channel shuffle layer. While pointwise convolution applies to all channels for each single point in the input, the *group convolution* layer reduces the computational cost by splitting the channels into groups; only intra-group channels are applied to each input point. To reach higher accuracy, the channels from different groups are later shuffled, so that the feature maps get a blend of different channel groups [Zhang2017; Ma2018].

Based upon the mentioned ideas, we developed a convolutional neural network to predict binding affinity values from the holo structure of the ligand-bound protein. The schematic in Figure 3 illustrates the main blocks in the model. The first block integrates information from atoms in the grid box. The second block is employed to extract features, and the third block predicts the binding affinity by taking average over the outputs.

Model Hyperparameters: They refer to parameters whose values are set by the developer before running the model, and not optimized by the model itself. The hyperparameters included grid box size, number of channels, grid resolution, occupancy type, and whether to take the average over augmented input during test. We tuned the model hyperparameters one at a time on a range of reasonable values. This iterative process was followed in order to improve the model's performance. Due to the long training time of the networks and the large number of hyperparameters, developing models with all combinations of hyperparameters is infeasible; therefore it is often assumed that the hyperparameters

are independent [Ballester2014]. Based on this assumption, one hyperparameter can be tuned while others get fixed at their reasonable values. In the following, further details are provided for each hyperparameter.

Grid box size: Because we planned to represent the 3D data on a grid, an initial step was to determine a reasonably large box around the ligand in the binding site. While the first choice could be to follow the recommended values in the literature, e.g. a cube with 24 Å sides [Ragoza2017; Jiménez2018], we analyzed the distribution of end-to-end distances for all ligands in the PDBbind v2016 *core* and *refined* subsets [Wang2005]. This gave us clue to define a larger box size of 32 Å. This is the same as the end-to-end distance for the longest ligand in these two datasets, so there is no need to filter out any. The distribution of ligand lengths in these two subsets is illustrated in Fig. 1. It can be argued that, even if a ligand is larger than the box size, it may fit into the box if its longest axis is not parallel to the grid box planes. However, keep in mind that grid-based deep learning models almost always use augmented data by applying translation and rotation to their original input; therefore it is very likely that the ends of large ligands be cropped by the grid box if its size is not defined large enough.

Atom types: We tried different combinations of atom types for protein and ligand atoms. This was intended to see if different complementary features would improve the model's ability to predict the binding affinity. On the one hand, we described the protein atoms by 40 ANOLEA atom types [Melo1997; Melo1998] and the ligand atoms by the 9 elements (C, N, O, F, P, S, Cl, Br, I); one the other hand, both protein and ligand atoms were also described by 11 Arpeggio atom types [Jubb2017], the most significant ones being H-bond acceptor, H-bond donor, positive / negative ionizable, hydrophobic, and aromatic properties. This combination provided 60 atom types to describe the atoms in the complex. Another choice for the featurization was to use 24 atom types; it describes protein and ligand by 11 Arpeggio atom types and one excluded volume channel, and discriminates between

protein and ligand channels, i.e. 2 x (11 + 1). Models were developed based on either 11 Arpeggio types, the whole 60 atom types, or the 24 channels.

Grid resolution: The *van der Waals* radius of the mentioned 9 heavy atoms is equal to or greater than 1.5 Å; hence the resolution of the individual grid voxels must be smaller that this value in order to differentiate two atoms from each other. Models were developed with grid resolution values of 1.0, 0.5 and 0.375 Å. The 0.375 Å is taken from AutoDock 4, the popular molecular docking package [Morris2009]. While a finer resolution may enhance the model performance, it is also expected to drastically increase the computational cost because the input size scales with N^3 where N is the number of voxels at each (X, Y, Z) direction. Consequently there is a trade-off between performance and computational cost. The optimized value for this hyperparameter is not necessarily the one yielding the highest performance, rather the value which, on balance, optimizes both performance and computational cost.

Occupancy types: How an atom affects its neighborhood in a grid is determined by its occupancy type. We compared binary and distance-dependent occupancy types; in the former, a voxel gets 1 if its center overlaps with an atom's *van der Waals* radius $\mathbf{r}_{vdw}$ in a specific channel, and zero otherwise. This gives rise to a sparse representation. As an example, one of the most dense ANOLEA channels corresponds to the backbone alpha carbon for all protein residues except Glycine, around the binding site of a complex. In the complex with PDB ID 1A30, only 0.07 of the total voxels overlap with the backbone $\mathbf{C_\alpha}$ atoms of this channel. On the other hand, in the distance-dependent scheme the contribution of each atom to its nearby voxels depends on the distance between them, as described below [Jiménez2017]:

$$n(r) = 1 - \exp\left(-\left(\frac{r_{vdw}}{r}\right)^{12}\right)$$

where $\boldsymbol{r}$ represents the distance between a neighbor grid point and the atom with *van der Waals* radius of $\boldsymbol{r}_{vdw}$.

Mitigate Overfitting: An issue that appears with various machine learning tasks is overfitting. It is similar to *remembering* versus *learning*. In simple words, it occurs when the model has learned not only those features of the input data necessary for the prediction task, but also the non-relevant features and details. This helps a model improve its accuracy on the training data, but will reveal a large gap in model's accuracy when evaluated on independent data not used in its training. Different strategies can be applied to alleviate this problem. Among the most straightforward ways is to augment the data itself. To implement this strategy, we generated random rotations of the ligand with respect to the binding site grid box. Furthermore we allowed a small translation of the ligand, up to 1.0 Å in an arbitrary direction, from the grid box center. Applying both translation and rotation, up to 36 grid boxes were sampled for each original binding site input. The second modification that helps decrease overfitting in a model is the use of dropout strategy. With a given dropout probability, the output from each of the nodes in the dropout layer will be given a weight of zero, effectively removing that node from that training cycle. However the nodes in the dropout layer are always present during test time, and their weights needs to get multiplied by the same dropout probability. This makes the model resistant to the loss of its connections and at each forward pass generates a potentially different architecture. This is in nature similar to running an ensemble of models because each epoch sees a different network architecture. Use of a dropout layer has therefore been reported to prevent a model from overfitting [Srivastava2014; Goodfellow2016]. An additional strategy to reduce overfitting is the use of regularization. It is implemented by introducing a penalty term in the loss function; this will make a balance in the model's level of complexity, i.e. the new loss function disfavors both too simple models which yield high error as well as too complex models which lead to overfitting and yield high penalty. We added L2 regularization to all variables in our models except the biases.

2.3. Analysis of Results

Enrichment Analysis: The test set in PDBbind v2016 dataset includes complexes from core and refined subsets, while the samples in the benchmark 2018 dataset may belong to either of the core,

refined, or general subsets of PDBbind v2018, or Binding MOAD dataset. In order to examine the effect of data source on predictions, enrichment analysis was carried out considering only PDBbind subsets in the test data. Enrichment factor was defined as the ratio of the observed percentage of complexes divided by the expected percentage of complexes across each PDBbind subset, for the 5%, 10%, and 20% most accurate predictions. The same analysis was performed using both DeepAtom and RF-Score models.

Analysis of Prediction Errors: Contrary to enrichment analysis, the *largest* prediction errors were analyzed across the PDBbind subsets. The error factor was defined similar to the enrichment factor, but for the 5%, 10%, and 20% largest prediction errors.

3. RESULTS AND DISCUSSION

The binding affinity of a protein-ligand complex is defined as the difference between the free energy of complex and the sum of free energies of protein and ligand in their free form. The standard way to formulate this measure has been to split the contributing phenomena into short-range and long-range interactions and then to define their parts in a functional form. This requires extensive feature engineering; as an example, refer to [Sotriffer2008] where 66 expert-defined features constitute the SFCscore objective function. On the contrary, deep learning models avoid the functional form assumption and mostly rely on the data itself for feature generation and extraction.

3.1. Model Architecture

The model consisted of three blocks, starting with fusing the atom information across different channels using a pointwise (PW, 1x1x1) convolution layer. To decrease the input dimension and provide more translational invariance for the features, a 3D max pooling layer followed, leading to an output size of 16x16x16. To extract features from the previous subsampling layer, the second block employed multiple consecutive 3D *shuffle* units to hierarchically extract the latent features. Three groups of these units were used in this architecture based on how many channels they had in their output. Similar to residual networks, the input of feature channels were split equally by a channel split operator at the beginning of the unit. While one branch was sequentially processed by a pointwise convolution, a 3x3x3 depthwise (DW) convolution and another pointwise convolution, the other identity branch got concatenated with the output from the first branch. The blending of information across the two branches was achieved by the channel shuffle operation. More specifically, the feature maps in each of the branches were divided into subgroups, followed by mixing of the branches with different subgroups. For these groups, downsampling layer occured in only the first shuffle groups, whereas the input dimensions was kept the same in the remainder of units. Subsequently after the three

shuffle groups stacked, the original 3D data included 3 grids along x, y, z axes and 1024 channels. The model's third block consisted of 8 vectors (2x2x2) with dimension 1024. These vectors then fed into a shared-weight fully connected layer, thus yielding the regression loss to train the network.

The leaky rectified linear unit (leaky ReLU) was adopted as the activation function. Speeding up of training was achieved by a batch normalization layer (batch size: 128) appended after each convolution operation. The mean squared error (MSE) was set as our affinity regression loss for model learning.

3.2. Model Refinement

The initial hyperparameter values in training the DeepAtom model were: number of channels: 11; grid resolution: 1.0 Å; occupancy type: binary; and no averaging during test time. Each step of hyperparameter tuning is explained in more details below. Table 1 reports the performance of the DeepAtom model in terms of Pearson's correlation coefficient between the predicted and experimental pK values when we optimize the hyperparameter values one by one. In addition, Figure 4 illustrates the impact of optimizing each hyperparameter.

***Grid box size*:** We defined a box size of 32 Å based upon our analysis of the end-to-end distance for the ligands in our training dataset. Figure 1 illustrates the distribution of ligand end-to-end distances for the PDBbind v.2016 *refined* and *core* subsets. Among these two subsets as the train and test datasets, 59 of them are longer than the 24 Å box size defined in a recent work by Jiménez *et al.* [Jiménez2018]. This issue gets more serious when data augmentation is applied; it is possible that the too long ligands have their most distant atom pair not originally parallel to the box side plane and could fit in the box, but these atom pairs are more likely to get out of the box after rotation and translation. It should also be remarked that the 24 Å box size may not be enough even with ligands with shorter end-to-end distance, because it may not include the interacting residues on the protein side. The threshold for Hydrogen bond, aromatic, and ionic interactions is about 4 Å, and the cutoff for hydrophobic

interactions is 4.5 Å [Jubb2017], so this cutoff distance should also be taken into account when defining the grid box to accommodate the residues surrounding the ligand.

***Grid channels*:** Representation of the input and type of channels is expected to largely influence a model's performance. We trained models on three different set of features, including 11, 24, and 60 features. The former describes both protein and ligand atoms with the same 11 Arpeggio atom types; the second one discriminates between protein and ligand atom types, with an additional "excluded volume" feature that includes all ligand or protein atoms, thus resulting in 2 x (11 + 1) = 24 features; finally the 60 features include 40 ANOLEA atom types to describe protein atoms, 9 element types to describe ligand atoms, and 11 Arpeggio atom types to describes both protein and ligand atoms. Comparison of the results indicated that, by not differentiating between protein and ligand atoms, 11 features are unable to make a reliable prediction. This is expected and can be explained by the binding affinity definition. The binding affinity energy is defined as the energy of complex minus the sum of energies of the free protein and free ligand. In the case of PDBbind as the training dataset, only the *holo* structure of protein is known and thus the *apo* protein structure as well as the ligand are extracted from the complex. The underlying assumption is that the protein conformational change is negligible and the extracted protein structure can represent the *apo* structure. If this assumption holds, we can continue with $\Delta G_{complex} - (\Delta G_{protein} + \Delta G_{ligand})$. Because the intramolecular interactions are the same for the extracted protein and the holo protein structure, they will cancel out. The same holds true for the ligand. As a result, when the free structures of protein and ligand are inferred from their complex structure, the only contribution to binding energy comes from the intermolecular interactions between the protein and ligand atoms. The model trained with 11 features ignores the distinction between protein and ligand atoms and is doomed to show relatively low performance. The model trained with 24 features mitigates the mentioned issue and achieves the highest performance among the three sets of features. Finally adding the ANOLEA protein types and ligand element types indeed lowered the

performance. This is consistent with the report by Ballester *et al.* [Ballester2014] who performed a systematic study and concluded that a more precise description of a protein-ligand complex does not necessarily lead to better prediction accuracy.

***Atom occupancy type*:** Various functions have been suggested to describe how much an atom influences its surrounding environment. These include binary, Gaussian [Ragoza2017] and pairwise correlation [Jiménez2017; Jiménez2018]. The binary occupancy does not depend on the distance between the atom center and the voxel center as long as this distance is shorter than the atom's van der Waals radius. In contrast, the Gaussian and pairwise correlation schemes are distance dependent. To see the impact of occupancy type on the model performance, models were trained using either binary or pairwise correlation occupancy. The results are in favor of the richer description offered by the pairwise correlation. While we did not test the Gaussian occupancy, we expect it to achieve similar favorable results, although the calculation needed for Gaussian occupancy is slightly more computationally expensive.

We would like to elaborate on the feature extraction step. It may be argued that the deep learning models are supposed to automatically extract useful features with little or no human intervention; on the other hand, our initial featurization step is based on domain knowledge of the biological interactions. While this is true, let us draw the reader's attention to how deep learning models function in computer vision applications such as image classification and segmentation. In these applications, the content of an input 2D image is quantified by the values of the three Red-Green-Blue (RGB) channels; then a deep learning model automatically extracts features from these values. Using this analogy, we need a similar input prior to automatic feature extraction. The input structures are first represented using different combinations of channels (in our work: 11, 24, and 60, similar to RGB channels) and different occupancy types. In image recognition applications, there is no need to take the last step, i.e. each pixel does not impact its adjacent pixels which could blur the image. In contrast, the occupancy is considered in our case based on the electron cloud distribution concept in chemistry; in a

simplified manner, the occupancy type defines how much each protein / ligand atom affects its surroundings. By considering both channels and occupancy effects, we prepare the input structure for automatic feature extraction. To summarize, the initial step (including channels and occupancy types) is more related to data representation, rather than feature extraction. The same holds true for related grid-based works in this field [Ragoza2017; Jiménez2018].

***Grid resolution*:** There is a trade-off between performance and computation time depending on the grid resolution. Three different values were tested for resolution: 0.375, 0.5, and 1.0 Å. The grid data resulted from the former is about 20 times larger in size than the representation with 1.0 Å resolution as it scales proportionate to N^3. The DeepAtom models trained on grid data with 0.375, 0.5, and 1.0 Å resolution respectively yielded Pearson's correlation coefficients of 0.79, 0.78, and 0.78 between the predicted and experimental pK values (see Table 1). The slight improvement in model performance comes with a large computational cost, so we picked the 1.0 Å resolution as the optimal value for this hyperparameter.

***Data augmentation at training and test time*:** It is desirable to train the model on a larger dataset, especially for a deep learning model. Nonetheless, the number of protein-ligand complexes with available experimental binding affinity data is limited. One solution is to expand the dataset by applying transformations such as random translation and rotation to each of the samples. Preliminary results revealed the need to always use augmented data during the training stage. We generated 36 such augmented samples for each original complex in the training dataset. Although each epoch in the absence of augmented data took less time due to fewer input, network convergence is much more guaranteed when trained on augmented samples. The next comparison was made between two cases; one with data augmentation at training time only, and the other with data augmentation applied during both training and testing. The model performance raised when we get from the former to the latter, i.e. when not only the available data was extended during training, but also several predictions were made for each of the augmented test samples. In this final case, the model's output was obtained by taking

average over all these predictions. As shown in Figure 2, data augmentation during test time improved the Pearson's R from 0.81 to 0.83.

***Dropout and Regularization*:** Besides data augmentation, adding dropout and regularization proves to avoid overfitting. From the beginning, we tested different values for dropout probability and achieved the best performance with dropout values of 0.5 and 0.8. Including regularization always improved the performance, so it was incorporated into all models.

3.3 Model Evaluation

To evaluate the generalization power of our DeepAtom model, we compare it with the published algorithms for binding affinity prediction as the baseline, namely RF-Score [Ballester2014] and K_{DEEP} [Jiménez2018]. Our model and the data used to evaluate it also try to fix the flaws in K_{DEEP} paper. In the following, we discuss the flaws as well as the measures we took to resolve them.

If the data used in training two models is different, the models' performance cannot be compared directly. Indeed, two of the three models used as baseline in K_{DEEP} paper [Jiménez2018] are trained on smaller datasets because of less data available at the time they were developed. The latest versions of X-Score [Wang2002] and CyScore [Cao2014] available to academic users were trained in 2003 and 2014, respectively. Their smaller training datasets make it unfair to compare them against models trained on PDBbind v2016 data. RF-Score model performs better than K_{DEEP} model in all four additional test sets used in the original report, and is either the best model or among the top two compared to the other models the authors evaluated [Jiménez2018]. Additionally, RF-Score is the only open source model among the four; this makes it a reasonable baseline for our work.

To compare DeepAtom model against K_{DEEP} and RF-Score, we first followed the splitting reported by K_{DEEP}. To make a fair comparison, we also optimized the hyperparameters of the RF-Score model, not carried out in the K_{DEEP} paper. We noticed that the source code for RF-Score sets the hyperparameters different from their optimized values reported in the original RF-Score paper

[Ballester2014] for the PDBbind v2007 data. For instance, the default distance cutoff is set as 5.0 Å, whereas the paper reported its optimized value as 12.0 Å. A more subtle issue is that, by default, RF-Score extracts the features from the pocket files pre-generated from the whole protein structures; this is with the aim of improving the computational efficiency. However, these pocket files in the PDBbind dataset include protein atoms up to 8.0 Å from the ligand. If using the default settings, RF-Score with a distance cutoff larger than 8.0 Å does not consider those protein-ligand atom pairs beyond 8.0 Å. To resolve these issues, we set distance cutoff as a hyperparameter, and also generated pocket files consistent with the distance cutoff used. It should be noted that in this splitting scheme no hyperparameter tuning was carried out for RF-Score model due to absence of a validation set. Instead we set their values as the optimized values reported in the RF-Score paper [Ballester2014], i.e. 12 Å distance cutoff, *elements* descriptors, 2 Å bin size, and feature selection threshold (spr=1). Using Mean Squared Error (MSE) as the evaluation metric, the Pearson's R correlation coefficient achieved by DeepAtom, K_{DEEP}, and RF-Score model on the *core* subset of PDBbind v2016 is 0.83, 0.82, and 0.81, respectively. Figure 4 shows how much the DeepAtom performance improves by tuning each of its hyperparameters.

As mentioned before, lack of validation set is discouraged in machine learning field. Being open source and showing superior performance makes RF-Score the ideal baseline. In the second splitting scheme, we combined the *refined* and *core* subsets of PDBbind v2016 and then split the whole dataset using 80-10-10 random splitting. This resulted in 80% (3,245 complexes) for training, 10% (406 complexes) for validation, and 10% (406 complexes) for testing. This yielded Pearson's R correlation coefficient (and MSE in pK units) of 0.79 (1.5) and 0.75 (1.8) for DeepAtom and RF-Score model, respectively. The correlations are provided as scatterplots in Fig. 5.

The benchmark dataset we compiled is composed of 10,673 complexes. Following 80-10-10 random splitting, 8,538 complexes were used in training both DeepAtom and RF-Score model. 1067 of the remaining complexes were used as validation to tune the hyperparameters of both models, and 1068

complexes were used to evaluate them. The performance in terms of Pearson's R (and prediction error, in terms of pK units) achieved for DeepAtom and RF-Score model were 0.78 (1.5) and 0.73 (1.8), respectively. Figure 6 illustrates the predicted pK values versus experimental pK values for DeepAtom and RF-Score model. The performance of the models on the mentioned three splittings is tabulated in Table 2.

It is noteworthy that the original RF-Score model trained on PDBbind v2007 data had shown the best performance when the *elements* descriptors (the simplest among the three descriptor schemes) were used [Ballester2014]. In contrast, we found *Sybyl* features (the most detailed among the three feature schemes) to yield the lowest prediction error for RF-Score models trained on 80-10-10 random splits of PDBbind v2016 and our extended benchmark dataset. However, consistent with the RF-Score model trained on PDBbind v2007, we found the lowest performance yielded from *Credo* descriptors. The training sizes for PDBbind v2007, PDBbind v2016 (K_{DEEP} split), PDBbind v2016 (80-10-10 split), and our extended benchmark dataset (80-10-10 split) are 1105, 3767, 3245, and 8538 complexes, respectively.

The two models were scrutinized by enrichment analysis; as the graphs in Figs. 7a and 7b illustrate, in the 80-10-10 random splitting of PDBbind v2016 dataset the complexes which belong to *core* subset are clearly over-represented in the 5% and 10% most accurate predictions by both DeepAtom and RF-Score models, whereas those complexes coming from *refined* subset occur as frequently as expected. The 80-10-10 random splitting of the benchmark 2018 dataset reveals an interesting difference between the two models; the performance of DeepAtom in prediction of *refined* and *general* subsets is balanced, with both around 1.0, while the RF-Score predictions are evidently in favor of the complexes from *general* subset. Due to much fewer number of complexes coming from *core* subset, their low occurrence in the enrichment analysis was expected (Figs. 7c and 7d).

Analysis of prediction errors for PDBbind v2016 complexes with the 5%, 10%, and 20% largest errors suggests higher errors than expected for complexes from the *core* subset (Fig. 8a). This behavior

is similar between DeepAtom and RF-Score, although the former model is less affected. Together with the enrichment analysis graphs in Figs. 7a and 7b, the error graphs in Figs. 8a and 8b may point to overfitting on the *core* subset for both models; this may root back to how the representative complexes from the *refined* subset have been assigned to the *core* subset in the PDBbind v2016 dataset [Wang2005]. Even a sharper difference is observed for the *core* complexes in the benchmark 2018 dataset between the DeepAtom and RF-Score models (Figs. 8c and 8d). The abundance of complexes from different subsets is almost balanced in DeepAtom, whereas the complexes from *core* subset are highly over-represented in the largest prediction errors yielded from the RF-Score model.

The training data puts a limitation on model performance. There has been debate whether the inclusion of low quality data leads to an improvement in a model's generalization power. Li and coworkers reported that their Random Forest (RF) model benefited from training on additional low quality structural and binding data. Their study suggests that a machine learning model should not be restricted on a small training set of high quality data, and it will get a boost from including lower quality data [Li2015]. Nevertheless, a recent work reported no significant change in model performance when trained on PDBbind's both *general* and *refined* subsets [Jiménez2018]. Both sides seem to have a point, so we planned to consider their justifications in compiling our extension to the *refined* set as the standard training dataset. As Figs. 9a and 9b illustrate, the median of errors in the PDBbind v2016 is almost the same in both models; meanwhile the Inter-Quartile Range (IQR) is smaller in the case of DeepAtom, meaning that the spread of errors in DeepAtom are smaller for the 50% top predictions, compared to the RF-Score model. Similarly for the benchmark 2018 dataset the errors have narrower spread in the DeepAtom predictions, which also gives rise to more outliers (Figs. 10a and 10b). In addition to these boxplots dissected by the source, comparison of all predictions for the two models confirms the previous finding, i.e. the DeepAtom model yields narrower error spread; more outliers in DeepAtom can in turn be attributed to this smaller spread, because the Absolute Error (AE) is almost the same for the outliers in DeepAtom and RF-Score (Figs. 11a and 11b). The median

prediction error (and IQR) for DeepAtom and RF-Score on the PDBbind v2016 dataset are 0.747 (0.994) and 0.84 (1.16) in pK units, respectively. Additionally the median prediction error (and IQR) for DeepAtom and RF-Score on the benchmark 2018 dataset are 0.663 (0.912) and 0.81 (1.125), respectively. As graphs in Figs. 11a and 11b indicate, inclusion of lower quality data from *general* subset for training and testing on even a larger test set has improved the median prediction error as well as the Inter-Quartile Range for both DeepAtom and RF-Score models.

Finally we analyzed the trend of errors yielded from the DeepAtom and RF-Score models. The errors from the former model were sorted in descending order, and the RF-Score prediction errors for the same order were plotted, as provided in Fig. 12. The raw prediction errors are shown in Figs. 12a and 12c. Furthermore the averaged errors with bin size of 5 are illustrated in Figs. 12b and 12d. The error trend are reasonably consistent, meaning that the error for the same complex has correlation between the two models. This correlation is depicted in Figures 13a and 13b, where the outputs from DeepAtom is plotted versus the RF-Score predictions. As the high Pearson's R correlation coefficients indicate, highly similar predictions are expected from the two models for the binding affinity of an input structure. The Pearson's R coefficients in the case of the PDBbind v2016 and the benchmark 2018 datasets are 0.9 and 0.87, respectively.

Limitations: We close this section by raising the discussion about a few limitations of our models. These limitations also apply to many machine learning algorithms for scoring the protein-ligand binding affinity, with different levels of severity:

- An inherent drawback for the models is related to the data they have been trained on. The input training data includes only the bound structure of protein and ligand. The free forms of ligand and protein are then inferred, assuming that the differences between bound and unbound conformations are negligible. This is not always the case; as an example, it is well known that protein kinases exist in an ensemble of conformations, with relatively large movements in their kinase domain [Huse2002; Rabiller2010]. This makes kinases a challenging case for binding

affinity prediction based on the bound conformation only. It also justifies why the competitions for binding affinity prediction, such as D3R (https://drugdesigndata.org), often test the models on kinase proteins [Gaieb2017].

- The second limitation relates to defining the binding box around ligand. Models which rely on grid box definition cannot take advantage of the structural information beyond the grid box. Through binding to an allosteric site, allosteric compounds impact the binding affinity of a ligand at a protein's binding site. In these cases, the binding affinity is not only affected by the ligand at the binding site, but also by the allosteric compound beyond the grid box.
- Finally, although different techniques are used to overcome the small training data, the number of complexes with reliable experimentally determined binding affinity data is relatively small, e.g. compared to the image recognition field. This limits the use of powerful deep learning architectures because overfitting is very likely to occur. The light-weight CNN architectures mitigate this issue to some extent.

4. CONCLUSION

Deep learning models leverage the data itself to generate and extract features in as much autonomous as possible. This work reports an improvement in the prediction performance of the model, compared to the state of the art models developed for protein-ligand binding affinity prediction. The correlation between predicted and experimental binding affinity data on the train and test dataset are reasonably high, given that the only input is the *holo* structure of the protein and ligand in the complex, and no further information about the protein conformational change or the other regions in the protein is used for the prediction, except for the binding site area.

The second contribution of this study is to extend the standard datasets used for training the binding affinity scoring algorithms. We propose our extended dataset which includes protein-ligand complexes with only K_d, K_a, or K_i experimental data. The proposed dataset offers greater than double the size of the PDBbind v.2016 *refined* subset as the standard training dataset. We observed superior performance of our DeepAtom model trained on this dataset, compared to RF-Score model as the state-of-the-art in the literature; therefore we recommend it as a benchmark dataset especially for machine learning scoring functions, where more samples potentially improve a model's generalization power.

Future directions may include the use of kernels which make the convolutional layers invariant to translation and rotation; this will make overfitting a less severe problem and will potentially boost the performance.

5. REFERENCES

[Ahmed2015] Ahmed A., Smith R.D., Clark J.J., Dunbar J.B. Jr., Carlson H.A. "Recent improvements to Binding MOAD: a resource for protein-ligand binding affinities and structures." Nucleic Acids Res. 2015; 43(Database issue):D465-9.

[Ain2015] Ain Q.U., Aleksandrova A., Roessler F.D., Ballester P.J. (2015) "Machine-learning scoring functions to improve structure-based binding affinity prediction and virtual screening." Wiley Interdiscip Rev Comput Mol Sci. 5(6):405-424.

[Ashtawy2015a] Ashtawy H.M., Mahapatra N.R. (2015) "A comparative assessment of predictive accuracies of conventional and machine learning scoring functions for protein-ligand binding affinity prediction." IEEE/ACM Trans Comput Biol Bioinform. 12(2):335-47.

[Ashtawy2015b] Ashtawy H.M., Mahapatra N.R. (2015) "BgN-Score and BsN-Score: bagging and boosting based ensemble neural networks scoring functions for accurate binding affinity prediction of protein-ligand complexes." BMC Bioinformatics. 16 Suppl 4:S8.

[Ballester2010] Ballester P.J., Mitchell J.B. (2010) "A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking." Bioinformatics. 26(9):1169-75.

[Ballester2014] Ballester P.J., Schreyer A., Blundell T.L. (2014) "Does a more precise chemical description of protein-ligand complexes lead to more accurate prediction of binding affinity?" J Chem Inf Model. 54(3):944-55.

[Cao2014] Cao Y., Li L. (2014) "Improved protein-ligand binding affinity prediction by using a curvature-dependent surface-area model." Bioinformatics. 30: 1674-80.

[Cha1975] Cha S. (1975) "Tight-binding inhibitors-I. Kinetic behavior." Biochem Pharmacol. 24(23):2177-85.

[deAzevedo2008] de Azevedo W.F. Jr., Dias R. (2008) "Computational methods for calculation of ligand-binding affinity." Curr Drug Targets. 9(12):1031-9.

[Du2016] Du X., Li Y., Xia Y.L., Ai S.M., Liang J., Sang P., Ji X.L., Liu S.Q. (2016) "Insights into protein–ligand interactions: mechanisms, models, and methods." Int J Mol Sci. 17(2), 144.

[Dunbar2013] Dunbar J.B. Jr., Smith R.D., Damm-Ganamet K.L., Ahmed A., Esposito E.X., Delproposto J., Chinnaswamy K., Kang Y.N., Kubish G., Gestwicki J.E., Stuckey J.A., Carlson H.A. (2013) "CSAR data set release 2012: ligands, affinities, complexes, and docking decoys." J Chem Inf Model. 53(8):1842-52.

[Fukushima1980] Fukushima K. (1980) “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.” Biol Cybernetics. 36(4):193-202.

[Gaieb2017] Gaieb Z., Liu S., Gathiaka S., Chiu M., Yang H., Shao C., Feher V.A., Walters W.P., Kuhn B., Rudolph M.G., Burley S.K., Gilson M.K., Amaro R.E. (2017) "D3R Grand Challenge 2: blind prediction of protein-ligand poses, affinity rankings, and relative binding free energies." J Comput Aided Mol Des. 32(1):1–20.

[Gilson2007] Gilson M.K., Zhou H.X. (2007) "Calculation of protein-ligand binding affinities." Annu Rev Biophys Biomol Struct. 36:21-42.

[Gomes2017] Gomes J., Ramsundar B., Feinberg E., Pande V.S. (2017) "Atomic convolutional networks for predicting protein-ligand binding affinity." eprint arXiv:1703.10603.

[Goodfellow2016] Goodfellow I., Bengio Y., Courville A. (2016) "Deep Learning." The MIT Press.

[He2015] He K., Zhang X., Ren Sh., Sun J. (2015) "Deep residual learning for image recognition." http://arxiv.org/abs/1512.03385.

[Heck2017] Heck G.S., Pintro V.O., Pereira R.R., de Ávila M.B., Levin N.M.B., de Azevedo W.F. (2017) "Supervised machine learning methods applied to predict ligand- binding affinity." Curr Med Chem. 24(23):2459-2470.

[Hernández-Rodríguez2016] Hernández-Rodríguez M., Rosales-Hernández M.C., Mendieta-Wejebe J.E., Martínez-Archundia M., Basurto J.C. (2016) "Current tools and methods in molecular dynamics (MD) simulations for drug design." Curr Med Chem. 23(34):3909-3924.

[Howard2017] Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. (2017) "MobileNets: Efficient convolutional neural networks for mobile vision applications." https://arxiv.org/abs/1704.04861.

[Hu2005] Hu L., Benson M.L., Smith R.D., Lerner M.G., Carlson H.A. "Binding MOAD (Mother Of All Databases)." Proteins. 2005; 60(3):333-40.

[Huang2010] Huang S.Y., Grinter S.Z., Zou X. (2010) "Scoring functions and their evaluation methods for protein-ligand docking: recent advances and future directions." Phys Chem Chem Phys. 12(40):12899-908.

[Huey2007] Huey R., Morris G.M., Olson A.J., Goodsell D.S. (2007) "A semiempirical free energy force field with charge-based desolvation." J Comput Chem. 28(6):1145-52.

[Huse2002] Huse M., Kuriyan J. (2002) "The conformational plasticity of protein kinases." Cell. 109(3):275-82.

[Iandola2016] Iandola F., Han S., Moskewicz M., Ashraf K., Dally W.J., Keutzer K. (2016) “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size.” https://arxiv.org/abs/1602.07360.

[Ioffe2015] Ioffe S., Szegedy C. (2015) "Batch normalization: accelerating deep network training by reducing internal covariate shift." https://arxiv.org/abs/1502.03167.

[Jiménez2017] Jiménez J., Doerr S., Martínez-Rosell G., Rose A.S., De Fabritiis G. (2017) "DeepSite: Protein binding site predictor using 3D-convolutional neural networks." Bioinformatics. 33(19):3036-3042.

[Jiménez2018] Jiménez J., Škalič M., Martínez-Rosell G., De Fabritiis G. "KDEEP: Protein-ligand absolute binding affinity prediction via 3D-convolutional neural networks." J Chem Inf Model. 58(2):287-296.

[Jubb2017] Jubb H.C., Higueruelo A.P., Ochoa-Montaño B., Pitt W.R., Ascher D.B., Blundell T.L. (2017) "Arpeggio: A web server for calculating and visualising interatomic interactions in protein structures." J Mol Biol. 429(3):365-371.

[Khamis2015] Khamis M.A., Gomaa W., Ahmed W.F. (2015) "Machine learning in computational docking." Artif Intell Med. 63(3):135-52.

[Kim2008] Kim R., Skolnick J. (2008) "Assessment of programs for ligand binding affinity prediction." J Comput Chem. 29(8):1316-31.

[Koes2013] Koes D.R., Baumgartner M.P., Camacho C.J. (2013) "Lessons learned in empirical scoring with smina from the CSAR 2011 benchmarking exercise." J Chem Inf Model. 53, 1893-1904.

[Li2014a] Li H., Leung K.S., Wong M.H., Ballester P.J. (2014) "Substituting random forest for multiple linear regression improves binding affinity prediction of scoring functions: Cyscore as a case study." BMC Bioinformatics. 15:291.

[Li2014b] Li Y., Liu Z., Li J., Han L., Liu J., Zhao Z., Wang R. (2014) “Comparative assessment of scoring functions on an updated benchmark: 1. Compilation of the test set.” J Chem Inf Model. 54(6):1700-16.

[Li2015] Li H., Leung K.S., Wong M.H., Ballester P.J. (2015) "Low-quality structural and interaction data improves binding affinity prediction via random forest." Molecules. 20(6):10947-62.

[Li2017] Li Y., Yang J. (2017) "Structural and sequence similarity makes a significant impact on machine-learning-based scoring functions for protein-ligand interactions." J Chem Inf Model. 57(4):1007-1012.

[Li2018] Li H., Peng J., Leung Y., Leung K.S., Wong M.H., Lu G., Ballester P.J. (2018) "The impact of protein structure and sequence similarity on the Aaccuracy of machine-learning scoring functions for binding affinity prediction." Biomolecules. 8(1). pii: E12.

[Liu2015] Liu J., Wang R. (2015) “Classification of current scoring functions.” J Chem Inf Model. 55(3):475-82.

[Liu2017] Liu Z., Su M., Han L., Liu J., Yang Q., Li Y., Wang R. (2017) "Forging the basis for developing protein-ligand interaction scoring functions." Acc Chem Res. 50(2):302-309.

[Ma2018] Ma N., Zhang X., Zheng H.-T., Sun J. (2018) "ShuffleNet V2: Practical guidelines for efficient CNN architecture design." https://arxiv.org/abs/1807.11164.

[Melo1997] Melo F., Feytmans E. (1997) "Novel knowledge-based mean force potential at atomic level." J Mol Biol. 267(1):207-22.

[Melo1998] Melo F., Feytmans E. (1998) "Assessing protein structures with a non-local atomic interaction energy." J Mol Biol. 277(5):1141-52.

[Montalvo-Acosta2016] Montalvo-Acosta J.J., Cecchini M. (2016) "Computational approaches to the chemical equilibrium constant in protein-ligand binding." Mol Inform. 35(11-12):555-567.

[Morris2009] Morris G.M., Huey R., Lindstrom W., Sanner M.F., Belew R.K., Goodsell D.S., Olson A.J. (2009) "AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility." J Comput Chem. 30(16):2785-91.

[Rabiller2010] Rabiller M., Getlik M., Klüter S., Richters A., Tückmantel S., Simard J.R., Rauh D. (2010) "Proteus in the world of proteins: conformational changes in protein kinases." 343(4):193-206.

[Ragoza2017] Ragoza M., Hochuli J., Idrobo E., Sunseri J., Koes D.R. (2017) "Protein-ligand scoring with convolutional neural networks." J Chem Inf Model. 57(4):942-957.

[Ryde2016] Ryder U., Söderhjelm P. (2016) "Ligand-binding affinity estimates supported by quantum-mechanical methods." Chem Rev. 116(9):5520-66.

[Schietgat2015] Schietgat L., Fannes T., Ramon J. (2015) "Predicting protein function and protein-ligand interaction with the 3D neighborhood kernel." In: Japkowicz N., Matwin S. (eds) Discovery Science. Lecture Notes in Computer Science, vol 9356. Springer, Cham

[Schreyer2009] Schreyer A.M., Blundell T.L. (2009) "CREDO: a protein–ligand interaction database for drug discovery." Chem Biol Drug Des. 73:157–167.

[Schreyer2013] Schreyer A.M., Blundell T.L. (2013) "CREDO: a structural interactomics database for drug discovery." Database (Oxford). 2013:bat049.

[Simonyan2014] Simonyan K., Zisserman A. (2014) "Very deep convolutional networks for large-scale image recognition." https://arxiv.org/abs/1409.1556.

[Sotriffer2008] Sotriffer C.A., Sanschagrin P., Matter H., Klebe G. (2008) "SFCscore: scoring functions for affinity prediction of protein-ligand complexes." Proteins. 73(2):395-419.

[Srivastava2014] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. (2014) "Dropout: A simple way to prevent neural networks from overfitting." J Mach Learn Res. 15(1):1929-1958.

[Torng2017] Torng W., Altman R.B. (2017) "3D deep convolutional neural networks for amino acid environment similarity analysis." BMC Bioinformatics. 18(1):302.

[Wang2002] Wang R., Lai L., Wang S. (2002) "Further development and validation of empirical scoring functions for structure-based binding affinity prediction." J Comput-Aided Mol Des. 16:11−26.

[Wang2005] Wang R., Fang X., Lu Y., Yang C.-Y., Wang S. (2005) “The PDBbind database: Methodologies and updates.” J Med Chem. 48: 4111-4119.

[Wang2013] Wang J.C., Lin J.H. (2013) "Scoring functions for prediction of protein-ligand interactions." Curr Pharm Des. 19(12):2174-82.

[Willander2009] Willander M., Al-Hilli S. (2009) “Analysis of biomolecules using surface plasmons.” In: Foote R., Lee J. (eds) Micro and Nano Technologies in Bioanalysis. Methods in Molecular Biology™ (Methods and Protocols), vol 544. Humana Press, Totowa, NJ

[Wójcikowski2017] Wójcikowski M., Ballester P.J., Siedlecki P. (2017) "Performance of machine-learning scoring functions in structure-based virtual screening." Sci Rep. 7:46710.

[Zhang2017] Zhang X., Zhou X., Lin M., Sun J. (2017) “ShuffleNet: An extremely efficient convolutional neural network for mobile devices.” https://arxiv.org/abs/1707.01083.

Fig. 1. Ligand end-to-end distance distribution in the PDBbind v2016 *refined* + *core* subsets

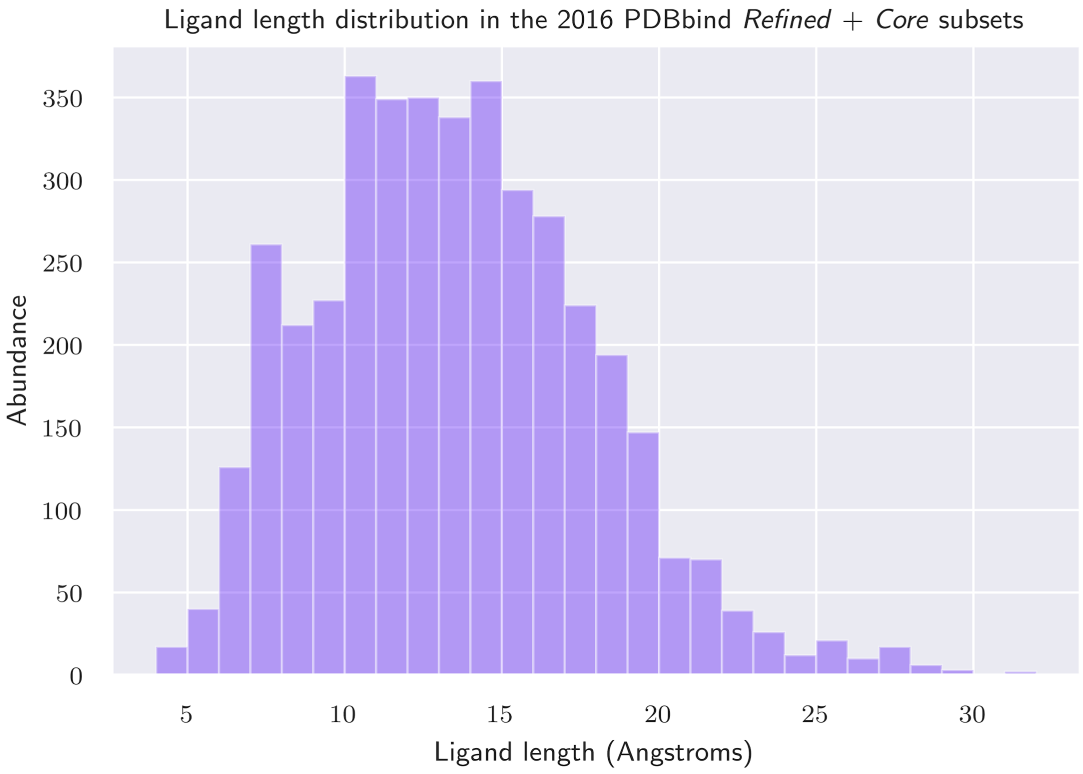


Fig. 2. Featurization of an input protein-ligand structure. A grid box is defined around the ligand in the binding site. Each channel includes only a specific view of the atoms inside of the grid box; the three channels shown here from left to right represent ligand's excluded volume, as well as the protein's hydrophobic and aromatic channels.

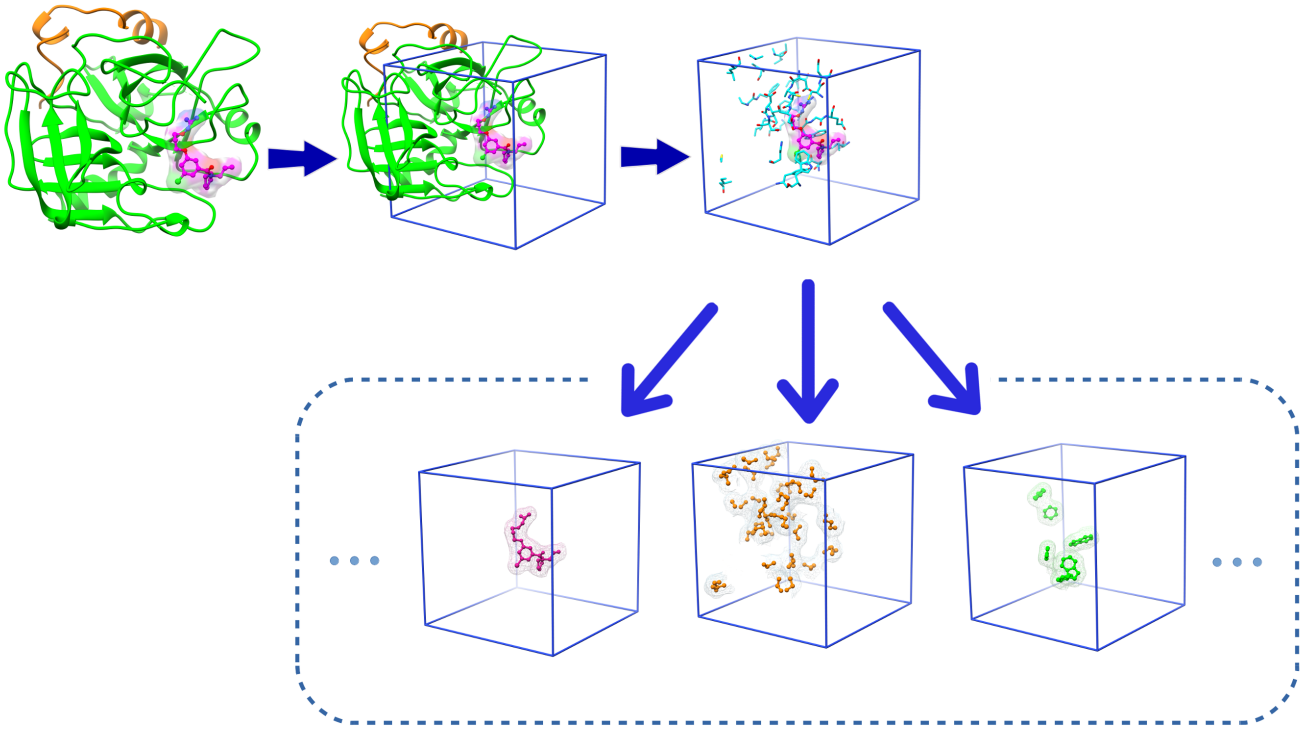


Fig. 3. Architecture of DeepAtom Model. Each Convolutional layer is specified by its number of channels, kernel size and stride. The 3D MaxPool layer has kernel size 3 and stride 2. For the 3D Shuffle Groups, the numbers in parentheses denote the number of output channels and repeat time of the unit. Only the first unit has down sampling layer, where the depthwise (DWConv) layer has kernel size 3 and stride 2. In the remaining units, DWConv with kernel size 3 and stride 1, as well as pointwise (PWConv) layer with kernel size 1 and stride 1 are utilized. Eight losses are calculated based on the shared weight fully connected layer output.

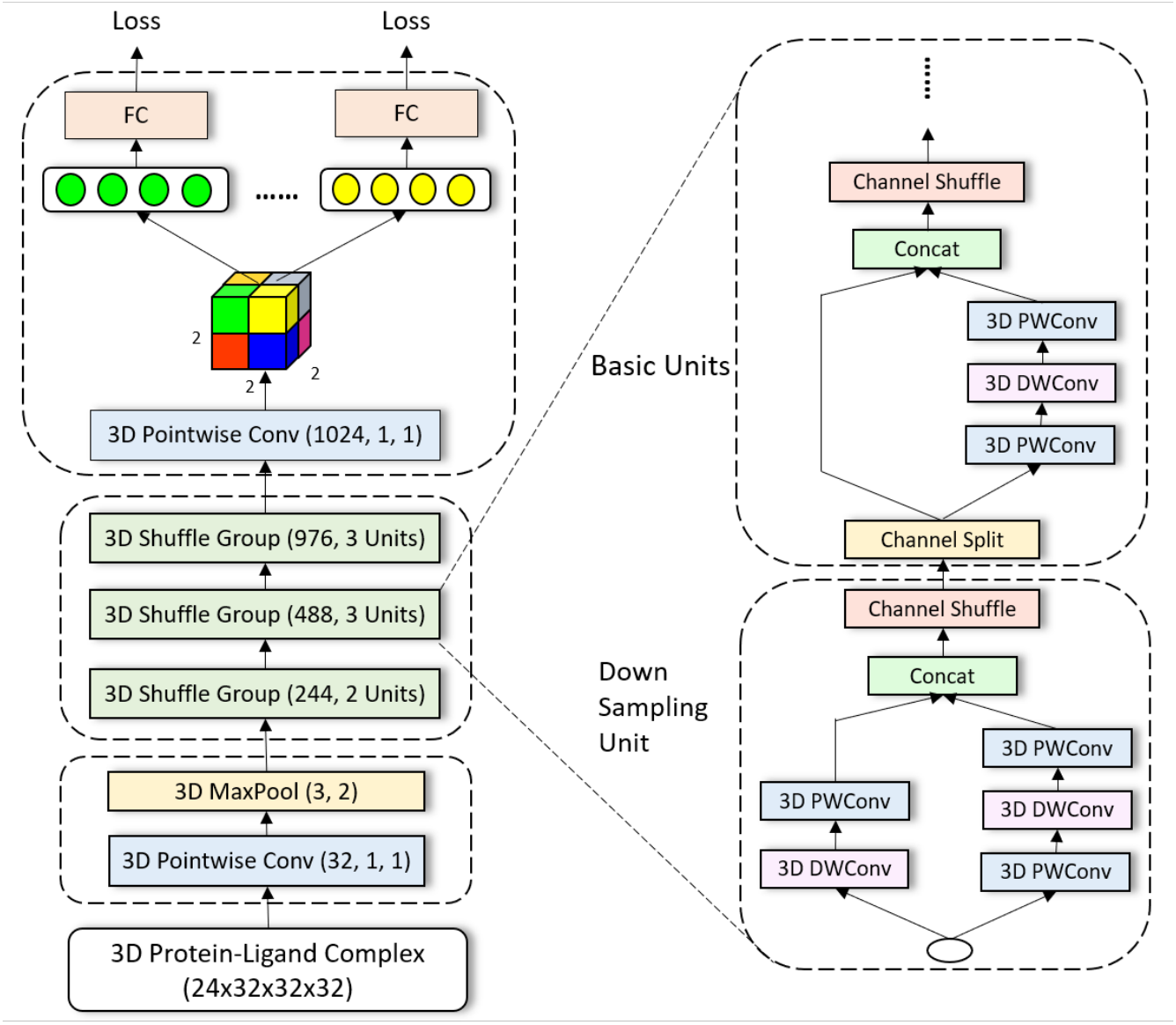


Fig. 4. Performance of DeepAtom model during hyperparameter tuning. The bars represent the performance of DeepAtom trained on the PDBbind v2016 *refined* subset and tested on the *core* subset. The performance is defined in terms of Pearson correlation coefficient between the predicted and experimental pK values.

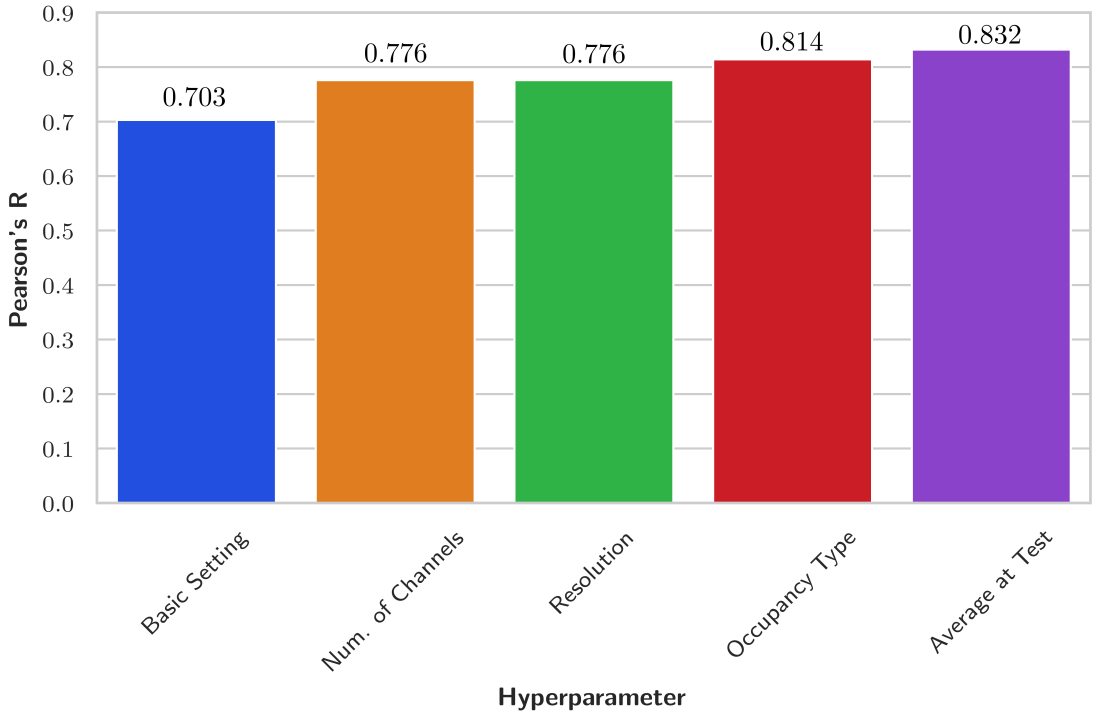


Fig. 5. Comparison of predictions in random splitting of PDBbind v2016. a) scatterplot for DeepAtom; b) scatterplot for RF-Score.

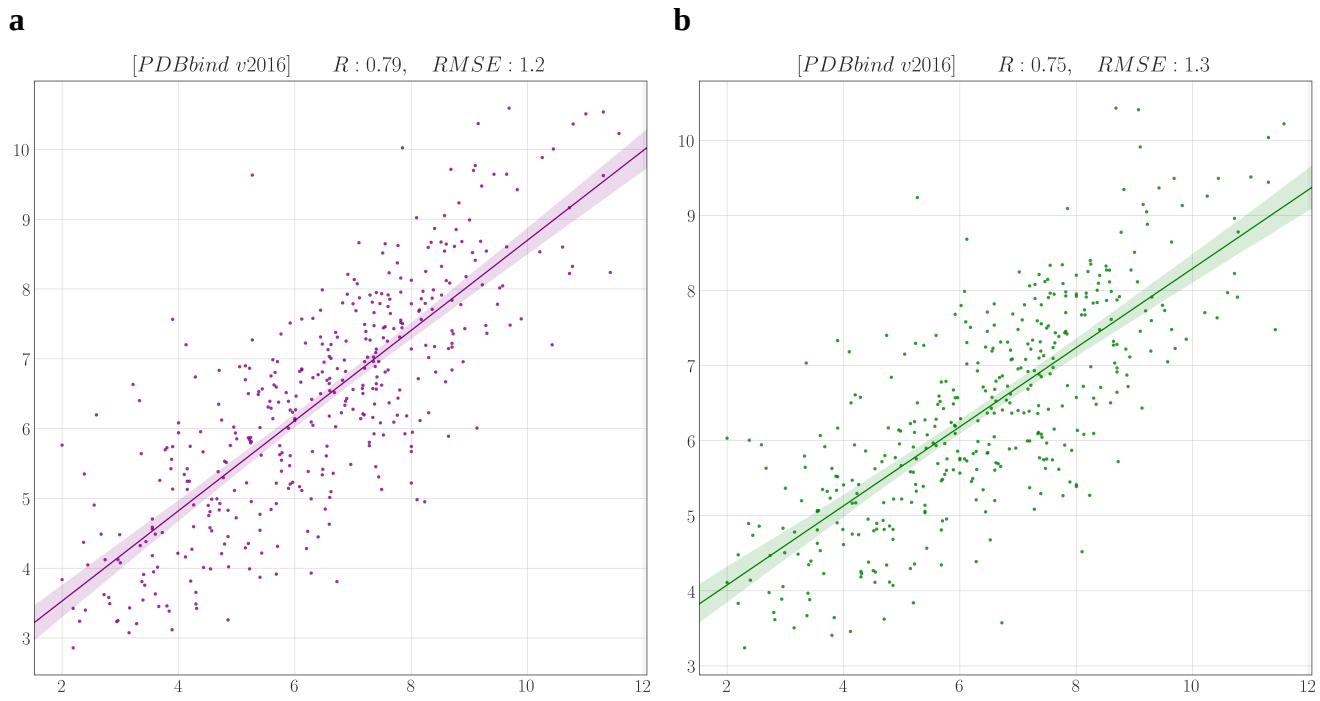


Fig. 6. Comparison of predictions in random splitting of benchmark 2018 dataset. a) scatterplot for DeepAtom; b) scatterplot for RF-Score.

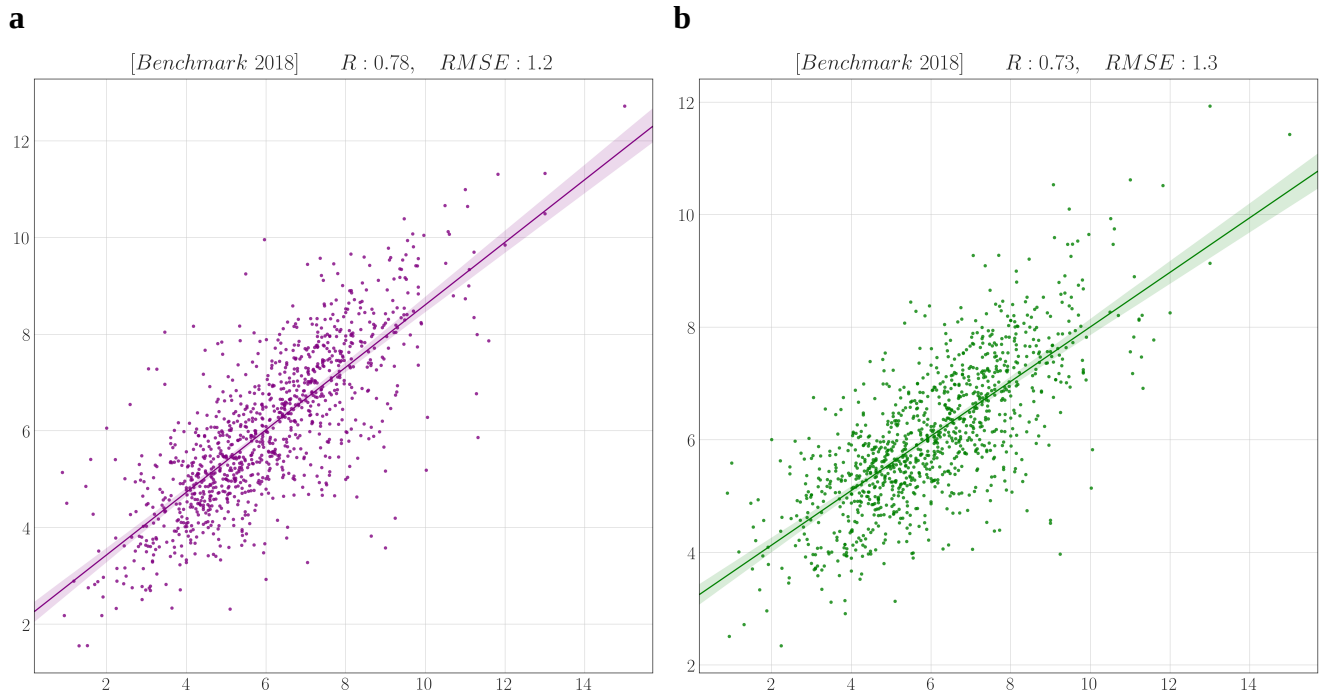


Fig. 7. Enrichment analysis for the top 5%, 10%, and 20% most accurate predictions dissected by source. a) DeepAtom on PDBbind v2016 dataset; b) RF-Score on PDBbind v2016 dataset; c) DeepAtom on benchmark 2018 dataset; d) RF-Score on benchmark 2018 dataset.

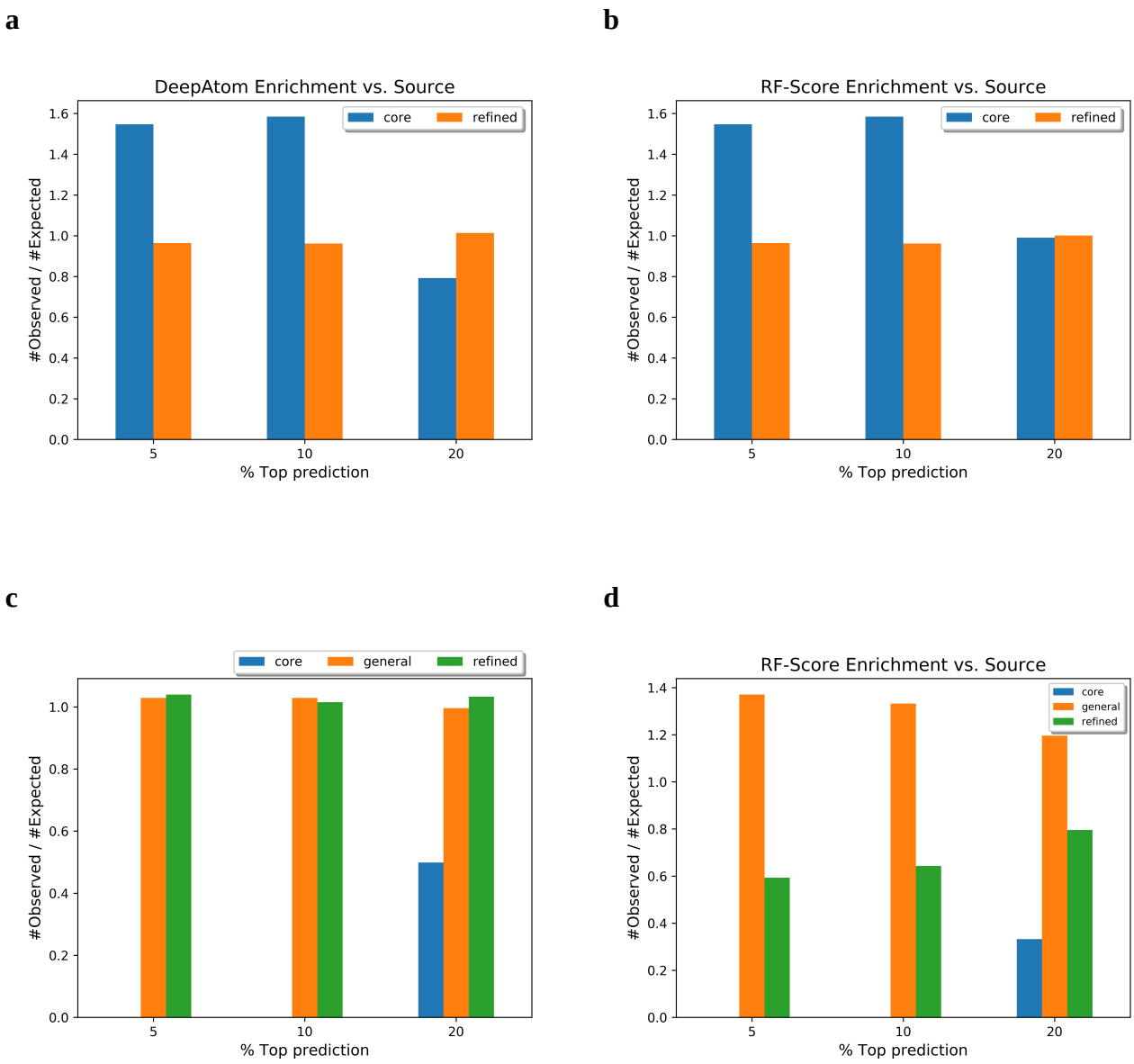
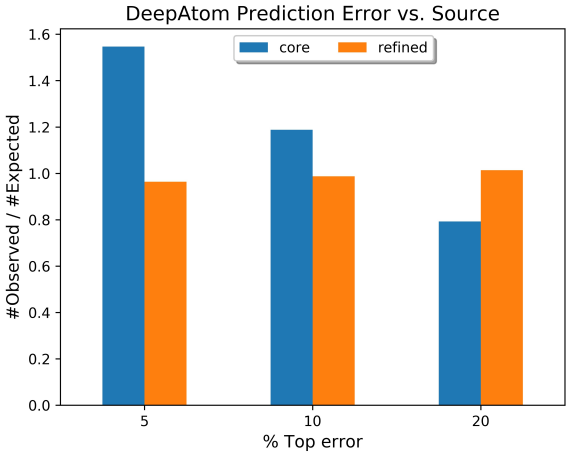
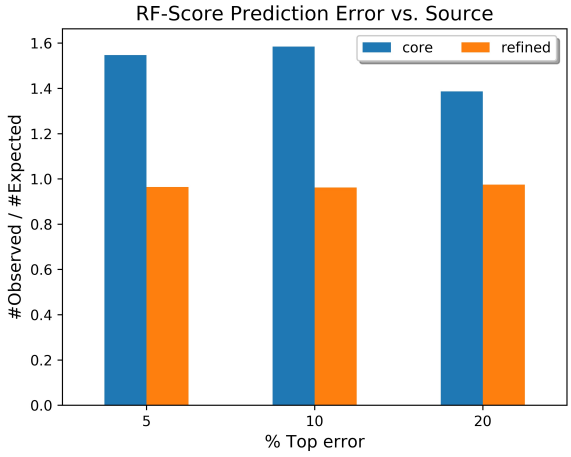


Fig. 8. Error analysis for the top 5%, 10%, and 20% largest prediction errors dissected by source. a) DeepAtom on PDBbind v2016 dataset; b) RF-Score on PDBbind v2016 dataset; c) DeepAtom on benchmark 2018 dataset; d) RF-Score on benchmark 2018 dataset.

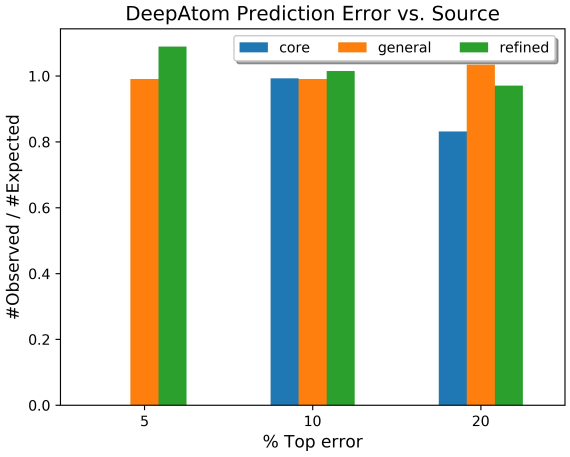
a



b



c



d

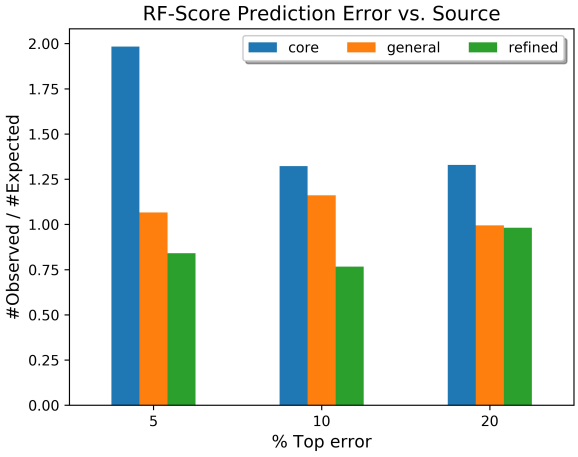
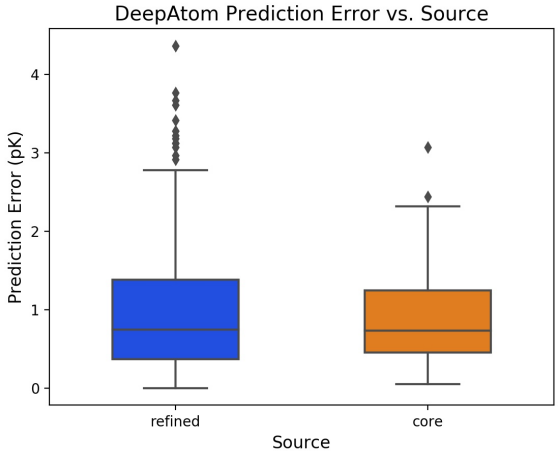


Fig. 9. Spread of prediction errors on the PDBbind v2016 dataset dissected by source. a) DeepAtom errors; b) RF-Score errors.

a



b

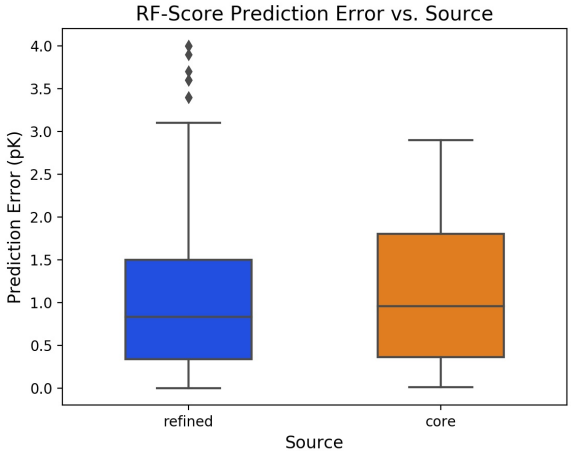
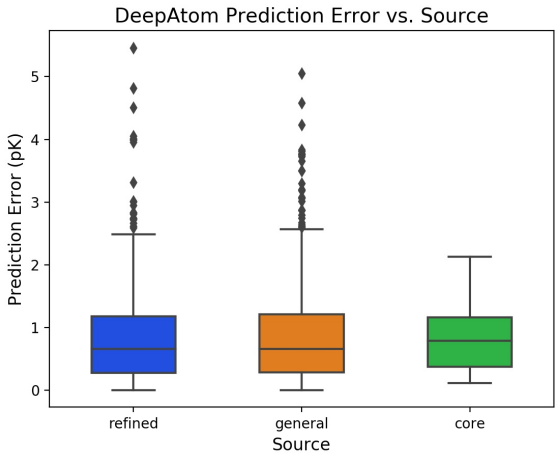


Fig. 10. Spread of prediction errors on the benchmark 2018 dataset dissected by source. a) DeepAtom errors; b) RF-Score errors.

a



b

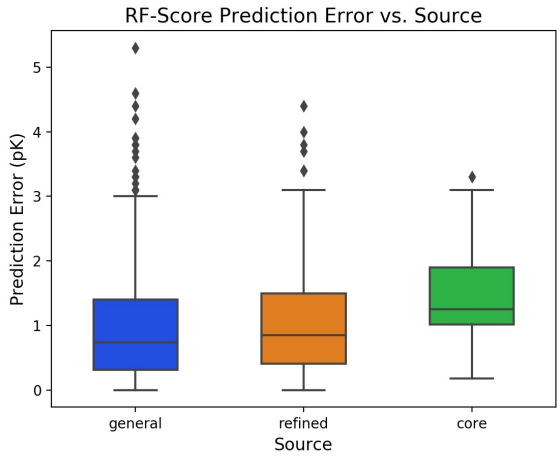
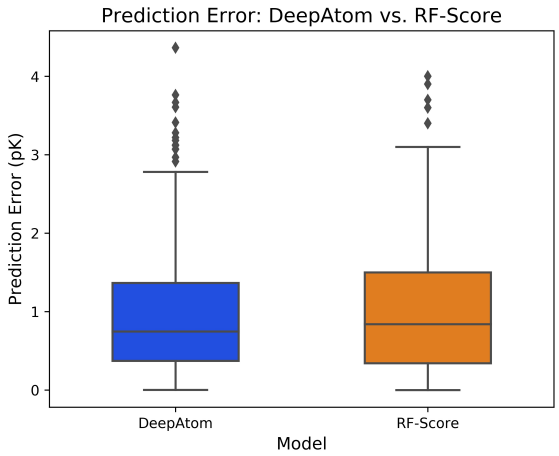


Fig. 11. Spread of prediction errors for the DeepAtom and RF-Score models. a) PDBbind v2016 dataset; b) benchmark 2018 dataset.

a



b

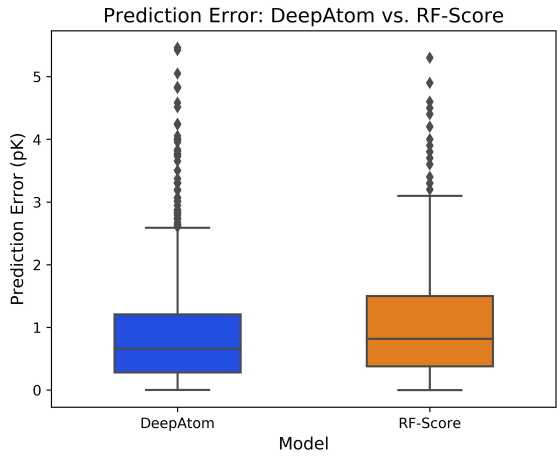


Fig. 12. Trend of prediction errors. The errors for DeepAtom are sorted in descending order, and the RF-Score errors are shown for the corresponding complexes. In the binned plots with bin size of 5, both error trends are plotted as the bin average values. a) raw trends on PDBbind v2016 dataset; b) binned trends on PDBbind v2016 dataset; c) raw trends on benchmark 2018 dataset; d) binned trends on benchmark 2018 dataset.

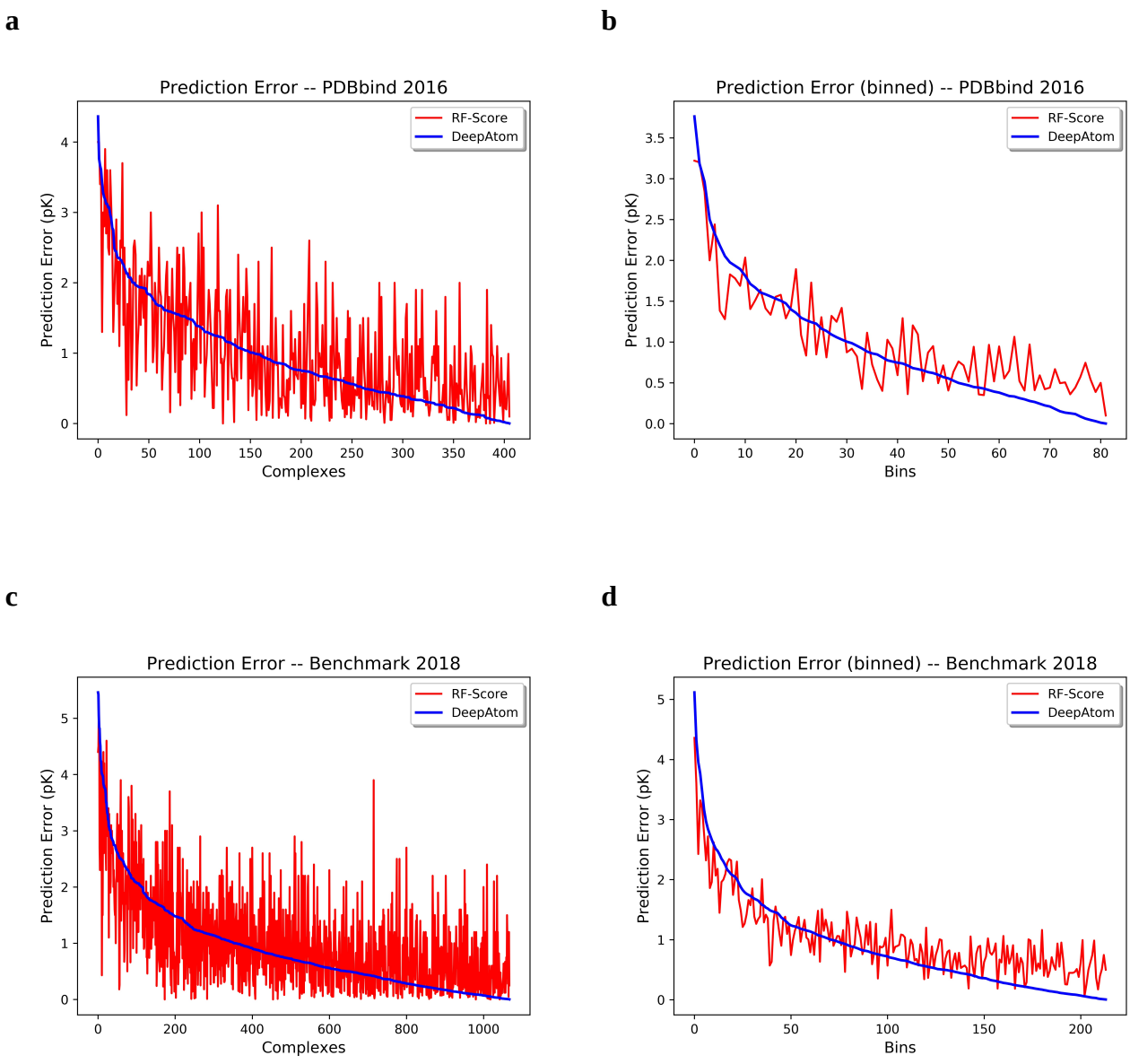


Fig. 13. Correlation of predictions between DeepAtom and RF-Score. Th predictions in pK units are plotted for DeepAtom versus RF-Score model. a) PDBbind v2106 dataset; b) benchmark 2018 dataset.

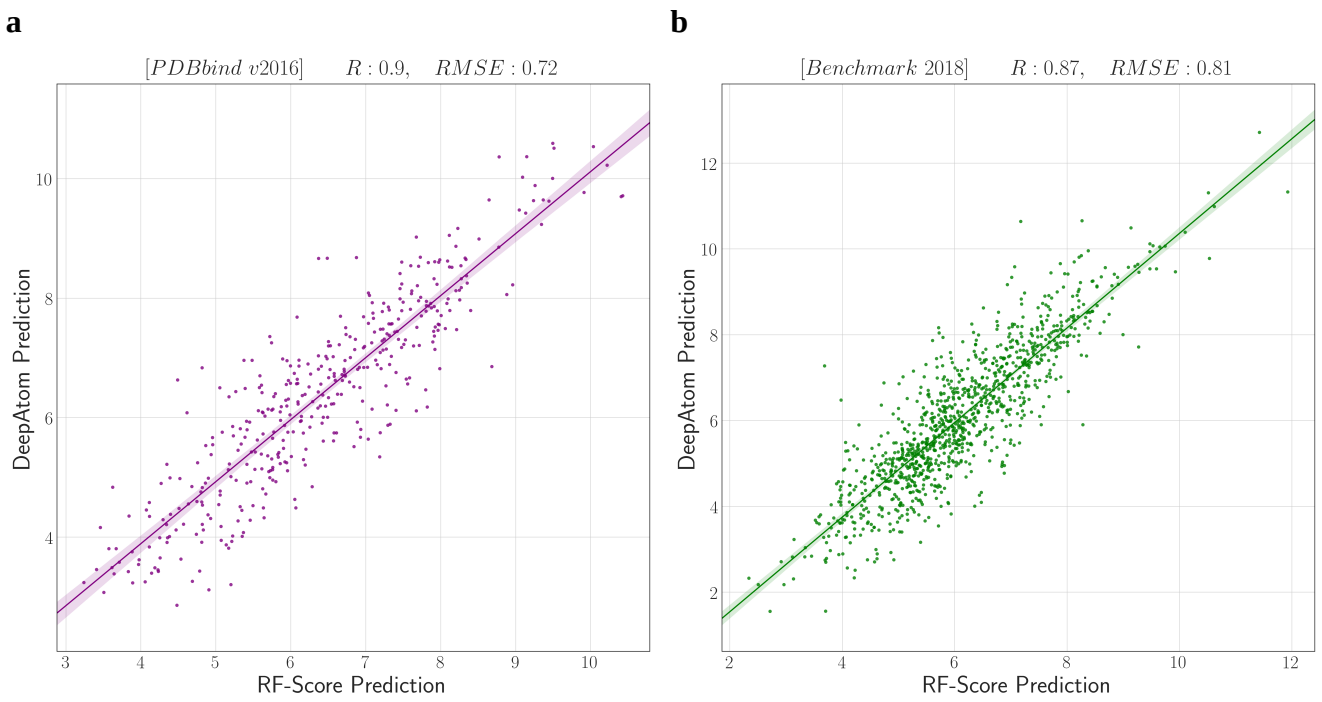


Table 1. Hyperparameter tuning. The DeepAtom model trained on PDBbind v2016 *refined* subset.

Number of Channels	Resolution (Å)	Occupancy Type	Averaging at Test Time	Pearson's R
11	1.0	Binary	0	0.703
24				**0.776**
60				0.776
24	0.375	Binary	0	0.788
	0.5			0.784
	1.0			0.776
24	1.0	Binary	0	0.776
		PCMax		**0.814**
24	1.0	PCMax	0	0.814
			36	**0.832**

Table 2. Comparison of model performance in terms of Pearson correlation coefficient between the predicted and experimental *pK* values. The DeepAtom model was developed in this study. The number in parentheses shows the Root Mean Squared Error (RMSE) in *pK* units. The best Pearson's R and RMSE values for each test set are shown in **bold**.

	PDBbind v2016 *core* set	**PDBbind v2016 (80-10-10 split)**	**Benchmark 2018 (80-10-10 split)**
DeepAtom	**0.83 (1.1)**	**0.79 (1.2)**	**0.78 (1.2)**
RF-Score	0.81 (1.2)	0.75 (1.3)	0.73 (1.3)
K_{DEEP}	0.82 (1.3)	N/A	N/A