DLSCORE: A Deep Learning Model for Predicting Protein-Ligand Binding Affinities

Md Mahmudulla Hassan[1],[†,‡] Daniel Castañeda Mogollón[1],[‡] Olac Fuentes,[†] and Suman Sirimulla[*,‡]

†*Department of Computer Science, University of Texas at El Paso, El Paso, TX*

‡*Department of Pharmaceutical Sciences, University of Texas at El Paso, El Paso, TX*

E-mail: ssirimulla@utep.edu

Abstract

In recent years, the cheminformatics community has seen an increased success with machine learning-based scoring functions for estimating binding affinities and pose predictions. The prediction of protein-ligand binding affinities is crucial for drug discovery research. Many physics-based scoring functions have been developed over the years. Lately, machine learning approaches are proven to boost the performance of traditional scoring functions. In this study, a novel deep learning based scoring function (DLSCORE) was developed and trained on the refined PDBBind v.2016 dataset using 348 BINding ANAlyzer (BINANA) descriptors. The neural networks of the DLSCORE model have different number of fully connected hidden layers. Our model, an ensemble of 10 networks, yielded a Pearson R^2 of 0.82, a Spearman Rho R^2 of 0.90, Kendall Tau R^2 of 0.74, an RMSE of 1.15 $kcal/mol$, and an MAE of 0.86 $kcal/mol$ for our test set. This software is available on Github at `https://github.com/sirimullalab/dlscore.git`

[1]These authors contributed equally.

Introduction

The use of information technology along with artificial intelligence in the drug discovery field has become critical over the past years. The use of biochemical high-throughput protein-ligand assays has the advantage of providing accurate results, however, they are usually expensive and are time- consuming. This is where chemoinformatics has an important role in drug discovery. One scope of chemoinformatics focuses on ligand identification and discovery of potential compound candidates that could help to prevent, cure, or even eradicate certain diseases. Furthermore, virtual screening and an appropriate molecular docking system of potential protein-ligand candidates helps by not only saving valuable time, but also reducing the cost of the research as well. Some popular docking scoring functions include AutoDock Vina[1] , GOLD[2] , SurFlex Dock[3] and Glide[4] among others. In our recent work we incorporated halogen bonding and chalcogen bonding interactions in to vina scoring function[5,6] . In this paper, we present a novel deep learning based scoring function to predict binding affinities in a protein-ligand complex: DLSCORE.

Nowadays, many researchers in both cheminformatics, bioinformatics and computer science, are using different approaches of AI to mimic the experimental biochemical high-throughput results of a protein-ligand interaction, aiming to evaluate the binding geometries of a putative ligand with a known protein target. One of the most recurrent approaches for accurate and fast molecular docking prediction is employing machine learning techniques for docking.

Machine learning (ML) is the scientific branch of artificial intelligence that focuses on how computational algorithms learn from empirical data[7] . There are different machine learning methods that are used for predicting protein-ligand affinities, for instance, Durrant *et al.*[8] (NNScore 2.0) used a neural-network scoring function, Ballester *et al.*[9] created a random forest scoring function for the same purpose, while other authors such as Kinnings *et al.* employ support vector machines (SVMs) to improve docking scoring functions for drug repurposing[10] , or Gomes *et al.* using atomic convolutional networks for predicting protein-

ligand binding affinity[11] . In here, we used a deep learning approach to accurately predict the binding affinities by using an ensemble of fully connected dense neural networks. Deep learning (DL) has shown great success in multiple fields, such as computer vision, speech and image recognition, natural language processing, and now in the development of potential ligands for novel drug discovery[12,13] . The salient feature of DL is, with the large numbers of hidden layers (with a bigger number of nodes) used, building higher-level representations of the data progressively and reducing the need for carefully hand-crafted features in contrast with conventional neural networks.

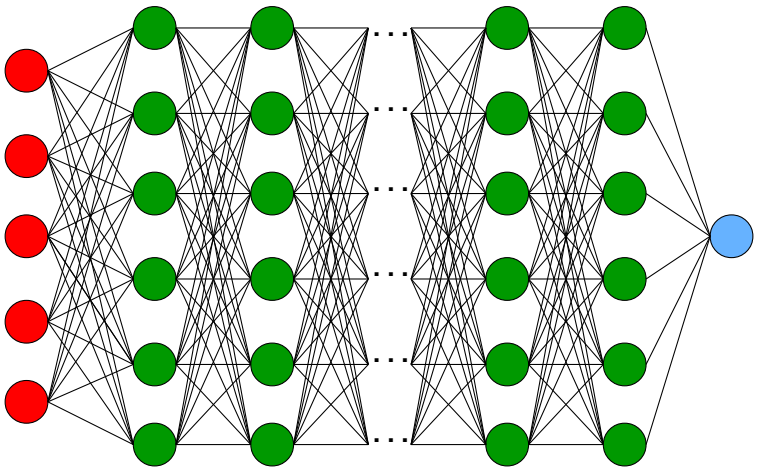


Figure 1: The basic representation of a fully connected dense neural network architecture. The nodes in red represent the input layer, the nodes in green represent the hidden layers, and the blue node represents the output layer. In DLSCORE, the input layer represents different molecular descriptors, while the output layer (one single node) is the pKd binding affinity predicted value.

In recent years, the scientific community has witnessed a colossal increase in biological data, including protein structures, genomic sequences, genes, SNP's, and active ligands for specific proteins. Thanks to this data growth, deep learning has become an attractive tool to be applied in industry and research[14] , including molecular docking and virtual screening. When it comes to molecular docking and compound activity prediction, DL methods can be adopted due to its capability of using a large number of nodes in the hidden layers, as well as the high number of molecular descriptors to build those models[12] . Recently, Ragoza

et al.[15] employed a protein-ligand scoring function by using convolutional neural networks. Jimenéz *et al.*[16] used convolutional neural networks (CNN) in the development of K_{DEEP} and obtained a Pearson correlation coefficient of 0.82, with a root mean square error ($RMSE$) of 1.27 in pK units between the predicted affinity and the experimental values[16] . Gomes *et al.*[11] developed an algorithm for learning atomic-level chemical interactions using spatial convolutional neural networks directly from atomic coordinates.

Materials and Methods

Workflow

The following automated workflow (Fig. 2) was used in order to test the results from DLSCORE against the predicted values from other scoring functions.

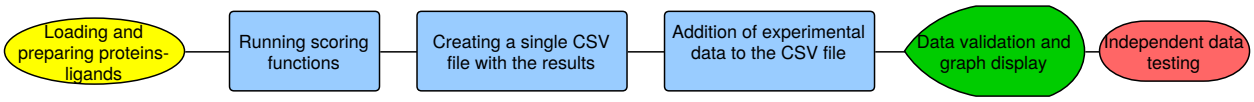


Figure 2: DLSCORE development workflow

Data Collection

PDBbind[17] dataset was used to train, test, and validate the model. The PDBbind database is a comprehensive collection of experimentally measured binding affinity data for biomolecular complexes deposited in the PDB data bank[17] . Here, we used the PDBbind (v.2016) refined set (n = 4154)[18] and their corresponding experimentally determined binding affinities obtained from PDBbind database[17] in terms of the equilibrium dissociation constant (Kd). The PDBbind (v.2016) refined set was used due to its high-quality data obtained after applying different filters regarding its binding features and resolution[18,19] . A total of 3191 crystallized complexes were utilized from the refined set to train and validate our DLSCORE model, while 300 were used for testing.

Protein-Ligand Preparation

The protein-ligand complexes were downloaded from PDBbind dataset and then transformed from `.pdb` to a `.pdbqt` format, which contains additional parameters, such as partial charges, and atom types. This conversion was necessary in order to obtain the BINding ANAlyzer (BINANA) features[20] that were used to train DLSCORE. Two different python codes, `"prepare_receptor4.py"` and `"prepare_ligand4.py"` from MGLTools[21] were used for the conversion of protein and ligand files respectively.

Intermolecular Features (descriptors)

We used the BINANA algorithm[20] implemented in NNScore 2.0[8] in order to characterize the binding of ligand-receptor complexes and used these descriptors for the input layer in our DLSCORE model. BINANA identifies ligand and protein atoms within a distance of 2.5 Å - 4.0 Å between them, as well as electrostatic interactions, binding pocket flexibility, hydrogen bonds, salt bridges, rotatable bonds, π interactions, among others[20] . A total of 348 features were considered for each protein-ligand complex.

DLSCORE Model Architecture

In this study, we employed fully-connected neural networks (FCNN). The networks in the model consist of multiple hidden layers with a different number of neurons. Each hidden layer has a weight matrix (W) with a dimension ruled by the input size and the number of neurons in that layer. Since we are dealing with a regression problem, the size of the output layer is 1. Each protein-ligand complex is represented by a feature vector ($f = f_1, f_2, f_3...f_n$) of size 348. The input layer takes the feature vector, performs a matrix multiplication with the weight matrix (W_1) and then it propagates the information after applying a non-linear activation function (Rectified Linear Unit (ReLU)[22] in our case) to it. The next hidden layer takes these values and performs the same operation before propagating these values to the

next layer. It is worth to mention that, the probability of each of the neurons information to be propagated depends on the dropout probability[23] . Here, the dropout probability was 20% for the input layer and 50% for the hidden layers. The network architecture can be expressed mathematically as follows:

$$\begin{aligned} h_1 &= ReLU(x \times W_1 + b_1) \\ h_2 &= ReLU(h_1 \times W_2 + b_2) \\ h_3 &= ReLU(h_2 \times W_3 + b_3) \\ &\dots \\ &\dots \\ h_n &= ReLU(h_{n-1} \times W_n + b_n) \end{aligned} \tag{1}$$

Where $h_1...h_n$ are the hidden layers, x is the input, $W_1...W_n$ are the weights and $b_1...b_n$ are the biases for each of the corresponding hidden layers.

In designing DLSCORE architecture, we divided the entire method into two parts:

First, we tried to find the number of fully connected hidden layers and the number of neurons in each of the layers that perform best. We chose the number of neurons in the hidden layers to be a subset of {128, 256, 512, 768, 1024, 2048}. By taking all possible combination and permutation, we came up with 55,986 different neural networks.

Second, we trained the networks with the following parameters:

Table 1: Training parameters

Optimization	Adam
Learning rate	0.001
Loss function	Mean Squared Error
Activation function	ReLU[22]
Dropout rate	20% (input layer), 50% (hidden layers)

We did not rely on a single best network, instead, we took an ensemble of multiple better performing networks. Since each network may capture different features, they might

be performing better for some but not for every system. So, to have consistent results, it is better to use the predictions from multiple networks and take the ensemble average. Therefore, after training the networks, we sorted them based on their performances on the validation set (see Results and Discussions) and put the best performing networks in the ensemble.

Evaluation metrics

The networks were evaluated using statistical metrics, including, standard deviation, mean square error (MSE), mean absolute error (MAE), RMSE, Pearson, Spearman rho and Kendall Tau correlation coefficients.The mathematical models for MAE and RMSE are given below:

$$\begin{aligned} MAE &= \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \\ RMSE &= \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2} \end{aligned} \tag{2}$$

Where y_j and $\hat{y}_j$ represent the experimental and the predicted binding affinity, respectively. The MAE measures the average magnitude of the errors in their binding affinity predictions, while the $RMSE$ measures the ability of DLSCORE to properly identify a small prediction range of the predicted vs the experimental values. We used a 1 - 2 kcal/mol confidence limits to test the scoring functions overall performance..

Results and Discussions

Initially, we trained DLSCORE with a total of 3,191 protein-ligand complexes using molecular BINANA descriptors[20] . During the training, we performed a 10-fold cross-validation in order to obtain unbiased results.

We chose the best 100 networks based on the performance (Pearson R^2) on the validation

set while training. The second step was to adjust the other hyperparameters like dropout rate, learning rate, $L1 - L2$ regularization, etc. However, we did not see any noticeable difference in the overall performance of these 100 networks while tuning the hyper-parameters mentioned. Therefore, we kept the initial configuration of these networks.

Since getting a prediction from an ensemble of 100 networks is time consuming, we analyzed different size of ensembles so that we can come up with a smaller ensemble that gives us the optimum performance. We looked at the comparative statistics (Pearson, Spearman Kendall, $RMSE$, and MAE) for different size of ensembles (Fig. 3) and noticed that the optimal performance (highest correlation coefficients and lowest $RMSE$ and MAE) is achieved with the top 10 networks. Moreover, choosing this subset of networks over a hundred gives us a 10x speedup of the program. The performance of the top 15 networks was looked at more deeply, and the results are available in the supplementary information (SI Fig. **??**). Based on the Pearson correlation coefficient, we chose the default number of networks for our model to be 10. An extra feature where the user is allowed to choose the number of networks has been added to DLSCORE.

When executing DLSCORE, NNScore 2.0, and Vina using the test set from the PDBbind v.2016 refined set , we obtained results where our deep learning scoring function outperformed NNScore 2.0 and Vina (Table 2, Fig. 4).

Table 2: Main statistics of binding affinity predictions of DLSCORE, NNScore 2.0 and Vina after testing it with 300 refined protein-ligand complexes.

Statistical value	DLSCORE	NNScore 2.0	Vina
N (sample size)	300	300	300
RMSE ($kcal/mol$)	1.15	2.78	3.17
MAE ($kcal/mol$)	0.86	2.03	2.50
Max possible correlation	0.98	0.98	0.98
Pearson R^2	0.82	0.21	0.15
Spearman rho	0.90	0.47	0.39
Kendall tau	0.74	0.33	0.27

When compared the three scoring functions (DLSCORE, NNScore 2.0, and Vina) with

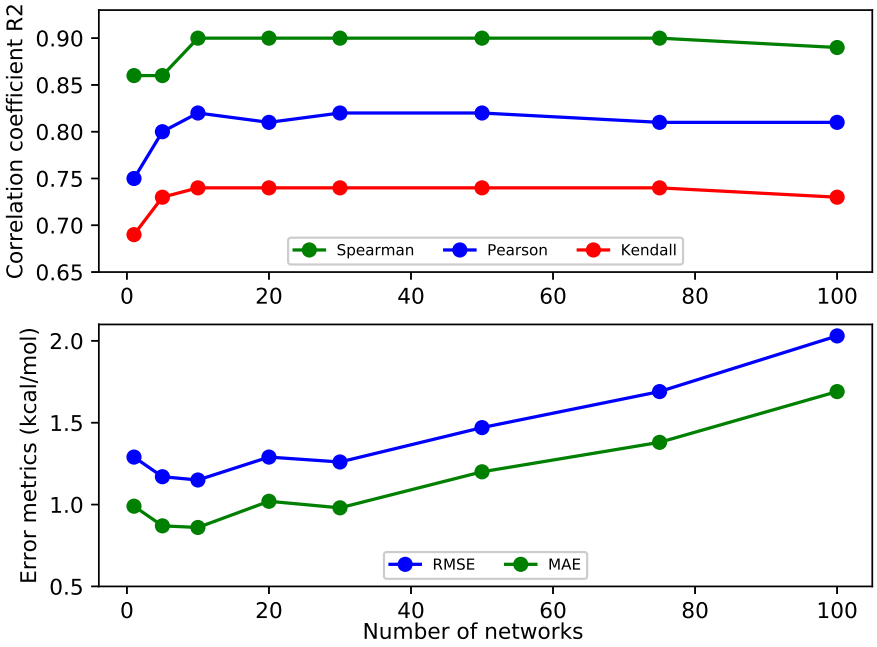
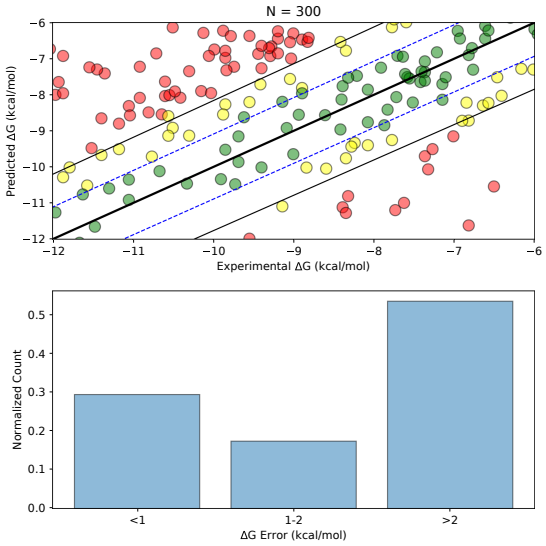
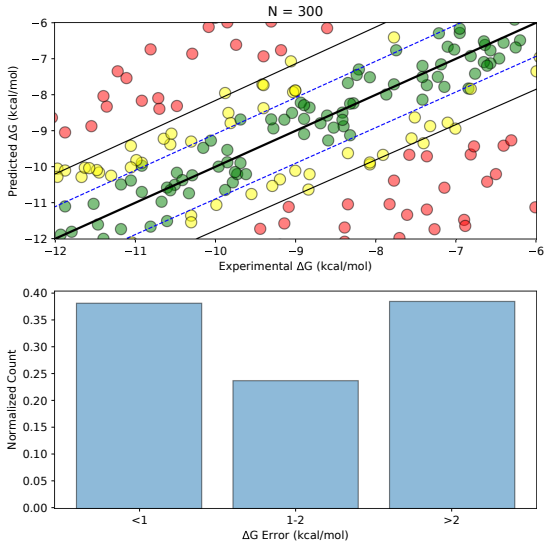


Figure 3: Plots displaying the statistical values of DLSCORE as a function of the number of networks. The first plot (above) shows the correlation coefficient values of Spearman, Pearson, and Kendall. The second plot (below) shows the $RMSE$ and MAE values in terms of $kcal/mol$

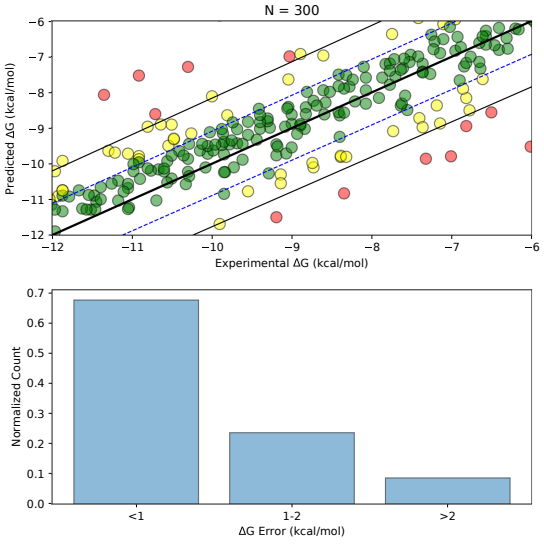
PDBbind v.2016 refined data, DLSCORE had the optimal performance, getting the closest values to the experimental data (Fig. 5a) in terms of ΔG values. Vina obtained 88 protein-ligand complexes (29.33% of the total data) with a difference less than 1 $kcal/mol$ of the experimental values, 52 data points (17.33%) within 1-2 $kcal/mol$ boundaries, and 160 (53.34%) were greater than 2 $kcal/mol$. NNScore 2.0 got 114 values (38%) less than 1 $kcal/mol$, 71 (23.67%) between 1-2 $kcal/mol$ and 115 (38.33%) were higher than 2 $kcal/mol$. On the other hand, DLSCORE outperformed the other scoring functions, where 203 data points (67.67% of the total data) were less than 1 $kcal/mol$ away from the experimental values, 71 (23.67%) were within 1-2 $kcal/mol$, and the 26 (8.66%) remaining were found outside the 2 $kcal/mol$ boundaries. Moreover, DLSCORE appears to have less variability, but a bigger number of outliers (Fig 5b), while NNScore 2.0 shows a greater standard deviation, but fewer outliers. Likewise, Vina displays slightly similar variability with NNScore



(a) Vina



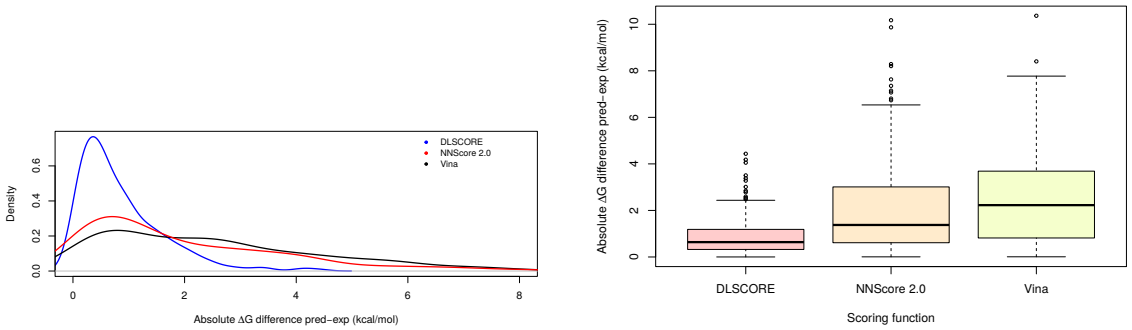
(b) NNScore 2.0



(c) DLSCORE

Figure 4: Graphs showing the predicted values within 1 $kcal/mol$ (dotted line) and 2 $kcal/mol$ (solid line) range. Green dots represent a predicted score less than 1 $kcal/mol$ away from the experimental value. Yellow dots represent a predicted score between 1 $kcal/mol$ and 2 $kcal/mol$ of the experimental value. Red dots represent a predicted score greater than 2 $kcal/mol$ away from the experimental value.

2.0, but fewer outliers. Both NNScore 2.0 and Vina have a max value of approximately 10.17 $kcal/mol$ and 10.36 $kcal/mol$ (respectively) between the predicted and experimental values, while DLSCORE has a max value of 4.43 $kcal/mol$.



(a) (b)

Figure 5: Graphs showing the absolute difference between the predicted and the experimental values in terms of ΔG $(kcal/mol)$ given three scoring functions (DLSCORE, NNScore 2.0 and Vina). Figure 5a displays the density plot behavior. Figure 5b shows the skewness, variabiblity and normality in a side by side boxplot representation.

Conclusion

Deep learning has been shown to be a useful approach of machine learning when it comes to protein-ligand binding affinity predictions. DL can take advantage of larger data sets in improving accuracy. Here, DLSCORE has proven to be a suitable DL machine learning algorithm for the accurate prediction of binding affinities of crystalized structures, outperforming NNScore 2.0 and Autodock Vina, yielding the best $RMSE$, MAE, and R^2 correlation coefficients, as well as other statistical values. Furthermore DLSCORE has proven to show more consistency in its results, as it has less variability, and a less max. value when in its absolute difference between the predicted affinity and the experimental data.

Implementation and Accessibility

We used Keras and Tensorflow to create and train all the neural networks. A system with 28 core (14 per socket) Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz processor, 128 GB of RAM and 8 GeForce GTX 1080Ti GPUs was used for training. The code is available for the public via GitHub at `https://github.com/sirimullalab/DLSCORE`. The workflow is available at `https://github.com/sirimullalab/CADD-Workflows/blob/master/scoring_function_workflow.ipynb` in the form of a JuptyterLab Notebook.

Acknowledgement

We would like to thank Dr. Patrick Walters at Relay Therapeutics for making his "metk" [24] code available on GitHub. We thank the High- Performance Computing support staff (Marc T. Hertlein and Leopoldo A. Hernandez) at The University of Texas at El Paso for assistance in using the Chanti cluster. We also acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. URL: `http://www.tacc.utexas.edu`

Funding

This work is supported by Dr. Suman Sirimulla's startup fund from UTEP School of Pharmacy.

Author Contributions

SS conceived and supervised the project. OF co-supervised the development of DLSCORE model. MH developed and trained the DLSCORE model. DCM performed the statistical

analysis, created the workflow, and prepared the files. All the authors contributed to the analysis of results. DCM, MH, SS wrote the manuscript. All authors reviewed and edited the manuscript.

Corresponding Author

E-mail: ssirimulla@utep.edu

Conflicts of Interest

The authors did not declare any conflicts of interest.

Supporting Information Available

The Supporting Information is available free of charge on the ACS Publications website at DOI:XXXX

References

(1) Trott, O.; Olson, A. J. *J Comput Chem* **2010**, *31*, 455–461.

(2) Jones, G.; Willett, P.; Glen, R. C.; Leach, A. R.; Taylor, R. *J Mol Biol* **1997**, *267*, 727–748.

(3) Jain, A. N. *J Med Chem* **2003**, *46*, 499–511.

(4) Friesner, R. A.; Banks, J. L.; Murphy, R. B.; Halgren, T. A.; Klicic, J. J.; Mainz, D. T.; Repasky, M. P.; Knoll, E. H.; Shelley, M.; Perry, J. K.; Shaw, D. E.; Francis, P.; Shenkin, P. S. *J Med Chem* **2004**, *47*, 1739–1749.

(5) Koebel, M. R.; Schmadeke, G.; Posner, R. G.; Sirimulla, S. *Journal of Cheminformatics* **2016**, *8*, 27.

(6) Koebel, M. R.; Cooper, A.; Schmadeke, G.; Jeon, S.; Narayan, M.; Sirimulla, S. *Journal of Chemical Information and Modeling* **2016**, *56*, 2298–2309, PMID: 27936771.

(7) Ain, Q. U.; Aleksandrova, A.; Roessler, F. D.; Ballester, P. J. *Wiley Interdiscip Rev Comput Mol Sci* **2015**, *5*, 405–424.

(8) Durrant, J. D.; McCammon, J. A. *Journal of Chemical Information and Modeling* **2011**, *51*, 2897–2903.

(9) Ballester, P. J.; Mitchell, J. B. O. *Bioinformatics (Oxford, England)* **2010**, *26*, 1169–1175.

(10) Kinnings, S. L.; Liu, N.; Tonge, P. J.; Jackson, R. M.; Xie, L.; Bourne, P. E. *J Chem Inf Model* **2011**, *51*, 408–419.

(11) Gomes, J.; Bharath, R.; Evan, N. F.; Vijay, P. **2017**,

(12) Chen, H.; Engkvist, O.; Wang, Y.; Olivecrona, M.; Blaschke, T. *Drug Discovery Today* **2018**,

(13) Pereira, J. C.; Caffarena, E. R.; dos Santos, C. N. *Journal of Chemical Information and Modeling* **2016**, *56*, 2495–2506.

(14) Cao, C.; Liu, F.; Tan, H.; Song, D.; Shu, W.; Li, W.; Zhou, Y.; Bo, X.; Xie, Z. *Genomics, Proteomics & Bioinformatics* **2018**, *16*, 17–32.

(15) Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. *J Chem Inf Model* **2017**, *57*, 942–957.

(16) Jiménez, J.; Škalič, M.; Martínez-Rosell, G.; De Fabritiis, G. *Journal of Chemical Information and Modeling* **2018**, *58*, 287–296.

(17) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. *Nucleic Acids Res* **2000**, *28*, 235–242.

(18) Renxiao, W. Beginner's Guide to the PDBbind Database (v.2017). 2017; `http://www.pdbbind.org.cn/download/pdbbind_2017_intro.pdf`.

(19) Khamis, M. A.; Gomaa, W. *Engineering Applications of Artificial Intelligence* **2015**, *45*, 136–151.

(20) Durrant, J. D.; McCammon, J. A. *J Mol Graph Model* **2011**, *29*, 888–893.

(21) Morris, G. M.; Huey, R.; Lindstrom, W.; Sanner, M. F.; Belew, R. K.; Goodsell, D. S.; Olson, A. J. *J Comput Chem* **2009**, *30*, 2785–2791.

(22) Nair, V.; Hinton, G. E. Rectified linear units improve restricted boltzmann machines. 2010.

(23) Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.

(24) Walters, P. metk. 2018; `https://github.com/PatWalters/metk`.

Graphical TOC Entry