OnionNet: a multiple-layer inter-molecular contact based deep neural network for protein-ligand binding affinity prediction

Liangzhen Zheng, Jingrong Fan and Yuguang Mu*

School of Biological Sciences, Nanyang Technological University, Singapore, 637551

* Corresponding author. ygmu@ntu.edu.sg

1. Neural networks model selections

We compared the performance of the fully-connected multiple layer perceptron (MLP) models with the CNN models, as well as the modified CNN models.

For the 1st CNN model (CNN-01), the input data was submitted to three two-dimensional (2D) convolutional layers (with 3 max-pooling layers after each one of the convolutional layers), and the output was flattened and subjected to 4 fully connected dense layers (with 400, 200, 100 and 1 neurons). A 2nd CNN model (CNN-02) was constructed based on three one-dimensional (1D) convolutional layers. The subsequent 4 dense layers were exactly the same as those in CNN-01 model. For both the CNN-01 and CNN-02 models, the filter sizes were 32, 64 and 128 for the 3 1D or 2D convolutional layers. The kernel sizes were 4, and stride was 1, and no padding was applied in the convolutional layers. For the max-pooling layers, kernel sizes were set as 2 and stride sizes were 1. The ReLu activation was used for each layer (including the final output layer with 1 neuron). Batch normalization and dropout layers were also added after the 3 convolutional layers and the first 3 dense layers. The SGD optimization was adopted with an initial learning rate as 0.0001.

For the MLP models, the first one (MLP-01) was constructed based on 4 dense layers (with 400, 200, 100 and 1 neurons), while the 2nd MLP model (MLP-02) has one more dense layer (1000 neurons) preceding the 4 dense layers as adopted in MLP-01 model. For both MLP models (MLP-01 and MLP-02), similarly, batch normalization and dropout layers were also added after the 3 convolutional layers and the first 3 dense layers. The SGD optimization was adopted with an initial learning rate as 0.0001.

Further, we also tested several modified CNN models. A first modified CNN model (mCNN-01) was constructed based on CNN-01, by removing the max-pooling layers after each one of the convolutional 2D layers, while other parts remained the same. Another modified CNN model (mCNN-02) was revised based on mCNN-01 by changing the filter sizes in 3 2D convolutional layers from (32, 64, 128) to (128, 64, 32), while kept all other parts remain unchanged. A 3rd modified CNN model (mCNN-03) was designed based on mCNN-02 model by changing the 2D convolutional layers into 1D convolutional layers but didn't change any other parts. The SGD optimization was adopted with an initial learning rate as 0.0001.

For each independent trial in all the above models, the loss was monitored to determine the stopping point of training. Here, we defined that for one specific network structure (either a DNN model, a CNN model or a modified CNN model), one combination of hyper-parameters (batch size, dropout rate and α) is one trial run. To compare the performance of different trials, R was used. Because with different α, for different trials, the loss values are not in the same scale, thus we used R values to compare the power of different models. Besides, we observed very large RMSE when α=1.0 in all related trials, therefore when we discuss the performance of different trials, we ignored all the trials with α=1.0.

Besides the DNN models, we also tested the performance of two random forest (RF) models with different number of decision trees. The RandomForest regression module in Sci-kit learn was used to train the two RF models (RF-01 and RF-02). In these two models, 1000 and 2000 trees were adopted respectively, and the "MSE" criterion was selected, while for all the other parameters default options had been chosen. 10 repeated training and validating were performed for the two models (Supplementary Table 2). For the validating performance, it is quite clear that the *R* values (0.725~0.753) for both RF-01 and RF-02 are smaller than those of the best mCNN-01 models (~0.78), meanwhile the *RMSE* values (1.372~1.435) of RF models are much higher than the RMSEs of the best mCNN-01 models (~1.29). Overall speaking, RF models trained with our training and validating set could generate less accurate results when comparing to mCNN-01 models.

Table S1. The DNN models tested in this study

Model name	Model type	With pooling	Filter-sizes in convolutional layers	Dense-layer-sizes	Convolution shape
CNN-01	CNN	Yes	32+64+128	400+200+100+1	2D
CNN-02	CNN	Yes	32+64+128	400+200+100+1	1D
MLP-01	MLP	N.A.	N.A.	400+200+100+1	N.A.
MLP-02	MLP	N.A.	N.A.	1000+400+200+100+1	N.A.
mCNN-01	Modified CNN	No	32+64+128	400+200+100+1	2D
mCNN-02	Modified CNN	No	128+64+32	400+200+100+1	2D
mCNN-03	Modified CNN	No	128+64+32	400+200+100+1	1D

Table S2. The best performance (based on the largest validating R^*) achieved in different models

Model name	Training RMSE	Training R	Validating RMSE	Validating R
CNN-01	0.5732	0.9553	1.4189	0.7237
CNN-02	0.4397	0.972	1.3479	0.7614
MLP-01	0.5282	0.9594	1.3169	0.7786
MLP-02	0.631	0.9459	1.32	0.7658
mCNN-01	0.2734	0.9895	1.2834	0.7809
mCNN-02	0.5175	0.9662	1.3159	0.769
mCNN-03	0.3435	0.9835	1.327	0.7632

* The conditions when alpha=1.0 were not considered.

We systematically evaluated the performance (*Loss*, *RMSE* and *R*) difference between the models with different batch sizes, dropout probabilities, and the α values. Through large scale comparison, we firstly confirmed that 2D-convoluational networks (CNN-01) outperform the 1D-convolutional networks (CNN-02). Next, by removing the pooling layers (mCNN-01, the performance of the networks improves further. We also evaluated the MLP networks and the results of the mCNN-01 networks are also better than those from the MLP-01 and MLP-02 models. Lastly, we tried to change the filter sizes in the modified CNN models (mCNN-02 and mCNN-03), however, the performance has not been increased further.

Therefore, our final OnionNet model was designed based on a CNN model without max-pooling layers. We systemically trained the final model (mCNN-01) with different batch size, dropout rate and customized loss function tuning factor α. The best model was obtained without dropout and with a 128 batch size and α=0.8. We then performed 20 independent repeated training of mCNN-01 model with shuffled training

samples (Supplementary Table 3). The performance of these mCNN-01 models was quite stable, the standard deviations of *RMSE* and *R* are 0.012 and 0.004 respectively.

2. Model training and hyper-parameter tuning for DNN models

The deep neural network (DNN) models were trained on NVIDIA GTX 1070 GPUs with Tensorflow.keras module [1, 2]. An early-stopping strategy was adopted to avoid over-fitting with a patience as 40 epochs and a min_delta = 0.01 for the validating set loss. Therefore, a best model is harnessed if the learning enters an overfitting stage and the decreasing of the loss smaller than min_delta for over 40 epochs. The best model containing weights was saved into an HDF5 file.

Here in this study, we introduced a customized loss function to specifically hope to obtain models with high *R* values. Therefore, we explicitly incorporated *R* into the loss function with a scaling factor α.

We fine-tuned the hyper-parameters (batch size, drop out probability, kernel size, and alpha value for the customized loss function). We used the performance (largest *R* without considering the conditions when α=1.0) of the validating set as an indication to determine the stopping condition for the training with different parameter combinations. The final model was selected based on the best performance for the testing set. The choices for batch size were 16, 32, 64, 128, 256 and 512. Whereas for dropout rate the choices were 0.0, 0.1, 0.2, 0.3, 0.4, 0.5 and 0.6. Lastly, α (0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0) was also screened. Therefore, we performed model training trials with 6 * 7 * 10 = 420 different combinations for batch size, dropout rate and α. The performance of these trials could be found in the Supplementary Table 2.

For the final model mCNN-01 with a batch size 128, α=0.8 and without dropout, after around 89 epochs (Figure S1), the loss is about to stay unchanged, and according the early stopping strategy, the model was terminated at epoch=89, and we collected the model at epoch=89 and used it as the best OnionNet model.

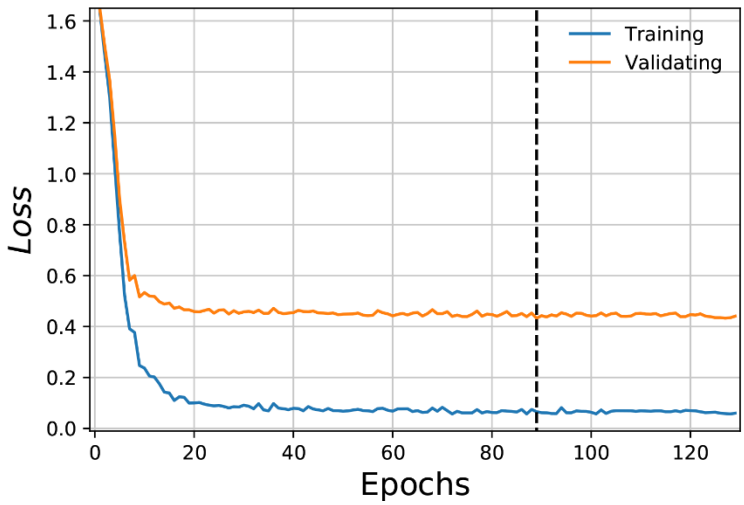

Figure S1. The decay of the training loss and validating loss along training epochs. The black dashed line indicates the position where validating loss stops decreasing according the early-stopping strategy we adopted.

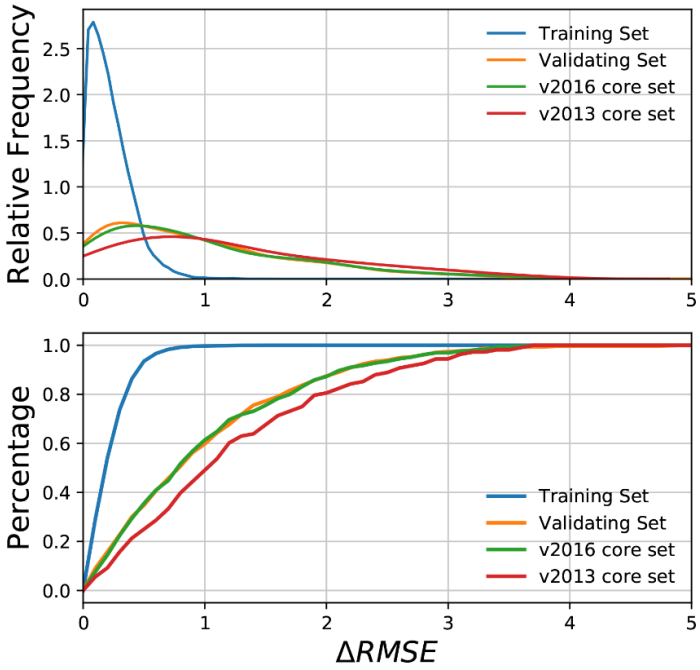

Figure S2. The $\Delta RMSE$ ($=|pK_{a\_true} - pK_{a\_predicted}|$) of the training, validating and testing sets.

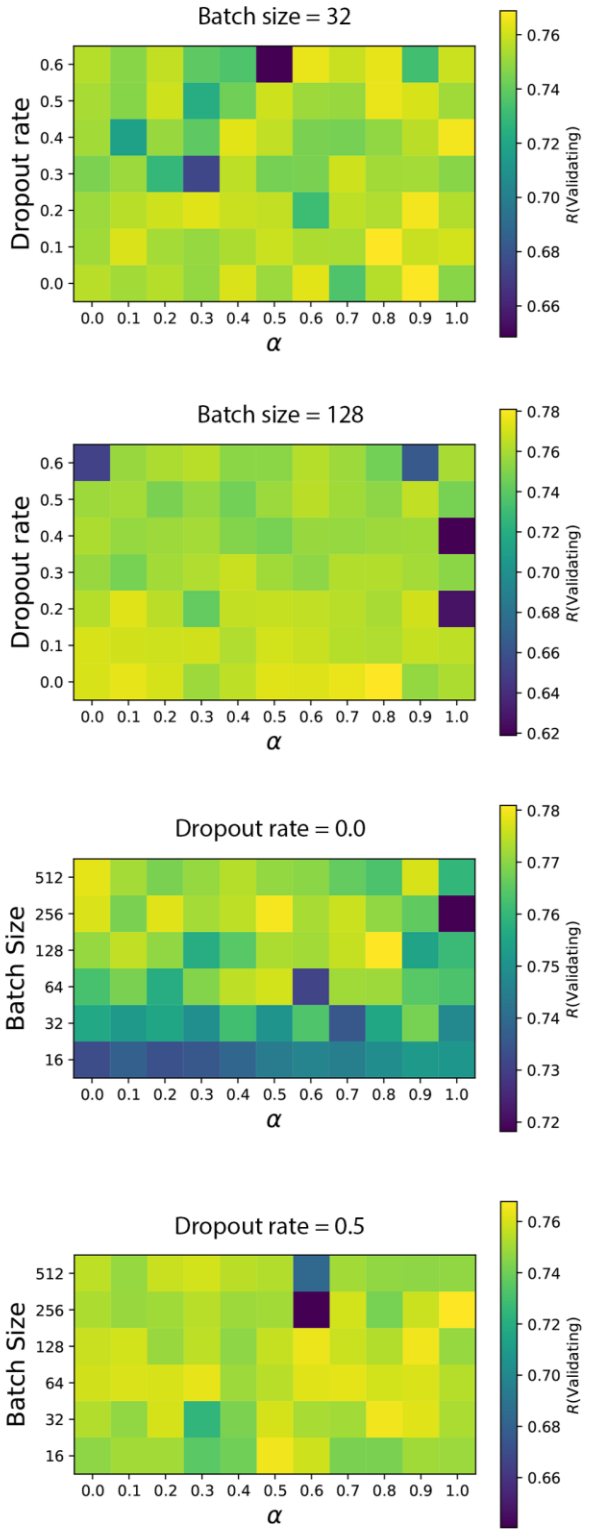

Figure S3. The performance of the mCNN-01 model with different batch size, dropout rate and α. When dropout rate equals to 0.0, the model was trained without applying dropout in the dense layers.

We also explored the influence of α in the model performance (based on *R*). Given that we adopted mCNN-02 model as the final OnionNet model structure, we screened the effects of α on the performance of different trials with different batch size and dropout rate. From Figure S3, it appears that larger α would lead to high *R* values. When batch size = 128 and dropout rate = 0.0, the performance is gradually increased when α increases. Similarly, when batch size = 16 and dropout rate = 0.0, the model could achieve higher performance when α adopts a larger value as well. Although the performance of the model is not strictly monotonic increasing with α, when batch size = 32, or dropout rate = 0.5, there are high-performance regions in the color map when α is between 0.5 to 0.9.

3. Model training with missing features

To test the stability and robustness of the OnionNet model, we artificially created different datasets with missing features. For the multiple-layer inter-molecular features, we generated a *M*×*N* matrix, containing samples (by rows) horizontally and features (by columns) vertically. In detail, for each of the 3840 features, if we need to remove one feature, we replaced the original values (in the specific column) to zeroes to maintain the dimensions of the dataset. This way the artificially generated dataset could be reshaped and trained with the same DNN architecture.

Firstly, we examined the stability of the model by removing the features (*n*-1) * 64 to *n**64 in shell *n* ($n \in [1, 64]$). For example, if n=1, we then remove features 1 to 64, and if n=6, we thus replaced the values in those features (from 320th to 384th) with zeros. For all the 60 shells, we generated 60 datasets with values being replaced by zero in specific features. Then for each one of the datasets, a mCNN-01 model was re-trained with the optimal batch size (128) and α (0.8) based on the same early stopping strategy. For each one of the 60 datasets, we re-trained the model 5 independent times. The Δ*Loss* of the 5 repeats was calculated based on the customized loss difference between the model and the best model without missing features. The mean $\Delta Loss$ and standard deviation of $\Delta Loss$ thus were also determined. Here this equation defines the $\Delta Loss$:$\Delta Loss = Loss_{missing} - Loss_{best}$

where the $Loss_{missing}$ and $Loss_{best}$=0.443 are the loss of the model with missing features and the best mCNN-01 model (with a batch size 128, α=0.8 and without dropout) we trained without missing features.

Similar, for models with missing specific element-type combination features, the values in the relevant features thus were replaced by zeroes. Since there are 64 different combinations, we generated 64 datasets and for each of them a model was trained using the same setting for 5 times independently with different random seeds. The model performance was evaluated in the same strategy.

In crystal structures, the coordinates of hydrogen atoms are missing. However, in PDBbind database, the positions of the hydrogen atoms were modeled. Therefore, we generated another dataset without any features based on element-type combinations containing hydrogen element. The dataset was fed to a DNN model (called OnionNet_HFree hereafter) and the performance was evaluated. It seems like that without including hydrogen elements, the model shows slightly lower performance for the validating dataset (Figure S4). Although the inclusion of hydrogen atom coordinates may introduce bias to the model, from the training results, however the model is robust enough to tolerate the bias.

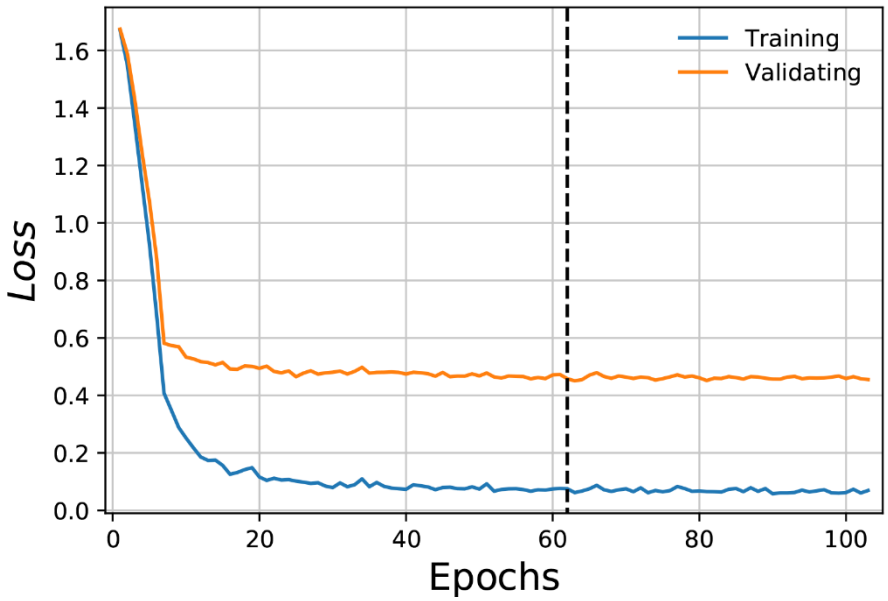

Figure S4. The decay curves of *RMSE* and *R* for the model training with a hydrogen-free dataset. The model (OnionNet_HFree) achieves the best performance at epoch = 62 (the black dashed line). The *RMSE* and *R* at epoch = 62 are 1.334 and 0.768 respectively.

4. Predicting pK_a of the "native-like" docking poses

The ability to predict the pK_a of a docking pose would enable the model a more powerful and practical tool as an alternative scoring function to enable large-scale virtual screening. Here, using AutoDock Vina [3], we docked a ligand into its native target binding pocket. And we selected the "good" protein-ligand docked complexes for pK_a prediction. If the root-mean-square-deviation (*RMSD*) between the docking pose and the native ligand conformation (in a crystal or NMR structure) is less than a *cutoff* = 2 Å, the complex of the target (protein) and the docking pose of the ligand thus is a "good" complex, or a "native-like" complex. For one protein-ligand complex, Vina was required to output 20 docking poses, out of which the best docking pose is selected based on minimum *RMSD*, and if the minimum *RMSD* is less than the *cutoff*, then the "good" complex is passed to generate multiple-layer element-type based inter-molecular features. This way a dataset was generated based on the docking poses and *RMSD*.

For the 290 complexes in the PDBbind v2016 core set, only 219 "good" complexes were generated. As for the rest 89 complexes in v2016 core set, the native-like poses have not been generated from docking. The 219 "good" complexes generated with Vina were submitted for inter-molecular featurization. An exhaustiveness= 32 and the box sizes=40 Å were adopted for docking, using the centroid of the native conformation of the ligand as pocket center. The full list of the 219 complexes was provided in the supplementary table 3.

The OnionNet_HFree model thus was used to predict the pK_a for the "good" complexes generated from docking simulations.

References

1. Chollet, F., *Keras*. 2015.
2. Abadi, M., et al. *Tensorflow: A system for large-scale machine learning*. in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016.
3. Trott, O. and A.J. Olson, *AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading.* J Comput Chem, 2010. **31**(2): p. 455-61.