Assessing the Reliability of AlphaFold3 Predictions forProtein-Ligand Affinity Prediction via Sfcnn

Guo Yu (2022533074) * **Linzheng Tang (2022533087)** *
Junlin Chen (2022533009) *

* *ShanghaiTech University (email: yuguo2022@shanghaitech.edu.cn, tanglzh2022@shanghaitech.edu.cn, chenjl2022@shanghaitech.edu.cn)*

Abstract: This project investigates the reliability of using AlphaFold3 (AF3)-predicted structures as an alternative. Sfcnn, a 3D-CNN based protein-ligand affinity prediction model, is reproduced using PyTorch, and its performance is validated on the PDBbind v2019 refined set for training and the CASF-2016 core set for testing. The AF3-derived protein structures of CASF-2016 core set is then evaluated and compared against the groundtruth and Sfcnn scores on the core set to assess the viability of AF3 predictions in PLA tasks.

Keywords: AlphaFold3, protein-ligand affinity, CNN scoring function, CASF-2016

1. INTRODUCTION

1.1 Sfcnn Background

Sfcnn is a 3D convolutional neural network based scoring function model proposed in 2022, which aims to provide accurate and reliable scores for binding affinities of protein-ligand complexes.

1.2 Data Methods

Dataset The Sfcnn network was trained with protein–ligand complexes from the refined set of the PDBbind database version 2019, which contains protein—ligand complexes and their corresponding binding affinities expressed with pKa values, the trained network is later tested on the CASF-2016 core set, which has 285 protein–ligand complexes.

Note that the overlaps between train set and test set (266 protein complexes) are excluded, leaving 4852 train complexes in total.

Augumentation To scale up the training set, each protein-ligand complex is rotated randomly for 9 times using random rotation matrices, those 10 complexes should bear the same PLA (protein-ligand affinity) score, resulting in total 48520 complexes for training

Featurization To capture the features of a protein-ligand complex, Sfcnn uses the method of grid mapping and one-hot encoding. Each complex is mapped to a 3D grid with resolution $20 \times 20 \times 20$, which is later transformed into a 4D tensor. Each cell within the grid is a formed by an encoding list of length 28, consists of 14 protein atom(isotope)[1] and 14 corresponding ligands, mapped with one-hot encoding method. The final training tensor size is therefore (48520, 20, 20, 20, 28).

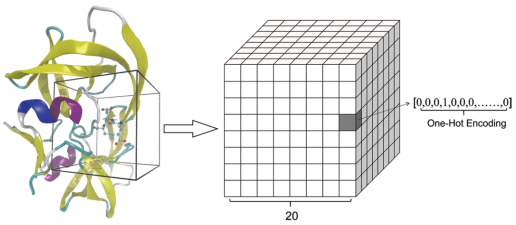

Fig. 1. Featurization of the protein–ligand complexes. PDB ID 1a30 is shown as an example. In the default case, the resolution of 20×20×20 and 28 categories of atomic types were used

1.3 Network

The original paper presents 4 different network structures along with 3 ways of featurization. The network shown in the figure, combing with the featurization method above achieved best performance on validation set.

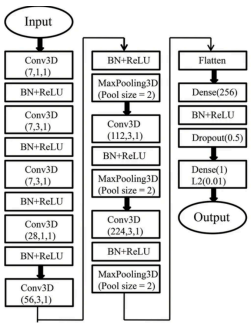

Fig. 2. Final CNN structure for Sfcnn network

[1] Please refer to the original Sfcnn paper for those atom types: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-022-04762-3#availability-of-data-and-materials

This network features 3D convolution layers with batch normalization and ReLU activation.L2 regularization was applied on the output layer to reduce the probability of overfitting and improve generalization.

2. REPRODUCTION

2.1 Data Method

Dataset and Featurization The reproduction pipeline uses the same dataset and featurization method, results in training 4D tensor, shaped (48520, 20, 20, 20, 28) testing 4D tensor, shaped (285, 20, 20, 20, 28).

Data storage It is worth noting that the original Sfcnn data storage uses the format of .pkl (pickle file), which features concatenate the full arrays first, then dump into the file at once. This approach requires to store and dispatch all the complexes' information within local memory, which would cause an extremely high memory consumptiondue to the high training data volume and is unfeasible on normal computers.

In alternative, our team switched to the format .h5 (h5py file), which supports instant writing and solve the issue, resulting in 40.1 GiB training grid.

2.2 Network

Structure The pytorch network structure is similar to the original tensorflow version except for two main difference:

- 1.Due to the Conv3D API requirement in pytorch, the input 4D tensor shape is permuted to (batch_size, 28, 20, 20, 20)
- 2.Pytorch lacks direct L2 regularization API, the final linear layer in the fully connected part is therefore set a weight decay to imitate the effect.

Training The training process is performed on training set and validation set, the validation set is partitioned from the training 4D tensor, indexed from 41000 to 48520, same as the original network. The final training set shape: (41000, 20, 20, 20, 28), validation set shape: (7520, 20, 20, 20, 28), the final dataset ratio is train : validation : test = 84.0% : 15.4% : 0.58%

Notice that the original training hyperparameters failed to converge in our experiments on the pytorch network, both the original hyperparameter and our current hyperparameter choice are presented in the following table:

Table 1. Original/Reproduced hyperparams

Param	Original	Reproduced
lr(learning rate)	0.004	0.002
batch size	64	32
dropout rate	0.5	0.15
L2 regularization/FC weight decay	0.01	0.01
epochs	200	200

2.3 Results

Metrics The performance of Sfcnn is measured by the following four main metrics:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} \left(y_{\mathrm{predict}} - y_{\mathrm{true}}\right)^2} \tag{1}$$

$$\mathrm{MAE} = \frac{1}{N} \cdot \sum_{i=1}^{N} |y_{\mathrm{predict}} - y_{\mathrm{true}}| \tag{2}$$

$$\mathrm{SD} = \sqrt{\frac{1}{N-1} \cdot \sum_{i=1}^{N} \left((a \cdot y_{\mathrm{predict}} + b) - y_{\mathrm{true}}\right)^2} \tag{3}$$

$$R = \frac{E\left[\left(y_{\mathrm{predict}} - \mu_{y_{\mathrm{predict}}}\right)\left(y_{\mathrm{true}} - \mu_{y_{\mathrm{true}}}\right)\right]}{\sigma_{y_{\mathrm{predict}}} \sigma_{y_{\mathrm{true}}}} \tag{4}$$

where a and b represent the slope and interception of the linear regression line of the predicted and measured values. $E[c \cdot]$ denotes the expectation. $\mu_{y_{\mathrm{predict}}}$ is the expectation of the predicted values. $\mu_{y_{\mathrm{true}}}$ is the expectation of the true values. $\sigma_{y_{\mathrm{predict}}}$ is the standard deviation of the predicted values. $\sigma_{y_{\mathrm{true}}}$ is the standard deviation of the true values.

The result is shown in the following table:

Table 2. Performance Metrics Comparison on CASF-2016 Core Set

Metrics	Reproduced Sfcnn	Original Sfcnn
Pearson R	0.7286	0.7928
RMSE	1.5481	1.3263
MAE	1.2579	1.0277
SD	1.4892	1.3252