Prediction of Protein-Ligand Binding Using Deep Learning
BS6207 Final Project

SONG YIFAN

1.0 Problem Statement

Many biological activities, such as membrane receptor signalling and enzyme catalysis, rely on molecular recognition between proteins and ligands. Two long-standing aims in molecular biophysics and medicinal chemistry are predicting the architectures of protein-ligand complexes and discovering ligands by virtual screening of small molecule databases [1, 2]. For modelling protein-ligand interactions, knowledge-based statistical potentials have been devised. They are based on intermolecular feature distributions in vast datasets of protein-ligand complexes. Deep learning in pharmaceutical research has evolved in recent years, and its value has gone beyond bioactivity predictions, showing promise in solving varied difficulties in drug discovery.

In this project, given the coordinate (X,Y,Z) and atom types ('C - Carbon', 'O – Oxygen', 'N – Nitrogen'). Here we only consider the hydrophobic ('h') and polar ('p'), where 'C' is interpreted as hydrophobic and the rest is interpreted as polar. We are required to train a neural network that input the coordinates and atoms types of a pair protein-ligand, and predict if they are bind at the output of the network.

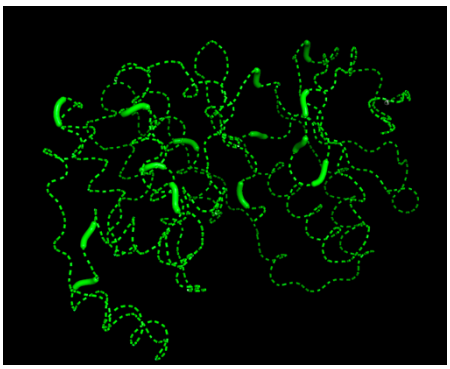


Figure 1:Protein structure in pyMOL (0001_pro.pdb)

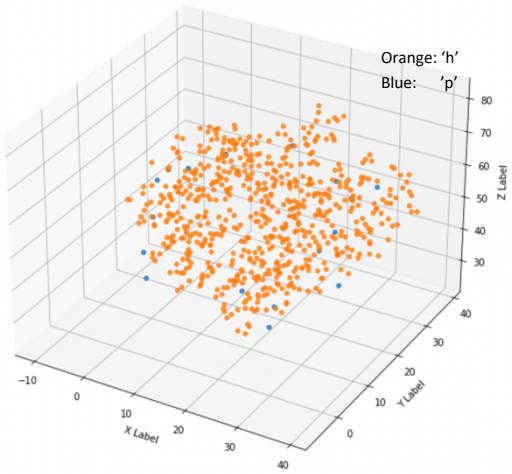


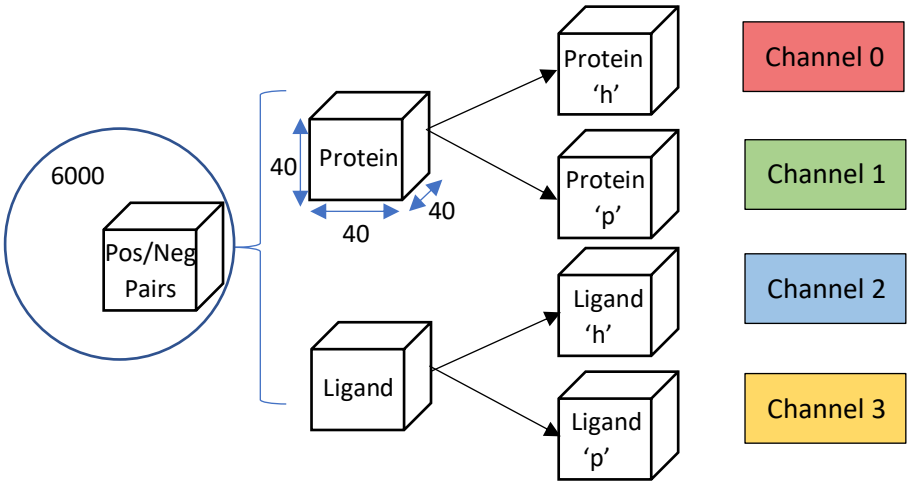
Figure 2: Protein structure in Voxel (0001_pro.pdb)

2.0 Dataset pre-processing

Input: A pair of protein and ligand with coordinates and atom type, which can be treated as a voxel. For each voxel, there are two types of atoms.
Label: Negative pair (0), Positive pair (1) --- Binary classification

Create Voxel:
First, create a voxel with width=20 around the centroid of ligand to contain the atoms as much as possible. If we choose smaller width, means some atoms will be exclude from the container. Then map the samples to the container with size [2*width, 2*width, 2*width]. The initial state of the voxel is 0, while a sample mapped, it is assigned 1. The final look of the 3D voxel could look like the Figure 2.

Second, combine ligand and protein with channels to form pairs. For both protein and ligand, there are 3D spaces, with channels for 'h' or 'p'. So, there are 4 channels:



For each pair of protein and ligand, if their file names of four digits (0001) are same, we assume they are a positive pair, otherwise, we assume they are negative pair. Since there are only positive pairs in the training set, we need to create negative pairs manually. I created the same number of positive and negative pairs to balance the data. And for train and validation split, I set ratio as 8:2, and shuffle was used. For the test data, there are 824*824 combinations.

	Training set (count)	Validation set (count)	Test set (count)
Positive pair	2400 labels=1	600 labels=1	824 *824
Negative pair	2400 labels=0	600 labels=0	

3.0 Highlights

1. In this project, a 3D CNN model was applied, and the structure shown below.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv3d-1       [-1, 32, 40, 40, 40]           3,488
       BatchNorm3d-2       [-1, 32, 40, 40, 40]              64
              ReLU-3       [-1, 32, 40, 40, 40]               0
         MaxPool3d-4       [-1, 32, 20, 20, 20]               0
            Conv3d-5       [-1, 64, 20, 20, 20]          55,360
       BatchNorm3d-6       [-1, 64, 20, 20, 20]             128
              ReLU-7       [-1, 64, 20, 20, 20]               0
         MaxPool3d-8       [-1, 64, 10, 10, 10]               0
            Conv3d-9      [-1, 128, 10, 10, 10]         221,312
       BatchNorm3d-10      [-1, 128, 10, 10, 10]             256
             ReLU-11      [-1, 128, 10, 10, 10]               0
        MaxPool3d-12        [-1, 128, 5, 5, 5]               0
           Conv3d-13        [-1, 256, 5, 5, 5]         884,992
       BatchNorm3d-14        [-1, 256, 5, 5, 5]             512
             ReLU-15        [-1, 256, 5, 5, 5]               0
        MaxPool3d-16        [-1, 256, 2, 2, 2]               0
          Flatten-17                [-1, 2048]               0
           Linear-18                 [-1, 512]       1,049,088
             ReLU-19                 [-1, 512]               0
          Dropout-20                 [-1, 512]               0
           Linear-21                   [-1, 2]           1,026
================================================================
Total params: 2,216,226
Trainable params: 2,216,226
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.98
Forward/backward pass size (MB): 64.86
Params size (MB): 8.45
Estimated Total Size (MB): 74.29
----------------------------------------------------------------
```

For each 3Dconvolutional layer:
Kernel size =3,
Stride =1,
Padding =1
Padding was used to avoid losing much original data.

2. Balance Positive and Negative pairs- Using the same size of pos and neg pairs
3. Using SoftMax to turn the prediction result into probabilities that sum to 1.

4.0 Training and testing procedure

Training:

Epoch	30
Learning rate	0.001
Batch size	64
Criterion	CrossEntropyLoss
Optimizer	Adam

When I am training the model, I found that when the epoch reaches about 15, it reaches high accuracy and starts to overfit. Therefore, I choose epoch as 30. For the criterion, I choose cross entropy loss, which is better than MSE loss in the classification problem. Since the decision boundary in a classification task is large compared with regression. MSE doesn't punish misclassifications enough but is the right loss for regression, where the distance between two values that can be predicted is small.

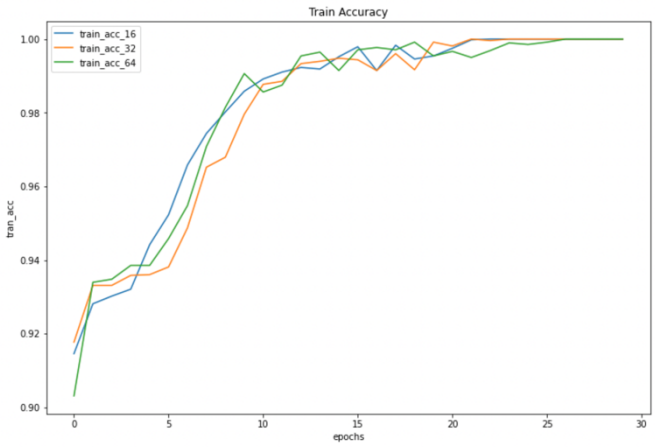
Testing:

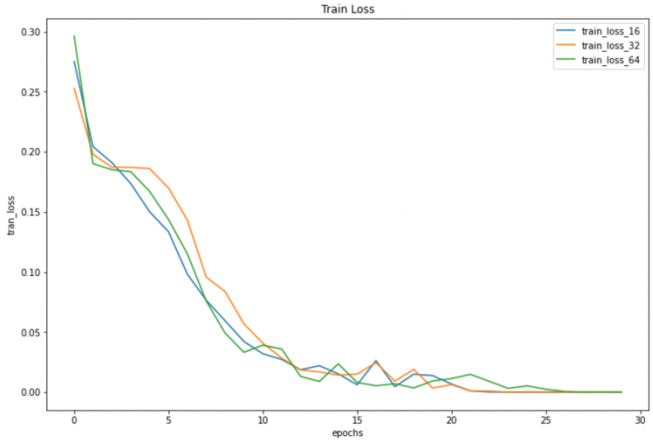
Load test data of 824*824 = 678976 pairs. Here I used no_grad because we are not training. Then apply the model to input test data, we got the output as a result. I used SoftMax activation function to replace the output as probabilities from 0 to1. This could make clearer insights of the result. While larger probability means easier to bind. So we just sort the result, and got the top 10 candidates.

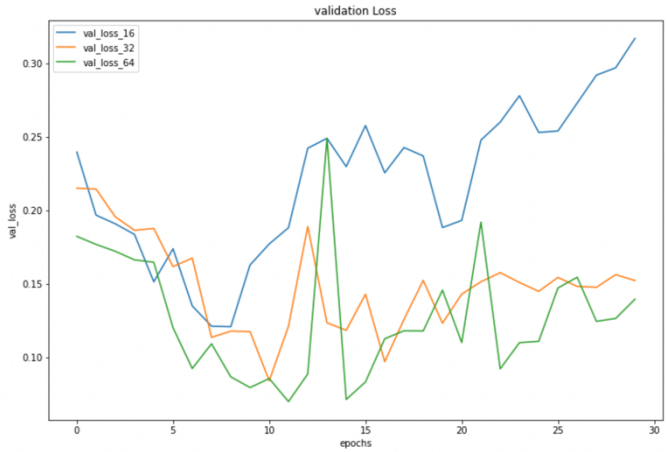
5.0 Experimental study

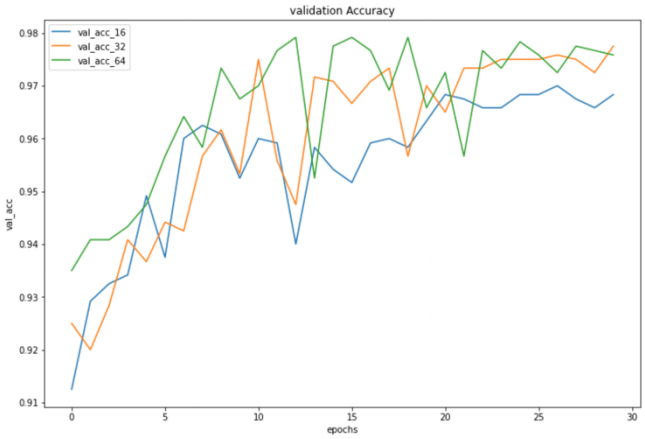
Try different batch size [16,32,64], and finally choose batch_size=64

From the training accuracy and loss, we can see that there is no obvious difference of the different batch size. The model started to overfit around the epoch = 13. For validation accuracy and loss, we can observe that the performance is worst with batch_size =16. Compared with batch_size=32 and batch_size =64, I think batch_size =64 is better, since the validation loss is lower than batch_size =32.









6.0 Results

pro_id	lig1_id	lig2_id	lig3_id	lig4_id	lig5_id	lig6_id	lig7_id	lig8_id	lig9_id	lig10_id
1	388	664	78	156	731	468	213	766	151	489
2	651	285	85	148	521	367	777	334	315	747
3	593	43	479	20	3	638	543	60	241	777
4	701	649	46	666	120	674	421	360	144	227
5	90	85	222	91	624	505	204	815	194	488
6	714	137	407	79	576	606	404	583	202	661
7	454	426	138	75	368	665	411	335	667	768
8	172	589	3	191	544	243	30	690	457	577
9	790	362	20	565	746	584	450	591	359	170
10	234	601	350	360	120	166	707	796	211	327

7.0 Future Work

There is a limitation of the voxel size, since my computer is not strong enough, I can only use small size (40*40*40) instead. Such a small voxel cannot guarantee that every molecule is included, so there is a risk that information will be lost, resulting in inaccurate model predictions. In the future, I may try different voxel size, and choose the best one. At the same time, I want to explore more on the trade-off between the model accuracy and computer memory.

The other direction of is to extract the features of each layer to get more insights of the 3D CNN model.

Reference

1) https://en.wikipedia.org/wiki/Protein
2) https://en.wikipedia.org/wiki/Protein_structure