BS 6207 Project- *Tian Siqi, G2101015G*

Github link: https://github.com/SiqiT/Assigmnment

1. Problem definition

In this project, we need to train a neural network that take in as input, the(x,y,z) coordinates of atoms and atom type of a pair of protein and the ligand, and then predict if they bind at the output of the network.

2. Highlights

Protein structure is the three-dimensional arrangement of atoms in an amino acid-chain molecule.[1] Since we are focus on the binding between ligand and protein, in this project we only focused on **a specific part of a protein structure**, which is closed to the ligand location. Thus we need to do some pre-processing to remove unimportant atoms.

Secondly, we need to form a dataset that contains both **positive records and negative records**, which means we have to perform different ligand-protein combinations, it's important to use a proper strategy to define the size of the NP dataset, or the label imbalanced problem would occur.

Last but not least, when everything is set down, we need to design a CNN to achieve the prediction. In this part, we need to design each layer carefully and also consider the proper size of input data or numbers of channels.

3. Dataset pre-processing description

(1) Relocate Protein & Ligand Atoms
Since we are trying to predict the relationship between the ligand and proteins, I think the first step is to relocate the ligands to the center of the coordinate and the new protein atoms' location are relocated by using the ligand atoms as the center points.
Figure 1. original location(pro0-lig0, pro0-lig1, pro1-lig1)

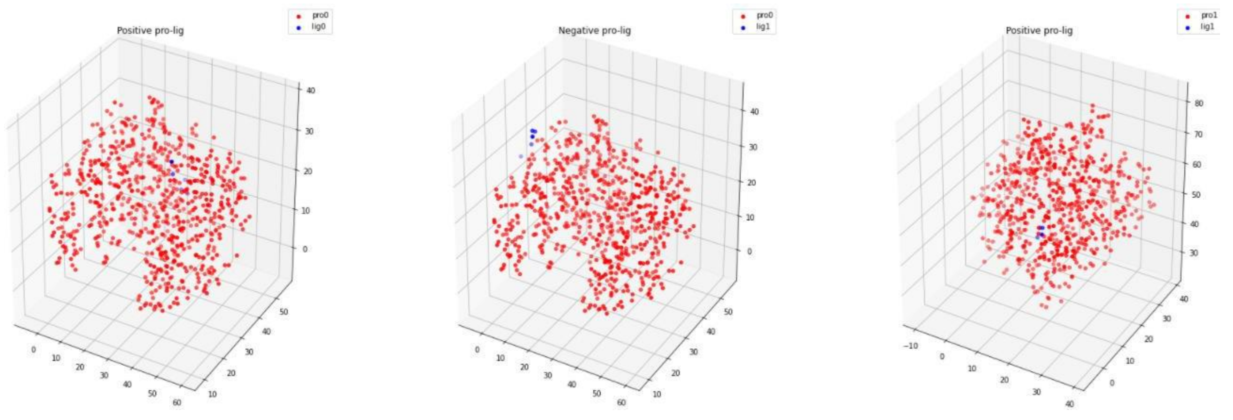
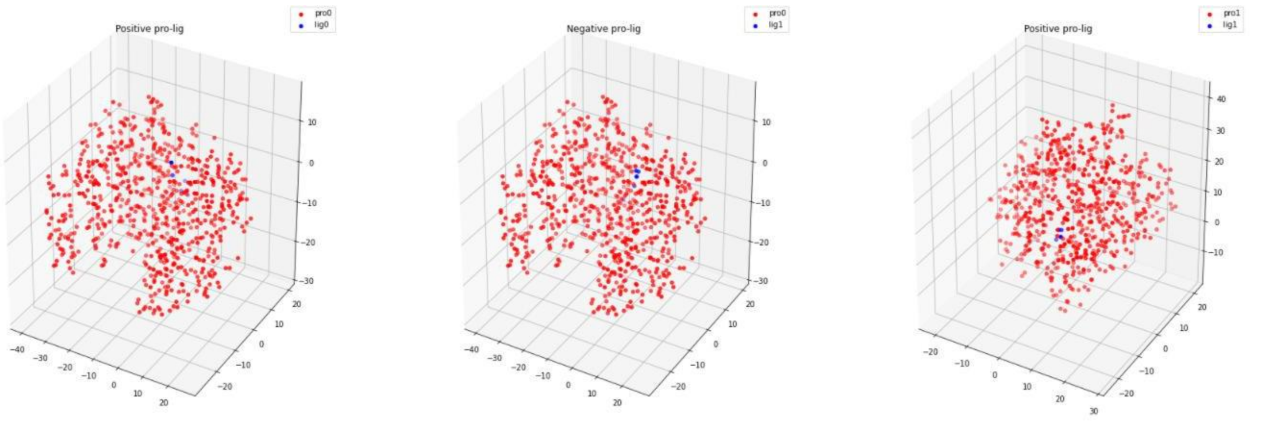


Figure 2. Relocated location(pro0-lig0, pro0-lig1, pro1-lig1)

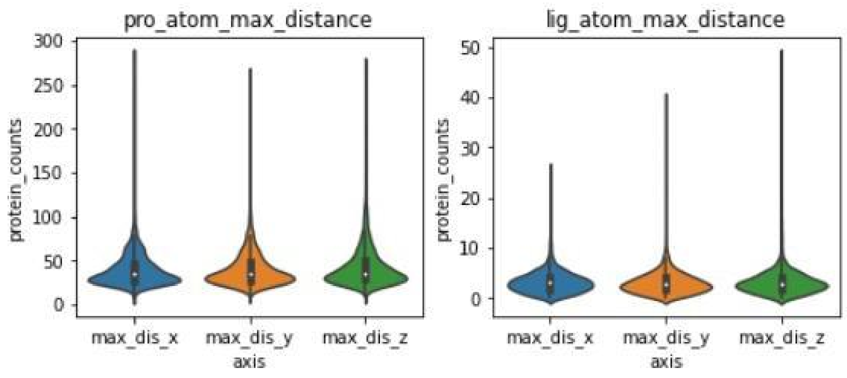


From the axis values and the location of ligand atoms(blue) and protein atoms(red), we can see after this processing, protein atoms are located around the ligand atoms, no matter they are matched or not, This is more convenient for further analysis, since we need to use the distance between protein atoms and ligand atoms.

(2) Pick subset of protein atoms

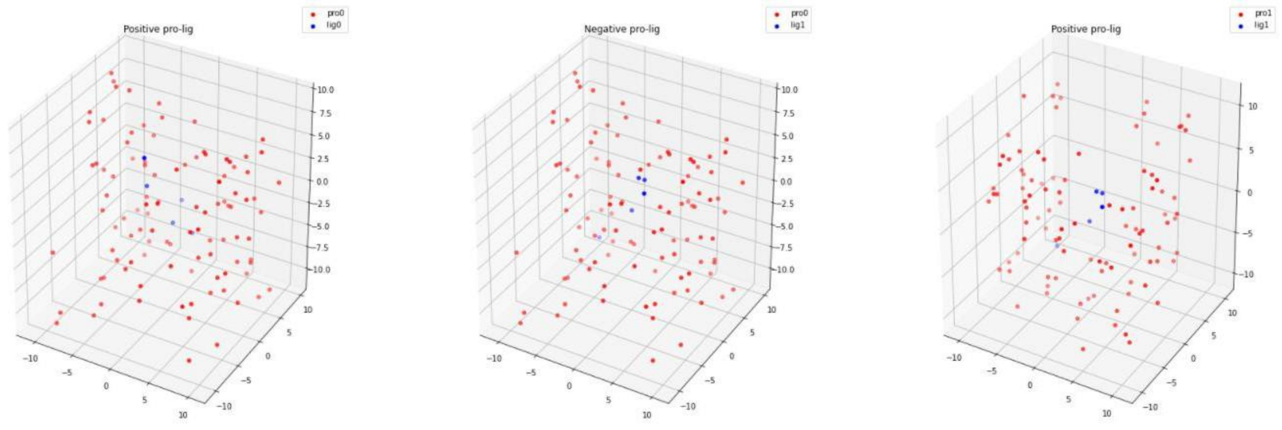
Here we focus on the area around the ligand atoms to do the prediction, so in this part we removed some unimportant protein atoms that are far from the ligand atoms. Here we keep the protein atoms that have a **distance < 11** from the center points of all ligand atoms.

Figure 3. Violin Plot of atom_distance to the ligand center



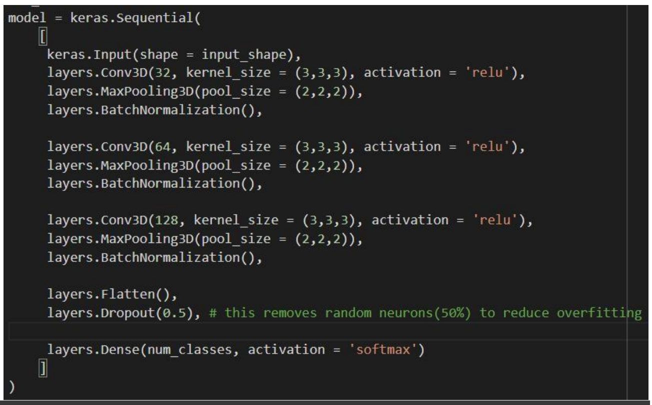
From this violin plot we can see most of the distance between ligand atoms are less than 10, the median distance between protein and ligand are about 25, but due to the GPU limitation, here I choose width = 11 as the threshold.

Figure 4. Selected atoms(pro0-lig0, pro0-lig1, pro1-lig1)

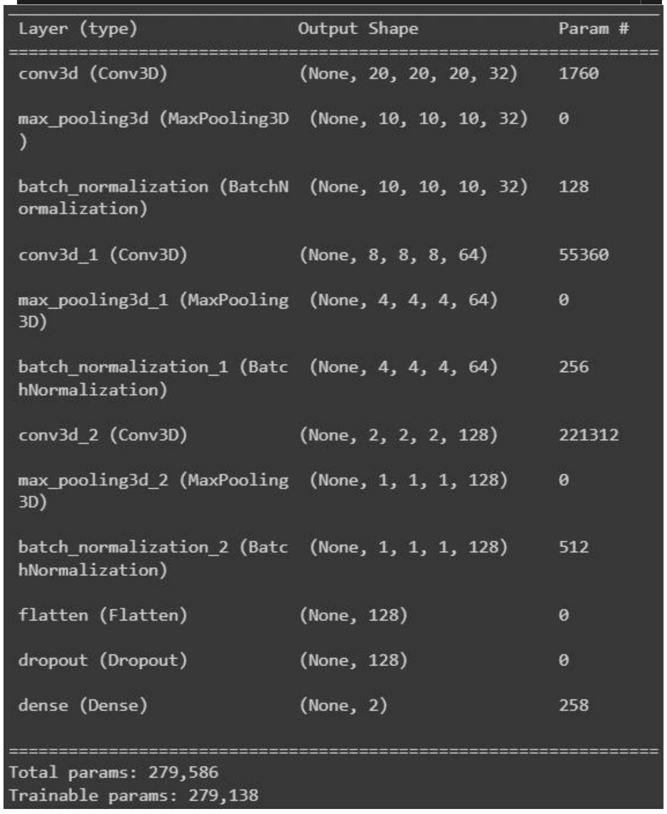


4. Training and testing procedure

(1) Transform dataset in proper shape
In the pre-processing part, I choose width = 11, thus for the input shape, I use 22*22*22*2, 22*22*22 represents the **(x,y,z)** location of a atom, the last channel represent the **protein** atoms or **ligand** atoms, the **hydrophobic** type or the **polar** type.(pro-h=1,lig-h=-1,pro-p=2,lig-p=-2)
(2) Form negative lig-pro pairs and positive lig-pro pairs
Here the positive lig-pro pairs mean the ligand is bound with the protein, the negative lig-pro pairs mean the ligand isn't bound with the protein. In the training set, the name of files tells whether they are bound or not. The label for negative pairs is **0**, for positive pairs is **1**. Since for one protein, only one ligand is matched, so the negative pairs are way more than the positive pairs, thus here I choose **10 negative pairs** randomly for one protein with **10 same positive pairs** for this protein at the same time to make the label **balanced**.
(3) Design CNN model
Figure 5. CNN model



```
model = keras.Sequential(
    [
     keras.Input(shape = input_shape),
     layers.Conv3D(32, kernel_size = (3,3,3), activation = 'relu'),
     layers.MaxPooling3D(pool_size = (2,2,2)),
     layers.BatchNormalization(),

     layers.Conv3D(64, kernel_size = (3,3,3), activation = 'relu'),
     layers.MaxPooling3D(pool_size = (2,2,2)),
     layers.BatchNormalization(),

     layers.Conv3D(128, kernel_size = (3,3,3), activation = 'relu'),
     layers.MaxPooling3D(pool_size = (2,2,2)),
     layers.BatchNormalization(),

     layers.Flatten(),
     layers.Dropout(0.5), # this removes random neurons(50%) to reduce overfitting

     layers.Dense(num_classes, activation = 'softmax')
    ]
)
```

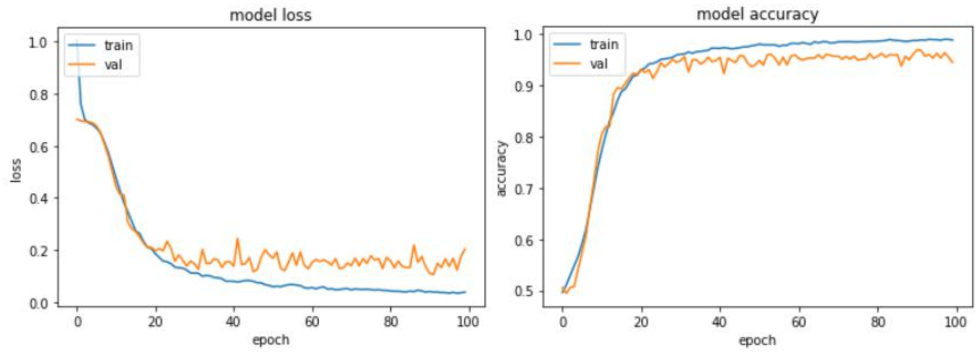


```
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 20, 20, 20, 32)    1760

 max_pooling3d (MaxPooling3D  (None, 10, 10, 10, 32)   0
 )

 batch_normalization (BatchN  (None, 10, 10, 10, 32)   128
 ormalization)

 conv3d_1 (Conv3D)           (None, 8, 8, 8, 64)       55360

 max_pooling3d_1 (MaxPooling  (None, 4, 4, 4, 64)      0
 3D)

 batch_normalization_1 (Batc  (None, 4, 4, 4, 64)      256
 hNormalization)

 conv3d_2 (Conv3D)           (None, 2, 2, 2, 128)      221312

 max_pooling3d_2 (MaxPooling  (None, 1, 1, 1, 128)     0
 3D)

 batch_normalization_2 (Batc  (None, 1, 1, 1, 128)     512
 hNormalization)

 flatten (Flatten)           (None, 128)               0

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 2)                 258

=================================================================
Total params: 279,586
Trainable params: 279,138
```

In this CNN model, I use **3** Conv3D layers with **kernel size=32,64,128**, after that with a **max-pooling** layer and a **batch normalization** layer. For the last layer, I use a dense layer with **softmax** as an activation function.

(4) Training Process

Figure 6. Training loss & accuracy curves

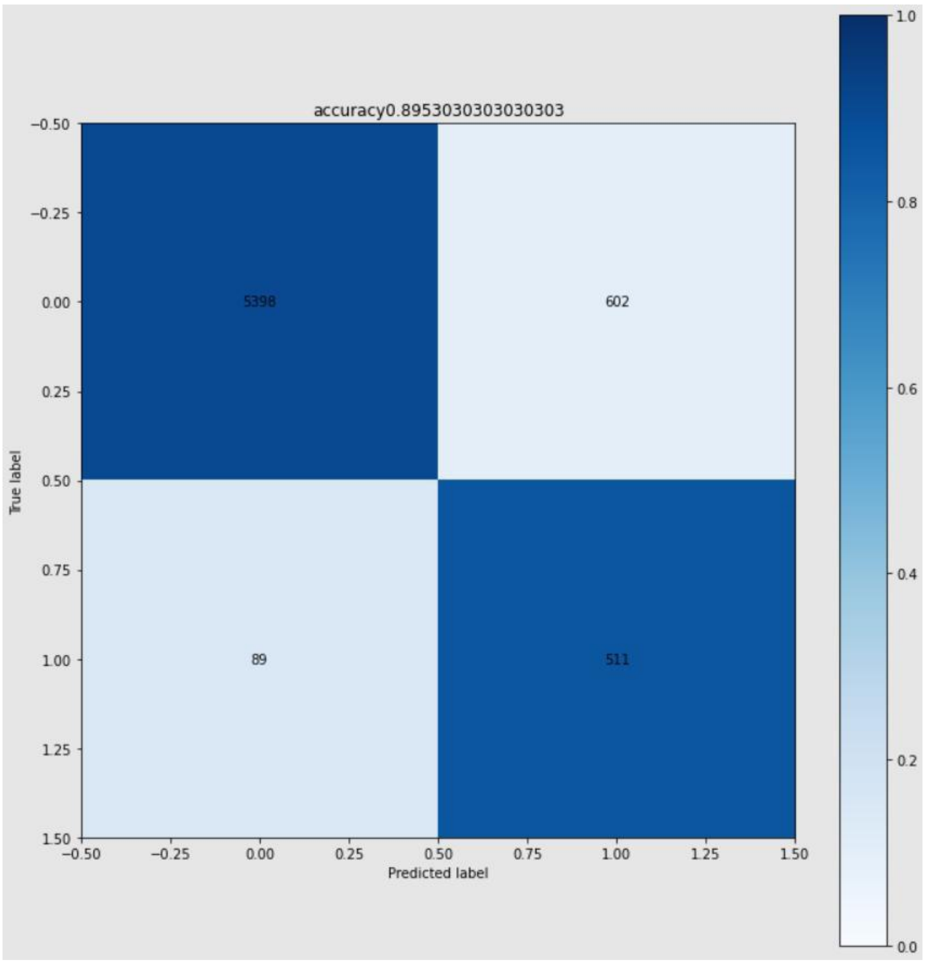


After the training, this model can get the accuracy for both training and validation set about 95%.

(5) Testing Process

There are 3000 training data in total, I randomly choose 600 of them as testing datasets. Then form **10** randomly negative pairs with **1** positive pair for one protein.

Fig 7. Confusion matrix for testing data



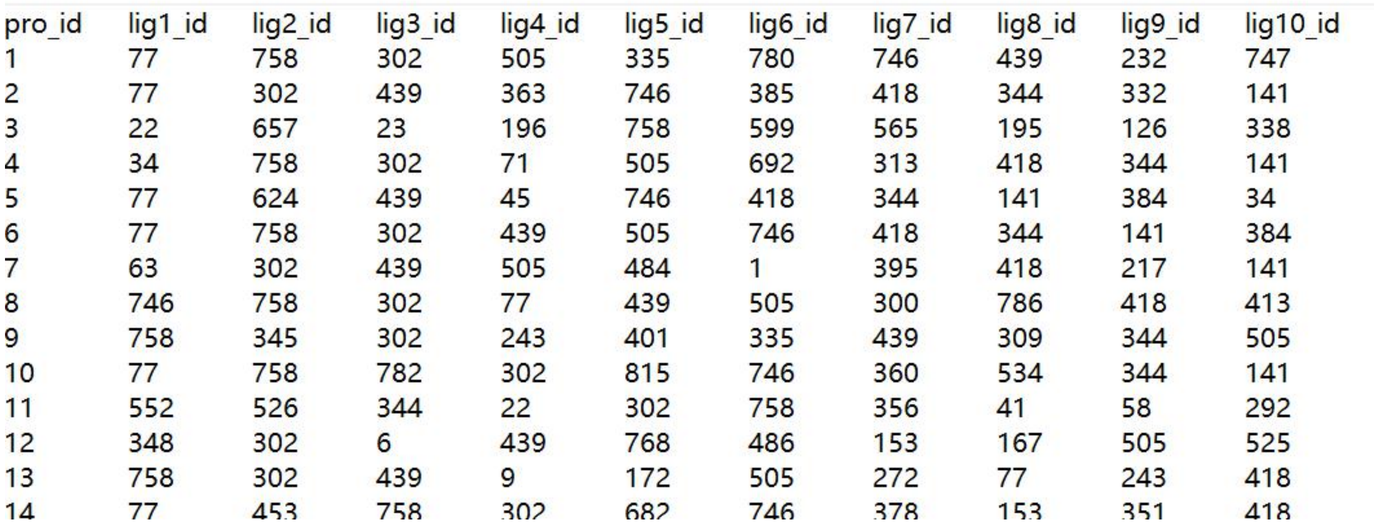
From this plot we can see the test accuracy is **89**%, there is no label-imbalanced problem for the prediction results.

5. Experimental study

(1)In this project, most of the efforts are on data pre-processing. In my attempts, the bost important part is to solve the **label imbalanced problem**.

For the first several models, I use **10** negative pairs with **1** positive pair for one protein in the training data, the training accuracy and testing accuracy looks fine, but when analyzing the probability of the prediction result, I found all the probabilities are very **low**(~10%), then by analyzing the confusion matrix, I realized this is cost by the imbalanced label problem-The model predict **all records to 0**, the model can still get high accuracy. Thus for the final model, I balanced the samples by **oversampling**.

(3) Though for the final model, the confusion matric and accuracy looks fine, when I try to create prediction result for the testing data, I noticed there are some **common ligands** that are in the top 10 **for most of the protein**. I think this may cost some error problems for the prediction accuracy. For future work, in data pre-processing part need to be more detailed to different ligand atoms, so there would have no ligand that has more compound impact n the prediction result.



pro_id	lig1_id	lig2_id	lig3_id	lig4_id	lig5_id	lig6_id	lig7_id	lig8_id	lig9_id	lig10_id
1	77	758	302	505	335	780	746	439	232	747
2	77	302	439	363	746	385	418	344	332	141
3	22	657	23	196	758	599	565	195	126	338
4	34	758	302	71	505	692	313	418	344	141
5	77	624	439	45	746	418	344	141	384	34
6	77	758	302	439	505	746	418	344	141	384
7	63	302	439	505	484	1	395	418	217	141
8	746	758	302	77	439	505	300	786	418	413
9	758	345	302	243	401	335	439	309	344	505
10	77	758	782	302	815	746	360	534	344	141
11	552	526	344	22	302	758	356	41	58	292
12	348	302	6	439	768	486	153	167	505	525
13	758	302	439	9	172	505	272	77	243	418
14	77	453	758	302	682	746	378	153	351	418