BS6207 Project Report

Xinxin | May - 2021

I. Problem Statement

Given the 3000 protein-ligand binding pairs with the atom type and the 3D coordinates, train a neural network to predict if a new protein-ligand pair would bind or not. The test set contains 824 proteins and 824 ligands. One protein would only bind to one ligand. For each protein, predict the top 10 ligands that are most likely to bind.

II. Highlights

There are two major challenges in the project: one is how to process the pdb data into a uniform format that can be feeded into a neural network, and the other is how to reduce the false positive rate during prediction, which is the result of the extremely imbalanced class in the test set.

1. Transform coordinate data into sparse 3D matrix

The given protein and ligand files have different sizes and scales, which cannot be used as input for neural networks directly. Hence, for each protein-ligand pair, the two pdb files were combined and converted into one sparse 3D matrix, which contains the encoded atom types based on the X/Y/Z coordinates. Detailed transformation process can be found in section III.

2. Imbalanced class and impact of false positive

We were asked to predict which protein and ligand will bind together, which can be translated into a problem of identifying the one binding ligand for each protein, out of the 824 candidates provided. The number of negative cases is much higher than that of positive cases (823: 1). This is the problem of class imbalance.

To tackle the imbalanced class problem, the following measures have been taken:

- Expose the model to more negative cases during training (see section IV-B for details)
- Employ cost-sensitive loss function (see section IV-D for details)
- Use the confusion matrix to measure the model performance instead of accuracy. As accuracy cannot reflect the real model performance with class imbalance. A model with 100% negative prediction would also have a high accuracy with 823:1 negative:positive class ratio. In this case, confusion matrix is a better matrix to measure the model performance.

III. Dataset Preprocessing Description

There are a few issues to be tackled in the dataset:

A. Lack of negative training data

The training data contains only binding pairs (i.e. positive cases) without any non-binding pairs. Hence, the protein and ligand files were randomly shuffled separately to generate non-binding pairs. All the protein-ligand pairs were then labeled as 1 (binding) or 0 (not-binding) accordingly.

B. Imbalance class in test set

The task of predicting the binding ligand for each protein in the test set suffers from an extreme class imbalance problem. For each protein, there are 823 negative cases and 1 positive case. Hence, training the model based on a balanced dataset might not yield good prediction performance. To tackle this issue, the dataset was randomly shuffled every 5 epochs during training to generate new non-binding pairs. With such a dynamic training set, the model can be exposed to much more negative cases and a fixed set of 2550 positive pairs, so that the false positive rate could drop.

C. Varying sizes of protein and ligand

The protein and ligand files in the training set have varying sizes. In general, the protein chain is much longer than the binding ligand chain. The length of protein chains varies from 38 atoms (2689_pro_cg.pdb) to 14644 atoms (0737_pro_cg.pdb), whereas the length of ligand varies from 1 atom (0687_lig_cg.pdb) to 24 atoms (1705_lig_cg.pdb). Hence, the pdb files cannot be simply converted to arrays for training. To solve this

problem, a sparse 3D matrix is constructed based on the X/Y/Z coordinates with the atom type as the matrix value. A sparse 3D matrix is a 3D matrix with many zeros. To differentiate atom types between protein and ligand chain, the hydrophobic and polar atoms in protein were encoded as 1 and 0, respectively, whereas the hydrophobic and polar atoms in ligand were encoded as 200 and 100.

D. Different scale of X/Y/Z coordinates

Different protein and ligand chains have different X/Y/Z coordinate ranges. To normalize the coordinates, every protein-ligand pair was centralized and shifted based on the centroid of the ligand atoms. This also ensures that the binding site is always included in the sparse 3D matrix. The coordinates are then moved and scaled to the positive space. Moreover, to unify the dimension of the matrix, a predefined box size was used to trim any protein atoms at the outskirt of the structure after scaling. This is based on the assumptions that protein-ligand binding could be predicted based on a relatively local structure closer to the actual binding site. Trimming the matrix also helps reduce the model training time. Theoretically speaking, with an infinitely large box size, all the information can be retained from the pdb files. All coordinates were also rounded to the closest integers. If two atoms share the same coordinates after the transformation, the two values will be added together. The transformation process is visualized as Figure 1 below.

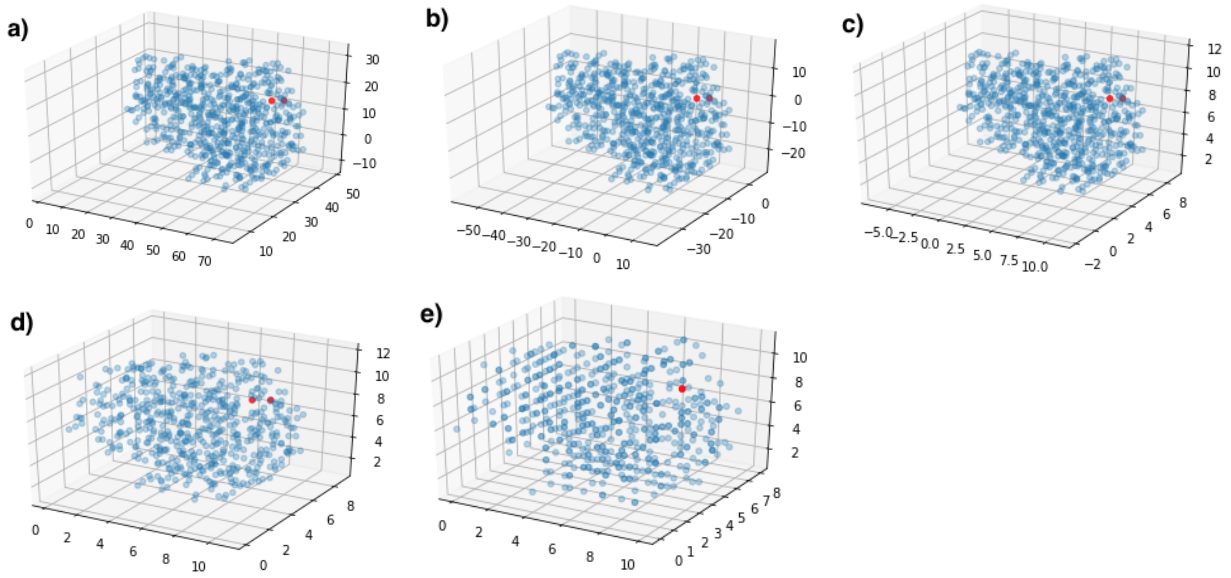

Figure 1. Protein-ligand matrix transformation (protein: blue, ligand: red). a) initial matrix. b) matrix centered by the centroid of ligand coordinates. c) matrix shifted and scaled to the positive space by the step of box size. d) matrix trimmed by the predefined box-size. e) matrix coordinates rounded to the nearest integer.

IV. Training and Experimental Study

To find the best neural network model, the model has been tuned from the following few perspectives in Table 1.

Table 1. Trials to tune the neural network

Approaches	Trial Set 1	Trial Set 2	Trial Set 3	Trial Set 4
A) Input Matrix Dimension	Small (16, 16, 16, 1)	Big (25, 25, 25, 1)		
B) Negative dat generation	Fixed before start training (pos:neg: 1:1 ~ 1:3)	Fixed before start training (pos:neg: 1:1 ~ 1:3)	Dynamically updated every few epoch	
C) Neural Network	CNN Model 1 (dropout = 0.4)	CNN Model 2 (dropout = 0.4)	CNN Model 2 (dropout = 0.1)	
D) Loss Function	Cross Entropy Loss	Cross Entropy Loss	Cost-sensitive	Cross Entropy

	(standard)	(standard)	Loss based on Cross Entropy	Loss (standard & weighted)
E) Optimizer	Adam (lr = 0.0005, weight decay = 0.005)	Adam (lr = 0.0005, weight decay = 0.005)	Adam (lr = 0.0005)	
F) Early Stopping	Based on validation loss (patience = 13)	Based on validation loss (patience = 13)	Based on validation loss (patience = 15)	

Tabel 2. CNN Model Structures

Layer	CNN Model 1			CNN Model 2		
	Kernel	Stride	Channel	Kernel	Stride	Channel
Input	(N, 1, 16, 16, 16)			(N, 1, 25, 25, 25)		
Conv3d-1	3x3	1	8	3x3	1	8
BatchNorm3d-1	-	-	8	-	-	8
ReLU-1	-					
MaxPool3d-1	2x2	1	-	2x2	1	-
Conv3d-2	3x3	1	16	3x3	1	16
BatchNorm3d-2	-	-	16	-	-	16
ReLU-2	-					
MaxPool3d-2	2x2	2	-	2x2	2	-
Conv3d-3	3x3	1	32	3x3	1	32
BatchNorm3d-3	-	-	32	-	-	32
ReLU-3	-					
MaxPool3d-3	2x2	**1**	-	2x2	**2**	-
Dropout	0.4			0.4 or 0.1		
Linear	Input: 32x2x2x2; Output: 32			Input: 32x4x4x4; Output: 32		
ReLU	-					
Linear	Input: 32; Output: 2			Input: 32; Output: 2		
Output	2 (logit scores for [bind, not-bind])			2 (logit scores for [bind, not-bind])		
Total Parameters	25 954			83 298		

A) Input matrix dimension

Two different sizes of sparse matrices were used as input. With a smaller matrix of 16 x 16 x16, information from the original pdb files is more condensed. Also, more information from the outskirts are trimmed and lost due to the smaller box size. Hence, theoretically, the smaller input matrix may not be able to train a model as good as the larger input matrix. Indeed, loss of models trained using a smaller matrix tends to oscillate much more than those trained using a larger matrix of 25 x 25 x 25. The example below shows the epoch-wise loss of models trained with class ratio of 1:3 (pos:neg) using smaller input matrix and larger input matrix, respectively.

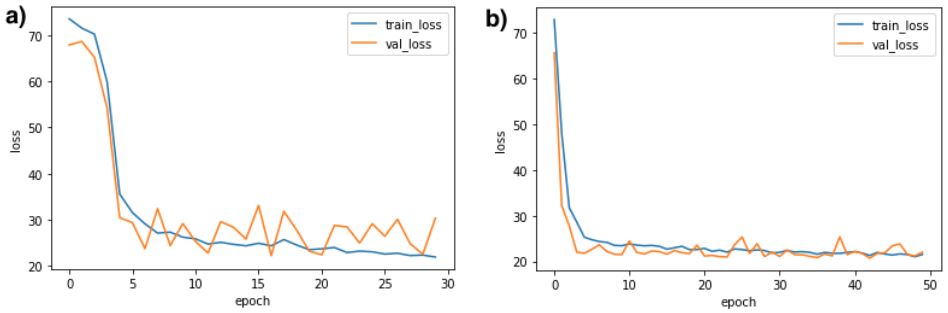

Figure 2. Epoch-wise loss of CNN model with small matrix input (a) and large matrix input (b)

On the other hand, the smaller matrix took much shorter to train. However, considering the model is more stable with larger matrices, the final model used the larger matrices of 25 x 25 x 25 as input.

B) Negative data generation

Due to the class imbalance problem in the test dataset, training the model using 1:1 class ratio may yield poor prediction performance. To solve this problem, the model was exposed to more negative cases during training by shuffling to generate new non-binding pairs once every 5 epochs. The 3000 binding pairs were first separated into 15% validation and 85% training beforehands, and the generation of new non-binding pairs were done separately within the validation/training set. In this case, within 50 epochs, the model can be exposed to 10 sets of different negative cases, which increases the training set ratio to 1: 10 (pos: neg). As shown in Figure 4, the model performance was measured in a validation set with 1:10 positive-negative class ratio. When the non-binding pairs were generated and fixed before training started, the model had higher false negative rate (i.e. lower recall). However, if the negative cases were refreshed during training, the recall improved to 100%. However, the false positive rate increased slightly, which will be handled by a cost-sensitive loss function (see section IV- D). Higher recall indicates that the model is able to predict almost all binding-pairs correctly. Hence, dynamic training set refresh was used in the final model.

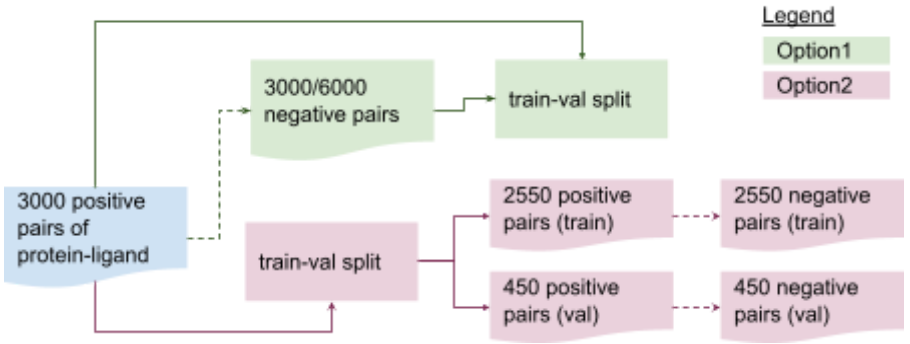

Figure 3. Two options of generating training and validation data (option 1 - generated negative cases first before splitting, fixed before training starts. Option 2 - split the train-val set first, and then generate negative cases within each set, refreshed every 5 epochs)

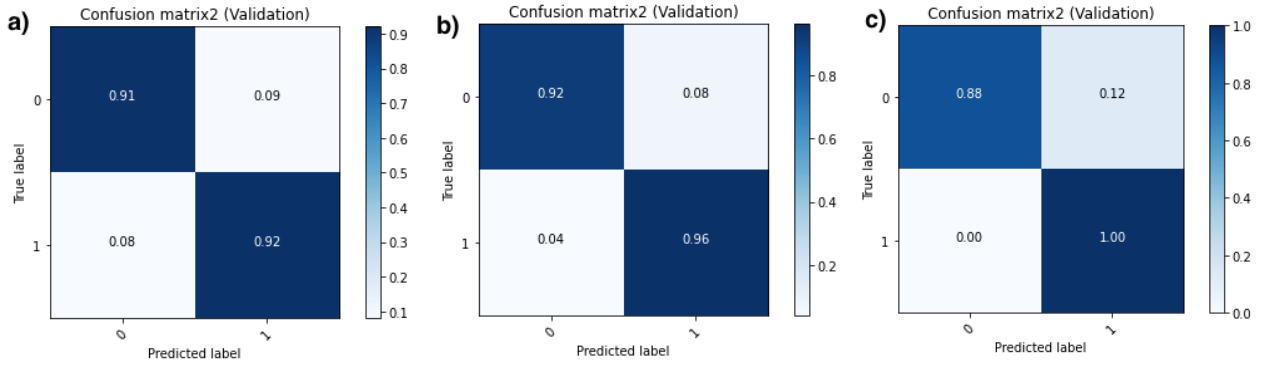

Figure 4. Confusion matrix of model performance in an imbalance set (pos: neg = 1: 10). a) model trained using 1:1 pos/neg ratio generated and fixed beforehand. b) model trained using 1:2 pos/neg ratio generated and fixed beforehand. c) model trained with training data refreshed every 5 epochs, the pos/neg ratio is 1:1 for each training set generated.

C) Neural network structure

There were two neural networks experimented in this project (see Table 2). CNN Model 1 was used with smaller input matrices, and CNN Model 2 was used for larger input matrices. They share almost identical structure with 3 CNN blocks, each containing a convolution layer, batch normalization, ReLU and MaxPooling. The only differences are the stride size in the last MaxPooling3D layer and the hidden nodes in the first dense layer, which were altered to control the complexity and the number of parameters in the model. Using Model 1 with larger input matrices would generate millions of parameters, which will be too complex and take too long to train. On the other hand, with a larger input matrix, a very shallow network

might not be able to learn all the features. Hence, Model 2 is still deeper than Model 1 with almost 3 times more parameters.

D) Loss function

Cross entropy loss has been used when training the CNN model with the use of class weights. This is to deal with the class imbalance problem in the test set as well. A higher weight is given to the negative class, indicating a heavy penalty if it's wrongly classified as positive, which helps reduce false positive rate (FPR). However, weighted cross entropy loss function impacted the model recall severely. As shown in Figure 5 a) ~ c), though the FPR has been reduced to 2% with weighted cross entropy, the recall has been reduced to 2% as well. Hence, this is not a good selection of loss function.

In addition, another cost-sensitive loss function was employed based on [1], which was developed on top of cross entropy loss and allows users to specify weight for True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), respectively. After a few trials, with the weight of [0, 10, 400, 200], the model reached the best prediction performance. With such a weight setting, a heavy penalty was given for FP, followed by FN. Besides, since the TP is the rarest during actual prediction, 0 loss was assigned for correct prediction of TP, which can also be viewed as an award. Figure 5 below shows the confusion matrix of the model in an imbalanced set (pos: neg = 1:10) trained with normal cross entropy loss, weighted cross entropy loss, cost-sensitive loss function with different weight ratio. All of these models were trained with training data shuffled and refreshed every 5 epochs. It can be observed that the cost-sensitive loss function in general performs much better than weighted cross entropy, especially for predicting the positive class (100% recall). Hence, the final model uses the cost-sensitive loss function with the weight assignment the same as Figure 5 e) with the lowest FPR.

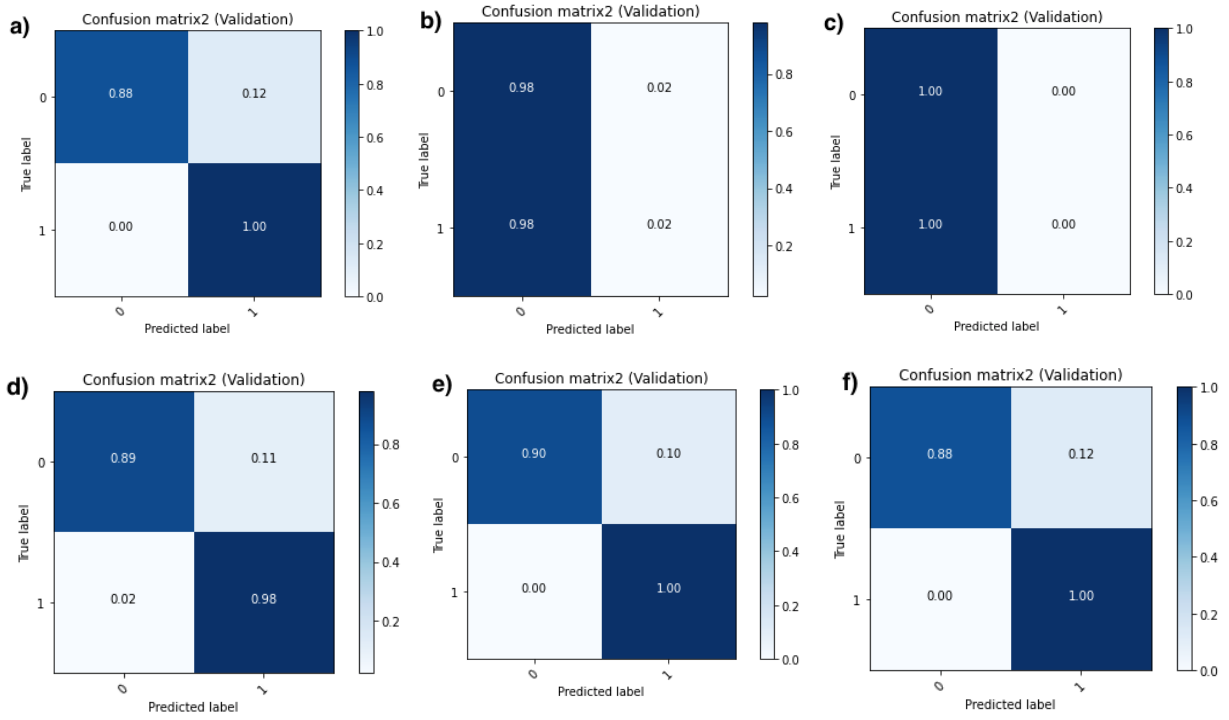

Figure 5. Confusion matrix measured in an imbalance validation set (pos:neg = 1:10) of models trained with a) normal cross-entropy loss, b) weighted cross-entropy loss (class weight neg: pos = 200: 1), c) weighted cross-entropy (neg: pos = 823:1), d) cost-sensitive loss function based on cross-entropy with weight of TP:TN:FP:FN = 0:10:100:50, e) cost-sensitive loss function based on cross-entropy with weight of TP:TN:FP:FN = 0:10:400:200, and f) cost-sensitive loss function based on cross-entropy with weight of TP:TN:FP:FN = 0:10:800:400.

E) Optimizer

Adam optimizer is used to optimize the model with a learning rate of 0.0005 and default values for all other parameters. A weight decay of 0.005 was added at first to prevent overfitting, which was then removed, as the validation loss in general was lower than the training loss without much overfitting. Figure 6 below shows

the epoch-wise loss for models trained with dynamic train-val split (pos: neg = 1:1) with normal cross-entropy loss and the cost-sensitive loss based on cross entropy, respectively.

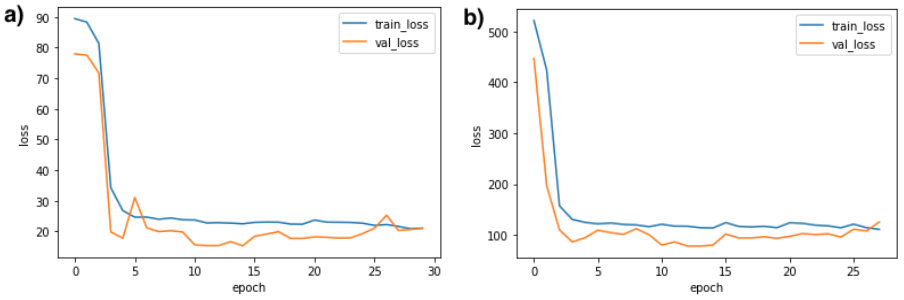

Figure 6. Epoch-wise train/validation loss of model trained without weight decay using a) normal cross-entropy loss, b) cost-sensitive loss function based on cross-entropy with weight of TP:TN:FP:FN = 0:10:400:200. Both show lower validation loss than training loss in general.

F) Early Stopping

To prevent the model from overfitting, early stopping was implemented with the patience of 13 ~ 15. In other words, if the validation loss is not improving in 13~15 epochs, the training will be stopped, and the model will retain the weight of the epoch with the lowest validation loss.

In conclusion, after tuning the model as mentioned above, the best performance was found using a bigger input matrix (1, 25, 25, 25) with CNN Model 2 with dropout of 0.1. The training data was refreshed dynamically once every 5 epochs, with a validation ratio of 15% (i.e. 5100 training pairs and 900 validation pairs). For each set of training/validation set, the class ratio is 1:1. Adam optimizer was used to train the model with a learning rate of 0.0005. Cost-sensitive loss function is employed, which is based on the cross-entropy loss with weight of TP:TN:FP:FN = 0:10:400:200. The training may stop early if the validation loss does not decrease in 15 epochs.

V. Testing Procedure

For each protein in the test set, it was paired up with all the ligands. Hence, there are 824 x 824 = 678 976 pairs of protein-ligand to be predicted. For each protein-ligand pair, the same transformation described in section II was applied to transform it into a sparse 3D matrix with the same size (i.e. 1, 25, 25, 25), where the coordinates were based on the X/Y/Z coordinates in the pdb file, and the value was the atom type (protein hydrophobic - 1, protein polar - 0, ligand hydrophobic - 200, ligand polar - 100). The best model found was then applied to the voxelized 3d matrices to predict if the protein and ligand would bind or not. The output of the prediction contains 2 logits scores, which were transformed into softmax probabilities. The class with larger probability is taken as the predicted result. To find the top 10 ligands that could possibly bind with each protein, the predicted-to-bind ligands were ranked by their binding probability, and the top 10 ligands with the highest probabilities were taken. Result was then exported into ‘test_predictions.txt’.

References

[1] Cost-Sensitive Regularization for Diabetic Retinopathy Grading from Eye Fundus Images Adrian Galdran, José Dolz, Hadi Chakor, Hervé Lombaert, Ismail Ben Ayed Medical Image Computing and Computer Assisted Intervention, 2020 (accepted).