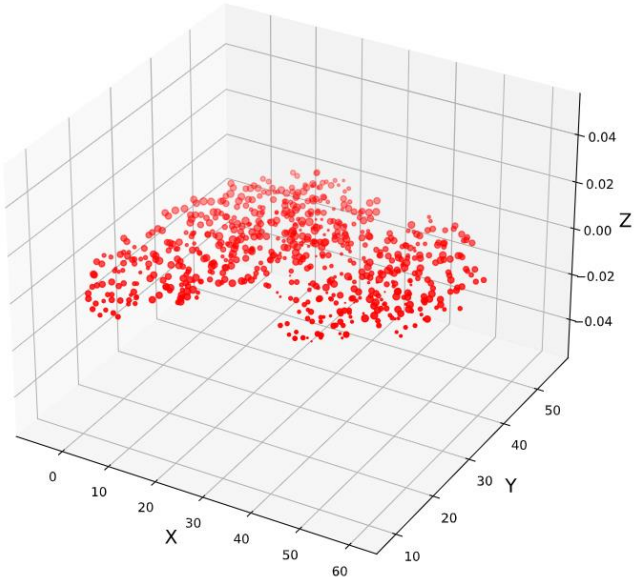
1. Data Exploration & Pre-processing

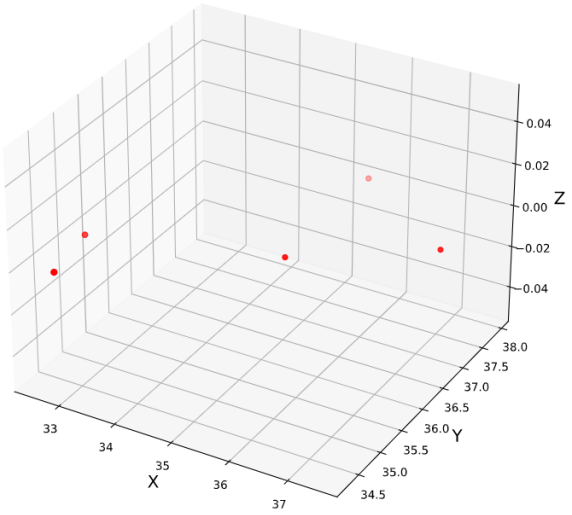
The data provided for this project are the coordinates data of all the vertexes which includes axis X, Y & Z. If we use the visualization to visualize the protein & the ligand, we may generate the 3D scatter plot like below, which is a visualization of protein 0001 in our training set, showing the overall structure of the protein itself.

Protein 0001 in Training Set

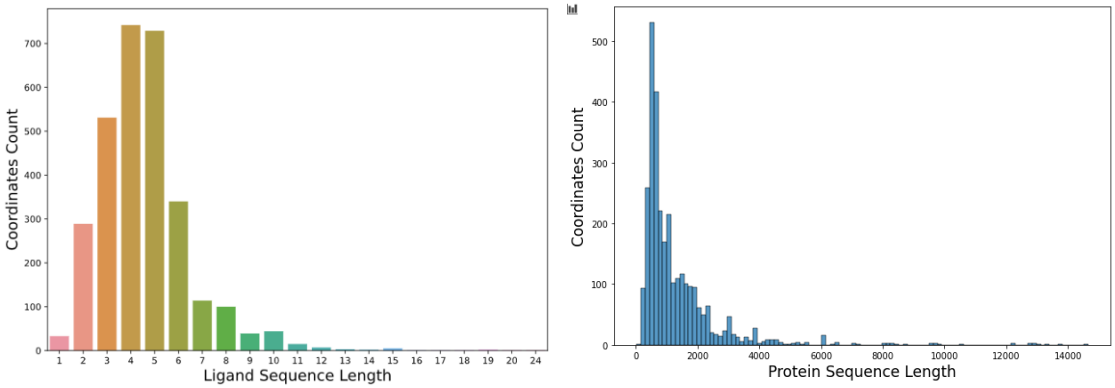


Similarly, we can also visualize the binding ligand structure for the protein above as below:

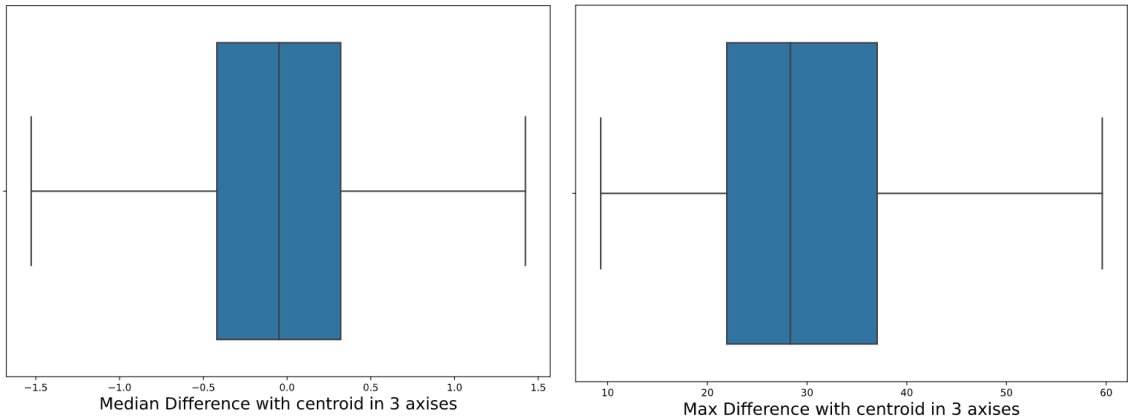
Ligand 0001 in Training Set



In this problem, while our objective of this project is to develop a model which is able to predict if the given protein & the ligand is binding, using their coordinates data, there is a challenge here, which is for different protein & ligand, the length of their coordinates list is quite differently distributed(as the below bar plot illustrates) so that there is no reliable length of data to pick, while the neural network are always requiring a constant number of channels, or neurons for data to flow inside the model.



To solve this problem, we have come up with a data pre-processing strategy finally, which is to process the coordinates into a cube with a new dimension. To decide the new size of the cube to generate, we have calculated the median and the maximum of all of the points, and their corresponding length of the points to the centroids on 3 axes. The result is as shown in the below 2 box plots, showing the median of that is only around 0 to 1.5, while the mean of the maximum is around almost 30.



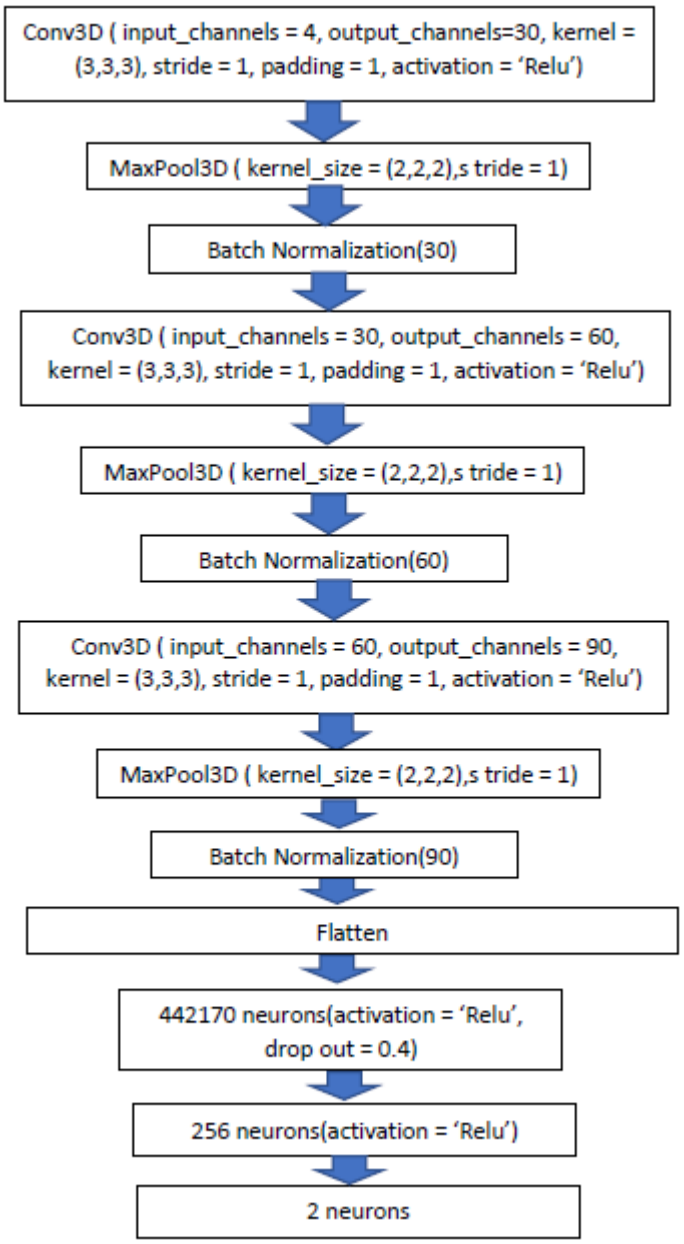
However, due to the CPU performance limitation, the model is not able to have too much input. So, we finally compromised to pick 10 as the width to pick all the points around the centroid on the 3 axes. So, our finally generated cube has the dimension as 20*20*20. The specific steps are as stated below:

1) Pick the centroid of the ligand, and only select the points which have the Euclidean distance to centroids less than 10, and correspondingly pick the points using the same rule for the protein.
2) For any coordinates indicated into the file, the pixel in the cube will be represented as 1, otherwise the value will be 0. For 2 kinds of different coordinates type, which are hydrophobic (‘h) & polar (‘p’), it will be located at 2 different channels respectively (similar to the RGB channel of image data).
3) The protein cube and the ligand cube will be merged at the channel axis. So, for each pair of protein and ligand, it will finally generate one combined cube with 20*20*20*4, while 4 is the ‘p’ & ‘h’ channels for both protein and ligand, respectively.

2. Model Construction, Training & Protein-Ligand Pair Pick

2.1 Model Construction

The constructed model structure is as shown in the below diagram. The data will go through 3 3D convolutional layers, with kernel 3*3*3, stride as 1, padding as 1, and activation function as Relu, and 3 3D max pooling layers, with kernel 2*2*2, and stride as 1, and 3 batch normalization layers with the corresponding generated channels in the previous convolutional layers. After that, the data will be flattened into 1D data, and pass

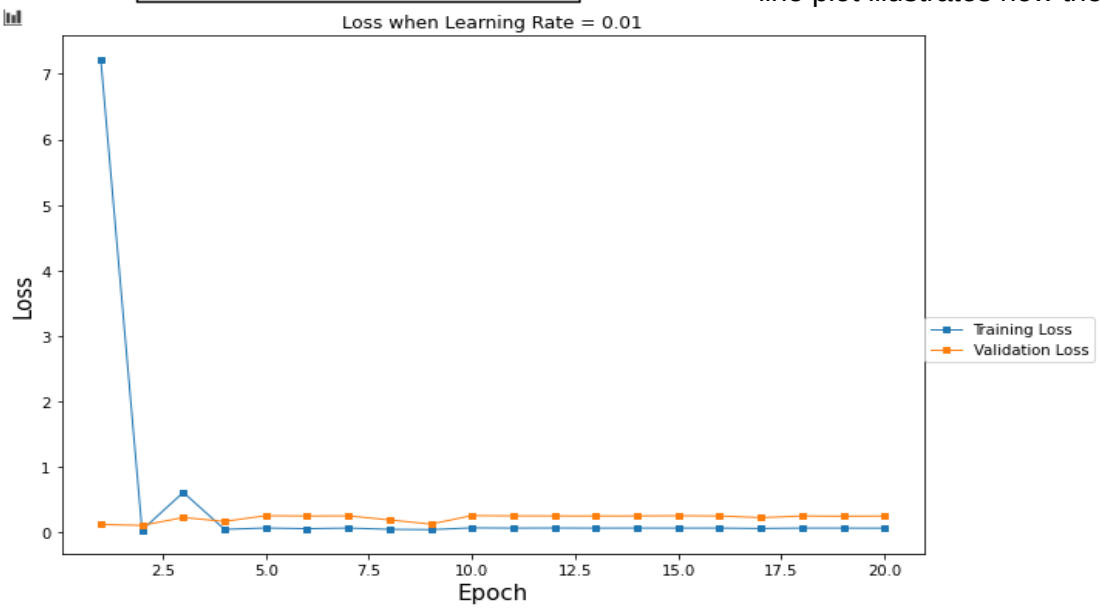


trough a deep neural network sub model, with the 1st layer having 442170 neurons, activated by Relu and using the drop out rate as 0.4, and the 2nd layer having 256 neurons, and 1 output layer with 2 neurons. The reason why the drop-out rate here is 0.4 which is higher than most of the situation, is due to that the 1st layer has tremendous neurons. So, increasing the drop-out rate here to 0.4 may reduce the risk of overfitting.

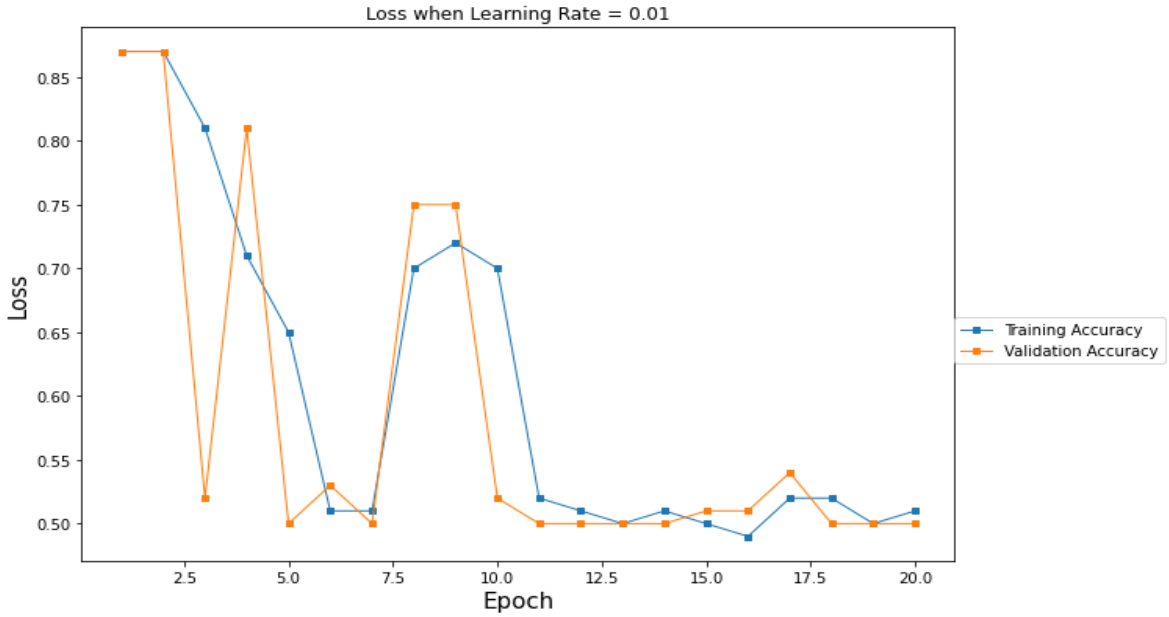
2.2 Training

In our training process, our data are divided into the training set and validation set by ratio of 4:1. The batch size we applied here is 20, and the cross-entropy loss function is also applied here to optimize the parameters inside the model.

We have finally set the epoch as 20, and saved the model during each epoch, and print the accuracy & loss for training set and validation set all the way. The below line plot illustrates how the



loss changed in each epoch. It is intuitive that after the epoch 2, the training loss and validation loss are all becoming stable.
We then further plot the relationship between the training accuracy & validation accuracy with the epoch. Maybe due to that our deep neural network part has set too many neurons, or the learning rate here is 0.01, which may be still a bit higher, both accuracies were fluctuating violently during the whole training process. However, this result could also be due to that maybe 1 or 2 epochs is already thoroughly enough for the model to be trained.



After the consideration between the loss and the accuracy, we finally picked the model generated at epoch 2, which have both reliable loss (down to 0.01) & accuracy (up to 90%) for our final protein ligand pair pick

2.3 Protein-Ligand Pair Pick

As Pytorch already embedded the Softmax function into the cross-entropy loss function package, so our model has not set the Softmax layer as our final output. However, while Softmax layer is using the below activation function to finally transform the output neurons to a probability using the below formula, it will be more understandable and reliable if we transform our output after one Softmax layer, which is more likely to generate the probability of the protein and ligand are binding or not.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

For testing data, we have pre-processed it using the same strategy as our training data set pre-processing process and flow the data into our model and one Softmax layer to generate the result and saved it as test_prediction.txt file.

3. Conclusion, Limitation & Future Work

3.1 Conclusion

- **The model generated at epoch 2 are the most reliable model during the whole process.**
- **Softmax is powerful in transform the output data into more interpretable probability like data**

3.2 Limitation & Future Work

- **From the plot of accuracy and loss, the model looks not stable. It may be due to that too many neurons have been set at the deep neural network part and reducing to violently to 256 neurons.**
- **Also, the instability of the training process can be possibly due the high learning rate. Further experiment on lower learning rate training can also be performed.**
- **During the data pre-processing, the width was set to 10 only, which may loss information somehow and influence our further training. Further experiment on different width could be applied to analyse the model performance.**