BS6207 Final Project

Problem definition:

The purpose of this project is to design a neural network to predict whether a given protein-ligand combination will bind, which can be considered as a binary classification problem, to predict and analyze the predicted labels.

Dataset:

The provided training set consists of 3000 pairs of protein-ligand combinations, where molecules are provided in .pdb format. Each molecule contains a list of atoms x, y, z coordinates of each atom and atom types, either hydrophobic or polar. On the other hand, the provided testing set contains 824 protein-ligand combinations. There are 3000 positive training examples among the datasets and 29992 negative training examples.

Data Pre-processing:

The dataset shows that each protein can contain 100s to 1000s of atoms, but each ligand can only have atoms less than 20, sometimes even one; thus, it won't be easy to pass this dataset as input to the neural network around various information. By doing research, I was able to implant a pre-processing algorithm called Voxelization, where Voxelization is the process of converting data structures that store geometric information in a continuous domain, such as a 3D triangular mesh, into a rasterized image, which is a discrete grid. I first merged and centred around the ligand for each protein and ligand and then moved to the nearest grid point. This step involved choosing a distance and grid resolution. Note that a high max distance and high grid resolution would theoretically allow the maximum amount of features to be captured in the input, but cost more memory. For this project, due to the fact that all atoms are centred around the ligand, a lower

max distance would also work. Thus, I set the max distance to be 20 and grid resolution to 4. After that, I pruned the atoms outside box size to get the atom types and where the molecule atom belongs to.

I also noticed the number of data in each class is not balanced. In order to solve this problem, I applied the Random Over Sampling technique to balance the dataset.

Training and testing procedure:

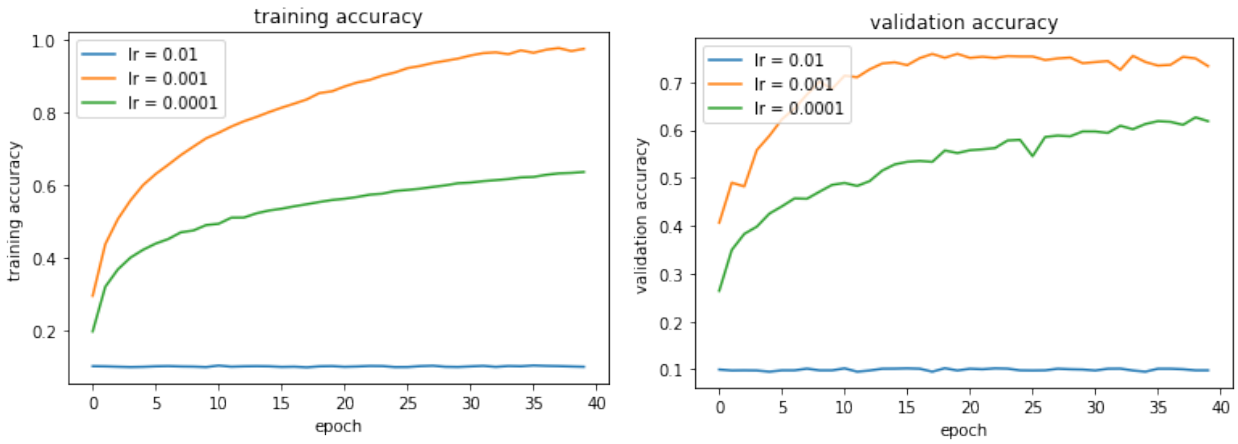
I divided the training set into train and validation set with a ratio of 0.77. Among the training set, I put the balance between positive and negative pairs as 1:1, where each consists of 10 teams of protein.

Experimental study - Model and Hyperparameter Tuning:

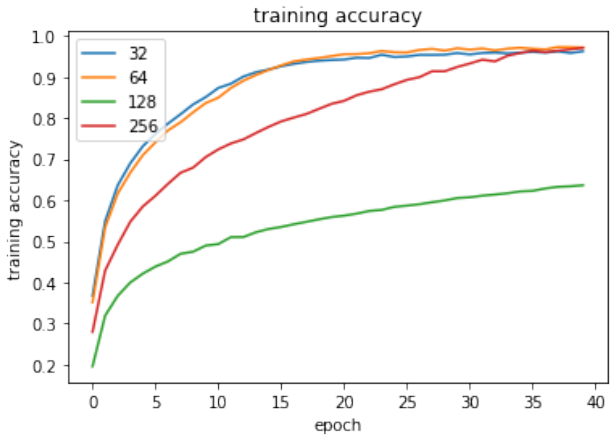
For the Neural Network, I designed a CNN with four layers, where the first layer is a Conv3D layer with 16 channels, kernel size equal to 3. I chose padding equal to “VALID” to leave out of it, and I decided to use the normal distribution as kernel initialization method and “ReLU” activation function. In the end, I added a batch normalization layer to normalize the hidden values. The rest of the layers are similar to this layer, with different channels equal to 32, 64 and 128. Then I added an Average Pooling 3D layer and a Flatten Layer to pass the values to Dense layers. I added two dense layers with channel equals 128 and 2 as output; the last activation function is the Sigmoid function. The model includes a random dropout with 50% to prevent overfitting.

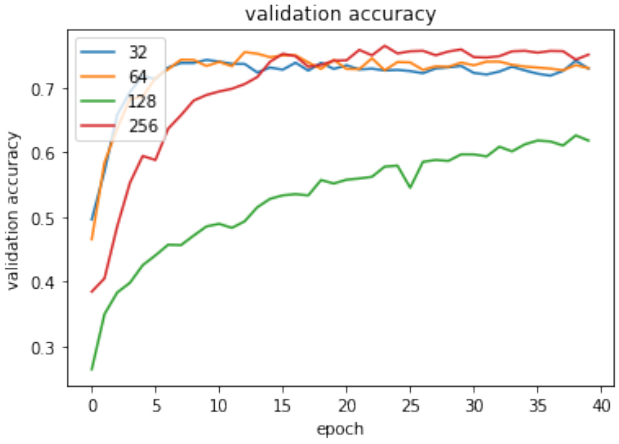
```
input_1 (InputLayer)        [(None, 21, 21, 21, 2)]   0

conv3d (Conv3D)             (None, 19, 19, 19, 16)    880

batch_normalization (BatchN  (None, 19, 19, 19, 16)   64
ormalization)

activation (Activation)     (None, 19, 19, 19, 16)    0

conv3d_1 (Conv3D)           (None, 17, 17, 17, 32)    13856

batch_normalization_1 (Batc  (None, 17, 17, 17, 32)   128
hNormalization)

conv3d_2 (Conv3D)           (None, 15, 15, 15, 64)    55360

batch_normalization_2 (Batc  (None, 15, 15, 15, 64)   256
hNormalization)

conv3d_3 (Conv3D)           (None, 13, 13, 13, 128)   221312

batch_normalization_3 (Batc  (None, 13, 13, 13, 128)  512
hNormalization)

average_pooling3d (AverageP  (None, 6, 6, 6, 128)     0
ooling3D)

flatten (Flatten)           (None, 27648)             0

dense (Dense)               (None, 128)               3539072

dropout (Dropout)           (None, 128)               0

dense_1 (Dense)             (None, 2)                 258
```

I tested the model with different learning rates and batch sizes for hyperparameter tuning. For learning rate, I tried the model with 0.01, 0.001 and 0.0001, and here is the result:

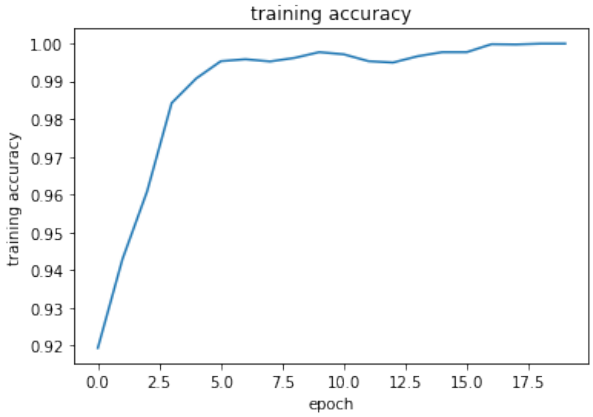


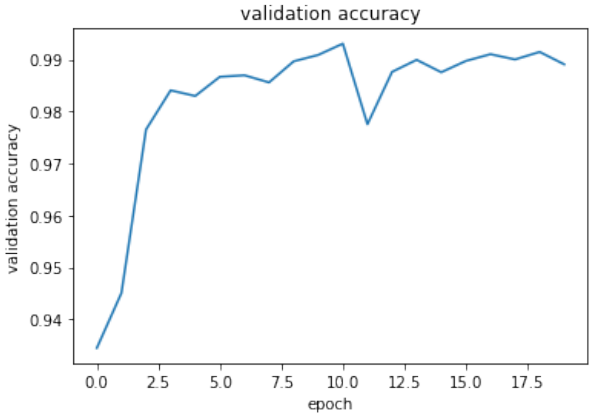
For the batch size, I tried a model with 32, 64, 128 and 256, and here are the results:





The model optimizer I decided to use the Adam optimizer since it is good at converging on a decent local optimum without needing to tune hyperparameters too much. Based on the outcome of my experiments, I decided to choose to train the model with a learning rate equal to 0.001 and a batch size equal to 64 to achieve the maximum training accuracy. The final training and validation accuracy reached 98% with the following trend.





Test File:

In order to generate the test file, I extracted the probability of each prediction of the test proteins. I sorted the probability in descending order, and then I saved the top 10 pairs of each protein and saved it into the test_prediction.txt file.

Reference:

- Yang, Jincai, et al. "Predicting or Pretending: Artificial Intelligence for Protein-Ligand Interactions Lack of Sufficiently Large and Unbiased Datasets." Frontiers in Pharmacology, vol. 11, 2020, https://doi.org/10.3389/fphar.2020.00069.
- Verma, Niraj, et al. "SSnet: A Deep Learning Approach for Protein-Ligand Interaction Prediction." International Journal of Molecular Sciences, vol. 22, no. 3, 2021, p. 1392., https://doi.org/10.3390/ijms22031392.