Work Description

The workflow was divided into two parts: solving the problem using the ESM-2 model, and solving it without applying the model.

At first, I performed the work without using the ESM-2 model. For this purpose, I used a RandomForest model. However, during both training and testing, it showed slow performance under the conditions of this problem, mainly because the number of data pairs is extremely large. In this case, I converted the data into .npz format. Due to limited computational resources, I was only able to train on a small dataset (100 .npz files), which I generated from .pdb files. Testing was also performed on a small dataset (8 .npz files), and the results were as follows:

Accuracy: 0.9234 Precision: 0.2106 Recall: 0.6719 F1-score: 0.3207

The main drawback of this approach is its computational heaviness. However, I believe this method could still be considered a potential solution under the given problem constraints.

In the second approach, I used the ESM-2 model to generate embeddings and then apply them in my own framework. The idea was the following: having a ground truth matrix and the ESM-2 model outputs, I passed protein pairs through a neural network to improve the results. The goal was to take all protein pair connections derived from ESM-2, reduce their dimensionality through my network, and finally output a single value: whether a connection exists between the pairs (1) or not (0).

For the model architecture, I implemented a simple neural network to enable faster training. The model consists of 3 layers with the following dimensions: (input_dim -> 512 -> 256 -> 1). ReLU activation was used between the layers, and a Sigmoid activation in the final layer to output 1 or 0. Binary Cross-Entropy loss was used as the loss function, and Adam optimizer with a learning rate of 0.001. Training was performed for 5 epochs.

Data Handling

For the first approach, I stored the data in .npz format, while in the second case, I used .txt format.

In the first case, I extracted connection matrices from the .pdb files, where each coordinate (i, j) indicated whether the amino acids in row i and column j were connected (1) or not (0). During training, I applied one-hot encoding to represent amino acid names.

In the second case, I generated .txt files from the .pdb structures. The `ESM-2/get_esm_output.py` script processes the .pdb folder, extracts all sequences, passes them through the ESM-2 model, applies a threshold to determine whether a connection exists (threshold = 0.1), and saves the result as a binary matrix in text format.

For evaluation in `ESM-2/test_esm_with_my_solution.py`, I used the .pdb folder to generate ESM-2 outputs, passed the embeddings through my network, and stored the resulting prediction matrices as .txt files. The ground truth was prepared using `ESM-2/get_ground_truth.py`, where amino acid pairs with a distance less than 8 Å were labeled as connected, and stored as .txt files.

Accuracy Evaluation

Since this is a classification task, I applied standard classification metrics (Precision, Recall, F1-score). Using the predicted matrices from the models and the ground truth matrices, I computed the metrics across the test dataset. The ESM-2 model used was `esm2_t6_8M_UR50D`. The results were:

ESM-2 only: Precision: 0.2156 Recall: 0.0351 F1-Score: 0.0604

My approach with ESM-2 embeddings and neural network: Precision: 0.5573 Recall: 0.3670 F1-Score: 0.4426

From these results, it can be concluded that my approach is effective in this problem setting. Since I used a smaller ESM-2 model with fewer parameters, the direct ESM-2 results were relatively weak. However, applying my method significantly improved performance even with a lighter model.

Time Spent

The first part (without ESM-2) was completed in 3–5 hours since I used an already available classifier. Researching and understanding the ESM-2 model took up to 5 hours, with additional challenges related to CUDA configuration. Designing the model and concept took about 1 hour (while walking and thinking). Coding and training took up to 10 hours. To speed up the process, I also used ChatGPT for certain coding parts.