1. Fine-Tuning the base model Prot_bert

For the first part of this assignment, I evaluated the base model over 30 epochs without any modifications, achieving an accuracy of approximately 0.201. Recognizing the need for improvement, I implemented several modifications to enhance the model's performance. Firstly, I redefined the model architecture to include additional layers. The updated structure is as follows:

- A linear layer that maps from 1024 to 512 features.
- A ReLU activation function for non-linear processing.
- A Dropout layer with a 0.5 rate to reduce overfitting.
- Another linear layer that maps from 512 features to the number of classes (n_classes).

In addition to architectural changes, I experimented with various learning rates within the range of 0.001 to 0.1. I also tested multiple optimizers, including SGD, Adam, and AdamW, to identify the most effective option for our scenario. Through these experiments, SGD emerged as the most suitable optimizer for optimizing this model. Lastly, I decided to remove the rare amino acids from the sequence instead of keeping them. However, the best accuracy that I can get from this Prot_bert model is around 0.4, which is not very optimized.

2. My best model facebook/esm-1b

I found this model in Hugging Face, and I found it very suitable for this assignment because the description says that it is a transformer protein language model, trained on protein sequence data without label supervision. The model is pretrained on Uniref50 with an unsupervised masked language modeling (MLM) objective, meaning the model is trained to predict amino acids from the surrounding sequence context.

I further enhanced the model by incorporating a checkpoint callback function, which enables tracking and saving the best-performing model during training. Additionally, I modified the model architecture to align with the 'facebook/esm-1b' structure, adjusting the input feature size from 1024 to 1280. To further improve training stability and potentially enhance performance, I incorporated a BatchNorm1d layer and replaced the traditional ReLU activation function with LeakyReLU. The dropout layer also is applied.

Regarding the optimizer, I continued to utilize SGD with a learning rate of 0.001. This decision was based on prior experiments indicating robust performance with this configuration. I also explored an alternative approach by attempting to implement the ProtENN + HMMER model, as detailed in a paper shared on Kaggle. This approach yields around 0.996 accuracy, which is not bad. However, the facebook/esm-1b out performs that combination model, reaching 1.0 accuracy after running forty-ish epochs.