Andrew Florial
Cameron Farzaneh

Task I:
For using the dataset with the architecture, we use every fifth line as the validation set and the remaining as the training set.

We use a standard size of 99 amino acids in a sequence, if the sequence size is smaller than that we pad the values with zeros. If the sequence size is larger than our standard size, we truncate the sequence. The amino acid sequences are encoding as 1 hot encoding vectors of 21 size for the amino acids and newline character.

Layer Type	***Activation***	***Neurons***	***Input Shape***	***Args***
LSTM	Tanh	512	(,99,21)	return_sequences=True
LSTM	Tanh	512		return_sequences=True
LSTM	Tanh	512		return_sequences=True
Dropout				p=0.5
TimeDistributed Dense	SoftMax	21		

Task II:
LSTM by default also initialized by the Glorot or also known as the Xavier initializer. The initial state is a vector of zeros.

Hyperparameters	***Value***
Learning Rate	0.001
Epochs	100
Loss	Categorical Crossentropy
Standard Sequence Size	100
Batch Size	256
Optimization Procedure	Adam

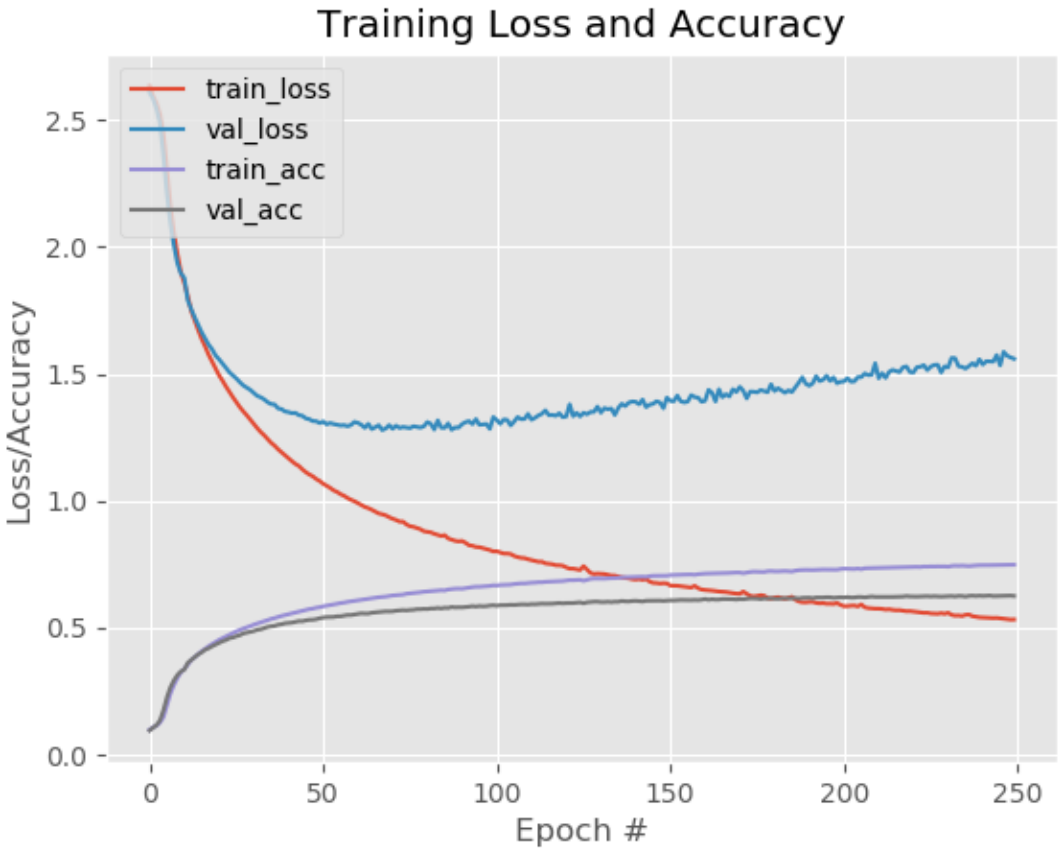
We used the empirical rules of thumbs to choose the appropriate hyperparameters for this problem. We chose a batch size of 256 based on experimental findings in our test runs where there was poor learning on 512 and accurately learning below 128.

Due to poor training accuracy on multiple epochs we decided we needed a higher hidden layer size to accurately model the neural network. 5,302,805 total trainable parameters improve both the optimization and generalization error of our model. Layers of LSTM with 512 neurons and a SoftMax layer of 21.

We started with a learning rate of 0.0001 and noticed unnecessary slow convergence so we increase the learning rate by a magnitude to 0.001 and noticed that it results in faster learning while still achieving convergence.

For the standard sequence size, we decided on a size of 100 to accurately model many of the amino acids without introducing too much noise into model with the padding.

Our model currently reports a training accuracy of 75% however, it appears that after 100 epochs, the model is beginning to overfit. The validation accuracy we were able to obtain was 62%. Maybe we regularization we can reduce the overfitting problem. Due to computation limitations, we did this training initially on the entire dataset, but then we used 10% of the sequences for training with the same percentage for validation instead.



Long Term Dependencies Estimation

To find out what the longest dependence our network is capable of capturing, we change one letter at a time in a sequence that are predicted right until the label is no longer predicted correctly. We have constructed a loop that tests this for 1000 protein sequences in the validation set. There are two experiments that we have done. First method was randomly choosing an index in the sequence, and then randomly choosing a letter that is not newline (from the available classes) to insert into the sequence. Doing this method, we discovered that the longest dependency the network was able to capture was of length 27. In addition, we experiment incrementally. This way, instead of randomly choosing an index, we begin from the beginning of the sequence and randomly choose a letter that is not newline and insert it into the sequence until the label was mis predicted. Because this both processes are stochastic, we do this process multiple times for 1000 protein sequences. Doing it the incremental method obtained the best

results, and it appears that the longest sequence captured was of length 68. Our model takes in 99 amino acids as input.

Task II:

K	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Because we had an input of 99 amino acids into our model, we feel like using small K values can yield poor results. However, it appears that with a K value of 11, we obtained the highest number of maximal matches. In addition, it appears that on average our model can obtain higher number of matches with larger K values. This is tested on 1000 samples in the validation set.