EE6123 Deep Learning

Fall 2018

Assignment 3

Report

Author: Varun Sundar

EE16B068

28th October 2018

1 Implementation and Technical Notes

Python 3.6 was used to code, with modularity of components being the main focus.

The code has been split into logical modules:

- **model_lib.py**: defines the MNIST model.
- **mnist_eager.py**: trains the MNIST model with *tf eager execution*.
- **README.md** : Contains relevant code documentation.

2 Question 1

We use *pytorch* to train the MNIST RNN Model of the following architecture:

The code to train the model is:

```
python3 q1.py --model LSTM --bi 1 -l 64
```

- Learning rate of $1e-3$, with Adam was used as the optimiser.
- We periodically run evaluation post an epoch.
- Loss is Cross Entropy
- Models testsed are LSTMs, GRUs, RNNs, with varying hidden unit sizes.
- L2 Regularisation 0.01 used always.

2.1 Accuracy and Loss Plots

For each model we try the following configuration:

- Hidden size 64
- Hidden size 128
- Hidden sizes (64, 64) : two layers

We report train loss, test loss and test accuracy with this. As usual, we compute the average loss across 10,000 samples for each experiment. We do so by running every evaluation step on the whole 10,000 samples. We find comparable performance across all models (except vanilla RNNs) , with more accuracy with more layers and with a LSTM. This is due to fewer parameters in a vanilla RNN, and the poor handling of long sequences. However, with more hidden layers, the accuracy improves.

We also experiment with bi directional models.

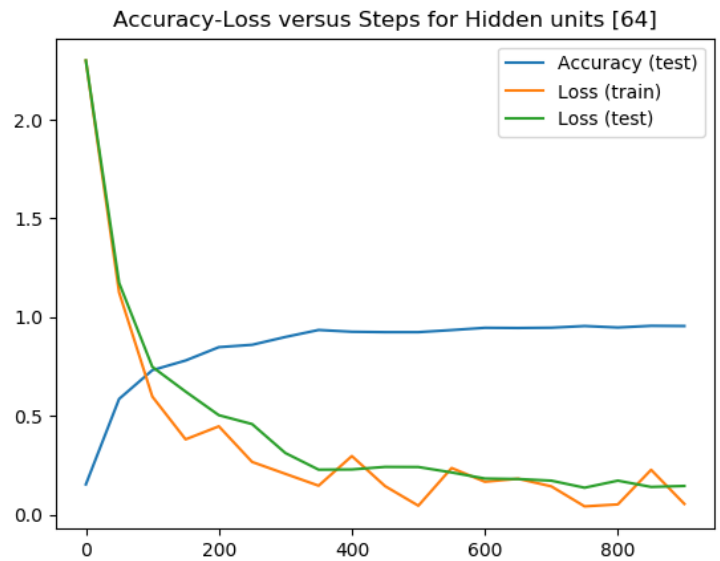


Figure 1: $H = 64$, LSTM, Accuracy Loss Curves

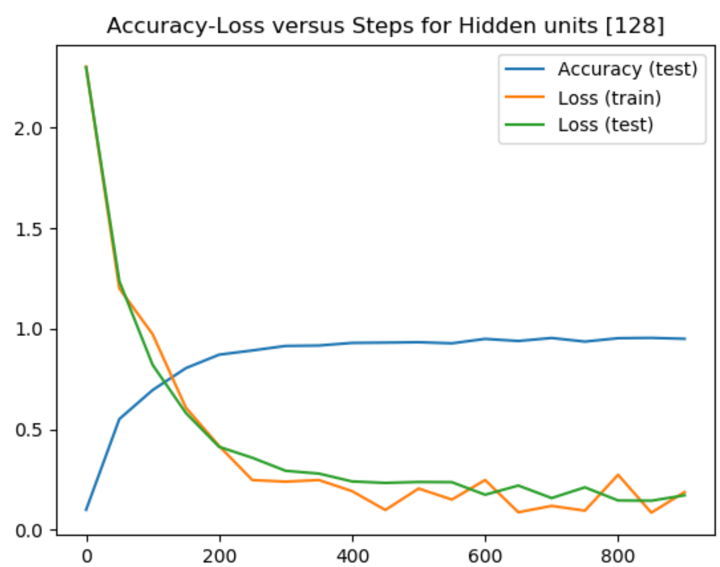
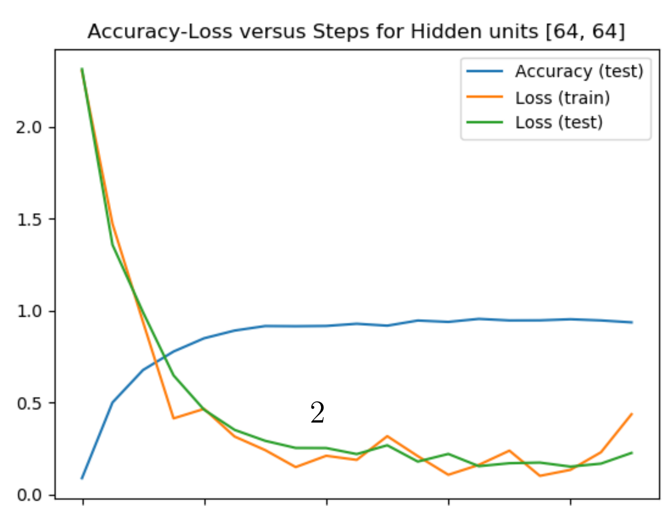


Figure 2: $H = 128$, LSTM, Accuracy Loss Curves



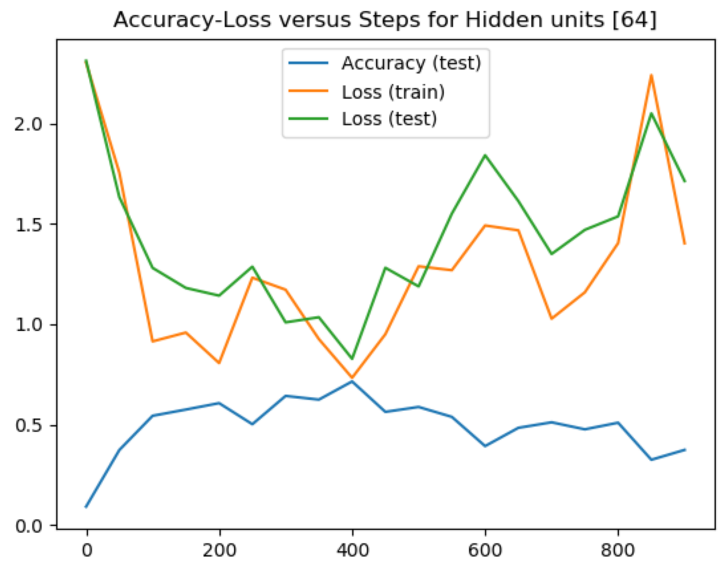


Figure 4: $H = 64$, RNN, Accuracy Loss Curves

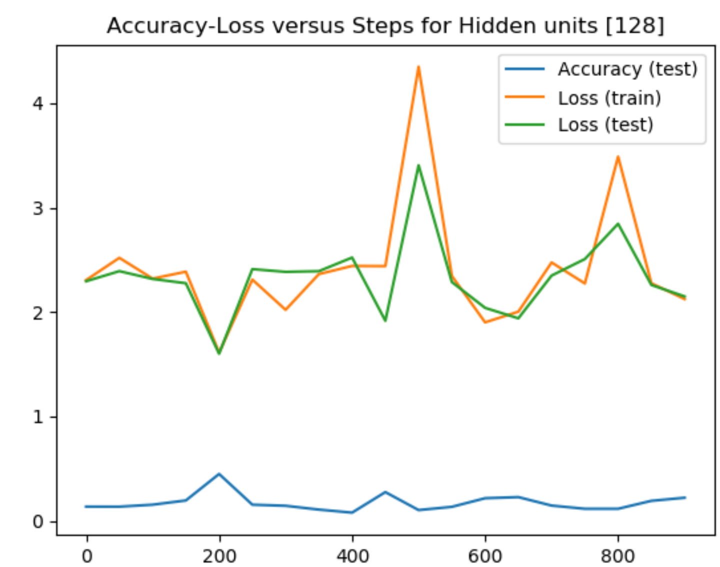
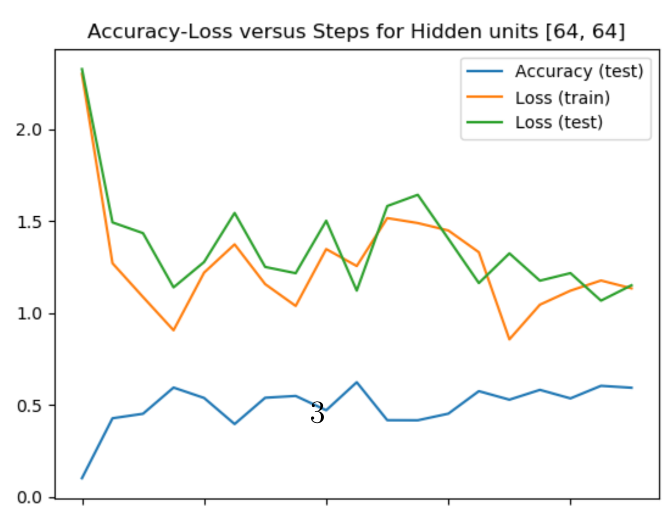


Figure 5: $H = 128$, RNN, Accuracy Loss Curves



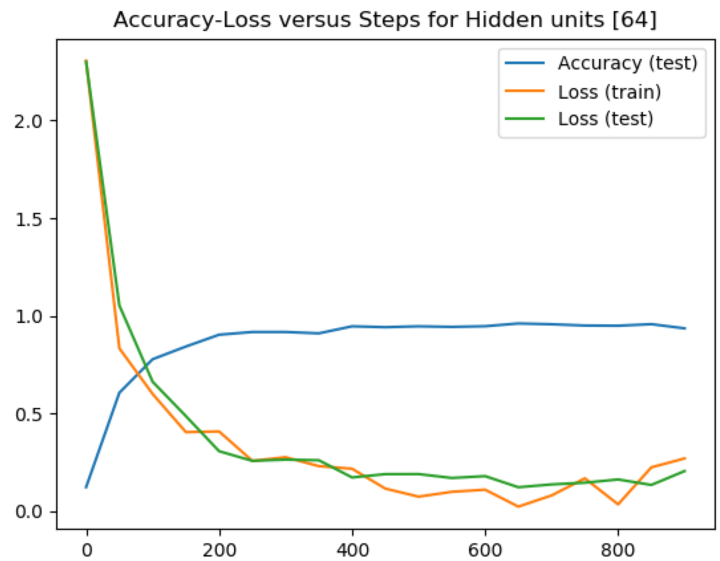


Figure 7: $H = 64$, GRU, Accuracy Loss Curves

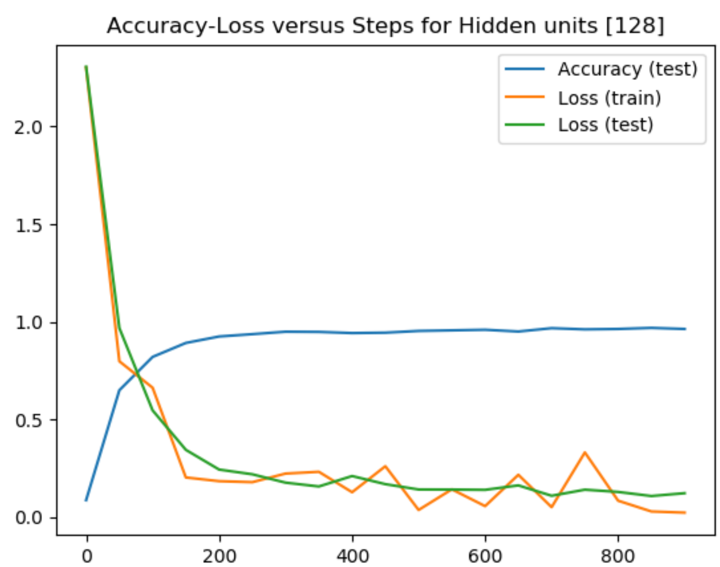
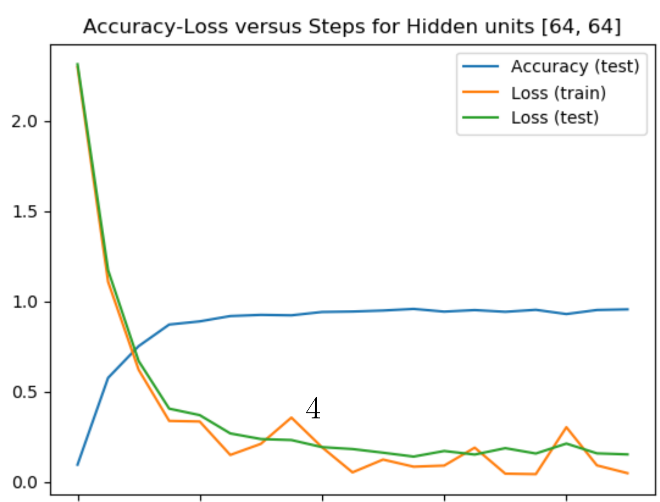


Figure 8: $H = 128$, GRU, Accuracy Loss Curves



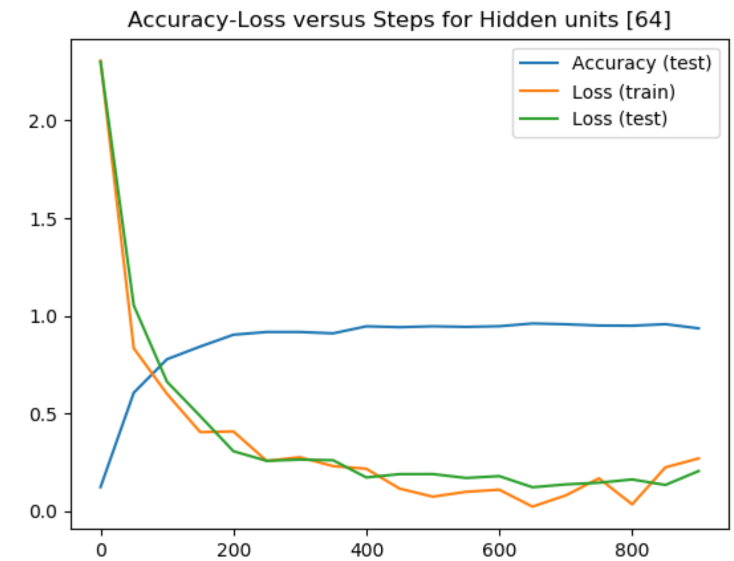


Figure 10: $H = 64$, GRU bidirectional, Accuracy Loss Curves

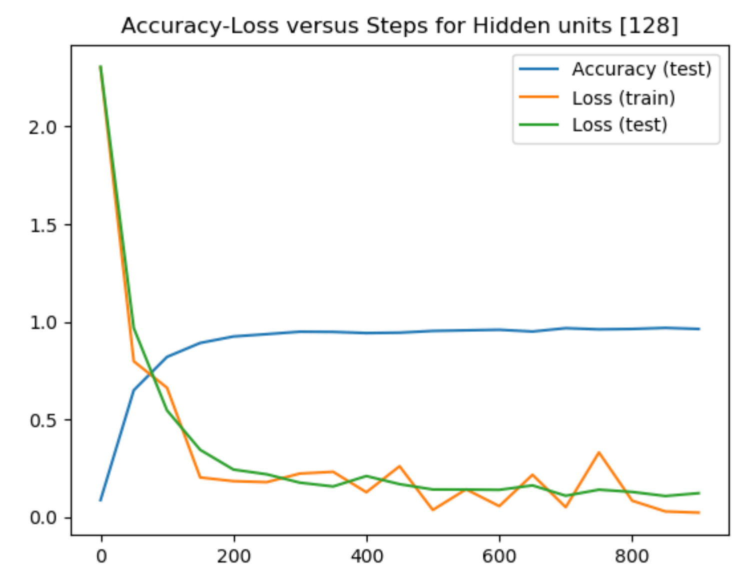
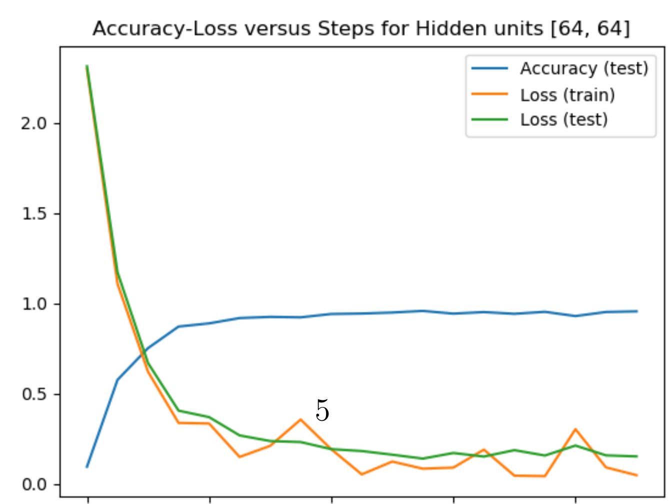


Figure 11: $H = 128$, GRU bidirectional, Accuracy Loss Curves



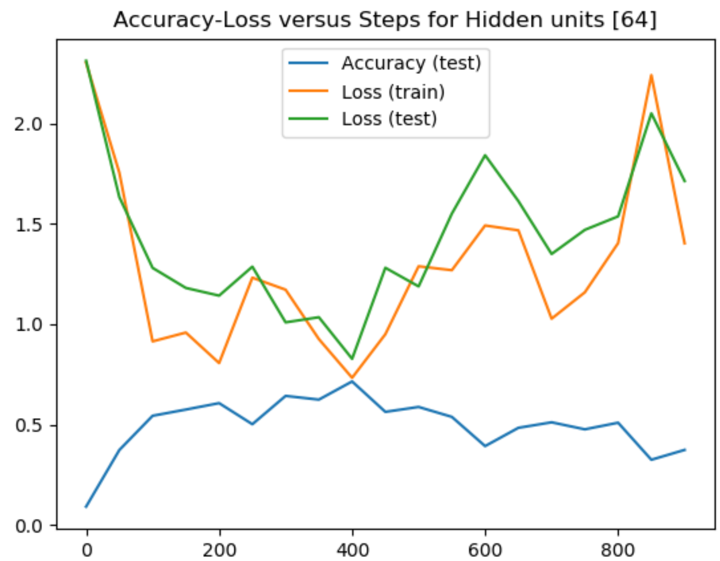


Figure 13: $H = 64$, RNN bidirectional, Accuracy Loss Curves

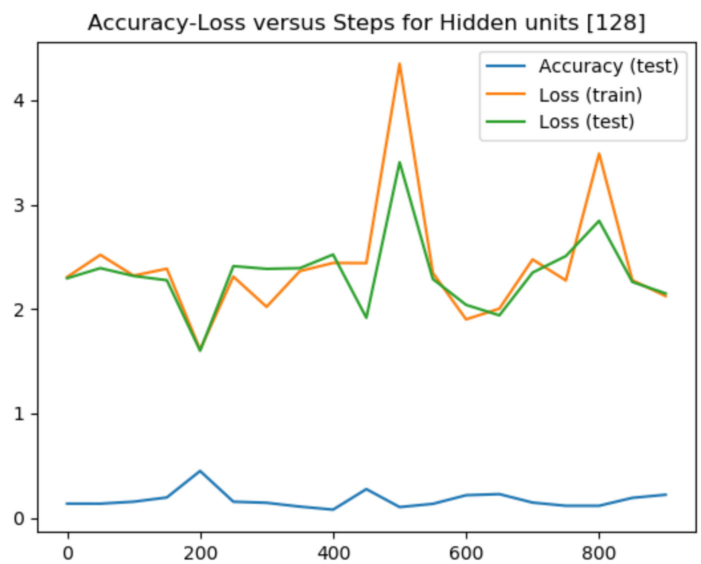
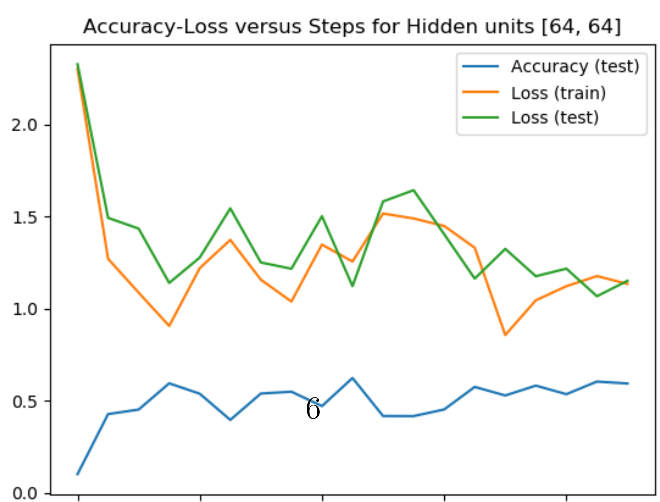


Figure 14: $H = 128$, RNN bidirectional, Accuracy Loss Curves



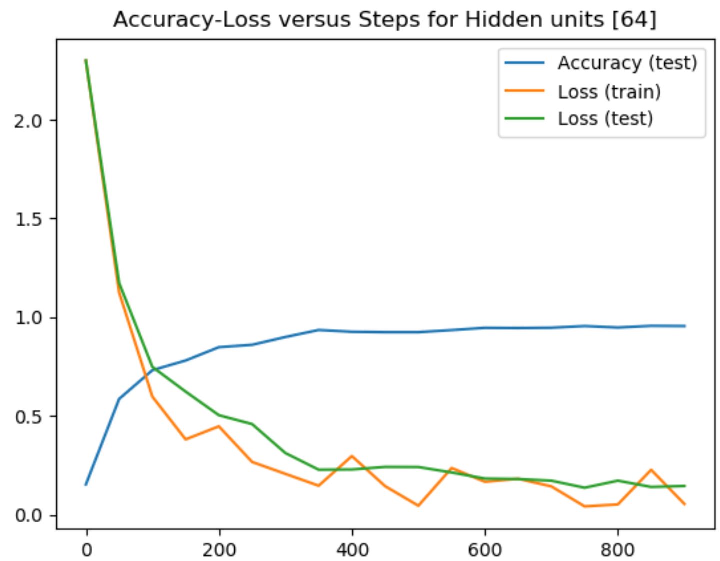


Figure 16: $H = 64$, LSTM bidirectional, Accuracy Loss Curves

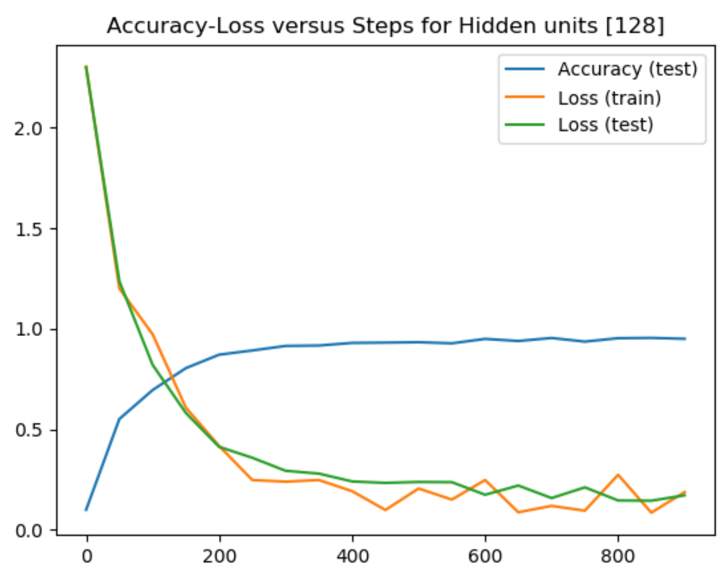
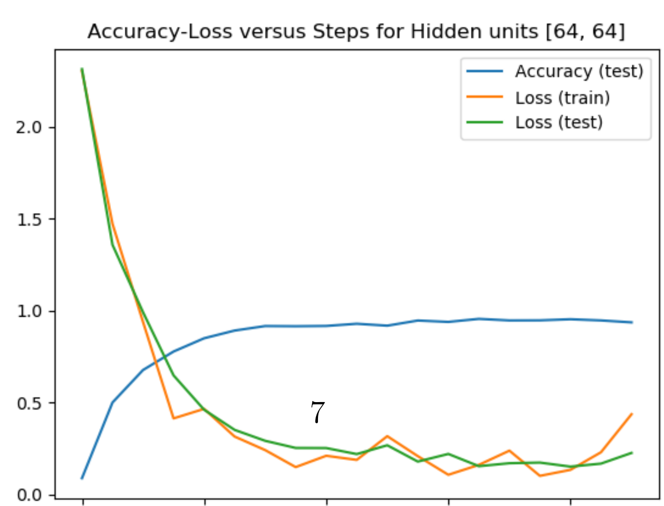


Figure 17: $H = 128$, LSTM bidirectional, Accuracy Loss Curves



3 Question 2

We use a LSTM with the following hyper-parameters:

- Adam optimiser, lr = 0.01, beta1 = 0.9, beta2 = 0.99
- Loss : Cross Entropy
- Single hidden layer, : 2,5,10 units wide.

The outputs of this question may be generated by:

```
python3 q3.py --H 5
```

We notice that increasing the hidden unit size increases the accuracy, due to better representation power.

We also perform randomised testing in the following format:

```
Input Sequence tensor([[3, 5, 8, 6, 8]], dtype=torch.int32)
Truth tensor([5])
Prediction 5
```

3.1 Loss and Accuracy Curves for H values

4 Question 3

We use a LSTM for this question with hidden (state) size as 5. We notice that increasing this to 10 improves accuracy (nearly 100%) and decreasing to 2 lowers accuracy. These plots however, aren't included here, and we experiment with state size 5.

The outputs of this question may be generated by:

```
python3 q3.py --loss_type MSE
```

Hyper-parameters:

- Adam optimiser, lr = 0.01, beta1 = 0.9, beta2 = 0.99
- Loss : MSE and Cross Entropy
- Single hidden layer, 5 units wide.

4.1 MSE vs Cross Entropy

We also note that MSE performs much better than Cross Entropy. This is because the output sequence is a binary number and their probabilities are correlated. Hence, we cannot use a "clean" version of cross entropy with L distinct distributions. We use a hackish method to use cross entropy, and for the reasons noted above, it performs poorly. We plot validation in blue, train in orange.

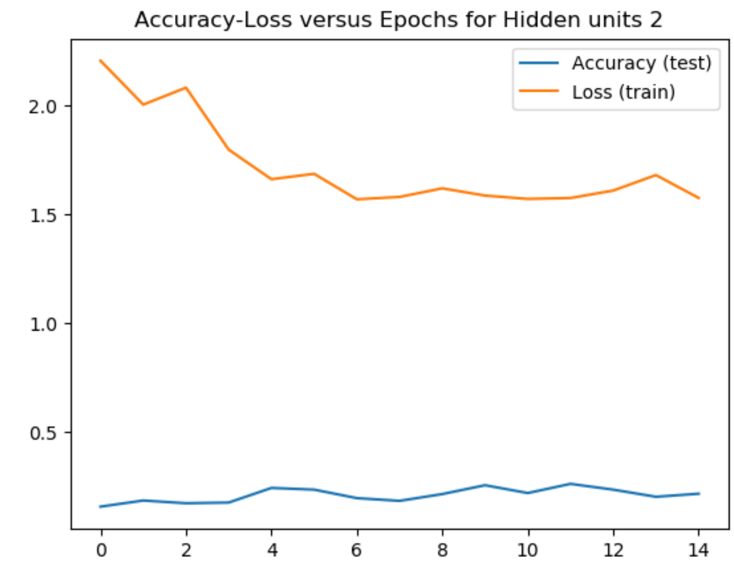


Figure 19: $H = 3$, Accuracy Loss Curves

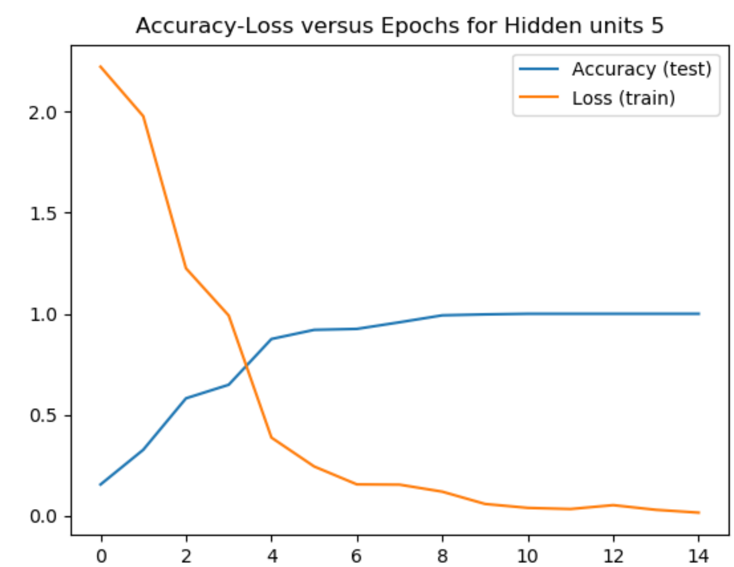
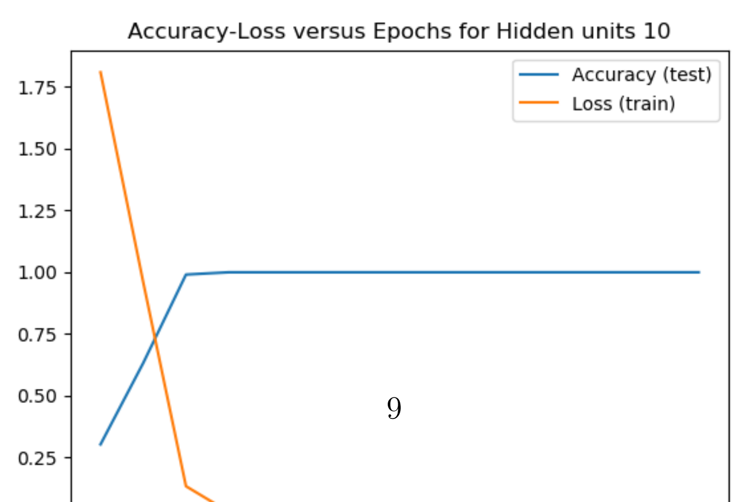


Figure 20: $H = 5$, Accuracy Loss Curves



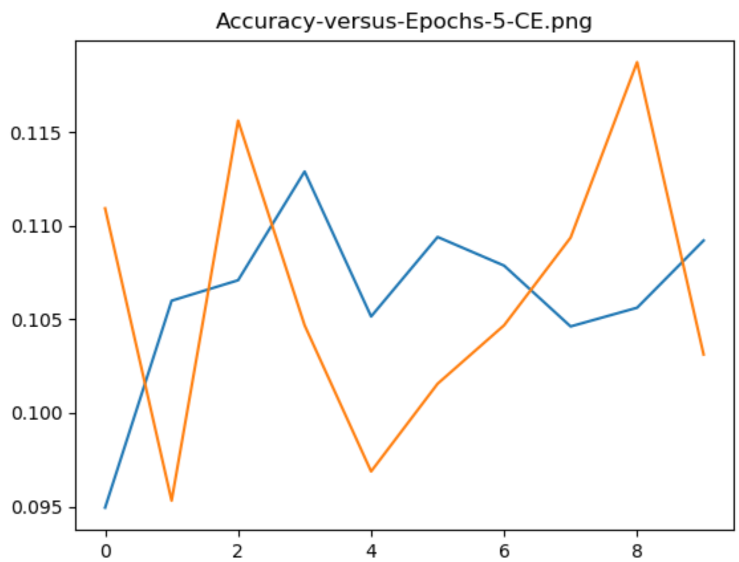


Figure 22: $L = 5$, Cross Entropy Loss

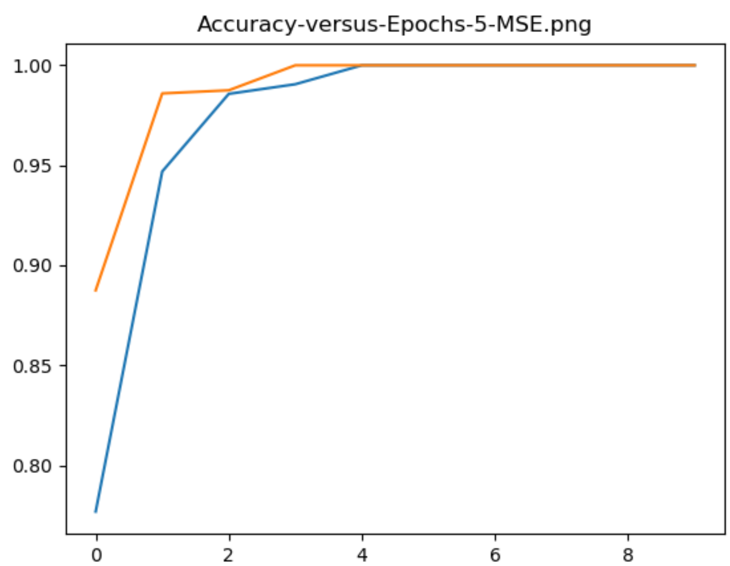


Figure 23: $L = 5$, MSE Loss

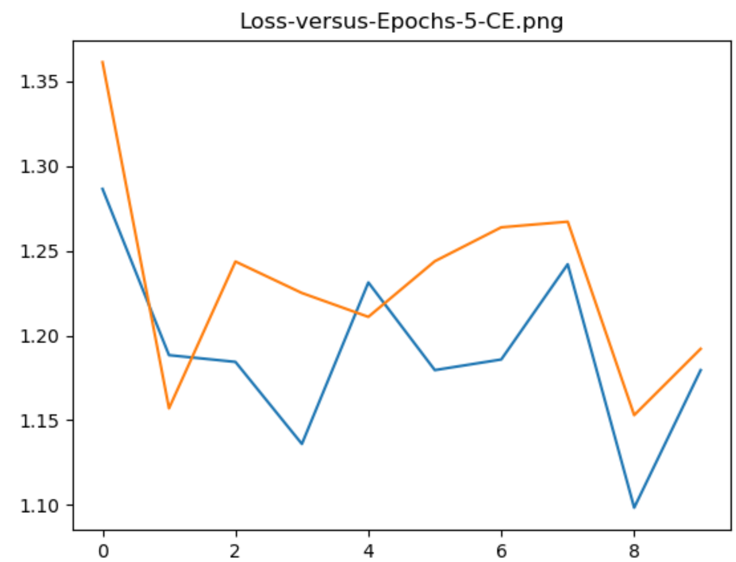


Figure 24: $L = 5$, Cross Entropy Loss

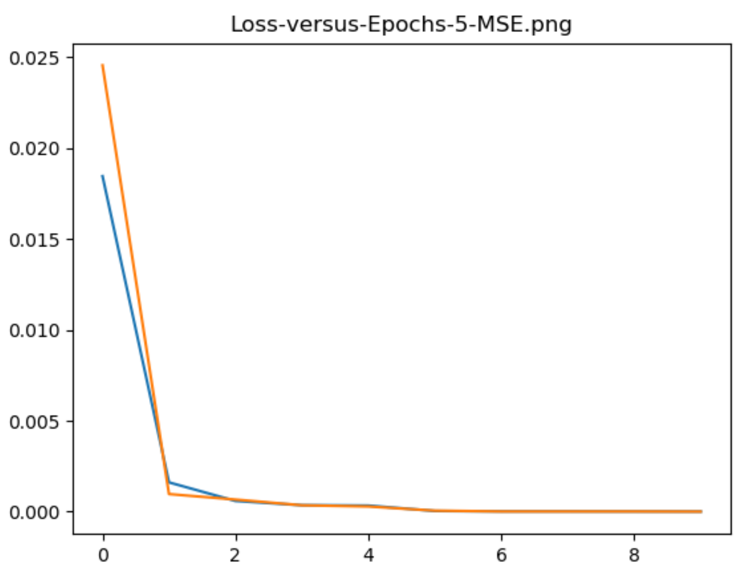


Figure 25: $L = 5$, MSE Loss

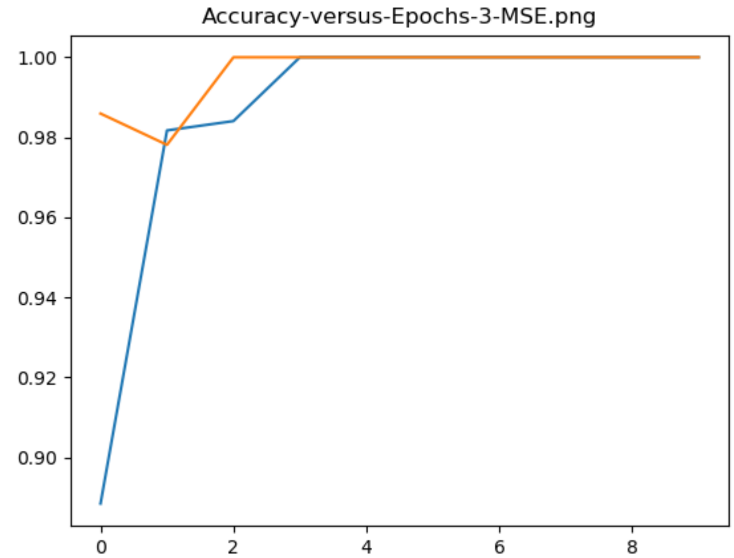


Figure 26: $L = 3$, Accuracy, MSE

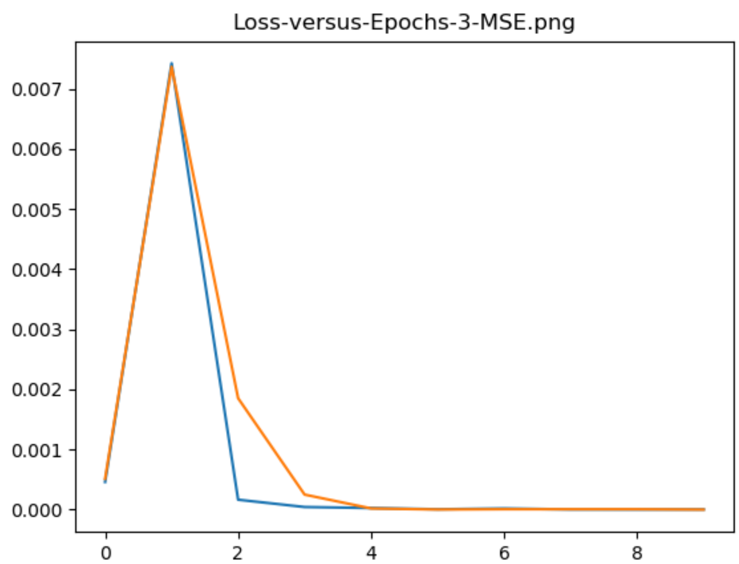


Figure 27: $L = 3$, Loss , MSE

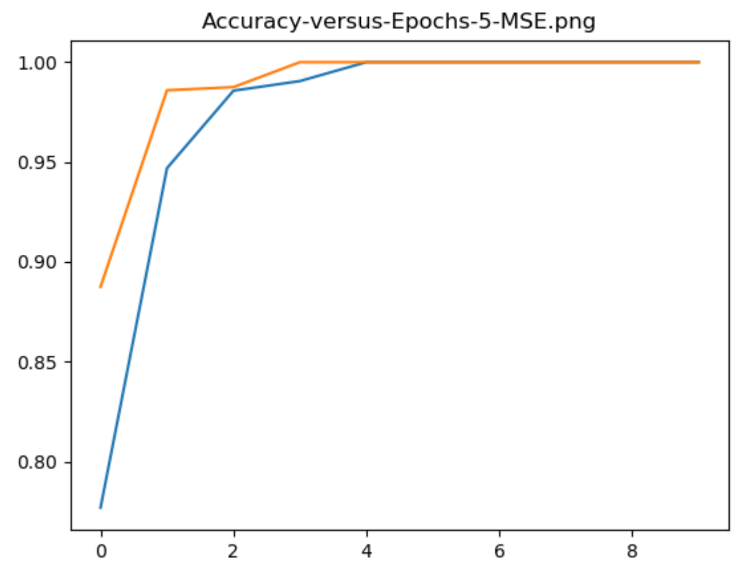


Figure 28: $L = 5$, Accuracy, MSE

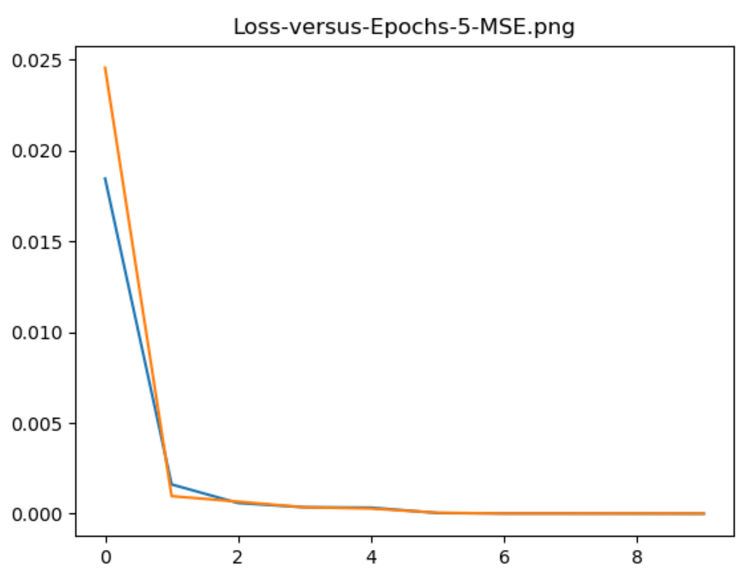


Figure 29: $L = 5$, Loss, MSE

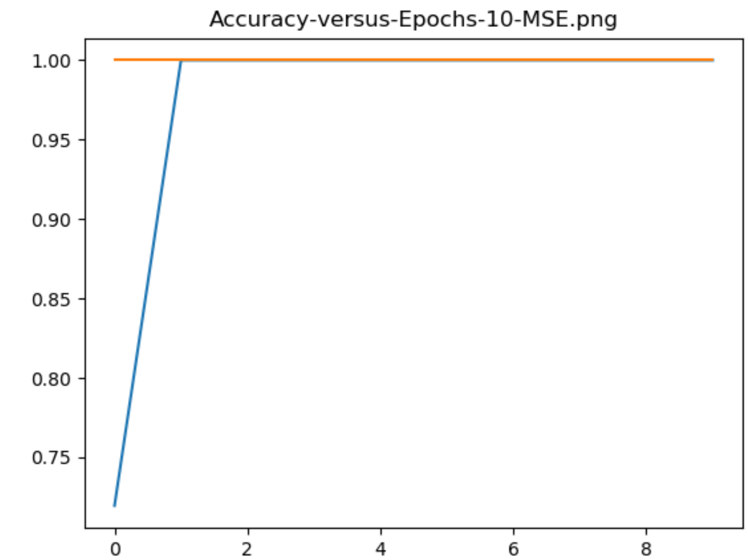


Figure 30: $L = 10$, Accuracy, MSE

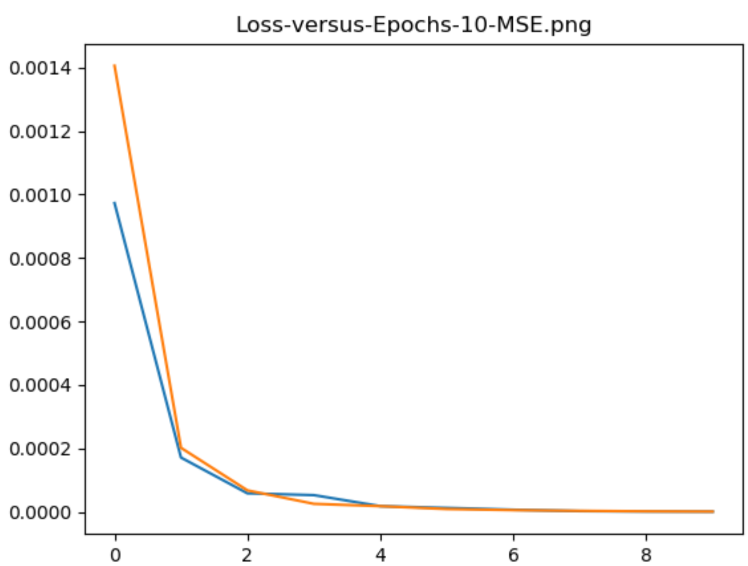


Figure 31: $L = 10$, Loss, MSE

4.2 MSE: Loss and Accuracy Curves

We include loss and accuracy curves for all sequence lengths (3,5,10) used.

4.3 MSE: Average Bit accuracy

We perform this for $L = 3, 5, 10$ as the training data.

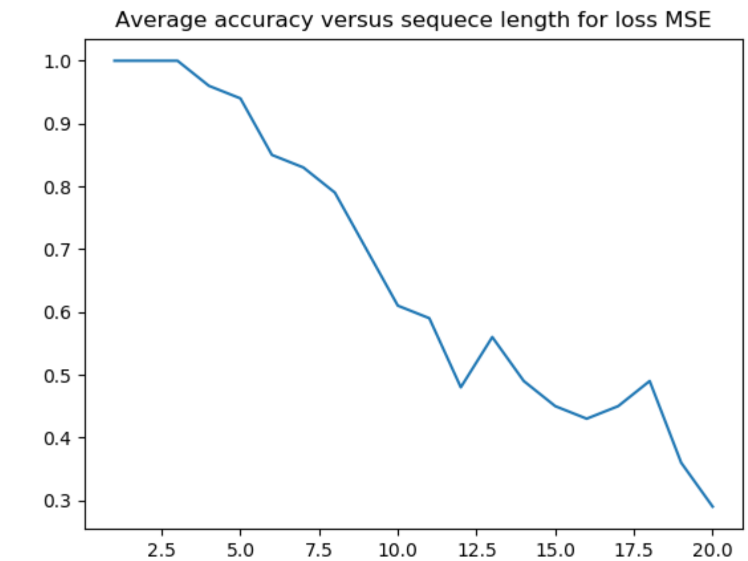


Figure 32: $L = 3$, Average Bit rate Loss

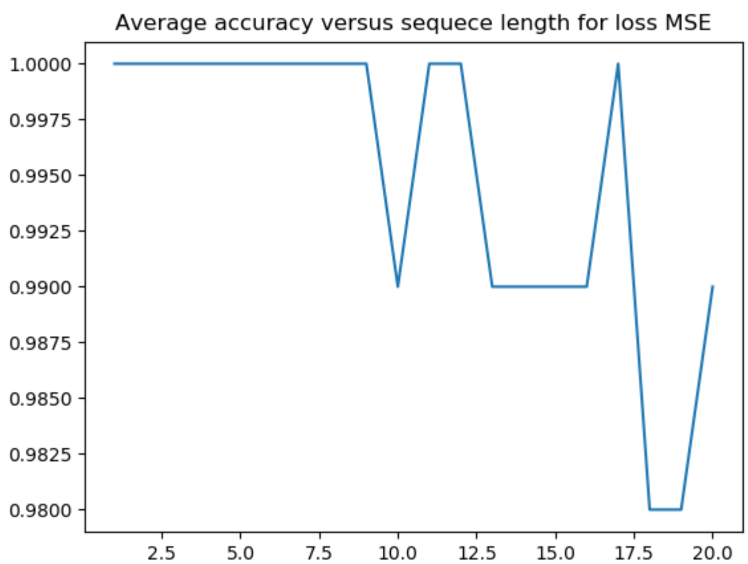


Figure 33: $L = 5$, Average Bit rate Loss