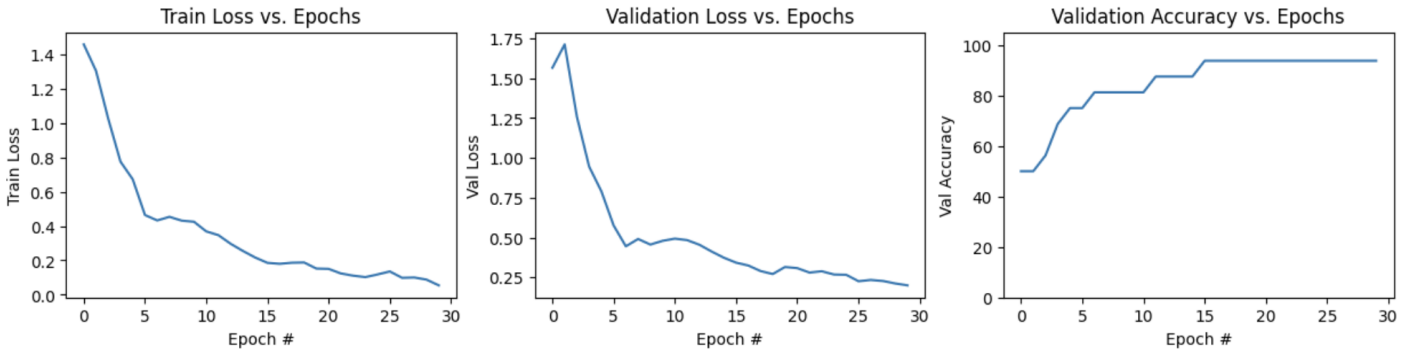
1. MNIST classification using RNN

Experiment 1 (Vanilla RNN)

Hyperparameters

- Hidden Dimension = 256
- SGD : lr=1e-3, momentum=0.9, weight_decay=5e-4
- Epochs = 30

Plots



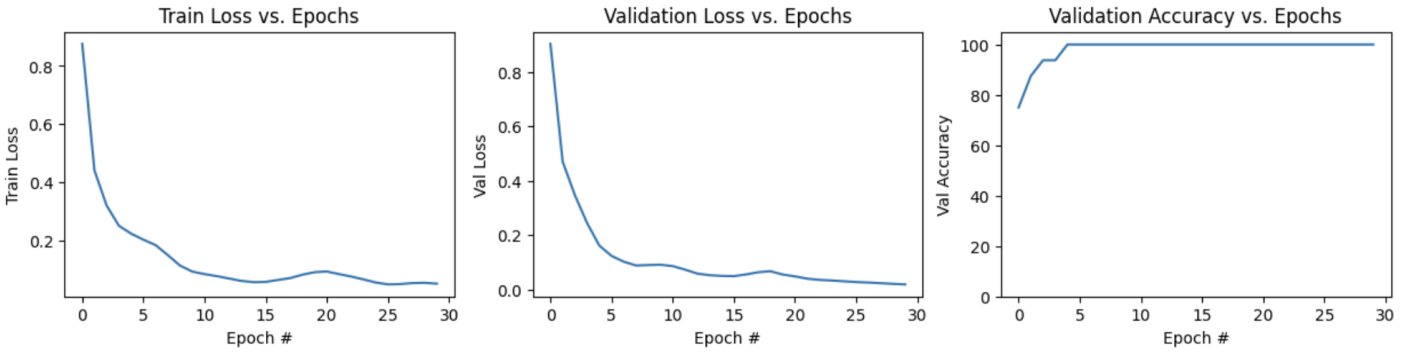
- **Average Test Prediction Accuracy = 100 %**
- Validation accuracy is 93.75%
- Train loss is converging to zero
- These are the best lr, momentum, and weight decay values I could find for hidden dim = 256, SGD

Experiment 2 (Vanilla RNN)

Hyperparameters

- Hidden Dimension = 256
- Adam : lr=1e-4, weight_decay=1e-6
- Epochs = 30

Plots



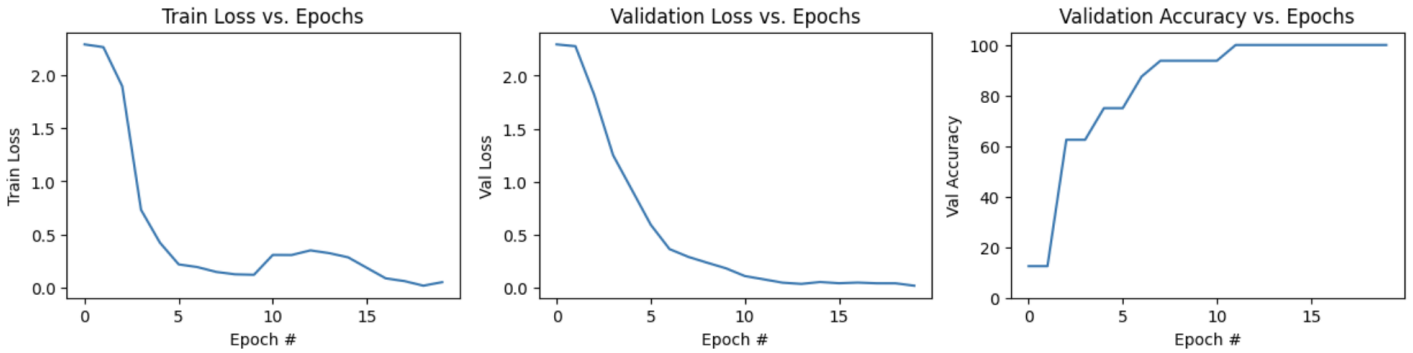
- **Average Test Prediction Accuracy = 100 %**
- Validation Accuracy is 100%
- Training loss is cleanly converging to zero, so the choice of learning rate is good.
- Validation loss is converging to zero, so there is no overfitting to training data.
- These are the best lr, and weight decay values I could find for hidden dim = 256, Adam

Experiment 3 (LSTM)

Hyperparameters

- Hidden Dimension = 128
- SGD : lr = 10 ** -2.5, momentum = 0.9
- Epochs = 20

Plots



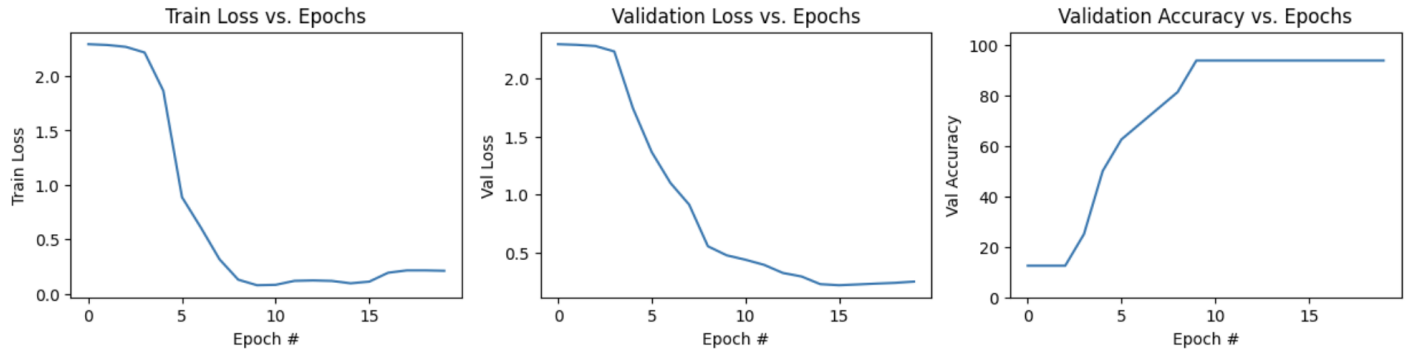
- **Average Test Prediction Accuracy = 93.75 %**
- Validation Accuracy is 100 %
- Although initially there are aberrations at the end, training loss & Validation loss are converging.

Experiment 4 (LSTM)

Hyperparameters

- Hidden Dimension = 128
- SGD : lr = 10 ** -2.7, momentum = 0.9
- Epochs = 20

Plots



- **Average Test Prediction Accuracy = 100 %**
- Validation Accuracy is 93.5 %
- For lr = 10**-2.5 validation accuracy and test accuracies are 100%, 93.75%, but for lr = 10**-2.7 they are 93.75% and 100%. How how they got interchanged

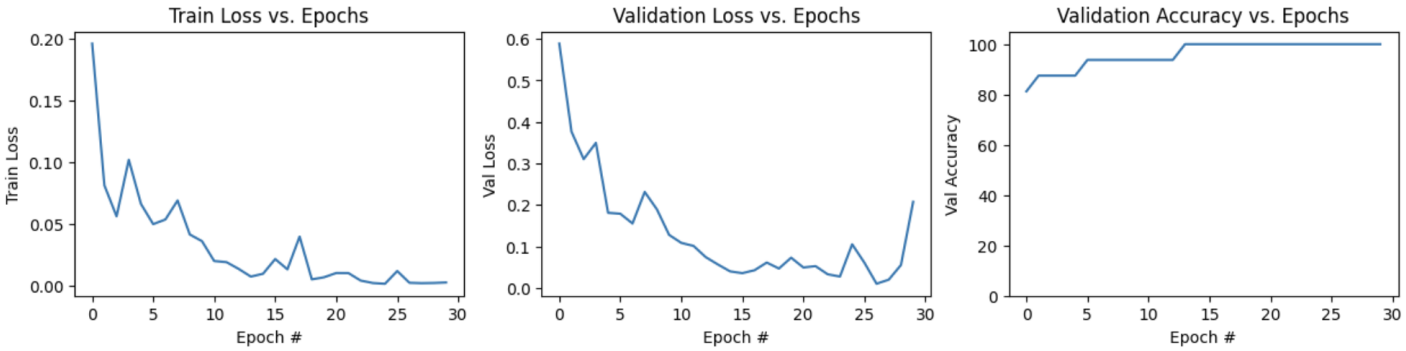
Experiment 5 (Bidirectional RNN)

Hyperparameters

- Hidden Dimension = 256, Num Layers = 2, Bidirectional
- Adam lr=10**-4, weight_decay=1e-6

- Epochs = 30

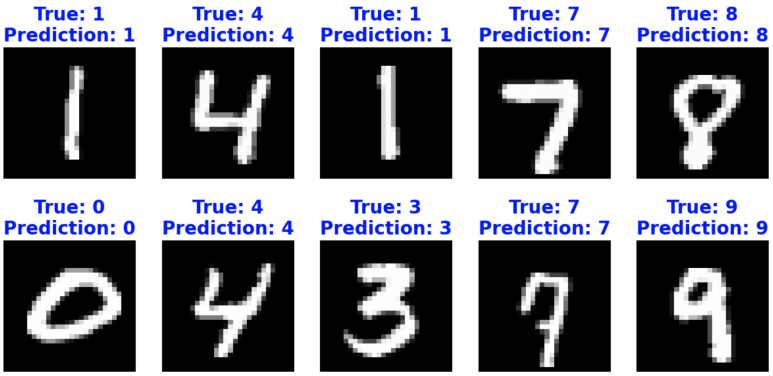
Plots



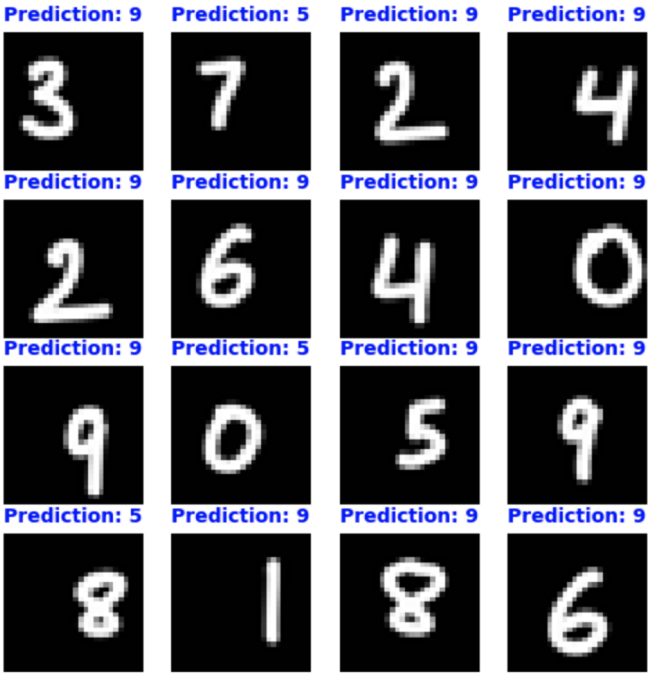
- **Average Test Prediction Accuracy = 100 %**
- Validation Accuracy is 100 %

Random Verification of Best Model

Experiment 2 with Vanilla RNN and Adam gave the best results, good loss curve and 100% accuracy on validation and training data.



Testing on My Handwriting



The accuracy of the model on my handwriting is really bad. Reason being, even though they look like digits to the human eye, the underlying distribution is different for MNIST and my handwriting. Hence the accuracies are bad.

A model that achieves “superhuman” performance when trained on a dataset can still make many basic errors when evaluated on another, possibly precisely because it is exploiting those dataset-specific quirks that humans are oblivious to (Geirhos et al., 2020).

2. Remembering number in sequence

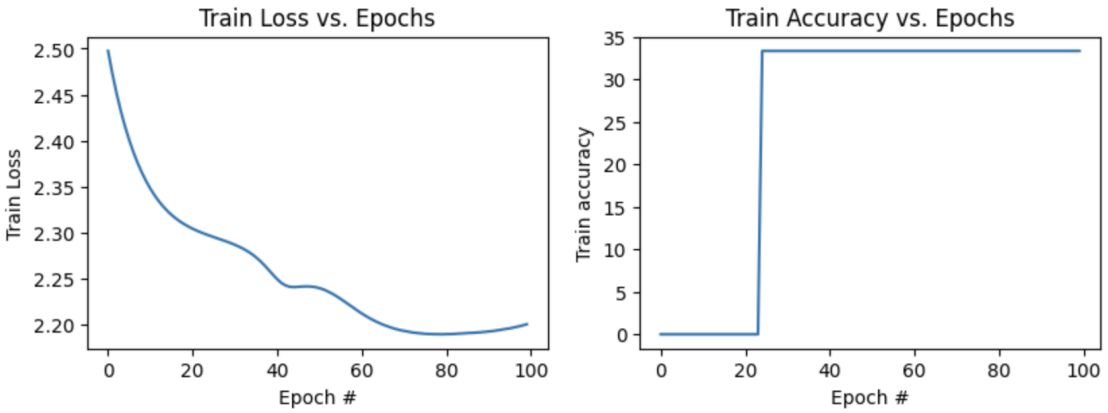
For hidden dim = 2, RNN cannot remember the number / its index because we are only passing two bits from one time step to another and a decimal number / its index requires more than 2 bits.

Model 1 (Hidden Dim = 2)

Hyperparameters & Other Info:

- LSTM
- Epochs = 100
- ADAM with lr = 1e-4, Weight Decay = 1e-4
- Cross Entropy
- Train Data Size = 1000 with variable sequence lengths picked from [3, 10]
- Test Data Size = 500 with variable sequence lengths picked from [3, 10]

Plots



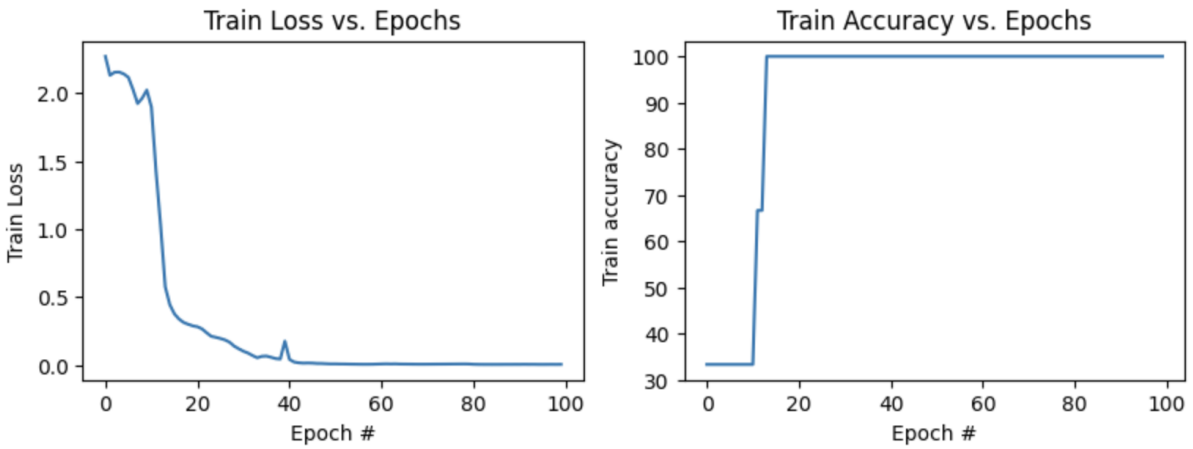
Test Accuracy = 33.33%

Model 2 (Hidden Dim = 5)

Hyperparameters & Other Info:

- LSTM
- Epochs = 100
- ADAM with lr = 1e-3, Weight Decay = 1e-4
- Cross Entropy
- Train Data Size = 1000 with variable sequence lengths picked from [3, 10]
- Test Data Size = 500 with variable sequence lengths picked from [3, 10]

Plots



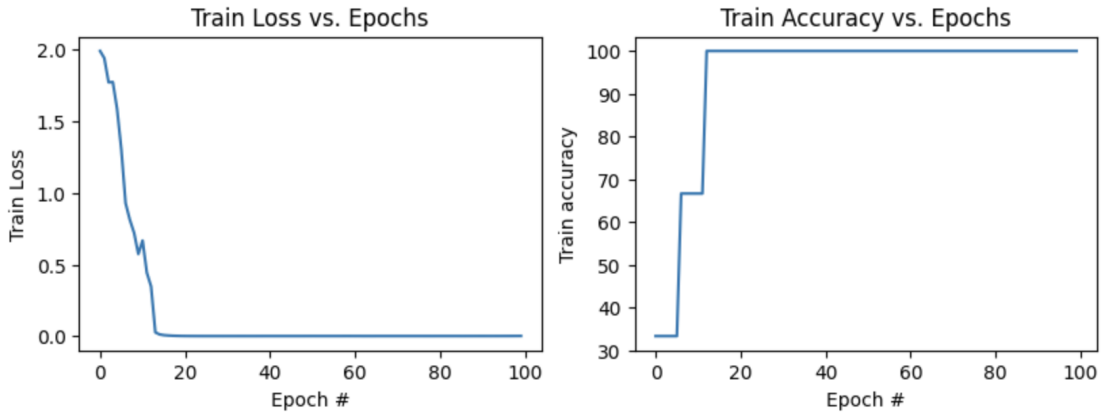
Test Accuracy = 100 %

Model 3 (Hidden Dim = 10)

Hyperparameters & Other Info:

- LSTM
- Epochs = 100
- ADAM with lr = 1e-3, Weight Decay = 1e-4
- Cross Entropy
- Train Data Size = 1000 with variable sequence lengths picked from [3, 10]
- Test Data Size = 10000 with variable sequence lengths picked from [3, 10]

Plots



Test Accuracy = 100%

Comparison

- Model 1, 2, 3 are having hidden size dimensions as 2, 5, 10 respectively
- Model 1 didn't converge to 0 training loss
- Model 2 and Model 3 converged to 0 training loss
- Model 3 converged quickly compared to model 2, reason might be: As model 3 has a large number of parameters we don't have to compress the patterns that much, but in model 2 we need to represent the pattern using fewer weights.
- Model 2 and Model 3 gave 100% accuracy on the test set, model 1 gave only 33.33% accuracy on test set.

Random Testing of Best Model

Best model is model 3. It was trained to remember the 3rd item(1-indexing) from the list. Here are some random tests conducted with different input lengths, and the model inference is correct in all the cases.

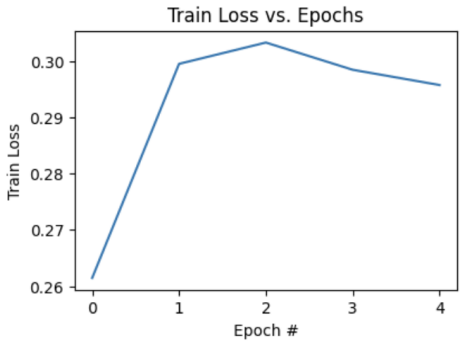
Input [1, 3, 6]
Output 6
Input [9, 0, 8]
Output 8
Input [5, 0, 7, 4]
Output 7
Input [4, 2, 8, 7]
Output 8
Input [3, 0, 6, 2, 2]
Output 6
Input [0, 5, 4, 0, 4]
Output 4
Input [6, 8, 7, 4, 6, 9]
Output 7
Input [4, 4, 7, 7, 0, 7]
Output 7
Input [0, 2, 2, 1, 8, 4, 3]
Output 2
Input [1, 2, 2, 2, 0, 6, 2]
Output 2
Input [7, 9, 5, 6, 9, 8, 6, 0]
Output 5
Input [5, 0, 1, 4, 3, 7, 9, 1]
Output 1
Input [0, 9, 8, 5, 1, 2, 6, 7, 2]
Output 8
Input [3, 9, 5, 0, 3, 5, 9, 3, 1]
Output 5
Input [5, 6, 0, 0, 3, 6, 3, 4, 0, 3]
Output 0
Input [0, 2, 4, 3, 8, 2, 3, 5, 3, 9]
Output 4

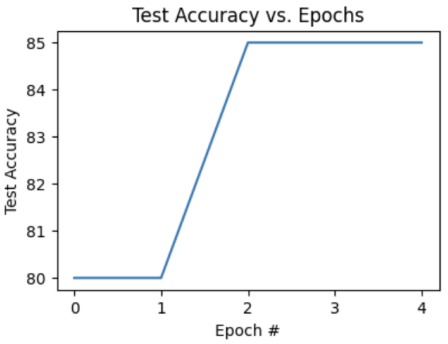
3. Adding two binary strings

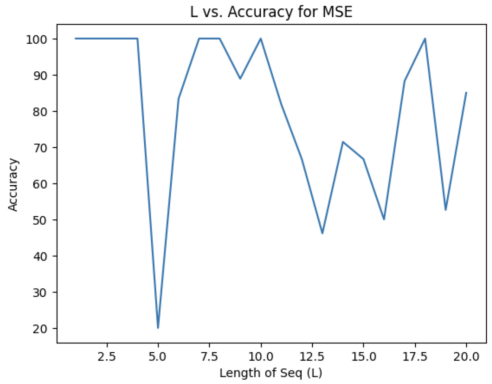
Experiment - 1

- Hyperparameters & Other Info:
  - Epochs = 5
  - ADAM with lr = 1e-3, Weight Decay = 5e-4
  - MSELoss
  - Train Data Size = 10000 with variable sequence lengths picked from [1, 20]
  - Test Data Size for each L in [1, 20] = 100

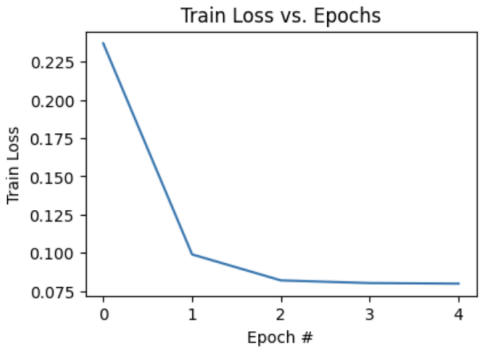
Plots (State Size = 1)

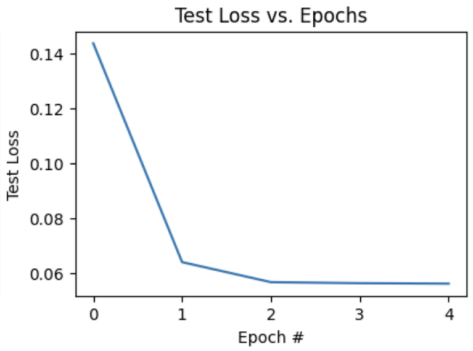


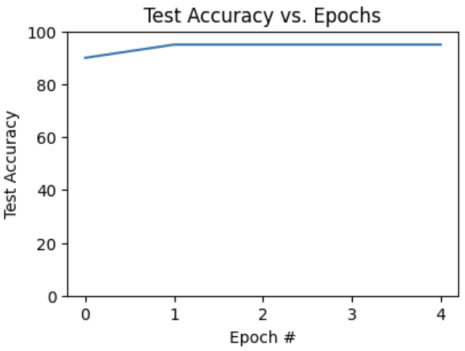


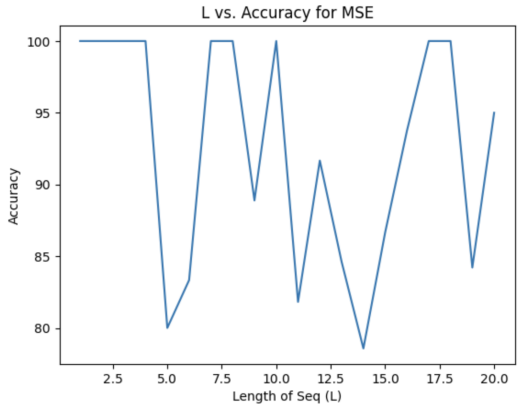


Plots (State Size = 3)

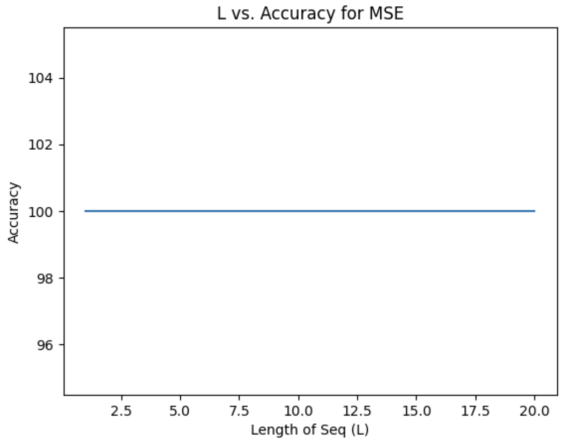








Plots (State Size = 5)



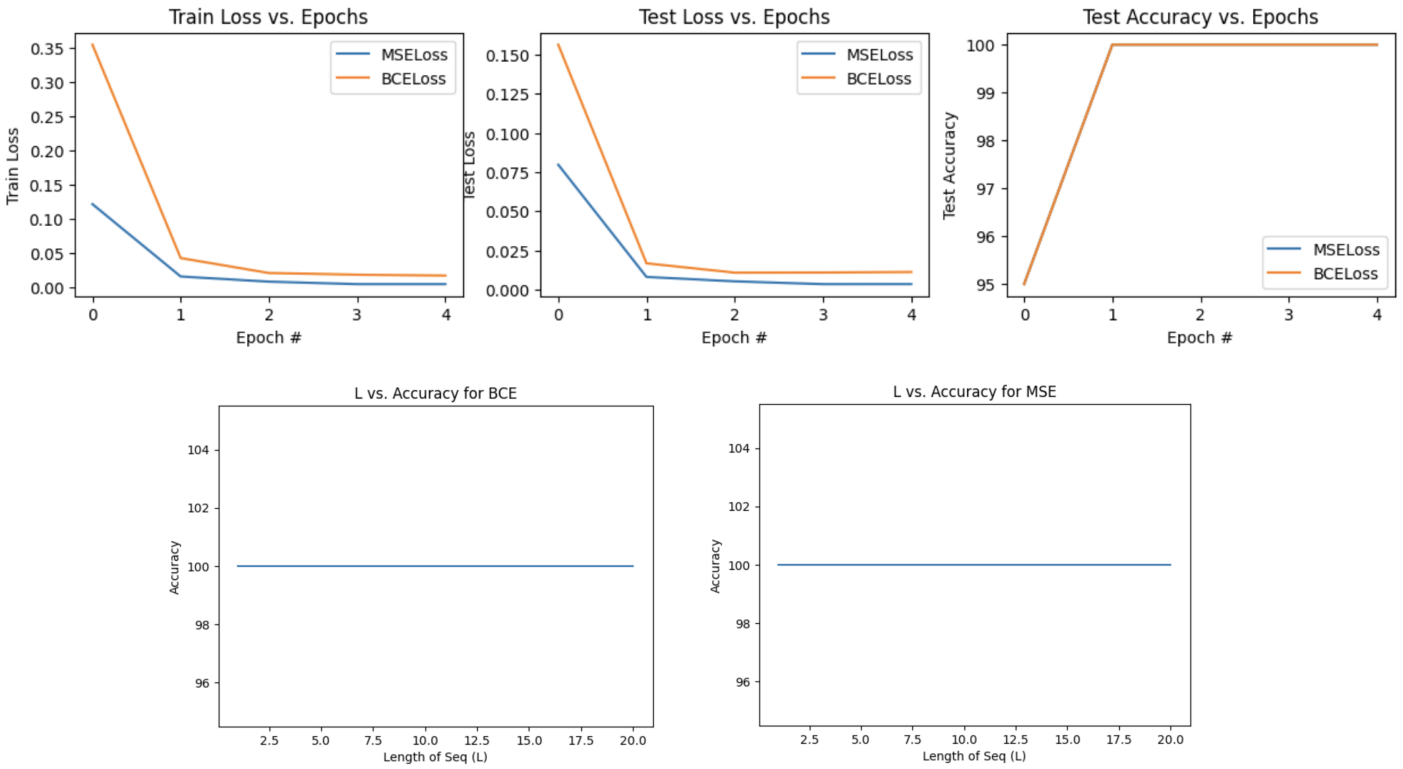
Observations

- With the same set of HyperParams, Test accuracy increased with the increase in hidden state size
    - Hidden Dim = 1, gave 85% accuracy
    - Hidden Dim = 3, gave 95% accuracy
    - Hidden Dim = 5, gave 100% accuracy
- Hidden dimension = 5 also gave a good generalisation for any length.
- The observations are consistent with the theoretical fact that greater the hidden states, greater the information that can be stored by the model.

Experiment - 2

- Comparing MSELoss and BCELoss
- Hyperparameters & Other Info:
    - Epochs = 5, Hidden Dim = 5
    - ADAM with lr = 1e-3, Weight Decay = 5e-4
    - Train Data Size = 10000 with variable sequence lengths picked from [1, 20]
    - Test Data Size for each L in [1, 20] = 100

Plots



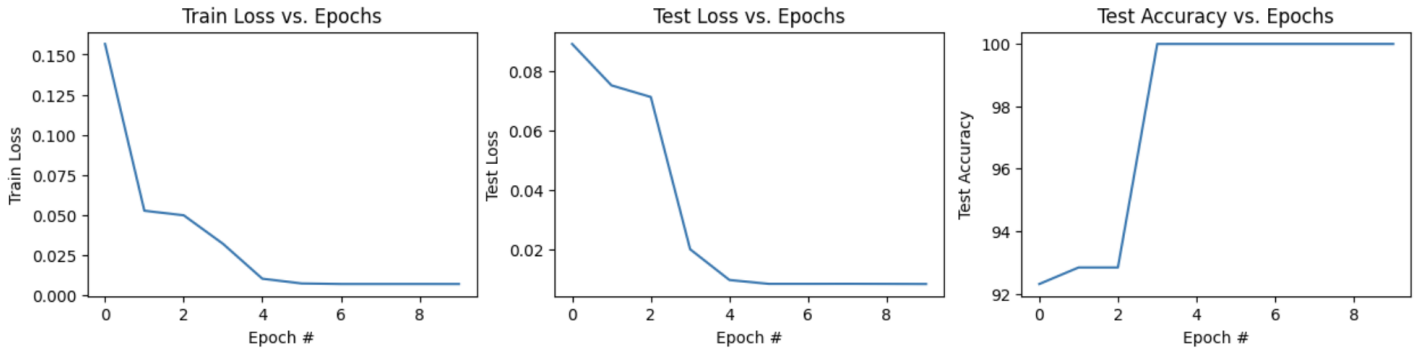
Observations

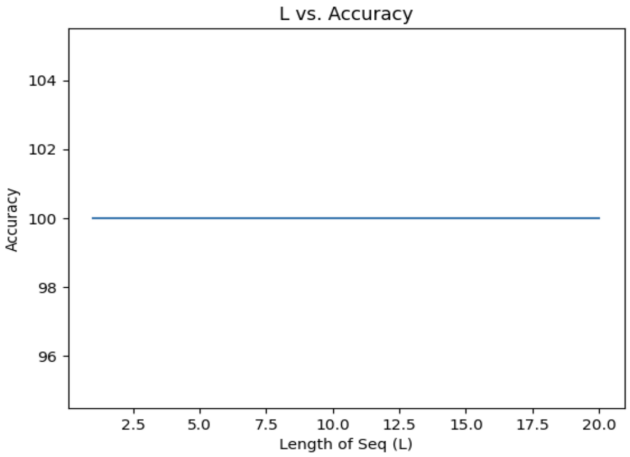
- Here we are dealing with a classification problem, we have to predict the class as '0' or '1' for each time step.
    - Theoretically BCELoss is well suited for classification problems, the reason being it leverages the fact that the output is either '0' or '1'.
    - Whereas MSELoss doesn't have any assumptions about the output hence making MSELoss more suitable for regression problems.
- Here we are dealing with a simple task of addition, so both MSELoss and BCELoss managed to achieve 100% Test Accuracy, and also managed to converge nearly in the same fashion.

Experiment - 3

- Trained the model only on L = 5 and tested on random length (1 to 20) sequences.
- Hyperparameters & Other Info:
    - Epochs = 10, Hidden Dim = 5
    - ADAM with lr = 1e-3, Weight Decay = 5e-4
    - MSELoss
    - Train Data Size = 10000, with L = 5
    - Test Data Size for each L in [1, 20] = 100

Plots





Observations

- The model is trained to do addition of L = 5 bit sequences, but it achieved 100% accuracy even on sequences with other lengths.
- It means the model learnt the concept of addition, like carrying the bit from one time step to another.