Source Code

Remote Repository: https://github.com/Mikestriken/Deep-Learning-Class
Name: Michael Marais
Student ID: 801177649
Homework Number: 3

Problem 1

In this homework, we focus on the language model we did in the lectures. However, we expand it to a much longer sequence. Here is the sequence:

"Next character prediction is a fundamental task in the field of natural language processing (NLP) that involves predicting the next character in a sequence of text based on the characters that precede it. This task is essential for various applications, including text auto-completion, spell checking, and even in the development of sophisticated AI models capable of generating human-like text.

At its core, next character prediction relies on statistical models or deep learning algorithms to analyze a given sequence of text and predict which character is most likely to follow. These predictions are based on patterns and relationships learned from large datasets of text during the training phase of the model.

One of the most popular approaches to next character prediction involves the use of Recurrent Neural Networks (RNNs), and more specifically, a variant called Long Short-Term Memory (LSTM) networks. RNNs are particularly well-suited for sequential data like text, as they can maintain information in 'memory' about previous characters to inform the prediction of the next character. LSTM networks enhance this capability by being able to remember long-term dependencies, making them even more effective for next character prediction tasks.

Training a model for next character prediction involves feeding it large amounts of text data, allowing it to learn the probability of each character's appearance following a sequence of characters. During this training process, the model adjusts its parameters to minimize the difference between its predictions and the actual outcomes, thus improving its predictive accuracy over time.

Once trained, the model can be used to predict the next character in a given piece of text by considering the sequence of characters that precede it. This can enhance user experience in text editing software, improve efficiency in coding environments with auto-completion features, and enable more natural interactions with AI-based chatbots and virtual assistants.

In summary, next character prediction plays a crucial role in enhancing the capabilities of various NLP applications, making text-based interactions more efficient, accurate, and human-like. Through the use of advanced machine learning models like RNNs and LSTMs, next character prediction continues to evolve, opening new possibilities for the future of text-based technology."

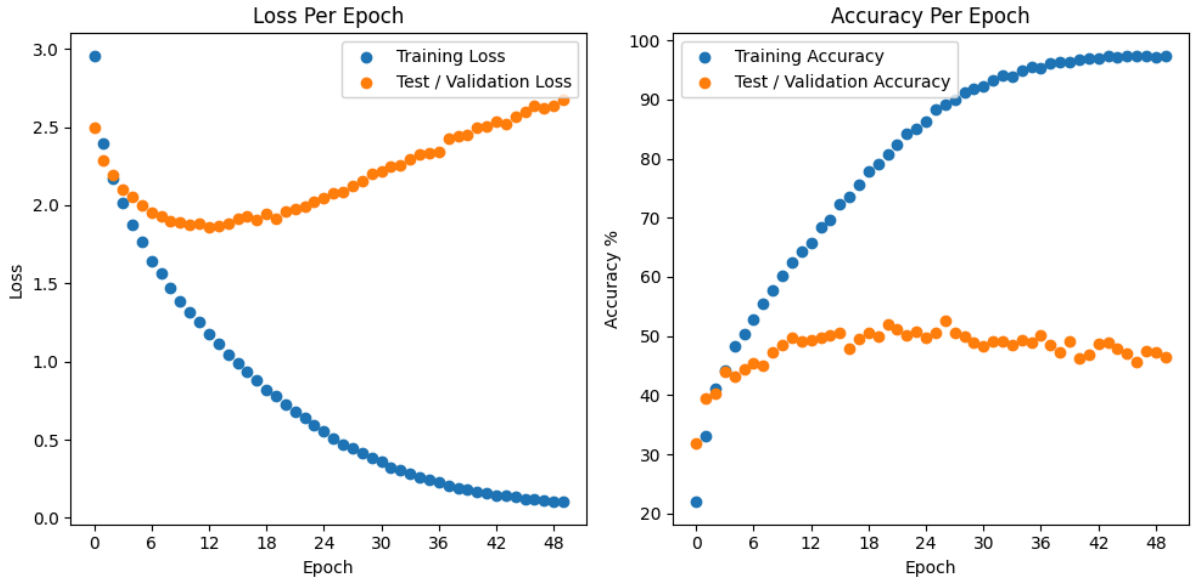
Inspired by the course example, train and validate rnn.RNN, rnn.LSTM and rnn.GRU for learning the above sequence. Use sequence lengths of 10, 20, and 30 for your training. Feel free to adjust other network parameters. Report and compare training loss, validation accuracy, execution time for training, and computational and mode size complexities across the three models over various lengths of sequence.

Preface

To estimate the computational complexity I will measure the number of FLOPS (FLoating-point OPerations per Second) and MACS (Multiply–ACcumulate Operations per Second). Size complexity will simply be the number of parameters. To do this I am using the calflops library.

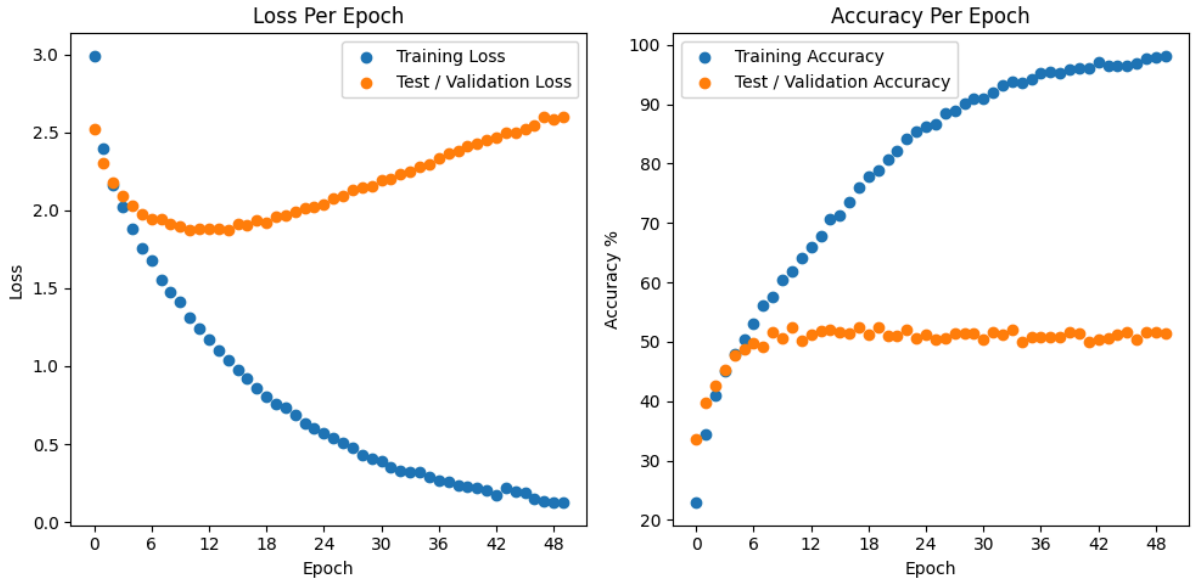
nn.RNN

Sequence Length 10



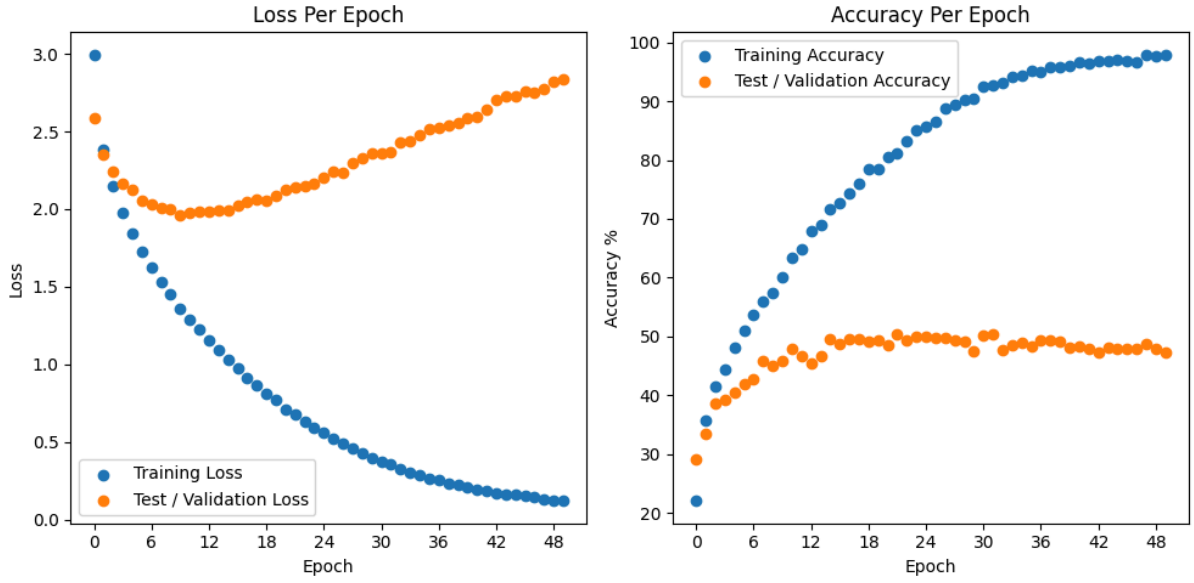
Last Best Epoch: 26, Train Time ≈ 0 hr, 0 min, 8 sec
Train Loss: 0.47051, Train Accuracy: 89.11%
Test Loss: 2.08796, Test Accuracy: 52.52%
Model Parameters: 44.332 K, FLOPS: 21.3729 M, MACS: 180.224 K

Sequence Length 20



Last Best Epoch: 10, Train Time ≈ 0 hr, 0 min, 8 sec
Train Loss: 1.30937, Train Accuracy: 61.89%
Test Loss: 1.87554, Test Accuracy: 52.53%
Model Parameters: 44.332 K, FLOPS: 42.3854 M, MACS: 180.224 K

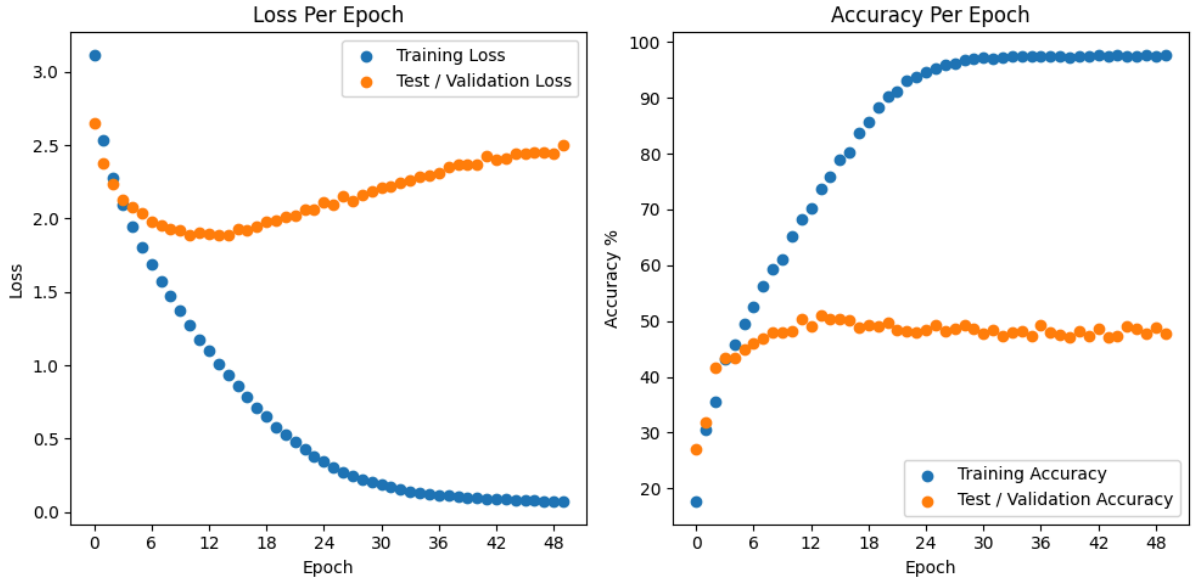
Sequence Length 30



Last Best Epoch: 21, Train Time ≈ 0 hr, 0 min, 8 sec
Train Loss: 0.67516, Train Accuracy: 81.16%
Test Loss: 2.14342, Test Accuracy: 50.42%
Model Parameters: 44.332 K, FLOPS: 63.3979 M, MACS: 180.224 K

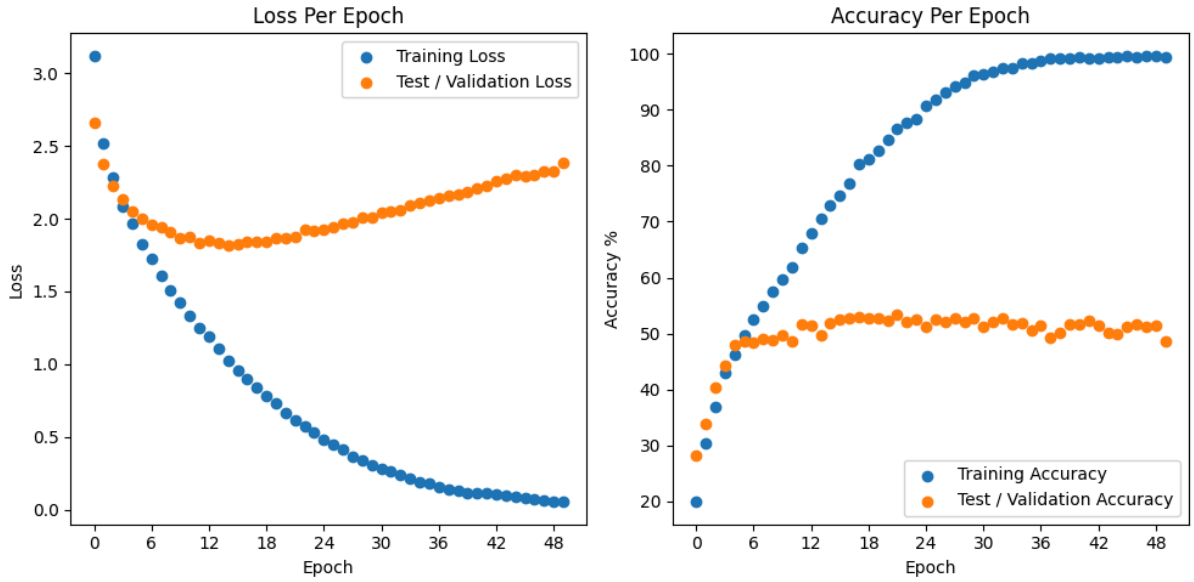
nn.LSTM

Sequence Length 10



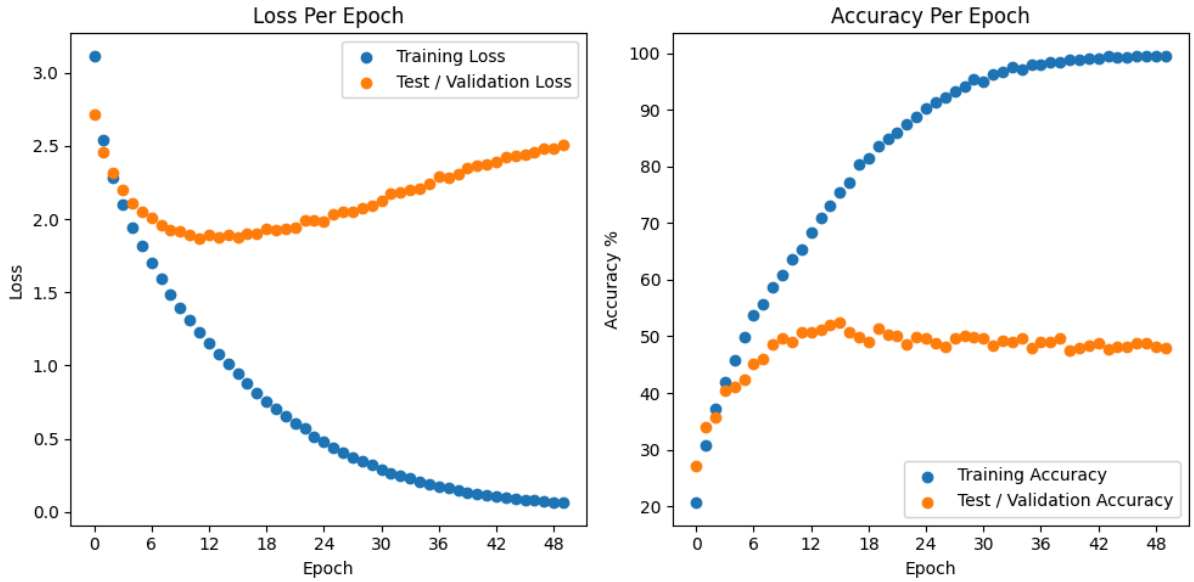
Last Best Epoch: 13, Train Time ≈ 0 hr, 0 min, 9 sec
Train Loss: 1.01199, Train Accuracy: 73.68%
Test Loss: 1.88519, Test Accuracy: 51.05%
Model Parameters: 143.404 K, FLOPS: 84.6561 M, MACS: 180.224 K

Sequence Length 20



Last Best Epoch: 21, Train Time ≈ 0 hr, 0 min, 9 sec
Train Loss: 0.61278, Train Accuracy: 86.52%
Test Loss: 1.87334, Test Accuracy: 53.38%
Model Parameters: 143.404 K, FLOPS: 168.952 M, MACS: 180.224 K

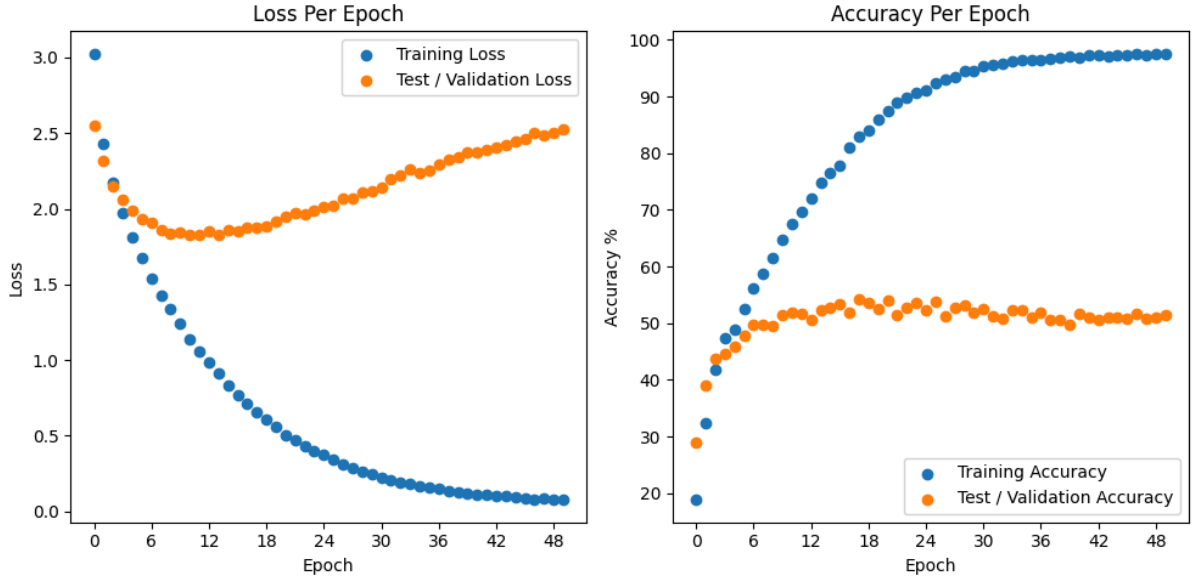
Sequence Length 30



Last Best Epoch: 15, Train Time ≈ 0 hr, 0 min, 8 sec
Train Loss: 0.94197, Train Accuracy: 75.48%
Test Loss: 1.87275, Test Accuracy: 52.54%
Model Parameters: 143.404 K, FLOPS: 253.248 M, MACS: 180.224 K

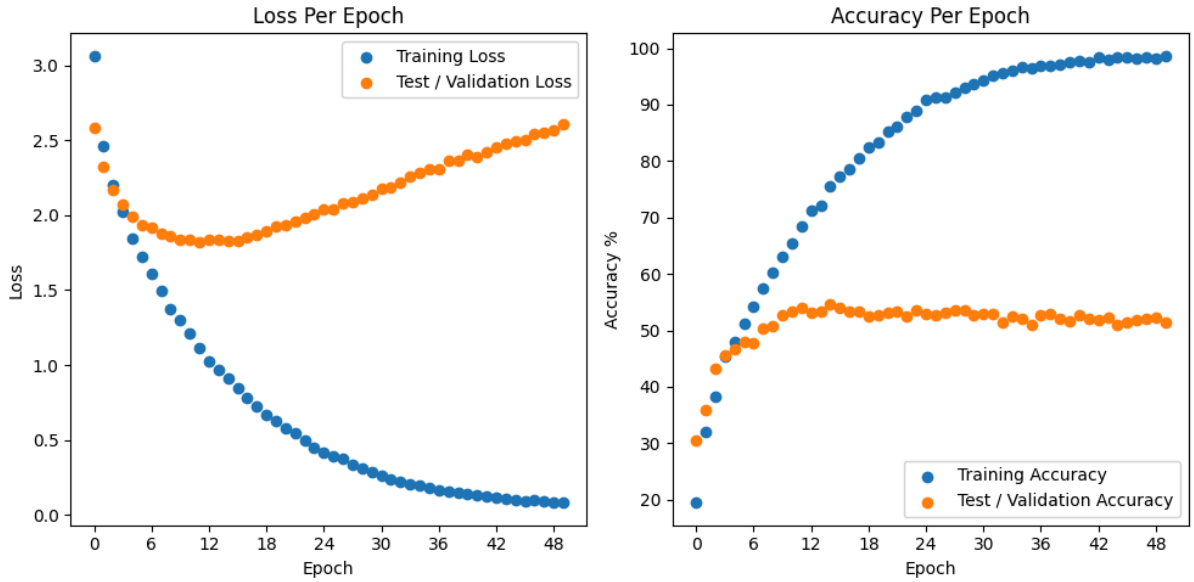
nn.GRU

Sequence Length 10



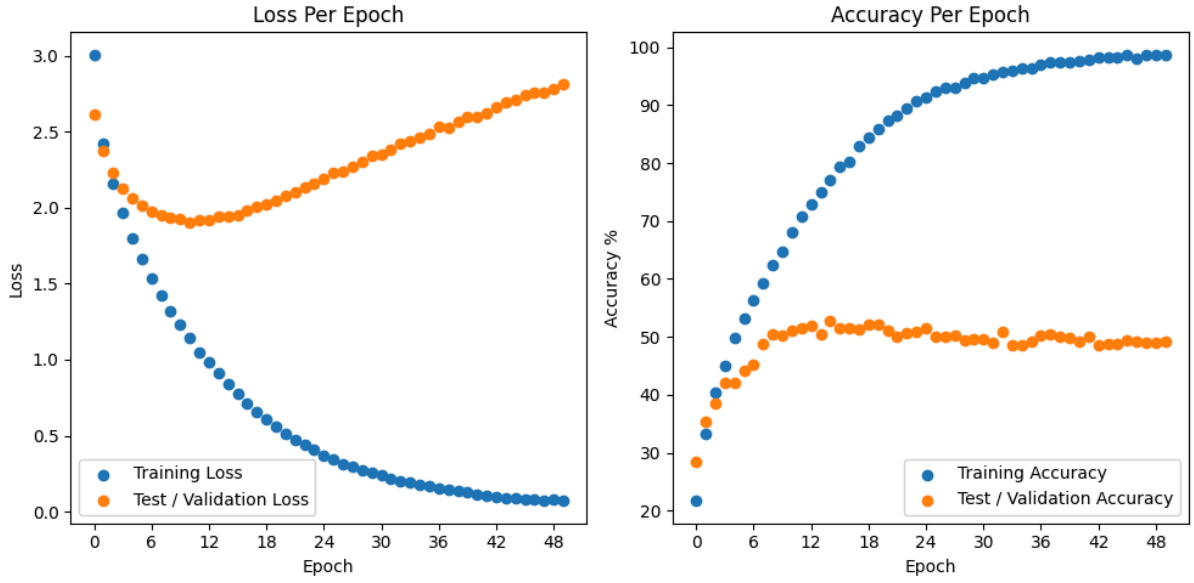
Last Best Epoch: 17, Train Time ≈ 0 hr, 0 min, 8 sec
Train Loss: 0.65683, Train Accuracy: 82.95%
Test Loss: 1.87684, Test Accuracy: 54.20%
Model Parameters: 110.38 K, FLOPS: 63.5617 M, MACS: 180.224 K

Sequence Length 20



Last Best Epoch: 14, Train Time ≈ 0 hr, 0 min, 8 sec
Train Loss: 0.91426, Train Accuracy: 75.53%
Test Loss: 1.82845, Test Accuracy: 54.64%
Model Parameters: 110.38 K, FLOPS: 126.763 M, MACS: 180.224 K

Sequence Length 30



Last Best Epoch: 14, Train Time ≈ 0 hr, 0 min, 8 sec
Train Loss: 0.84238, Train Accuracy: 77.12%
Test Loss: 1.93905, Test Accuracy: 52.75%
Model Parameters: 110.38 K, FLOPS: 189.964 M, MACS: 180.224 K

Conclusions

In terms of test/validation accuracy and training time, all models performed very similarly (within percent error). The GRU model performed slightly better, but a sample size of 3 isn't enough to make it definitively better than the rest.

In terms of model size complexity, the RNN model had the least number of parameters with around 44.332 thousand parameters. The GRU model had the 2nd least number of parameters with 110.38 thousand parameters. The LSTM model had the most number of parameters with 143.404 thousand parameters.

In terms of model computational complexity, the LSTM model has the biggest time complexity with an average of 168.9520333 million floating point operations per second. The GRU model had the 2nd highest computational complexity with 126.7629 million floating point operations per second. The RNN had the lowest computational complexity of 42.3854 million floating point operations per second.

Adjusting the sequence length had no observable impact (within percent error) on training the model. The only thing of note is that the number of floating point operations per second increased linearly with the sequence length

Problem 2

Build the model for.LSTM and rnn.GRU for the tiny Shakespeare dataset, the data loader code is already provided.

1. Train the models for the sequence of 20 and 30, report and compare training loss, validation accuracy, execution time for training, and computational and mode size complexities across the two models.
2. Adjust the hyperparameters (fully connected network, number of hidden layers, and the number of hidden states) and compare your results (training and validation loss, computation complexity, model size, training and inference time, and the output sequence). Analyze their influence on accuracy, running time, and computational perplexity.
3. What if we increase the sequence length to 50. Perform the training and report the accuracy and model complexity results.

Standard Vs. Adjusted Hyperparameters

My standard model consists of an embedding layer followed by an RNN layer that connects to a single dense output layer.

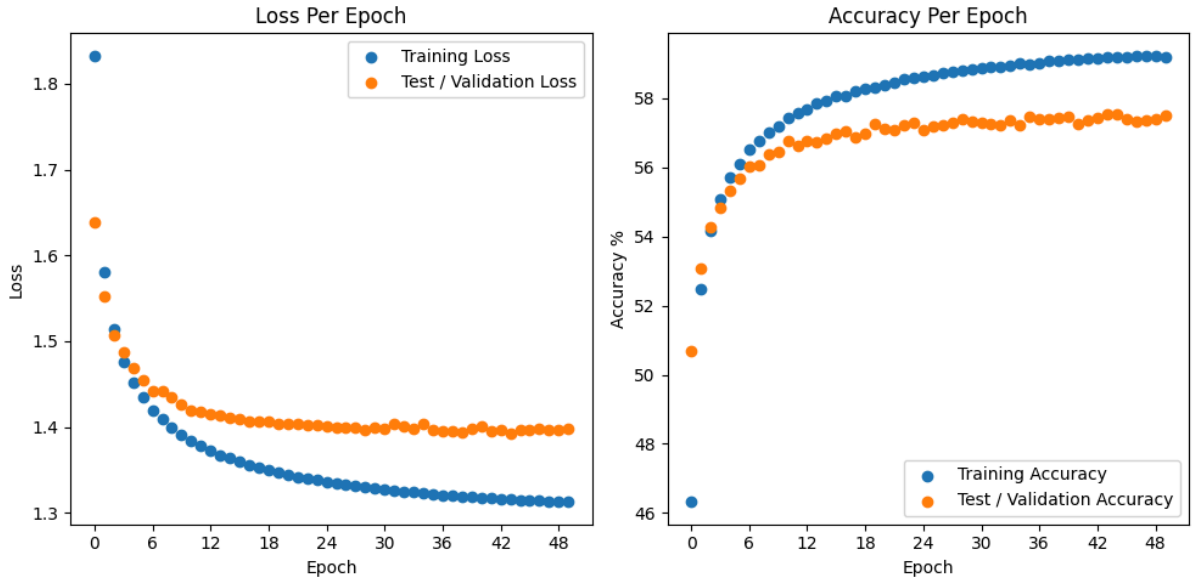
The Standard RNN layer has no dropout and is only a single layer, the hidden state size is 128 and the input is also 128.

For the adjusted hyperparameters implementation, I added an additional layer to the dense layer. Instead of the output dense layer being a single 128 to number of characters layer, it now is a 128 to 128 layer followed by batch norm then relu then a dropout block with a probability of 50% then into a 128 to number of characters layer.

nn.LSTM

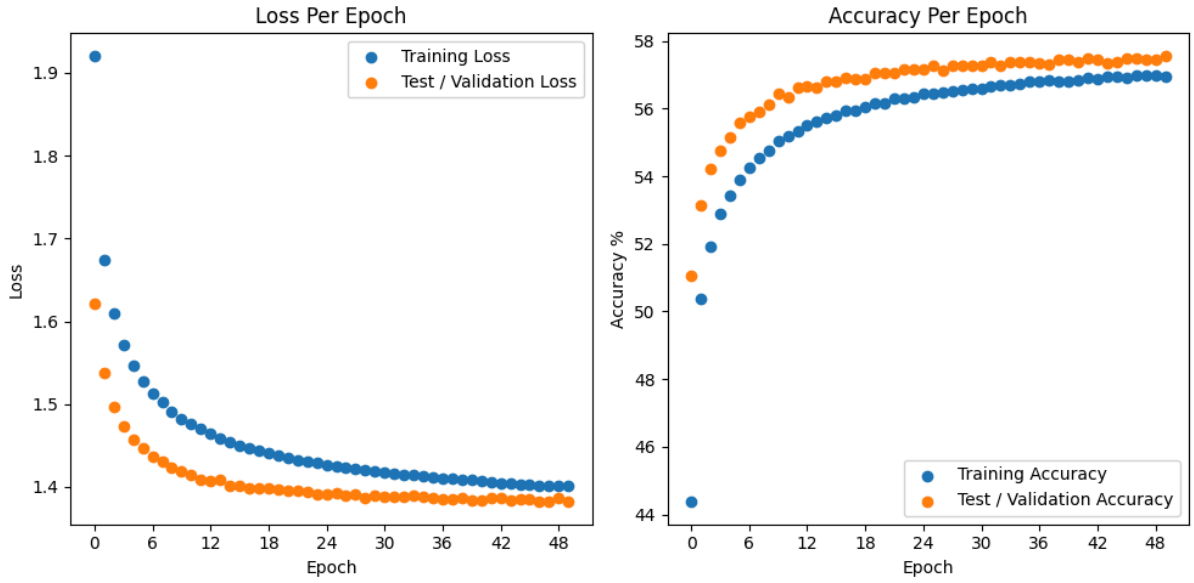
Sequence Length 20

Standard Hyperparameters



Last Best Epoch: 43, Train Time ≈ 0 hr, 18 min, 24 sec, Inference Time ≈ 1.993656 ms
Train Loss: 1.31522, Train Accuracy: 59.18%
Test Loss: 1.39276, Test Accuracy: 57.54%
Model Parameters: 148.801 K, FLOPS: 676.495 M, MACS: 1.065 M

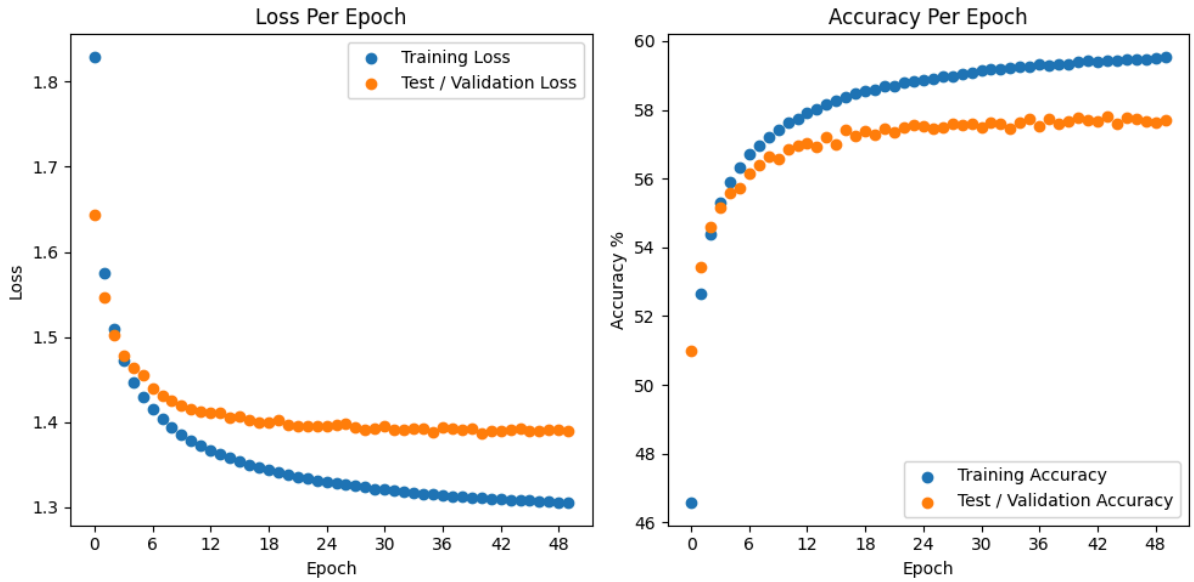
Adjusted Hyperparameters



Last Best Epoch: 49, Train Time ≈ 0 hr, 27 min, 25 sec, Inference Time ≈ 1.994848 ms
Train Loss: 1.40079, Train Accuracy: 56.96%
Test Loss: 1.38274, Test Accuracy: 57.56%
Model Parameters: 165.569 K, FLOPS: 687.112 M, MACS: 6.3242 M

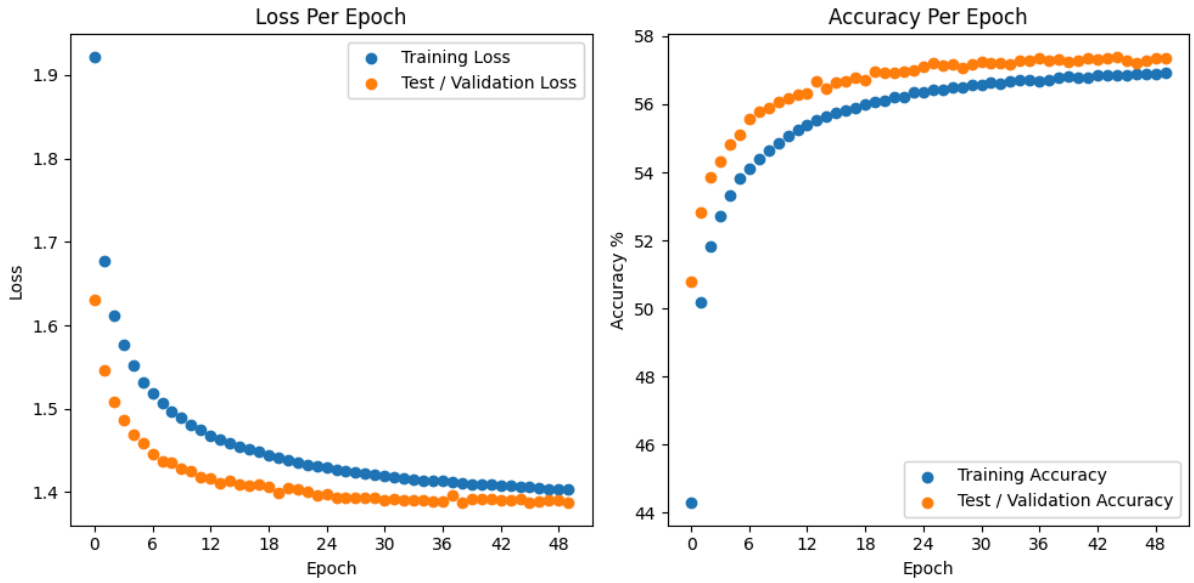
Sequence Length 30

Standard Hyperparameters



Last Best Epoch: 43, Train Time ≈ 0 hr, 21 min, 13 sec, Inference Time ≈ 0.997305
Train Loss: 1.30881, Train Accuracy: 59.44%
Test Loss: 1.39091, Test Accuracy: 57.81%
Model Parameters: 148.801 K, FLOPS: 1.0137 G, MACS: 1.065 M

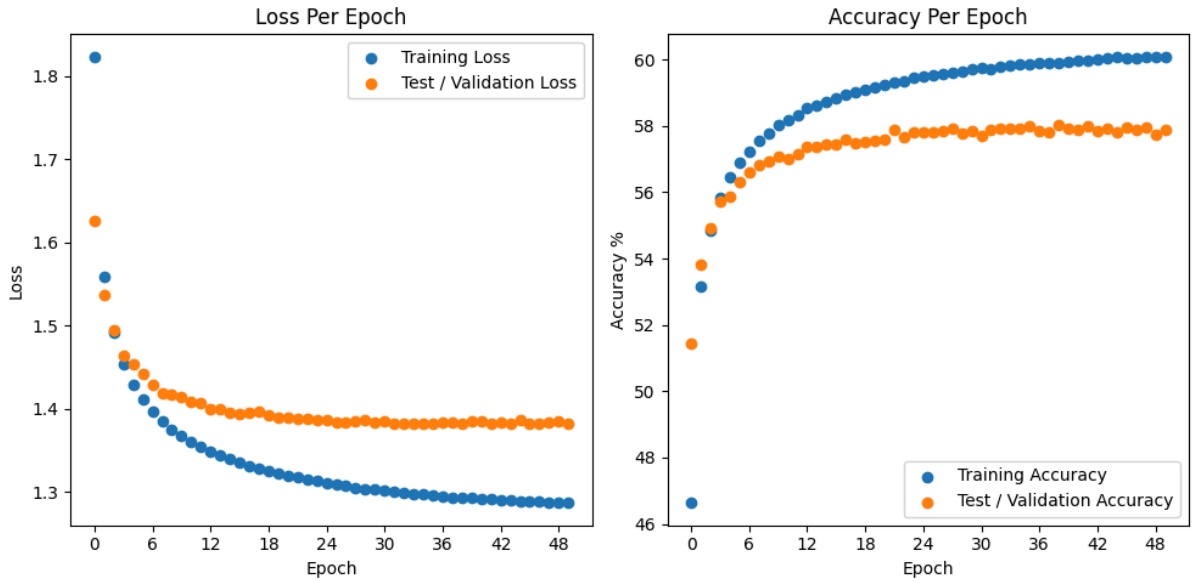
Adjusted Hyperparameters



Last Best Epoch: 44, Train Time ≈ 0 hr, 33 min, 17 sec, Inference Time ≈ 2.991199 ms
Train Loss: 1.40659, Train Accuracy: 56.85%
Test Loss: 1.39250, Test Accuracy: 57.40%
Model Parameters: 165.569 K, FLOPS: 1.0243 G, MACS: 6.3242 M

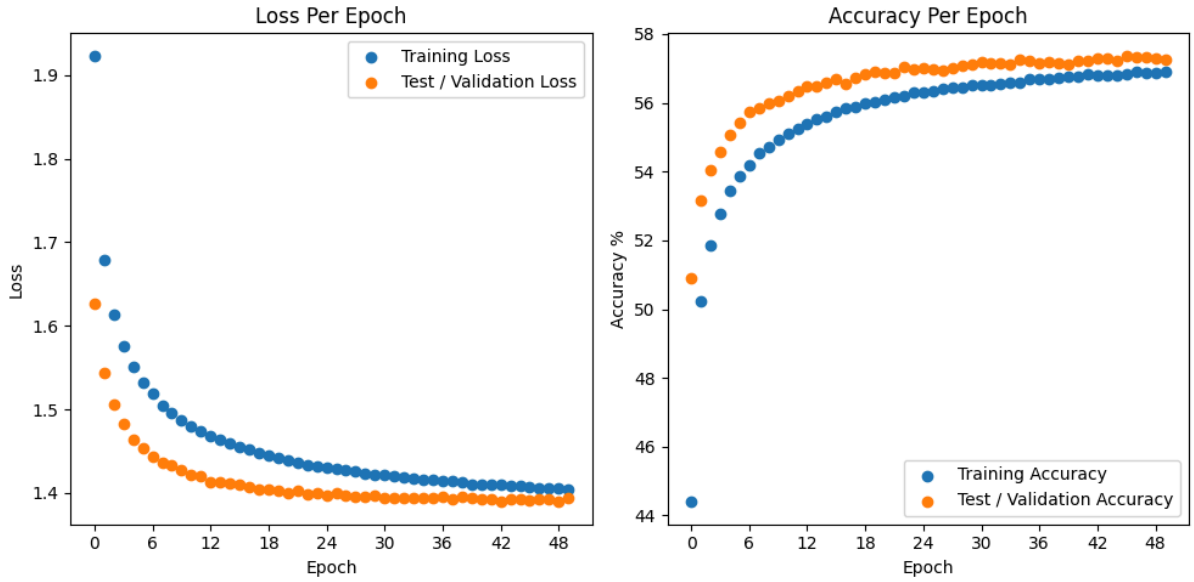
Sequence Length 50

Standard Hyperparameters



Last Best Epoch: 38, Train Time ≈ 0 hr, 22 min, 50 sec, Inference Time ≈ 1.994371 ms
Train Loss: 1.29351, Train Accuracy: 59.90%
Test Loss: 1.38267, Test Accuracy: 58.03%
Model Parameters: 148.801 K, FLOPS: 1.688 G, MACS: 1.065 M

Adjusted Hyperparameters

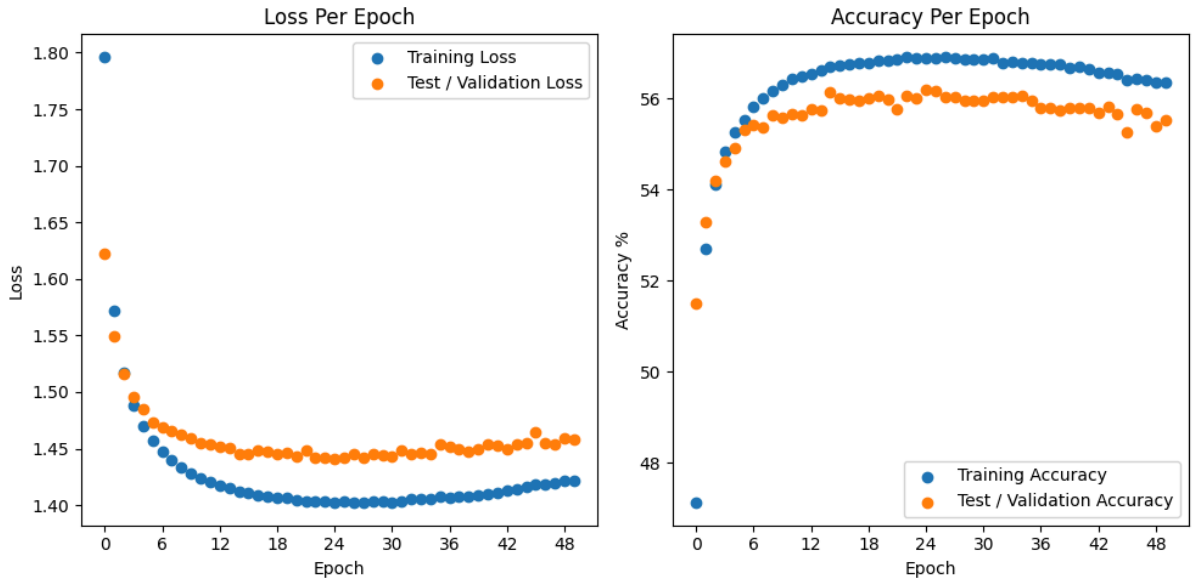


Last Best Epoch: 45, Train Time ≈ 0 hr, 32 min, 46 sec, Inference Time ≈ 2.992630 ms
Train Loss: 1.40666, Train Accuracy: 56.84%
Test Loss: 1.39049, Test Accuracy: 57.37%
Model Parameters: 165.569 K, FLOPS: 1.6987 G, MACS: 6.3242 M

nn.GRU

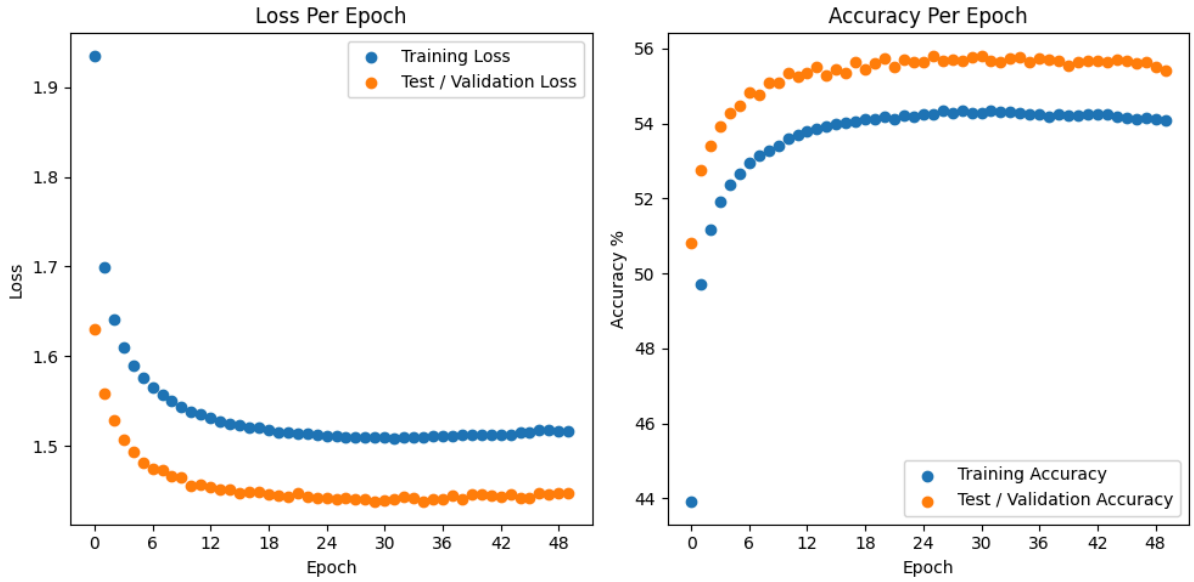
Standard Vs. Adjusted Hyperparameters
Sequence Length 20

Standard Hyperparameters



Last Best Epoch: 24, Train Time ≈ 0 hr, 21 min, 37 sec, Inference Time ≈ 1.994371 ms
Train Loss: 1.40207, Train Accuracy: 56.90%
Test Loss: 1.44055, Test Accuracy: 56.21%
Model Parameters: 115.777 K, FLOPS: 507.74 M, MACS: 1.065 M

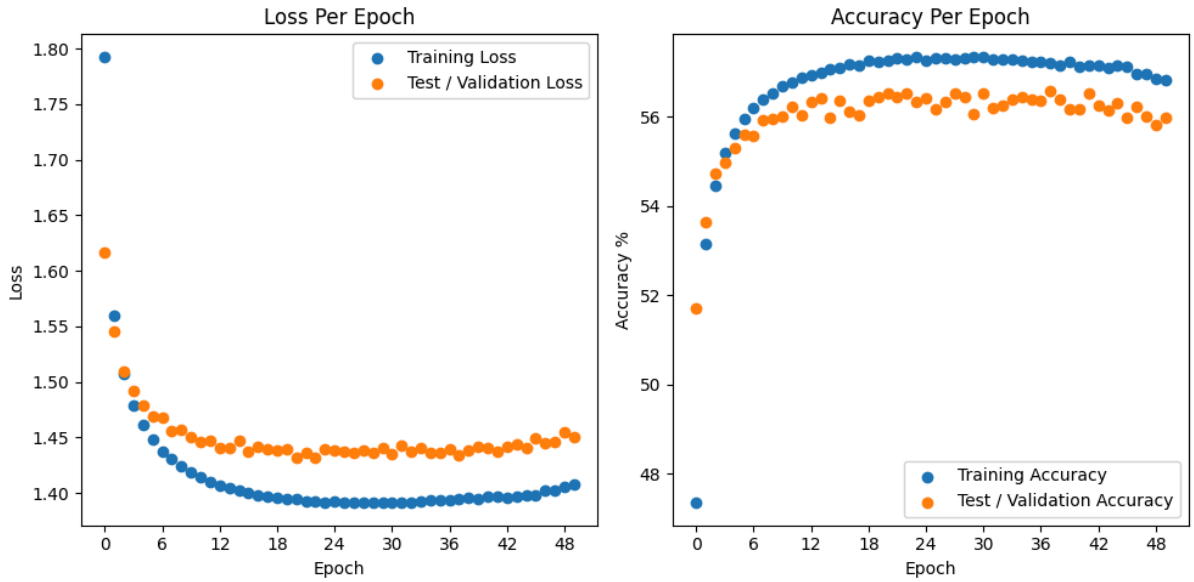
Adjusted Hyperparameters



Last Best Epoch: 30, Train Time ≈ 0 hr, 26 min, 30 sec, Inference Time ≈ 1.995325 ms
Train Loss: 1.50945, Train Accuracy: 54.30%
Test Loss: 1.43968, Test Accuracy: 55.81%
Model Parameters: 132.545 K, FLOPS: 518.357 M, MACS: 6.3242 M

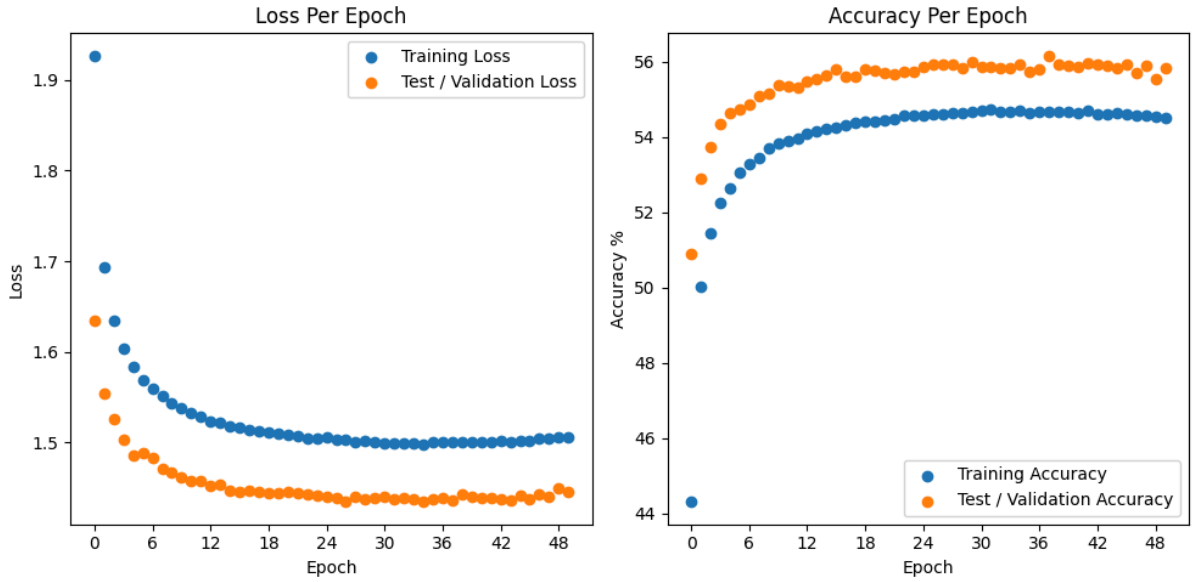
Sequence Length 30

Standard Hyperparameters



Last Best Epoch: 37, Train Time ≈ 0 hr, 21 min, 19 sec, Inference Time ≈ 0.997543 ms
Train Loss: 1.39453, Train Accuracy: 57.20%
Test Loss: 1.43341, Test Accuracy: 56.57%
Model Parameters: 115.777 K, FLOPS: 760.545 M, MACS: 1.065 M

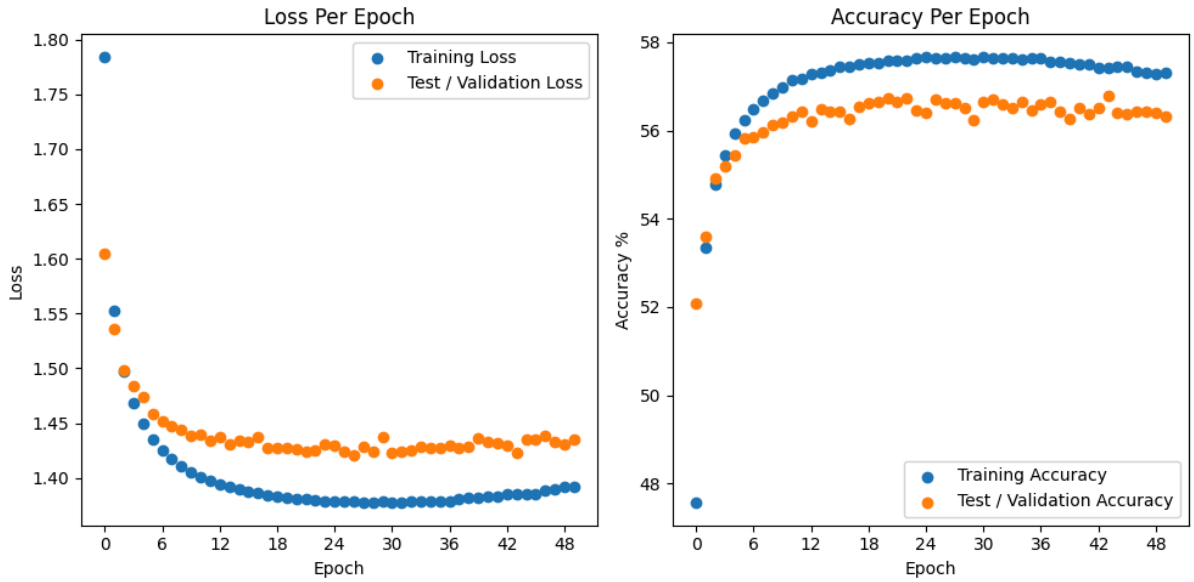
Adjusted Hyperparameters



Last Best Epoch: 37, Train Time ≈ 0 hr, 31 min, 40 sec, Inference Time ≈ 1.994848 ms
Train Loss: 1.50049, Train Accuracy: 54.66%
Test Loss: 1.43565, Test Accuracy: 56.17%
Model Parameters: 132.545 K, FLOPS: 771.162 M, MACS: 6.3242 M

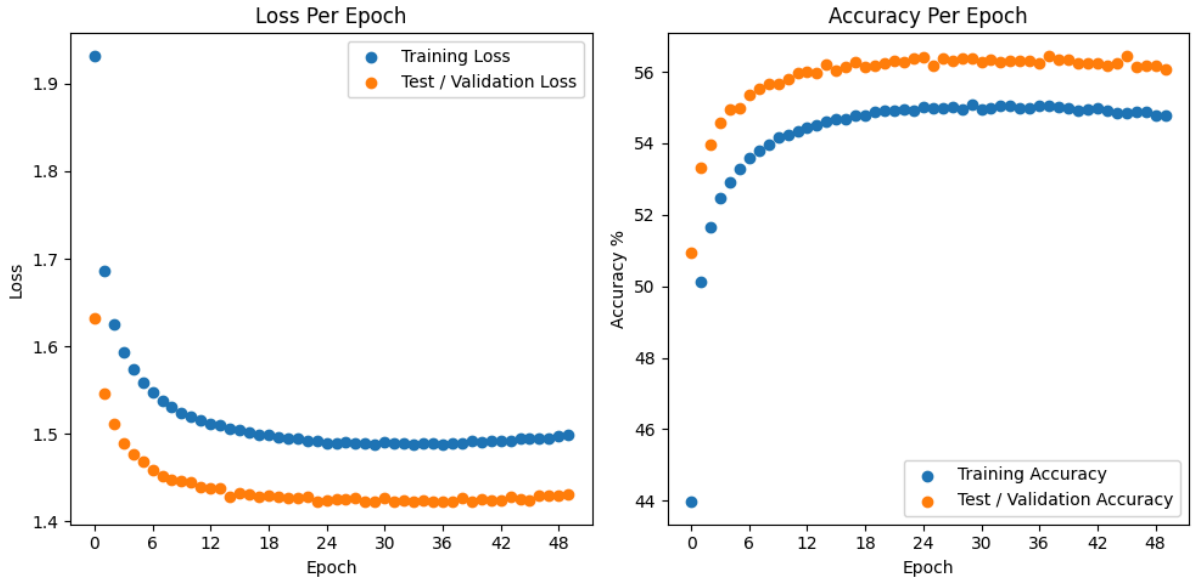
Sequence Length 50

Standard Hyperparameters



Last Best Epoch: 43, Train Time ≈ 0 hr, 22 min, 19 sec, Inference Time ≈ 1.996517 ms
Train Loss: 1.38558, Train Accuracy: 57.42%
Test Loss: 1.42354, Test Accuracy: 56.77%
Model Parameters: 115.777 K, FLOPS: 1.2662 G, MACS: 1.065 M

Adjusted Hyperparameters



Last Best Epoch: 45, Train Time ≈ 0 hr, 28 min, 35 sec, Inference Time ≈ 1.994133 ms
Train Loss: 1.49493, Train Accuracy: 54.86%
Test Loss: 1.42369, Test Accuracy: 56.45%
Model Parameters: 132.545 K, FLOPS: 1.2768 G, MACS: 6.3242 M

Conclusions

In terms of test/validation accuracy and training time, both models performed very similarly (within percent error). The GRU model performed slightly better, but a sample size of 3 isn't enough to make it definitively better than the rest.

In terms of model size complexity, the GRU model had the least number of parameters with an average of 124.161 thousand parameters across the 2 implementations. The LSTM model had the most number of parameters with an average of 157.1985 thousand parameters across the 2 implementations.

In terms of model computational complexity, the LSTM model has the biggest time complexity with an average of 1.0179214 billion floating point operations per second. The GRU model had the 2nd highest computational complexity with 764.8008 million floating point operations per second.

Adjusting the sequence length had no observable impact (within percent error) on training the model. The only thing of note is that the number of floating point operations per second increased linearly with the sequence length.

Inference time remained the same across all cases with some outlier exceptions that are most likely due to the quirks of Python being an interpreted language.

The adjusted hyperparameters implementation had a noticeable impact on training time in that it was increased by around 25 - 50%. In return, overfitting was completely removed with seemingly no impact on accuracy. It also had an increase in model complexity (both size wise and computationally).