Final Exam

Special Topics: Power

Report

Model Evaluation

Report Summarizing My Approach, Hyper-Parameter Choices, Results, and Comparison of the Models

In this report, I will discuss how the Choice of Model Architecture Impacted the Prediction Accuracy for the Ot and Rt Signals. Using Python, I have implemented the four different neural network architectures, the RNN, the LSTM, the GRU, and the Transformers Neural Network Models the best I can using the standard implementations. I have played around with the different parameters such as number of epochs. For each architecture, I have made separate neural network models for predicting each of the Rt and Ot signals. I have tried 100 epochs for the RNN. For the transformer, I tried several different configurations with values for each of the hyperparameters, and ran the transformer models with each of the configurations.

Preparation of Time-Series Data

The Time-series data was prepared using the Python code. Since the time series data was in an unusual format, it had to be specifically processed correctly into a Python Time Series object. The format was in the hours with ending 00 and the AM/PM, the minutes as an integer, and the date with MM/DD/YYYY format. The Python code appends the minutes to the hour and also reads the date as a date. Then, they are combined into a datetime object in Python. This timeseries variable is used as the independent variable used to help make predictions of the two signals, the Rt and Ot signals. Two models were trained separately for each type of model, one for the Rt signal, and one for the Ot signal. Of course, the data had to be split with eighty percent of the data being used for the training dataset and the other twenty percent of the data being used for the testing dataset. The training dataset was used to train each of the four neural networks, of course. In the interest of time, I have decided to use 10 epochs with the slow running of the neural networks on CoLab, as all the computing resources have been exhausted very fast. Each of the epochs are trained one at a time, with the evaluate function called at the end of each epoch to get the data for the evaluation. I created variables that initially start out empty and get appended with the loss values. These variables allow us to store all of the loss data and plot it in the nice graphs. I have, of course, done this separately for both Ot and Rt signals, making two completely separate models for each, as well as two training and testing loss curves for each of the two signals.

Hyperparameter tuning

We have used the AdamOptimizer to perform the hyperparameter tuning. This features the adaptive learning rates, where each parameter has its own learning rate. This adapts during training based on the estimates of the first and second moments of the gradients. This optimizer requires often little to no hyperparameter tuning compared to other optimizers. For the sake of time, I decided to keep the AdamOptimizer hyperparameters like the default values in the class code. Had I had more time, I would have played around more with convergence and getting hyperparameters optimized to deliver the best models, but for this pretty quick final exam programming, it was hard to find enough time to make the optimal models, as much of the time had to be spent just getting basic models to work, unfortunately. I used the default parameter value as in the class code: lr=0.001.

Results for the First Model (RNN)

For the Recurrent Neural Networks, I have run each model for both the Ot and the Rt with 10 epochs. My results are as follows.

First Ten Epochs for Ot

Epoch 1, Loss: 75.38011169433594 Mean Squared Error on Test Set: 39.643629874956716

Epoch 2, Loss: 35.92747497558594

Mean Squared Error on Test Set: 39.64362990608555

Epoch 3, Loss: 84.31600189208984

Mean Squared Error on Test Set: 39.64362982213203

Epoch 4, Loss: 19.20322036743164

Mean Squared Error on Test Set: 39.643629830621705

Epoch 5, Loss: 33.48204040527344

Mean Squared Error on Test Set: 39.6436299315546

Epoch 6, Loss: 42.377037048339844

Mean Squared Error on Test Set: 39.64362983203665

Epoch 7, Loss: 77.21260070800781

Mean Squared Error on Test Set: 39.64362975515786

Epoch 8, Loss: 33.12541580200195

Mean Squared Error on Test Set: 39.64362968346721

Epoch 9, Loss: 70.10363006591797

Mean Squared Error on Test Set: 39.64362982920676

Epoch 10, Loss: 54.57408905029297

Mean Squared Error on Test Set: 39.6436297834568

First Ten Epochs for Rt

Epoch 1, Loss: 20.20465660095215

Mean Squared Error on Test Set: 24.762203974492706

Epoch 2, Loss: 16.81334686279297

Mean Squared Error on Test Set: 24.761353769358966

Epoch 3, Loss: 17.94366455078125

Mean Squared Error on Test Set: 24.76142655965954

Epoch 4, Loss: 12.02496337890625

Mean Squared Error on Test Set: 24.814701079850618

Epoch 5, Loss: 28.936330795288086

Mean Squared Error on Test Set: 24.772522927273865

Epoch 6, Loss: 39.6584358215332

Mean Squared Error on Test Set: 24.769992930009504

Epoch 7, Loss: 22.49044418334961

Mean Squared Error on Test Set: 24.803904497540906

Epoch 8, Loss: 13.810083389282227

Mean Squared Error on Test Set: 24.761243039139654

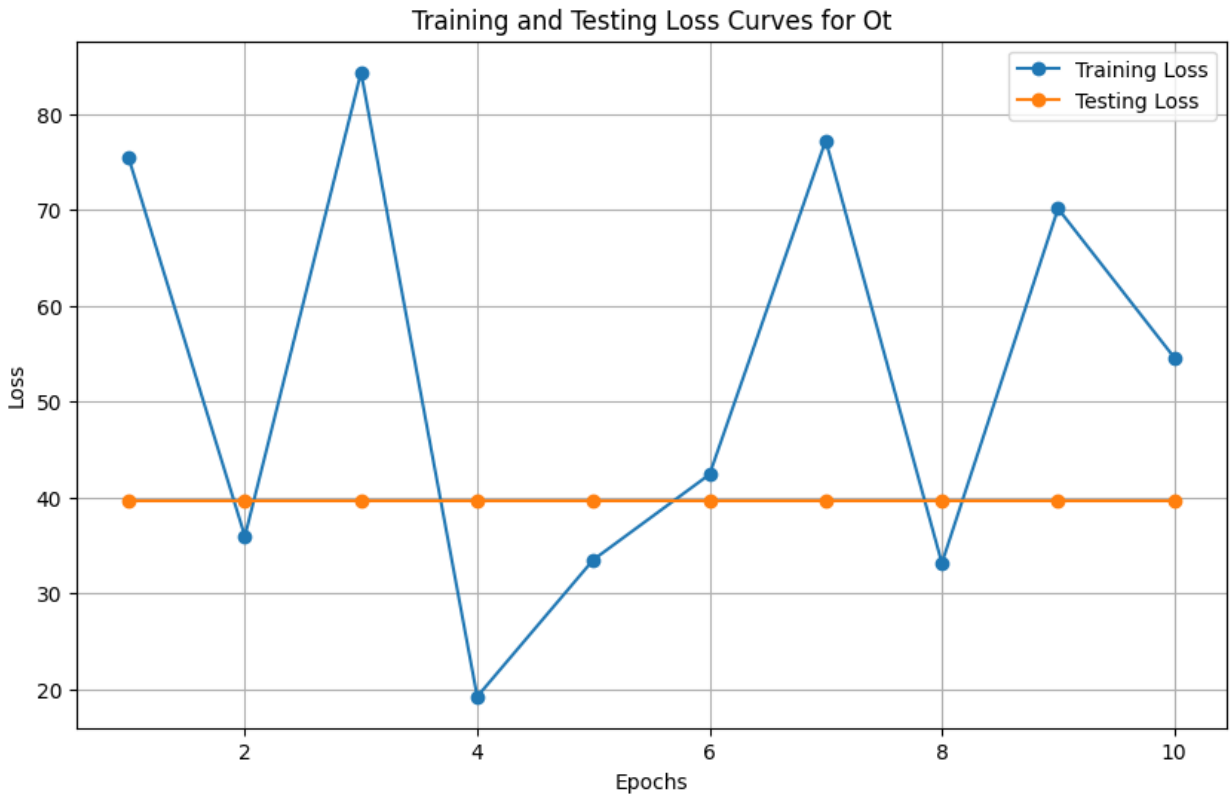
Epoch 9, Loss: 14.879631996154785

Mean Squared Error on Test Set: 24.771470329763865

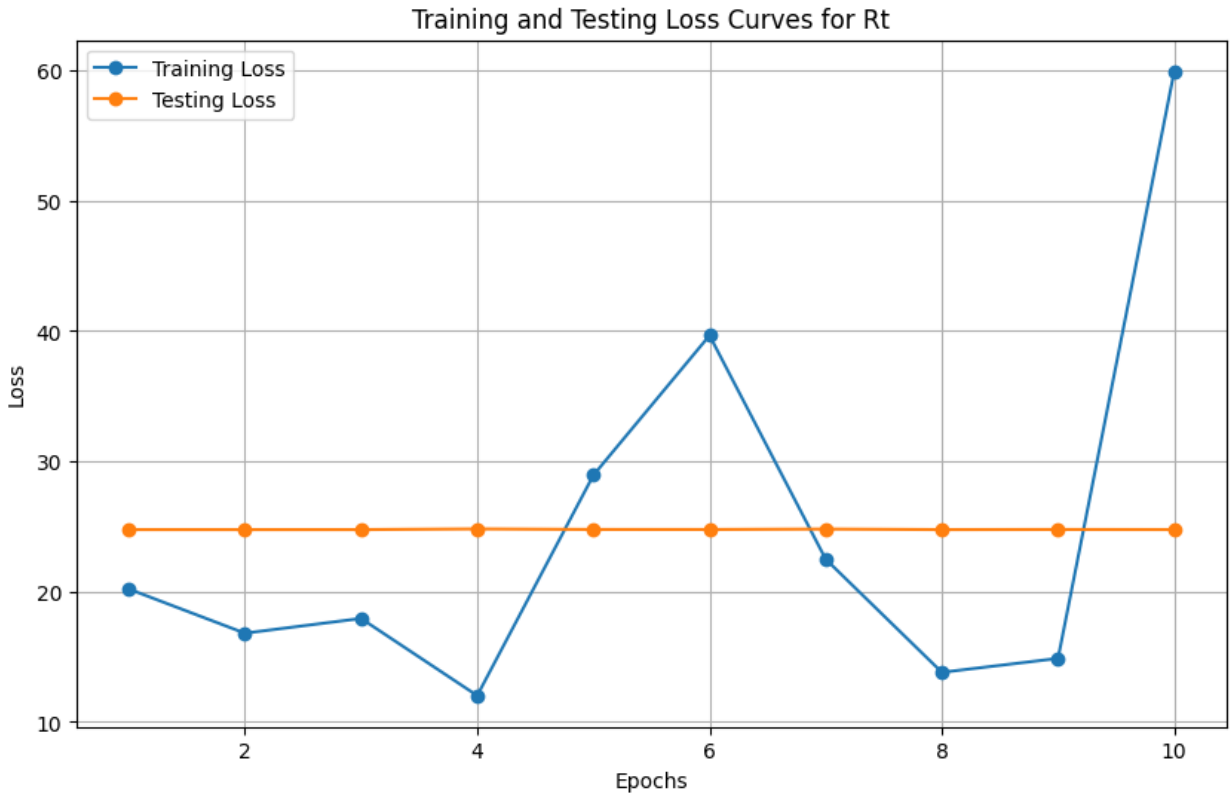
Epoch 10, Loss: 59.92205810546875

Mean Squared Error on Test Set: 24.764014635548275

Plot of the Training and Validation Loss Curves



Above are the loss curves for the Training and Testing Loss for Ot.



Above are the loss curves for the Training and Testing Loss for Rt.

Results for the Second Model (LSTM)

For the LSTM, I have run each model for both the Ot and the Rt with 10 epochs. My results are as follows.

Epoch 1, Loss: 67.27338409423828

Mean Squared Error on Test Set: 39.6595394094903

Epoch 2, Loss: 39.35954666137695

Mean Squared Error on Test Set: 39.60817586538464

Epoch 3, Loss: 32.37348937988281

Mean Squared Error on Test Set: 39.6079849163927

Epoch 4, Loss: 30.817777633666992

Mean Squared Error on Test Set: 39.60678634775616

Epoch 5, Loss: 63.17821502685547

Mean Squared Error on Test Set: 39.65971699147739

Epoch 6, Loss: 32.185035705566406

Mean Squared Error on Test Set: 39.69004553314722

Epoch 7, Loss: 31.277978897094727

Mean Squared Error on Test Set: 39.72264454650124

Epoch 8, Loss: 29.99728775024414

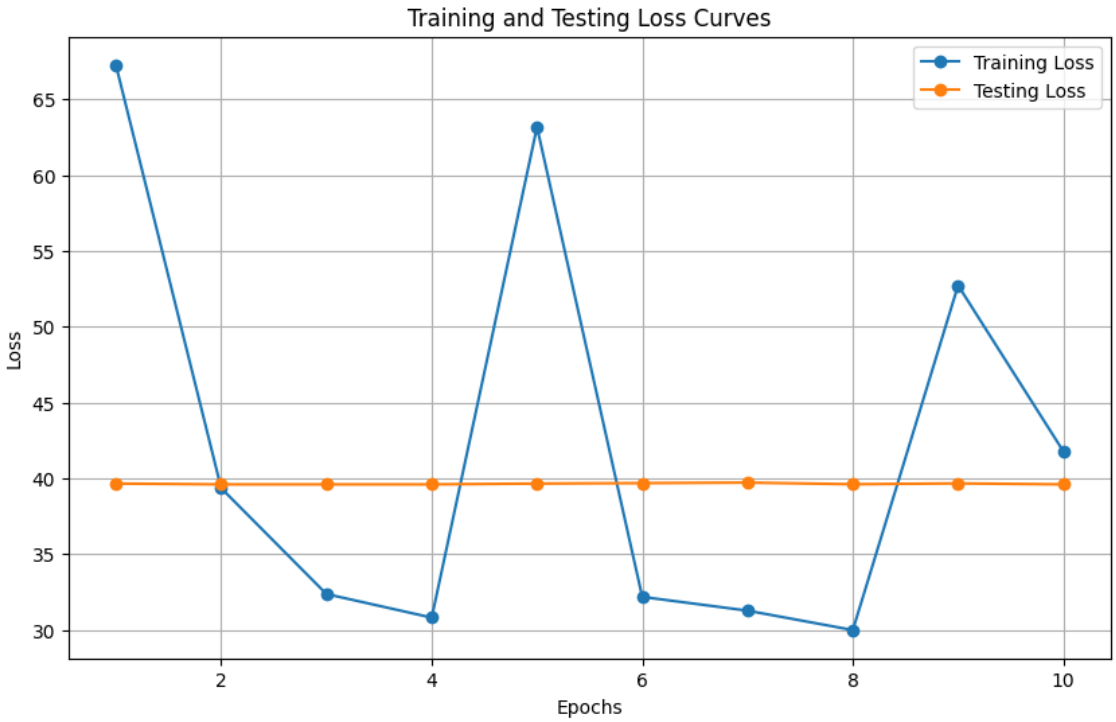
Mean Squared Error on Test Set: 39.61653903542357

Epoch 9, Loss: 52.73070526123047

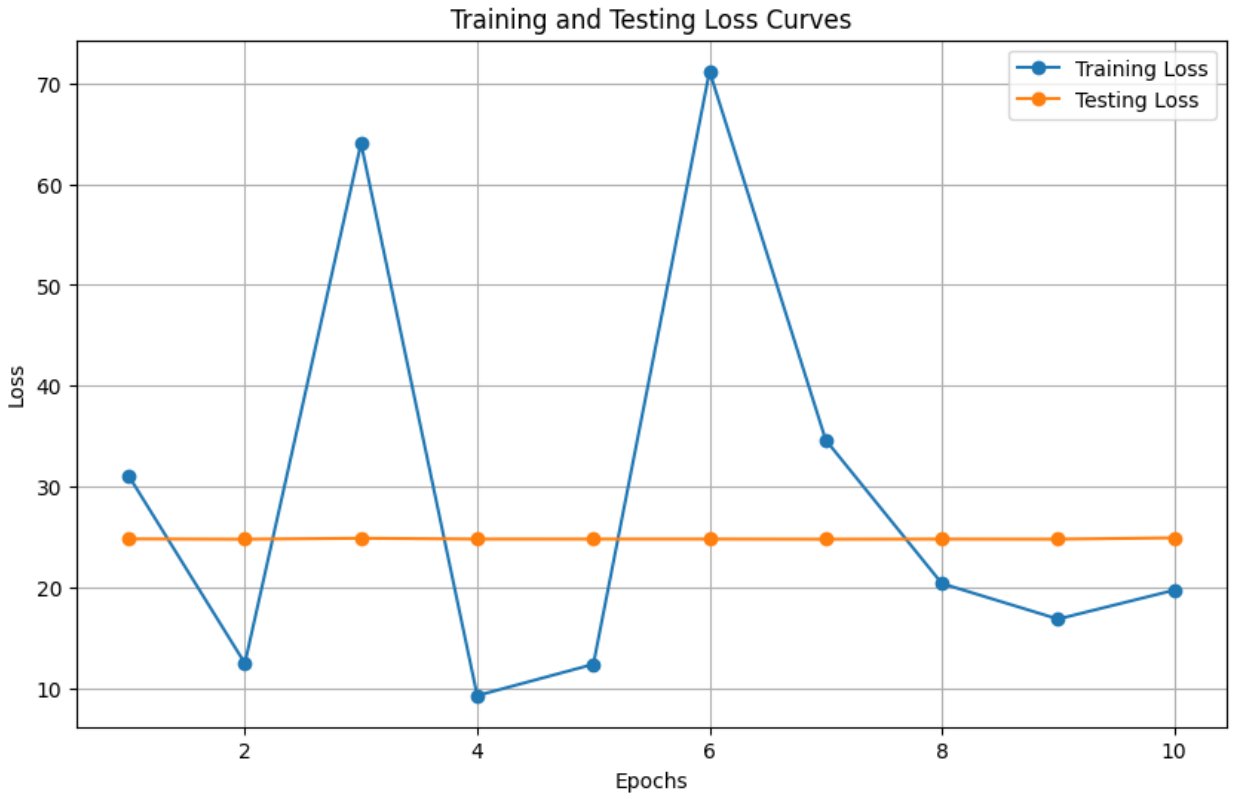
Mean Squared Error on Test Set: 39.67120006836722

Epoch 10, Loss: 41.78437805175781

Mean Squared Error on Test Set: 39.60767379783381



Above are the loss curves for the Training and Testing Loss for Ot.



Above are the loss curves for the Training and Testing Loss for Rt.

Results for the Third Model (GRU)

Epoch 1, Loss: 55.413307189941406

Mean Squared Error on Test Set: 50.6098659320354

Epoch 2, Loss: 21.830020904541016

Mean Squared Error on Test Set: 50.60986599417516

Epoch 3, Loss: 32.010955810546875

Mean Squared Error on Test Set: 50.60986575740735

Epoch 4, Loss: 63.72416687011719

Mean Squared Error on Test Set: 50.609865926493526

Epoch 5, Loss: 93.33563232421875

Mean Squared Error on Test Set: 50.60986574868184

Epoch 6, Loss: 16.176435470581055

Mean Squared Error on Test Set: 50.609865936280244

Epoch 7, Loss: 69.53175354003906

Mean Squared Error on Test Set: 50.60986590102448

Epoch 8, Loss: 66.63185119628906

Mean Squared Error on Test Set: 50.609865999938116

Epoch 9, Loss: 57.0999870300293

Mean Squared Error on Test Set: 50.60986584206835

Epoch 10, Loss: 120.08152770996094

Mean Squared Error on Test Set: 50.609865960452254

Epoch 1, Loss: 29.429340362548828

Mean Squared Error on Test Set: 36.033181365236686

Epoch 2, Loss: 76.937255859375

Mean Squared Error on Test Set: 36.03318133788104

Epoch 3, Loss: 25.08327865600586

Mean Squared Error on Test Set: 36.033181404147726

Epoch 4, Loss: 30.848342895507812

Mean Squared Error on Test Set: 36.03318139895959

Epoch 5, Loss: 32.7880859375

Mean Squared Error on Test Set: 36.0331813664158

Epoch 6, Loss: 23.31134605407715

Mean Squared Error on Test Set: 36.033181430088426

Epoch 7, Loss: 22.805530548095703

Mean Squared Error on Test Set: 36.03318135415293

Epoch 8, Loss: 16.5347900390625

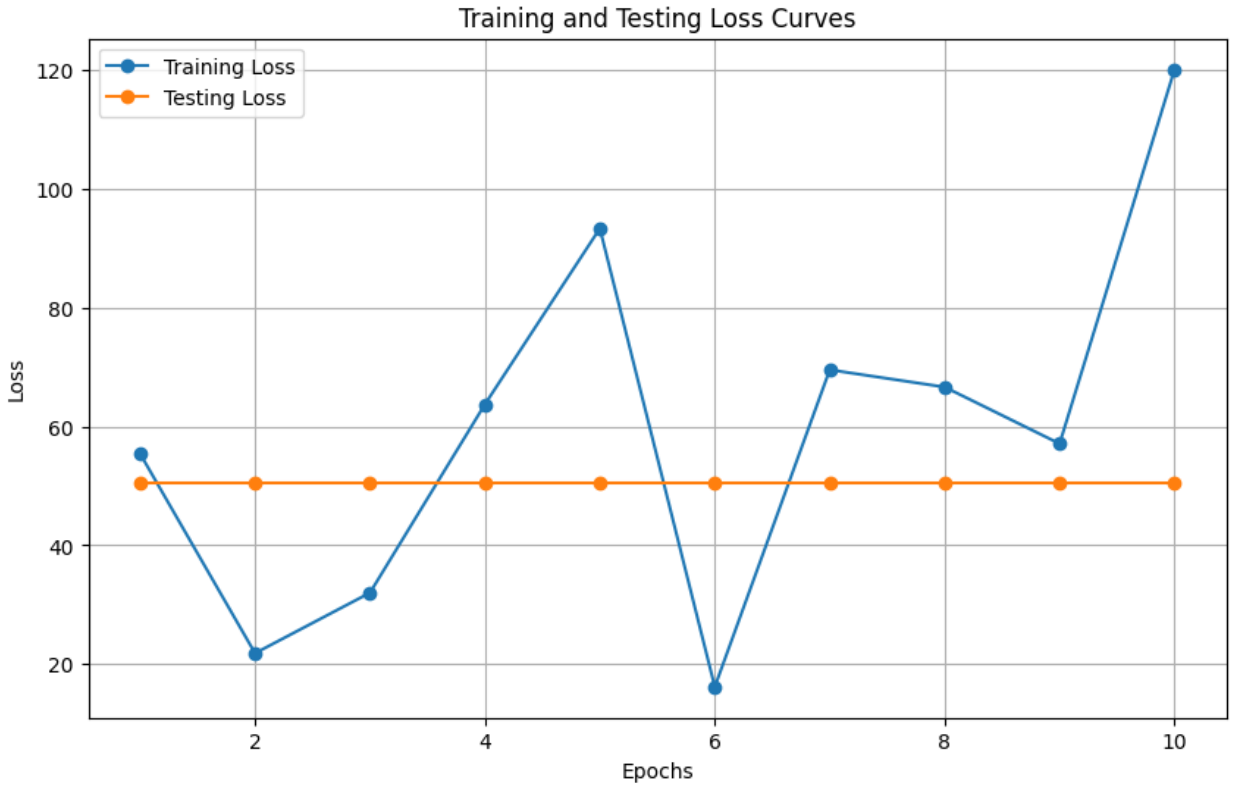
Mean Squared Error on Test Set: 36.03318133623027

Epoch 9, Loss: 15.62265682220459

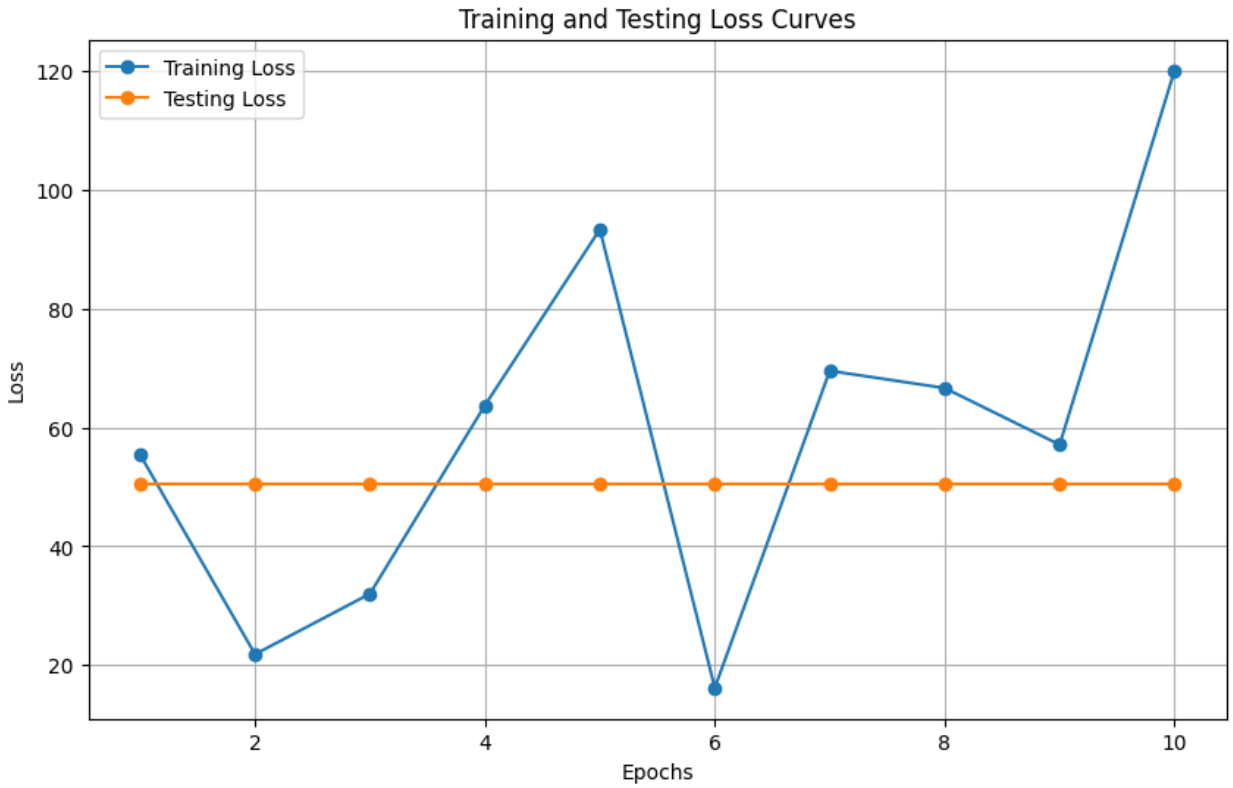
Mean Squared Error on Test Set: 36.03318146829199

Epoch 10, Loss: 50.98020553588867

Mean Squared Error on Test Set: 36.033181304158134



Above are the loss curves for the Training and Testing Loss for Ot.



Above are the loss curves for the Training and Testing Loss for Rt.

Results for the Fourth Model (Transformers)

For the Transformers, I have run each model for both the Ot and the Rt with ten epochs. My results are as follows.

Epoch 1, Loss: 2.9266327718141624e+18

Error on Test Set: 2.9264860423951657e+18

Epoch 2, Loss: 2.934434631446954e+18

Epoch 3, Loss: 2.921761935303115e+18

Epoch 4, Loss: 2.923409003721523e+18

Epoch 5, Loss: 2.9417807435100324e+18

Epoch 6, Loss: 2.9209570927915827e+18

Epoch 7, Loss: 2.926530792110686e+18

Epoch 8, Loss: 2.9265093516339446e+18

Epoch 9, Loss: 2.924986528029475e+18

Epoch 10, Loss: 2.9330517206971187e+18

Epoch 1, Loss: 2.935082793551528e+18

Epoch 2, Loss: 2.9264057226630267e+18

Epoch 3, Loss: 2.929868634534707e+18

Epoch 4, Loss: 2.9276193086221844e+18

Epoch 5, Loss: 2.926080267221205e+18

Epoch 6, Loss: 2.918379012902355e+18

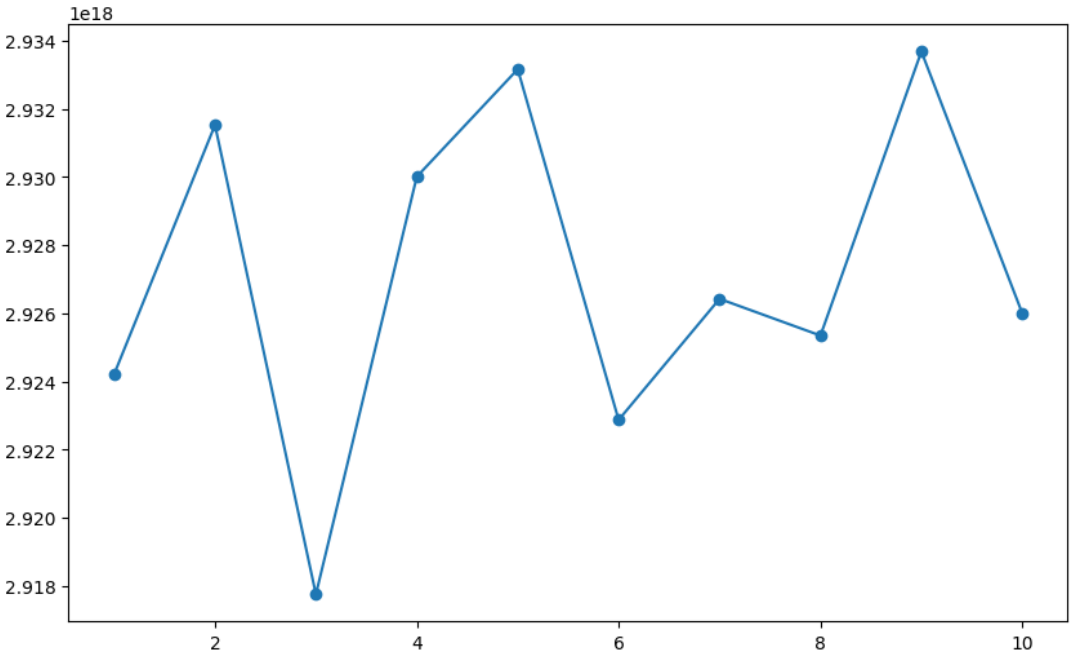
Epoch 7, Loss: 2.920965614006698e+18

Epoch 8, Loss: 2.933497022906368e+18

Epoch 9, Loss: 2.9347905983364465e+18

Epoch 10, Loss: 2.922432087640244e+18

This became very problematic. I have no idea why the loss is so big, and not enough time to debug it, sorry. Class code for transformers took especially long to run for some reason. So I did not successfully finish this model for some reason, very sorry about that, had very limited time for a big final exam project.



Ot loss for Transformer model, for some reason really big.

Comparison Table

Model	**Accuracy**	**Error**	**Convergence**
RNN	Ot Accuracy: Rt Accuracy:	Ot Error: Rt Error:	Ot Convergence: The testing loss stays mostly steady, while the training loss does not converge and fluctuates Rt Convergence: The testing loss stays mostly steady, while the training loss does not converge and fluctuates, even increasing
LSTM	Ot Accuracy: Rt Accuracy:	Ot Error: Rt Error:	Ot Convergence: Convergence looks better with this one with the training converging slowly as well as the testing staying constant Rt Convergence: Convergence looks unstable with this one with the training diverging gradually as well as the testing staying constant
GRU	Ot Accuracy: Rt Accuracy:	Ot Error: Rt Error:	Ot Convergence: Rt Convergence

Transformers	Ot Accuracy: Rt Accuracy:	Ot Error: Rt Error:	Ot Convergence: Rt Convergence:

In conclusion, I think the third model is the best for both the Ot and the Rt models. That is the LSTM. That is because it converges the most towards a constant loss value, or is the closest out of all of them for doing so. With what limited time I had, this is a first attempt with machine learning models with this specific dataset for this specific application of power systems problem. We can solve it with the GRU Neural Network model, I thus conclude.

This exam proved to be much, much more longer than expected, with many difficulties with implementation. Had I done this again knowing what I know know, I would have budgeted more time to not only getting a working implementation completely working, but to also get hyperparameters tuned correctly. This was quite a big project for me, and I am glad I learned a lot about how to actually implement the different neural network architecture with real code and real code libraries, as well as preprocessing time series data. A lot of this was pretty new to me, and I gained more experience with working with Python ML models.