Exploring Neural Network Architectures in Air Passenger Time Series Forecasting

1 Introduction

Neural networks are within the domain of machine learning and can work quite effectively in predicting future trends within time series forecasting. The International Business Machines Corporation, (IBM), states that there are Artificial or Simulated Neural Networks (ANN's vs. SNN's). A broad definition of ANN's include noting that they are comprised of a node layer which contains an input layer, additional hidden layers, and an output layer. A specified threshold must be stated which acts as the barrier between if data will be sent to the next individual node or not (IBM).

The international airports dataset includes a monthly index starting in 1949 and ending in 1960. The distribution of airline passengers across time can be plotted as follows:

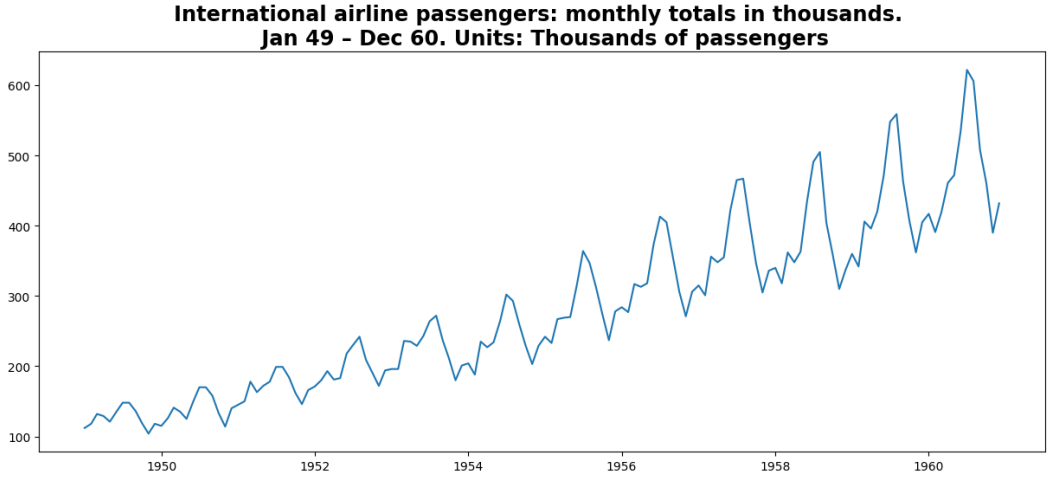

Figure 1. Distribution of Airline Passengers in Thousands

It is important to test various statistical models or neural network methodologies in order to see which has the most accurate representation of airline passengers across time. The five models that we will report on include ARIMA (AutoRegressive Integrated Moving Average), LSTM (Long Short-Term Memory), LSTM with Attention Mechanism, CNN (Convolutional Neural Network), and Transformer. A brief definition for each will be provided within the methods section. In addition to defining the neural networks or statistical models used, it's important to analyze various prediction accuracy measures. The measures we will use within this project include a non-rolling and rolling prediction.

Finally, after computing the various accuracy measures of the five models, we will redefine intricate details of the models that may improve results. Some of these re-configurations include regularization adjustments, experimentation of various dropout rates, normalization layer adjustments, layer parameter modifications, and architecture alterations.

2 Methods

As discussed within the introduction, the first method that will be analyzed is ARIMA (AutoRegressive Integrated Moving Average). This is noted to have exemplary results on capturing linear trends within time series data (Zhuang). As noted within lecture notes, ARIMA is a simple linear equation for stationary time series data. If there is dependence among values, it's often beneficial to employ ARIMA. The noteworthy parameters within ARIMA include p, d, and q. p is the number of Auto-Regressive terms, d is the number of nonseasonal differences, and q is the moving average, or lagged forecast errors (Zhuang). Within the results and discussion section, we will discuss what values were decided on for these parameters.

A type of neural network noted is Long Short-Term Memory (LTSM). This is a recurrent neural network that has been noted to have exemplary results on explaining long-term dependencies within time series data trends. This method works best on nonlinear time series sequences. Another article discussing this method states that LTSM's have feedback connections compared to standard feed-forward neural networks. It is noted that LTSM's were created to combat the issue of vanishing gradients that are encountered when training other RNN's (*Introduction*).

An extension on LSTM's are called LSTM with Attention Mechanisms. This attention mechanism extension allows for important predictors to have a higher weight within the final prediction model.

An additional type of neural network is known as Convolution Neural Networks (CNN). These are primarily used within image processing, but can also be used in capturing local patterns within sequential data (Zhuang). According to IBM, there are three layers within convolution neural networks. These include a convolution, pooling, and Fully-connected layer. It's stated that with each additional layer, the more complex the network becomes. Various hyperparameters must be set before training of the model begins. These include the number of filters, stride, and zero-padding (*What are convolutional*).

The final neural network to mention is called Transformer. According to a developing website called "Turing", Transformer neural networks learn the context of data through sequential data analysis. The general architecture of such a network includes an encoder-decoder representation. This is found within recurrent neural networks but is slightly different in the sense that it does not perform data processing in sequential order like RNN's do. N layers exist within the encoder, each of which have two sub-layers. The decoder has similarities to the encoder but has three sublayers for each layer. Transformation, linear algebra, and sequential processing help in achieving transformer neural netowrks (*Understanding*).

The two performance accuracy measures are known as "rolling" and "non-rolling" predictions. Rolling is said to be more realistic since it predicts each future step one at a time. The prediction is placed back into the input and used to predict the next observation. On the contrary, non-rolling predictions predict future steps all at once. The predictions are based upon historical data (Zhuang).

3 Results and Discussion

Baseline ARIMA vs. Modified ARIMA

We will begin analysis of the various methods mentioned above by employing baseline models presented within this assignment. The first model to look at is ARIMA. We can split the data into a testing and training set and calculate the root mean squared error (RSME). We find within the graphic below, that there is an RSME of 89.2228.

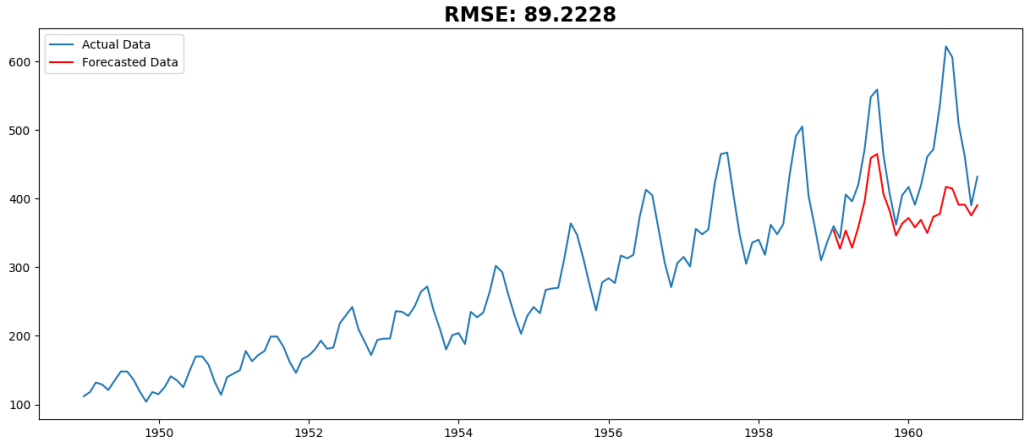

Figure 2. RSME using Baseline ARIMA

We can modify ARIMA in a few different ways. There are three parameters within ARIMA that can be modified. These include the number of auto-regressive terms (p), moving average terms (q), and differences (d). It is stated within Dr. Zhuang's notes that there are two methods we can use to decide on what to use for p and q. We can use an autocorrelation and partial autocorrelation function to understand these values. Based on these plots, an initial estimate for p and q would be 2 and 27, respectively. The RSME for this model is equal to 67.45. With some trial and error, however, we find that an even better RSME (equal to 42.2033), can be calculated if p, d, and q are equal to 0, 2, 25, respectively. This can be seen within the plot below:

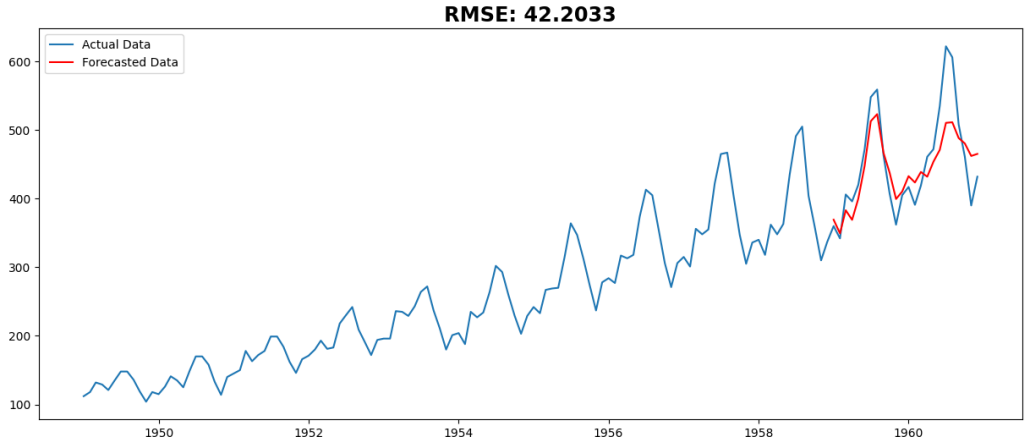

Figure 3. RSME using Modified ARIMA

Baseline LSTM vs. Modified LSTM

We move forward with the baseline LSTM model by first normalizing the input data and creating a sequential normalized dataset. We split the data into a testing and training set. LSTM uses the "Tensor Flow" package within Python. There are four interactive layers. The size of the hidden state is 200. The activation method specified is "relu". There is a dropout method of 20 percent, regularization, and the return sequence only returns the last output. We find that the non-rolling prediction RSME is equal to 24.7617 The rolling prediction of 39.3867 can be seen in the plot below:

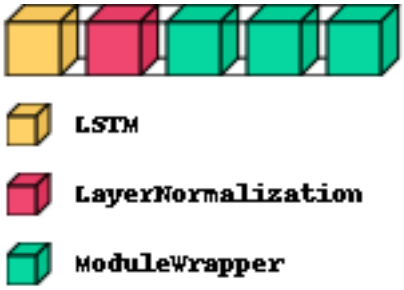

Figure 4. Baseline LSTM Architecture

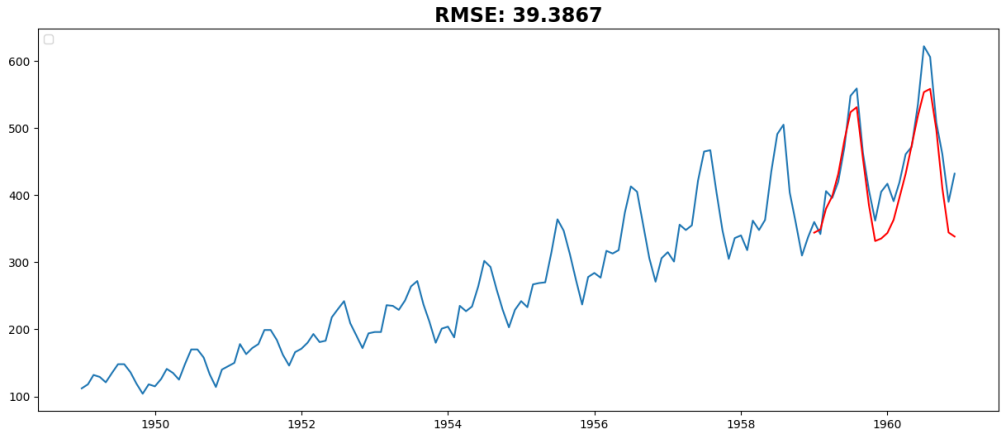

Figure 5. Rolling RSME using Baseline LSTM

Various modifications can be made within LSTM that may help accuracy results. Modifications began by altering the activation method to "sigmoid" or "tanh". We find, generally that "relu" performed the best and had the lowest RSME values. After conducting some research, it is also said that "relu" is simple to compute and efficient. Increasing the size of the hidden state helped RSME results. Normalization also increases the accuracy of the model. Regularization also helped results. Adding too high or low of a dropout parameter can negatively impact results. It's also interesting to note that adding an additional dense LSTM layer improved results. After trial and error, the best model that could be found is as follows: size of the hidden state = 700, dropout rate = 20 percent, two additional LSTM dense layers having units equal to 60 and 30, regularization and normalization performed, learning rate = .001, and an activation method of relu. The non-rolling RSME was 18.0774. The rolling prediction of 20.8812 can be seen within the plot below:

Figure 6. Modified LSTM Architecture

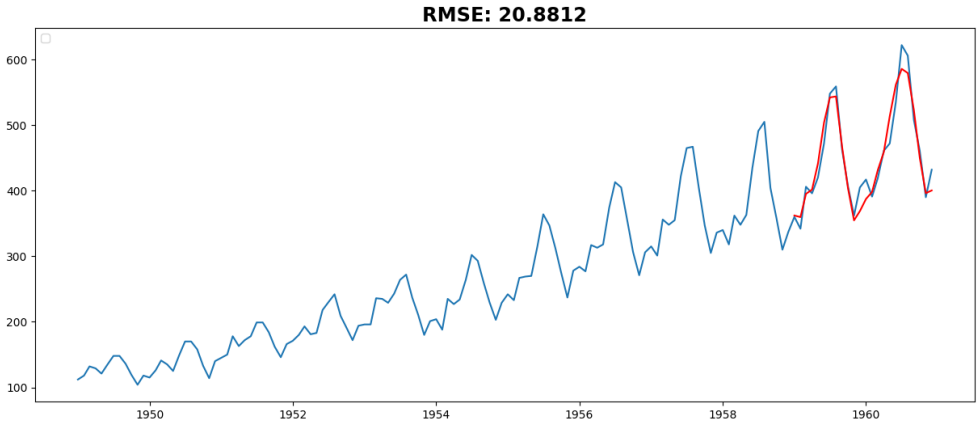

Figure 7. Rolling RSME using Modified LSTM

Baseline LSTM with Attention Mechanism vs. Modified LSTM with Attention Mechanism

An extension of LSTM includes an attention mechanism. Model building and training is quite similar to LSTM. The same parameters mentioned previously, are used within this model. The attention mechanism utilizes a connection from each timestamp, so that is why we must specify the return sequencies as true. When the attention mechanism is included within the baseline models, the RSME results are slightly better than when it is not used. The non-rolling RSME prediction is equal to 32.3013 and the rolling RSME prediction is equal to 58.7443, which can be seen in the plot below:

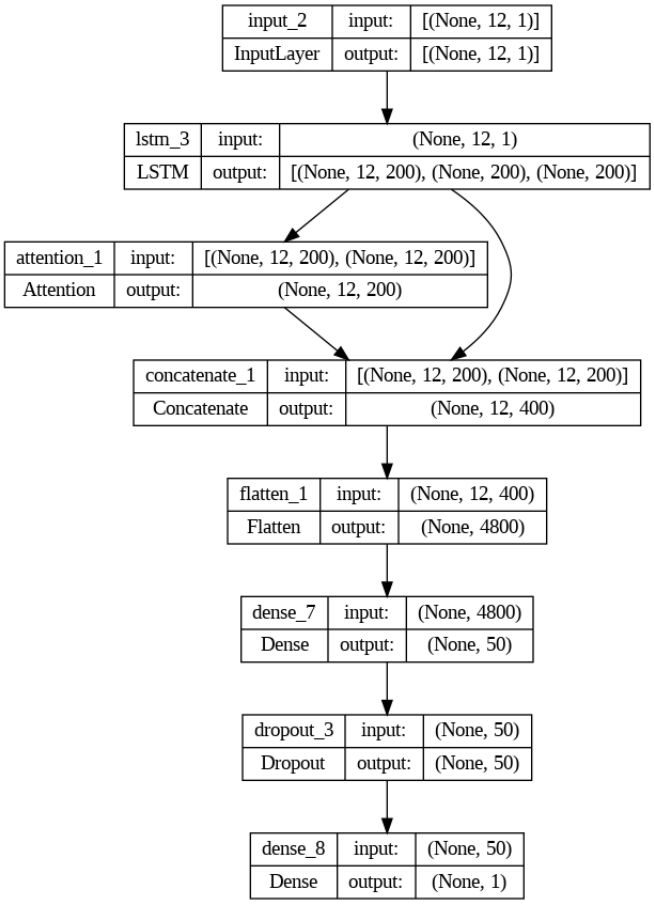

Figure 8. Baseline LSTM with Attention Mechanism Architecture

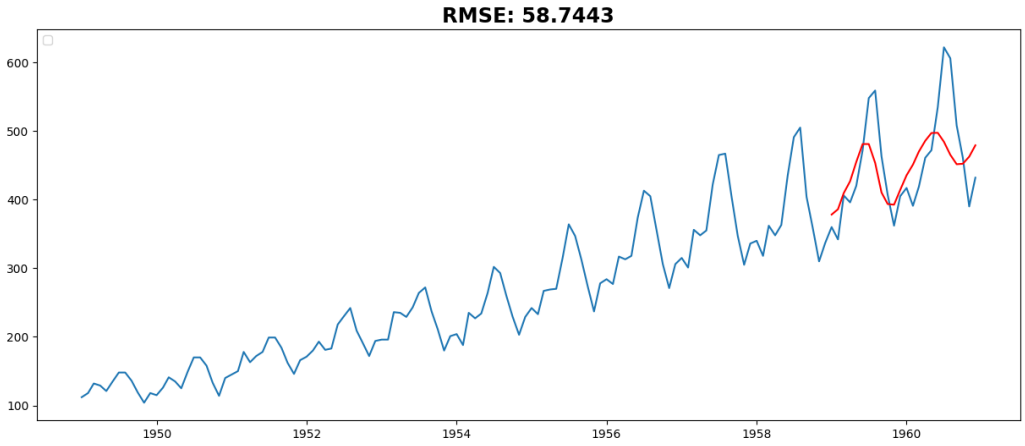

Figure 9. Rolling RSME using Baseline LSTM with Attention Mechanism

In a similar fashion, modifications can be added to the model to potentially improve accuracy. Modifications mentioned within the previous LSTM discussion will be noted. The activation method, Sigmoid, does not perform well. Tanh and relu perform more similarly. Regularization improved results. Decreasing the dropout rate improved results as well. The final model I could find to improve results was: lstm units equaling 291, a dropout rate

of 15.15 percent, regularization, no normalization, and three additional dense layers having units equal to 291 and an activation method of relu. The non-rolling RSME was equal to 24.9804. The rolling estimate of 29.1294 can be seen below:

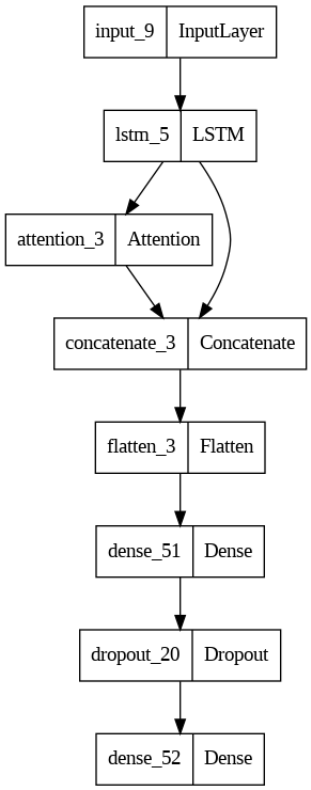

Figure 10. Modified LSTM with Attention Mechanism Architecture

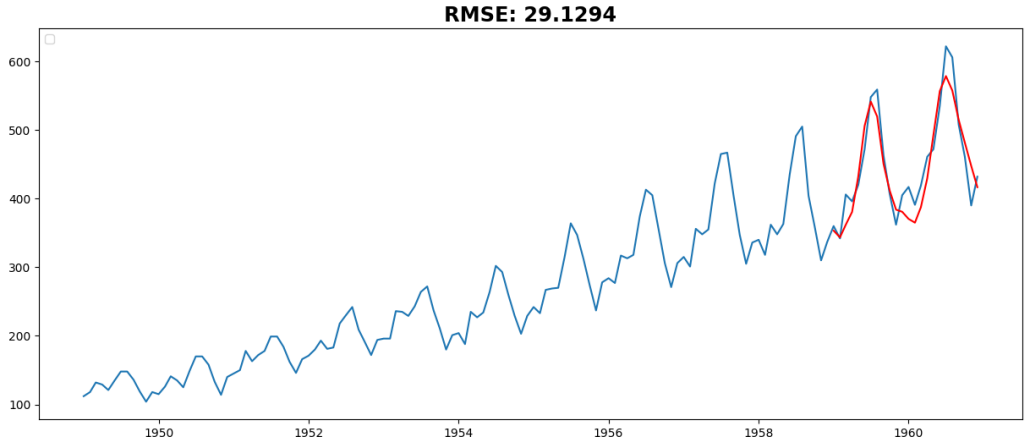

Figure 11. Rolling RSME using Modified LSTM with Attention Mechanism

Baseline CNN vs. Modified CNN

Model building and training within the baseline CNN model includes specifying 64 filters, kernel size of 5, activation method of "relu", and an additional convolution layer. Two dense layers are added within this model. The non-rolling RSME is equal to 18.6110

and the rolling RSME is equal to 18.9082 which can be seen below:

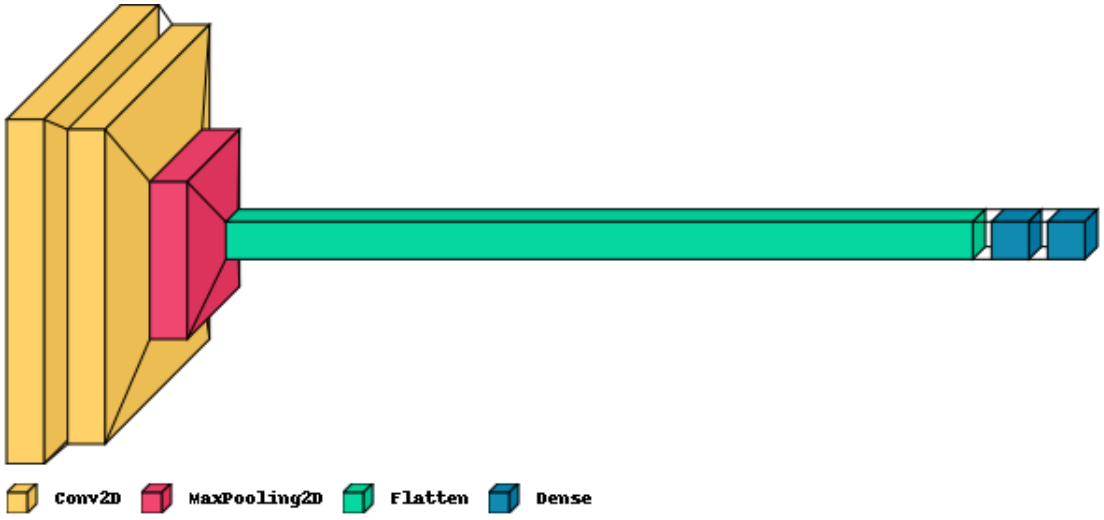

Figure 12. Baseline CNN Architecture

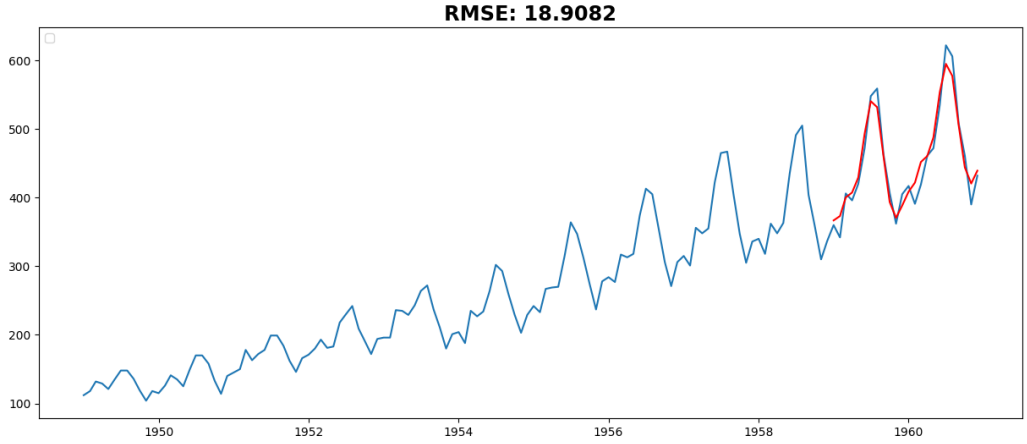

Figure 13. Rolling RSME using Baseline CNN

Various modifications to add include varying the amount of filters within the input and additional convolution layers. Alteration of the kernel size was attempted but a size of 5 produced the best results. The method tanh seemed to perform similarly to relu. Sigmoid still performed worse than these two methods, similar to what happened within LSTM. Adding additonal convolution layers with an activation method of relu and kernel size of 1 also improved results. Removing regularization from the input and additional convolution layers also improved results. Finally, reducing the pool size to 1 made the RSME decrease. Increasing the pool size rapidly increased the RSME. The final parameters that were chosen for this model included a filter size of 150 and kernel size of 5 within the input layer, 4 convolution layers having filters equal to 30 and three of which had a kernel size of 1 and the last had a kernel size of 5, removed regularization from input and additional convolution layers, a pool size of 1, and three dense layers having units 30 and an activation method of relu for prediction. The non-rolling RSME was equal to 17.2463. The rolling RSME was

equal to 16.4455 and can be seen below:

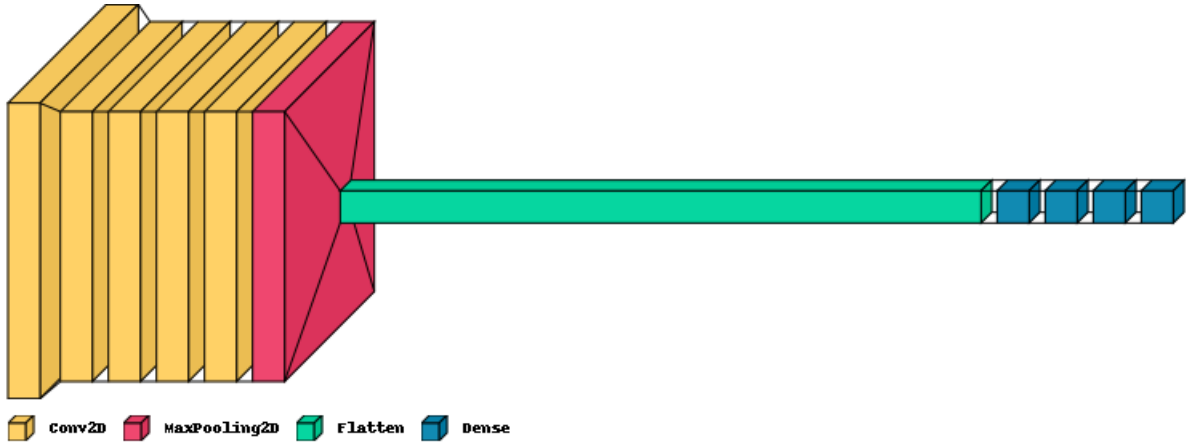

Figure 14. Modified CNN Architecture

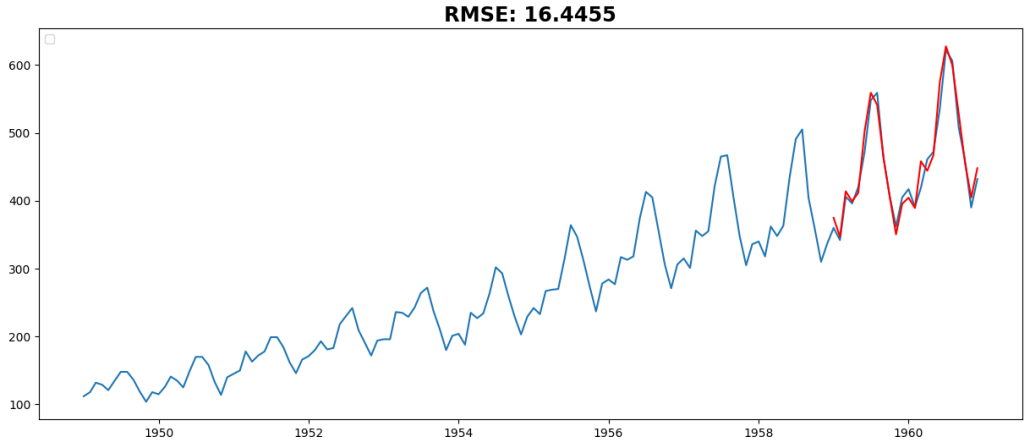

Figure 15. Rolling RSME using Modified CNN

Baseline Transformer vs. Modified Transformer

The final baseline model that will be assessed is the transformer. The current number of heads within the encoder is equal to 4, with a head size of 100. The hidden layer size within the feed forward network inside the transformer is equal to 64. Global average pooling is performed and 2 dense layers are added, with an activation method of "relu". The non-rolling RSME is equal to 31.5618 and the rolling RSME is equal to 50.9936 which can be seen below:

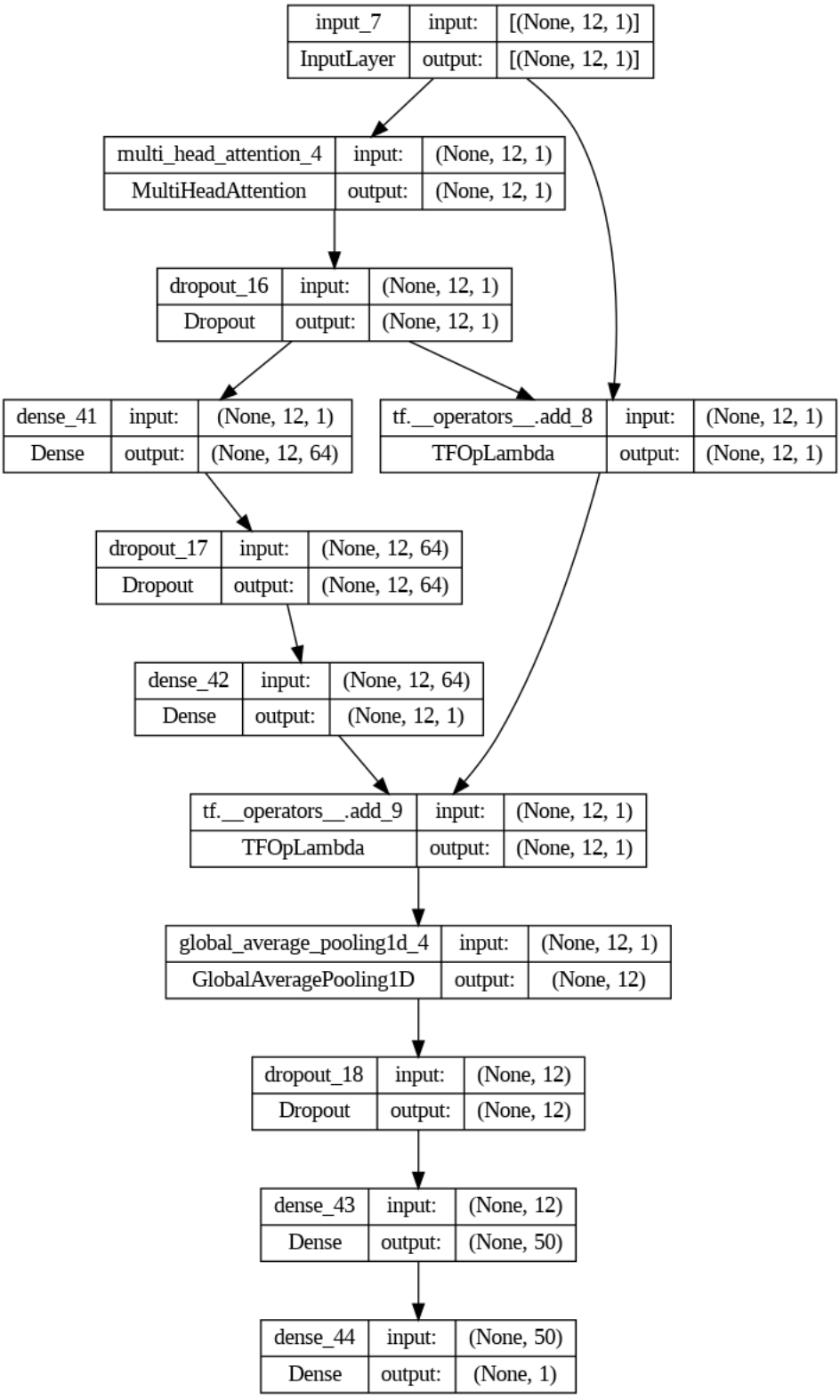

Figure 16. Baseline Transformer Architecture

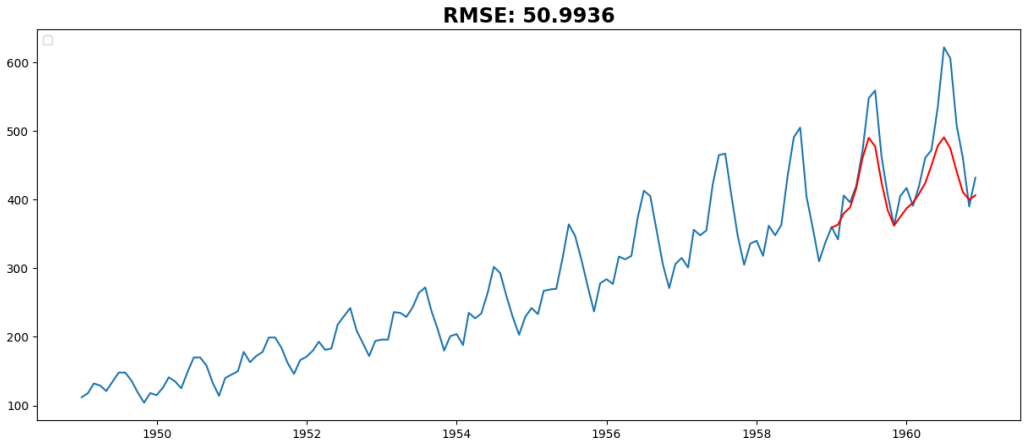

Figure 17. Rolling RSME using Baseline Transformer

Modifications can also be applied to this model. Various modifications include changing the number of heads. It is found that deviating too far from a hidden layer size of 64 decreases accuracy. Increasing the headsize was also helpful. Altering the dropout rate to .098 percent was useful. This method interestingly benefited from having dense layer activation methods of tanh. The final model that was produced with these modifications in mind was having a hidden layer size of 65, 3 heads, .098 percent dropout rate, a head size of 214, no normalization, a combination of some layers having regularization and others not, 3 additional encoders having activation methods of tanh and having sizes 49, 50, and 69, respectively, and two dense layer predictions having units 1, 30, and 150, respectively. 2 of the prediction layers utilized tanh, while the final used relu. The non-rolling RSME was equal to 23.3656 and the rolling RSME of 29.4385 can be seen below:

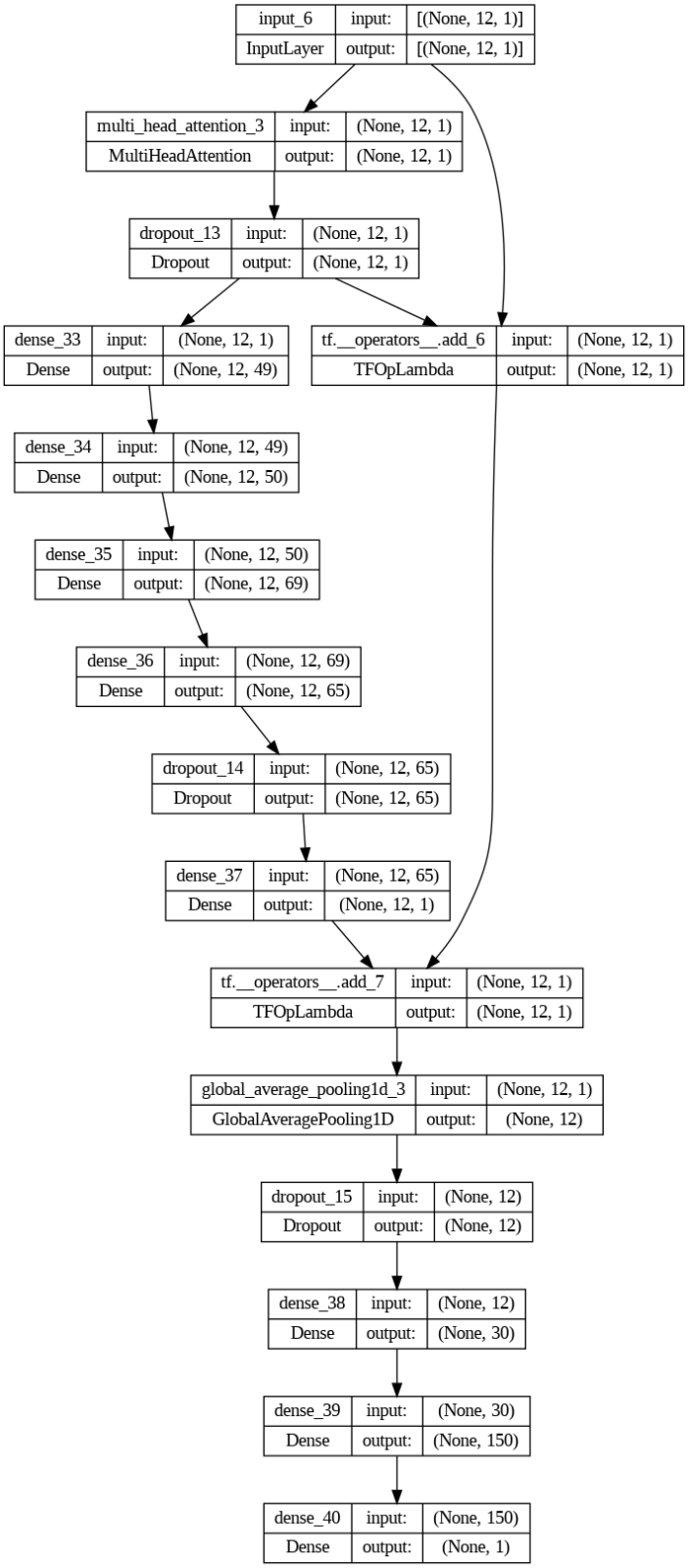

Figure 18. Modified Transformer Architecture

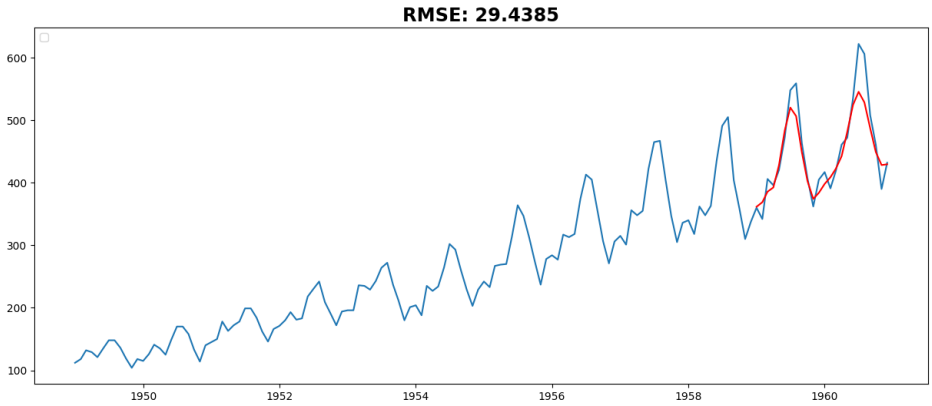

Figure 19. Rolling RSME using Modified Transformer

4 Conclusion

Based on the results provided, we find that the order in which the models performed the best include: CNN, LSTM, LSTM with Attention Mechanism, Transformer, and ARIMA, respectively (from best to worst). Transformer and LSTM with Attention Mechanism had very close results.

It was interesting to modify the various elements of the neural networks and parameters within ARIMA. It felt as though the network training within CNN performed the quickest compared to the other models. It was also interesting to note that, overall, the activation method of relu performed the most accurately. It also seemed as though normalization wasn't needed in many of the models. Adding additional convolution, dense, or encoder layers to the various models proved to be beneficial. In my opinion, I placed more importance on the rolling prediction results rather than the non-rolling since, as mentioned, it is a more realistic measure.

In addition to concluding on the positive aspects of modifications within neural networks that improved accuracy results, it's also important to mention the aspects that made results worse. Overall, I'd say that neural networks are quite finicky or tedious. Changing a parameter by just one unit can dramatically increase or decrease results. For example, adding too many or too little encoders, convolution, or dense layers can cause for negative results. Having too large or low of a dropout rate is not good either. Normalization and signmoid activation method didn't seem beneficial. It seemed as though there may be a dependency on the input layers size and the additional dense layer sizes that are added (just a hypothesis).

Further development within this analysis could include creating more sophisticated functions that loop through various parameters until the lowest RSME is achieved. It would also be interesting to test other neural networks capabilities.

5 References

Introduction to Long Short Term Memory (LSTM), Artificial Intelligence +, 26 June 2023, www.aiplusinfo.com/blog/introduction-to-long-short-term-memory-ltsm/.

Zhuang, Yong. *Exploring Neural Network Architectures in Time Series Prediction*, Github, 2023, yong-zhuang.github.io/gvsu-cis635/assignment4.html.

Zhuang, Yong. *A Comparative Study on the International Airport Passengers* Dataset, Github, 2023, yong-zhuang.github.io/gvsu-cis635/air-passenger-forecast.html.

What are convolutional neural networks?, International Business Machines Corporation (IBM), www.ibm.com/topics/convolutional-neural-networks.

What are neural networks?, International Business Machines Corporation (IBM), www.ibm.com/topics/neural-networks.

Understanding Transformer Neural Network Model in Deep Learning and NLP, Turing, 2023, www.turing.com/kb/brief-introduction-to-transformers-and-their-power.