Predicting Network Throughput Using Recurrent Neural Networks

Everett Tucker, North Carolina State University

26 November 2024

1 Abstract

With the advent of 5G and 6G high-frequency networks in urban areas, it is increasingly necessary to have shorter-distance relays to aid communication. Aerial Base Stations are one potential solution wherein UAVs would serve as relays between ground users and traditional cell towers. For this task, predicting users' throughput becomes vital for maximizing their communication with the UAVs. It allows the UAV to position themselves optimally not just for current needs, but future demand as well. In this paper, a Recurrent Neural Network (RNN) is used to predict throughput data gathered from a dataset of network anomalies. When combined with Standardization, Principal Component Analysis (PCA), and a smoothing convolution, the RNN model reduced test prediction error by 21.3% compared to a naive moving average baseline.

2 Background

Predicting network throughput is a pressing problem in machine learning with applications in streaming, low-latency communications, and energy efficiency. There are many promising algorithms for throughput prediction, including Autoregressive Integrated Moving Average (ARIMA), Recurrent Neural Networks, and Long Short-Term Memory (LSTM) Networks.

ARIMA uses both an Autoregressor, a regression that predicts future values based on a buffer space of past values, and a moving average, a simple convolution, to predict both seasonal and unseasonal data. It is the most explored of the three models for this task, with several papers finding excellent results for throughput prediction.[1,2]

The recurrent neural network approach, which was explored in this paper, involves feeding inputs from the data into different points in the neural network, and reusing outputs from nodes back into the neural network. The outputs of the nodes are also run through an activation function to introduce non-linearity. This architecture gives RNNs the ability to consider different points on the input buffer in varying ways, while keeping the complex pattern-detection abilities of

a neural network. This type of network has been explored for time series data, though they often involved more complex architecture than the design explored in this paper.[3]

The next technique most often explored in time series analysis is the Long-Short Term Memory algorithm. This model seeks to determine which input data points to remember, and which to forget. The architecture uses input, output, and forget gates together to construct an optimal memory of the previous data values. These values are then used for prediction through a neural network. This memory eliminates the need for a static buffer, and allows the model to choose which pieces of data to consider in each iteration. This method has been utilized in time series with high accuracy to predict the traffic at major shipping ports.[4]

3 Data Analysis

Before training a model, it is important to understand the dataset. The data was imported from the Kaggle "network-anomaly-dataset" and graphed over time to understand the general trends and variance within the data. Figure 1 shows different communication metrics displayed over time, including throughput, congestion, packet loss, latency, and jitter.

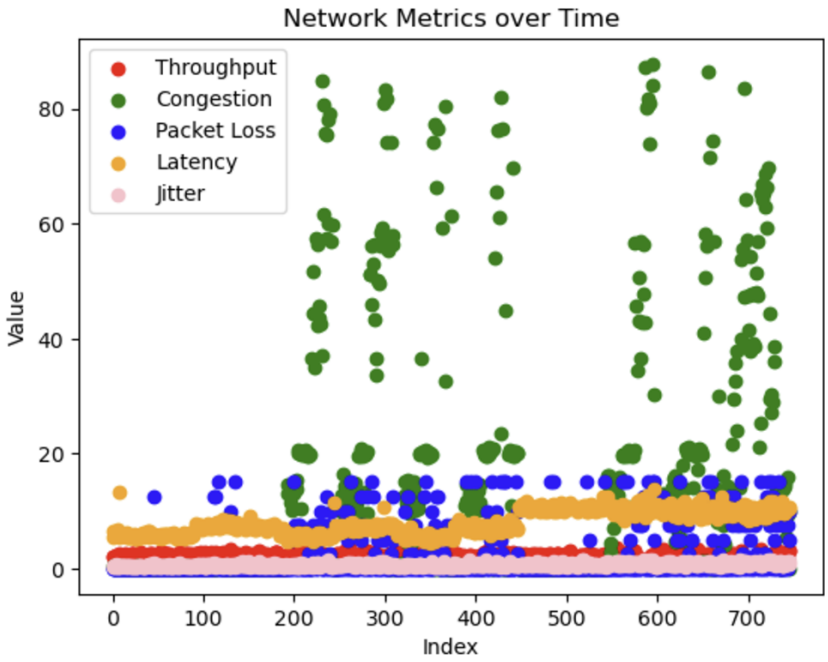

Figure 1: Communication Metrics Plotted over Time

Figure 1 shows that there is a high degree of variability in the data, and some somewhat clear period trends in some of the time series. This variability suggests that the different network metrics are not that related, and thus each contains useful information to the final prediction problem.

Next, a linear model was used to find the correlation between the different network metric time series. This simple model seeks to predict the throughput for a given time given the congestion, packet loss, latency, and jitter at that point. The linear model returned a correlation of 0.26, which suggests that they may be slightly correlated and that Principal Component Analysis may help reduce dimensionality while preserving their useful variation.

Finally, lag plots were created to show the distribution of consecutive network metrics. This is similar to plotting the distribution of the derivative of a time series for discrete points. Figure 2 shows a lag plot for the throughput time series with a lag of 1.

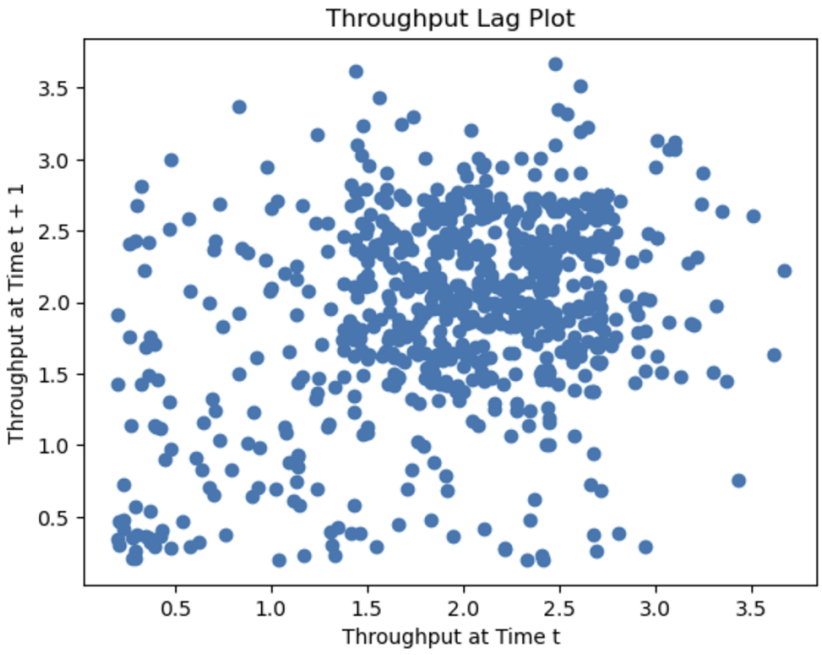

Figure 2: 1-Lag Plot for Throughput Data

Figure 2 shows a general cluster around the middle of the plot which suggests that most throughput values fall close to the mean. There is also a slight positive linear trend that means throughput values follow their previous values closely. In general, this lag plot is fairly random which suggests that throughput is not trivial to predict from its previous points.

4 Data Augmentation

The data was modified in several ways to increase the quality of data, and reduce the amount of unnecessary information to allow for more model training iterations. First, the data was pruned to remove samples that were marked as an anomaly in either throughput, congestion, packet loss, latency, or jitter. These points were likely not as indicative of the larger trend in the data, so they were less useful for model training and excluded. Next, the data was standardized to mean zero and variance one to help the neural network converge more readily and keep the parameter scalars small.

The next step involved using Principal Component Analysis to reduce the feature dimensionality from 5 to 3. Multiple values for the number of components were tested, and it was determined that using 3 components decreased training time enough while also maintaining enough information to make good predictions. The PCA was computed exactly because of the small training size. The standardization combined with PCA resulted in input data that had concentrated information both in scalar value and tensor shape.

5 Model Architecture

The recurrent neural network using for prediction uses a single line of k linear nodes, which each take in a single input vector with as many elements as PCA components. Additionally, each node takes in the output of the last linear node, which also is a vector with the number of components used in PCA. The first layer is defined to take in a vector of all zeros as the previous input. After the results are computed by the neural network and the entire time series is predicted, a modified moving-average convolution is applied to smooth the results and ensure less outlier values in the predictions.

Figure 3 shows a diagram of the recurrent neural network architecture, including the line of hidden linear nodes, the input data, and the final convolutional step.

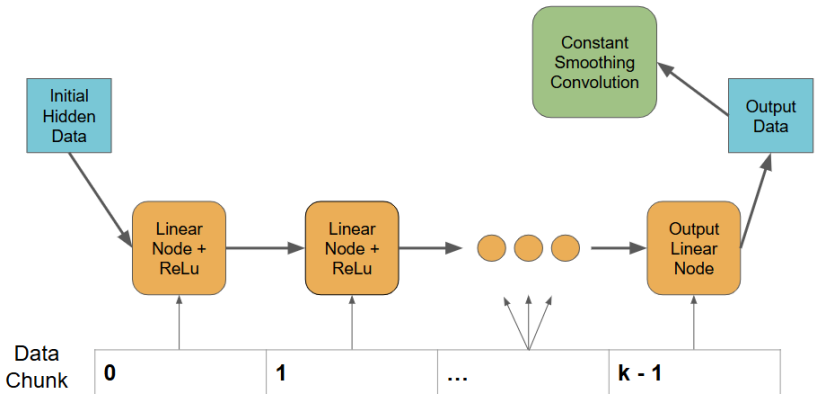

Figure 3: Complete Recurrent Neural Network Architecture

Figure 3 highlights the entire model design, except for the data preprocessing steps including pruning, standardization, and PCA. It is also important to note that the final hidden node outputs a single scalar as its output into the convolution, instead of the vector returned by the previous k - 1 hidden nodes.

6 Model Training

The model training procedure consisted of hyperparameter optimization first based on the loss function, then on training accuracy, and finally on the validation data. The test data was only used at the very end to verify the results of the model training.

The hyperparameters used in this model include the buffer size, learning rate, activation function, loss function, PCA components, convolution size, and initial hidden layer. The hyperparameters were trained in order of effect on the final model, with the most sensitive parameters being tuned first.

Table 1: Hyperparameter Chart with Optimal Values

Hyperparameter	**Optimal Value**
Buffer Size (k)	3
Learning Rate	0.0001
Activation Function	ReLu
Loss Function	MSE
PCA Components	3
Convolution Size	6
Initial Hidden Layer	Zero Vector

The final model was trained over 300,000 iterations of the original dataset, which consisted of about 750 samples of 5 features. Figure 4 shows the training

loss plotted every 1000 iterations over this training period.

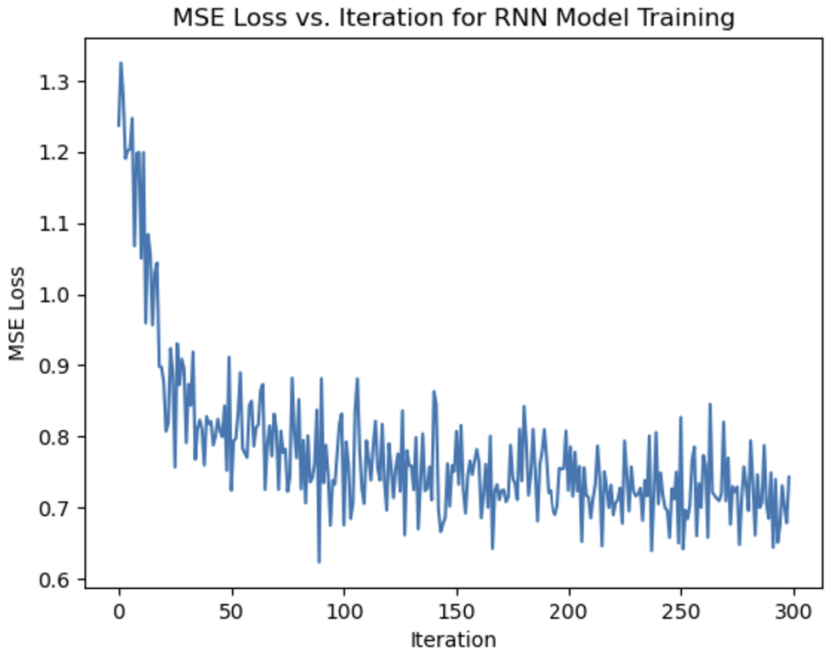

Figure 4: Final Model Training Loss for 300000 Iterations

Figure 4 shows a model training loss curve that is fairly standard, but has quite a bit of variation. This is likely due to the small dataset, which means that the random variations in model loss across test instances contribute to an average loss with a higher variation. The general trend of the loss decreases at a nice curve, and levels out at about 300,000 iterations which suggests that the model converges within the allotted steps.

7 Results

To validate the model, results were demonstrated for the testing set and test set. Additionally, statistics are computed for the Root Mean Squared Error (RMSE), and the Mean Absolute Percent Error (MAPE). Because the error statistics are hard to comprehend outside of comparison, a moving average prediction is introduced to serve as a baseline for the model. The moving average baseline is simply the mean of the k-length buffer. It assumes every data point is approximately the mean of the previous k data points. The RNN's predictions are compared to this baseline to give some intuition for model performance. The results are demonstrated in Figures 5 and 6, which display the Training and Test results, respectively.

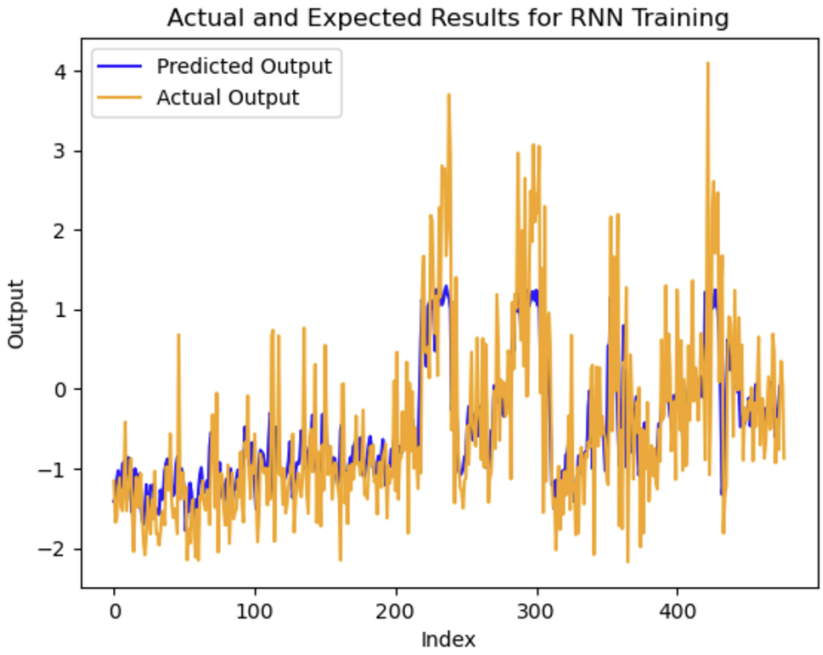

Figure 5: RNN-Predicted and Expected Results for Training Set

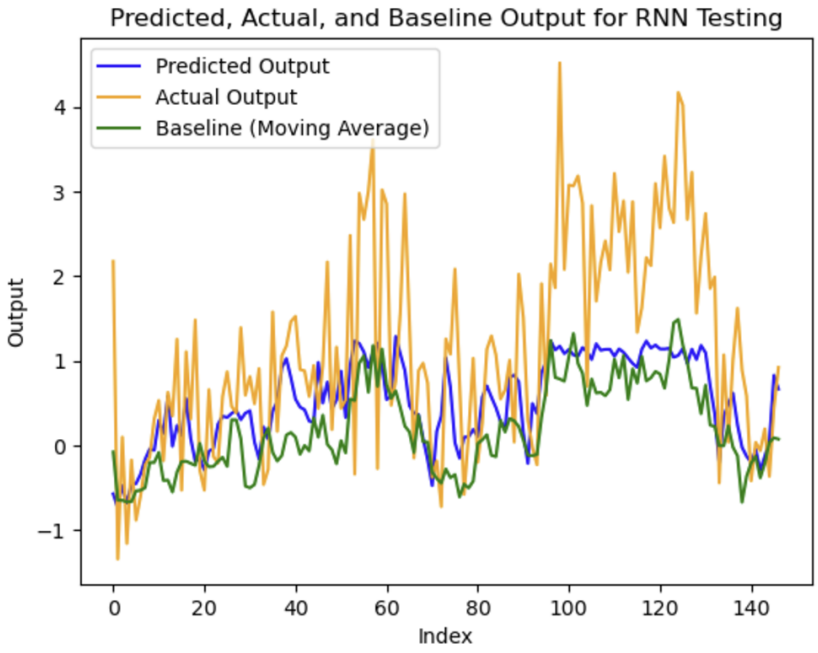

Figure 6: RNN-Predicted, Baseline-Predicted, and Expected Results for Testing Set

In addition to the graphs of the expected and predicted results, the RNN model achieved a RMSE of 0.8782 and a MAPE of 4.375 on the training data set. Furthermore, the model scored a Test RMSE of 1.5046, a Test MAPE of 2.0509. In comparison, the baseline scored a MAPE of 2.4889, so the RNN model offers a 21.36% decrease in the test MAPE compared to the moving average baseline.

8 Future Work

Future work will involve expanding the predictive capacity of this model including generalizing the predictive capacity to a dynamic range of data that could predict multiple data points in the future, instead of using a static buffer to predict only a single data point in the future.

Additionally, a more refined convolutional layer could be applied to the end of the model to aid in the predictions and make the model smoother and more predictive. In this work, the convolutional layer is static in both size and value. A future model could have a more dynamic convolution or more convolutional layers.

Building a more advanced model also involves training on more and higher quality data. Achieving truly useful results would involve collecting mobile

throughput data from an actual urban setting, and using more factors about user position to trace complex relationships the characterize when users are more likely to use data in a specific area or transportation method.

9 Stakeholder Acknowledgments

The potential stakeholders for communication through UAV aerial base stations include pedestrians that use data in urban settings, busses, above-ground trains, and other traffic infrastructure like traffic lights and signals. These users could use a higher level of connectivity to build smart city infrastructure and public transportation that would decrease travel times with network optimizations based on maintaining connectivity of traffic devices. In addition to the benefits in communication infrastructure, this development would also increase quality-of-life for ground users by increasing download and message speeds for all communication types.

This model could potentially have a negative effect on user privacy, because it requires collecting user communication and location data throughout their time in an urban setting. This data could potentially be mishandled by a variety of capital interests including advertising. While the model described here doesn't require any specific information about user data, it is likely that this data would be collected by adjacency by anyone who implements this or a similar algorithm in a commercial setting.

Furthermore, because of the scale and infrastructure of such a design, it would be difficult to test in practice without collecting a large amount of sensitive user data including location and network throughput. This could be tested in a beta setting with a few UAVs and a small area such as a single intersection, with users consenting to share their location and communication data. Simulation is also a promising tool, especially because of the reinforcement learning background of this model. However, this emulation ignores user experiences and privacy concerns that would be present in a realistic test of the model.

10 Conclusion

Based on Figure 4, the training loss plot, the model is likely converging to an optimal policy, but the variability is indicative of a learning rate that is too high. Furthermore, Figures 5 shows the actual and expected results for the model training data, which are very tightly correlated. This means that overfitting is highly probably in the training process. When this is observation is combined with some of the model's mispredictions in the testing sets, overfitting is a clear cause for some of the model's failures. However, because of the high degree of randomness in the throughput data, any prediction model will probably have a significant error term, so the model error and improvement of about 21% over the baseline is substantial for an initial model test. In the future, more complex method and higher quality data can be used to improve the model

for the specific scenario of predicting throughput in urban settings. Finally, user position could be infused into the model data to account for throughput changes that commonly occur when pedestrians enter a new area. Combining more diversified input data and a more nuanced model would likely improve results into a range usable for practical UAV communication optimization.

11 Works Cited

[1] X. Dong, W. Fan, and J. Gu, "Predicting LTE Throughput Using Traffic Time Series," ZTE Communications, vol. 13, no. 4, Nov. 2015, doi: https://doi.org/10.3969/j.%20issn.%201673-5188.

[2] D. Lee, D. Lee, M. Choi, and J. Lee, "Prediction of Network Throughput using ARIMA," Feb. 2020, doi: https://doi.org/10.1109/icaiic48513.2020.9065083.

[3] S. Shankar, P. V. Ilavarasan, S. Punia, and S. P. Singh, "Forecasting container throughput with long short-term memory networks," Industrial Management & Data Systems, vol. 120, no. 3, pp. 425–441, Dec. 2019, doi: https://doi.org/10.1108/imds-07-2019-0370.

[4] N. D. Tan, H. C. Yu, L. N. B. Long, and S.-S. You, "Time series forecasting for port throughput using recurrent neural network algorithm," Journal of International Maritime Safety, Environmental Affairs, and Shipping, vol. 5, no. 4, pp. 175–183, Oct. 2021, doi: https://doi.org/10.1080/25725084.2021.2014245.