Modelling of a Recurrent Neural Network based on Surrogate of Amplitude Simulation

Sonal Rajendra Palrecha - 11266082 Milankumar Hareshbhai Mavani - 11280466
Rishikesh Ravikiran Tiwari - 11257044

Abstract

Turbochargers enhance internal combustion engine performance by increasing power output, but high-speed operations introduce internal vibrations that require precise monitoring. Traditional methods using strain gauges face spatial limitations, necessitating alternative predictive approaches. This study explores the development of a Recurrent Neural Network (RNN) model as a surrogate for computationally expensive amplitude simulations. Through Exploratory Data Analysis (EDA) and feature engineering, we identified key parameters influencing nominal amplified amplitude prediction. A hybrid modeling strategy combining RNN for lower frequency ranges and Random Forest for higher frequencies significantly improved prediction accuracy, reducing Mean Squared Error (MSE) from 16 to 13. This approach minimizes reliance on physical simulations while ensuring effective vibration monitoring. Future work aims to refine deep learning models by integrating larger datasets and advanced feature selection techniques.

1. Introduction

Turbochargers improve the performance and efficiency of internal combustion engines by compressing air and increasing power output. However, during high-speed operation, turbochargers encounter internal vibrations that, if not adequately monitored and regulated, can cause mechanical problems. Previously, engineers have used strain gauges to detect vibration amplitudes, but these sensors cannot be placed everywhere, limiting the efficiency of direct measurements.

Simulations are used to estimate critical vibration amplitudes based on experimental data to address this issue. However, these simulations are computationally expensive and consume a large amount of time and resources. The purpose of this research is to replace costly simulations with a neural network capable of predicting maximum nominal amplitude values based on observable data, thus reducing the need for physical testing.

The dataset provided to us contained a variety of parameters gathered from turbocharger tests. The primary goal was to create a Recurrent Neural Network (RNN) model that predicts nominal amplitude using measured parameters. This predictive technique would decrease the need for expensive simulations while assuring precise vibration monitoring for turbochargers.

2. Exploratory Data Analysis

The initial stage of the project involved performing Exploratory Data Analysis (EDA) on the provided dataset to gain insight into the data structure, identify potential issues, and derive patterns. Key aspects of the EDA process are highlighted below:

2.1. Data Overview

The dataset consisted of 20,185 rows and 87 columns, with a mix of numerical and categorical variables. Data types were identified and missing values were checked. The target variable for optimization was defined as `N`[Nominal Amplified Amplitude]. The features `M`[Amplitude] and `R` were removed from the dataset and then the following preprocessing steps were executed.

2.2. Corelation and Feature Importance

In our dataset, we had two categorical features, `C`[Series] and `F`[Strain gauge], which we encoded using the Ordinal Encoder(1) to convert them into a numerical format suitable for modelling. To assess feature importance, we employed a Random Forest model(2), which evaluates the contribution of each feature to the predictive power of the model. Upon analyzing the correlations between the features and the target variable, `N`, we found that while some features exhibited moderate correlations, none showed a particularly strong relationship. This observation suggested the necessity of retaining a diverse set of features to capture potential complex interactions within the data.

Further, we performed feature importance analysis using the Random Forest model and plotted the top 40 features to visualize their contributions. The resulting plot revealed that the most significant feature was `J`[frequency], indicating its substantial influence on the target variable, as seen in

Figure 2. This insight underscores the importance of J in our modelling efforts and highlights the value of feature importance analysis in understanding the underlying data dynamics.

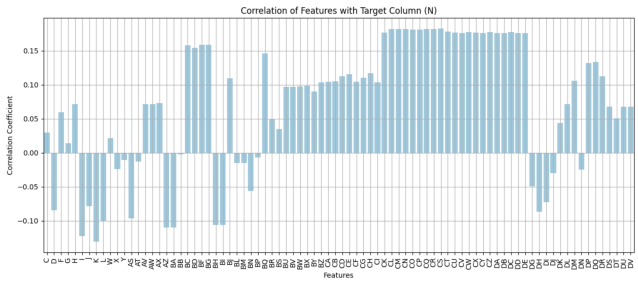

Figure 1. Correlation

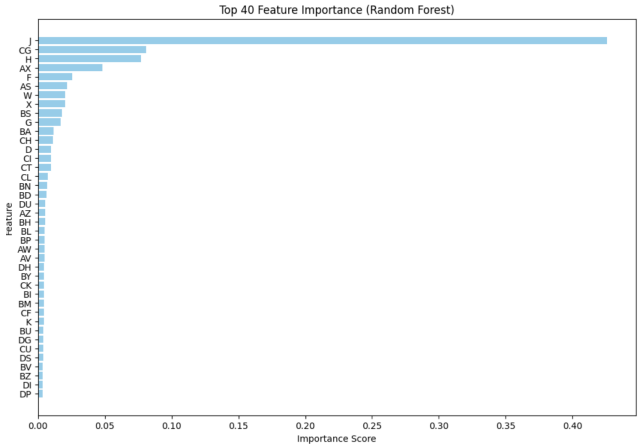

Figure 2. Feature Importance

2.3. Outlier Analysis

Outliers were analyzed using the Isolation Forest(3) algorithm with a contamination threshold of 5%. Although 1,006 rows were identified as outliers, their removal showed no significant change in the results during model training. Consequently, the full dataset, including outliers, was used for model development to retain maximum data variability.

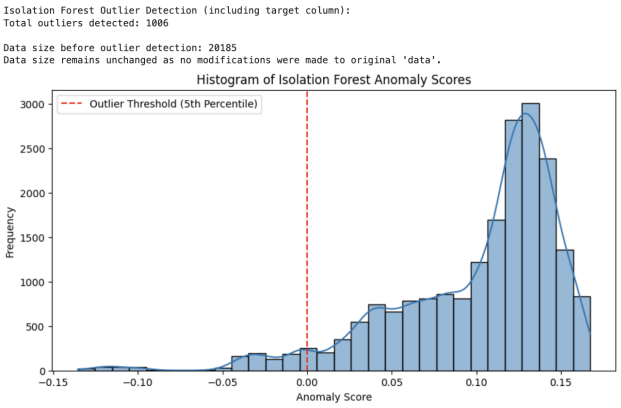

Figure 3. Outlier Analysis

2.4. Key Insights

- Outlier removal was deemed unnecessary as it did not significantly affect model performance.
- Retaining all features except M and R ensured a balanced dataset with adequate predictive power.
- Moderate correlations between variables and the target (N) emphasized the importance of keeping multiple features for capturing complex relationships.
- EDA established a strong foundation for the subsequent modelling phase, ensuring the dataset was clean and well-prepared.

By leveraging the full dataset, comprehensive feature set, and grouped analyses, we aimed to maximize the model's ability to learn and generalize effectively.

3. Feature Engineering

Feature engineering played a crucial role in optimizing our predictive model for the target variable N . Our approach was iterative, leveraging insights from feature importance analysis, binning techniques, and tailored modeling strategies to improve performance. Below is a step-by-step breakdown of our process.

3.1. Initial Feature Selection

Upon receiving the data set, we performed an initial analysis to determine the most relevant features for predicting N. After dropping features M and R, we observed that J had the highest feature importance. This insight guided our subsequent steps in feature transformation and model selection.

3.2. Feature Binning

To better understand the impact of J, we implemented feature binning. Initially, we divided the frequency range into

five bins, ensuring each contained an equal number of data points(4). By evaluating feature importance within each bin, we noticed that different bins exhibited distinct patterns of importance, which can be seen in Figure 4

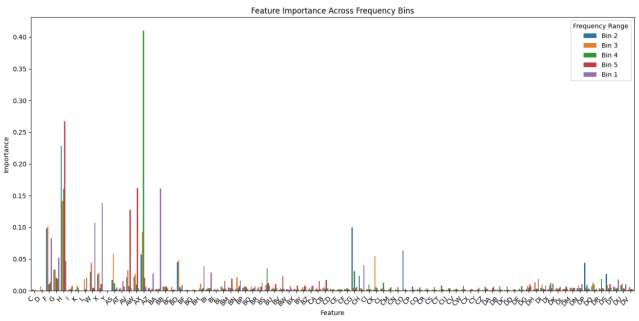

Figure 4. Feature Importance across bins

This observation led us to explore a neural network(5)-based approach, where the data was segmented based on frequency bins. For each bin, the model was trained using the top 40 most important features identified within that specific bin. However, the neural network struggled to generate accurate predictions, yielding a high overall Mean Squared Error (MSE) of 52.

To improve performance, we then implemented a Recurrent Neural Network (RNN)(6) model, leveraging its capability to process sequential data and capture temporal dependencies. While, the RNN demonstrated better predictive capabilities compared to the initial neural network, we observed that bins 4 and 5 consistently exhibited significantly higher MSE values than the other bins. This suggested that the model was unable to generalize well for data points in these frequency ranges, possibly due to data sparsity. To address this, we experimented with a Random Forest model, which led to improvement, resulting in MSE of 16.The higher error rates in bins 4 and 5 persisted, indicating the need for further refinements or alternative modelling strategies to enhance prediction accuracy in these challenging J ranges.

3.3. Refinement Based on frequency J Distribution

To investigate the discrepancies in prediction accuracy, we plotted various features against frequencyJ. As seen in the Figure 5, the data distribution is clearly imbalanced, there is a high concentration of data points below 10,000 Hz and a sparse distribution above this threshold. Given this imbalance, we hypothesized that a single-model approach might not be optimal.

3.4. Hybrid Modeling Strategy

Based on our observations, we adopted a dual-model approach:

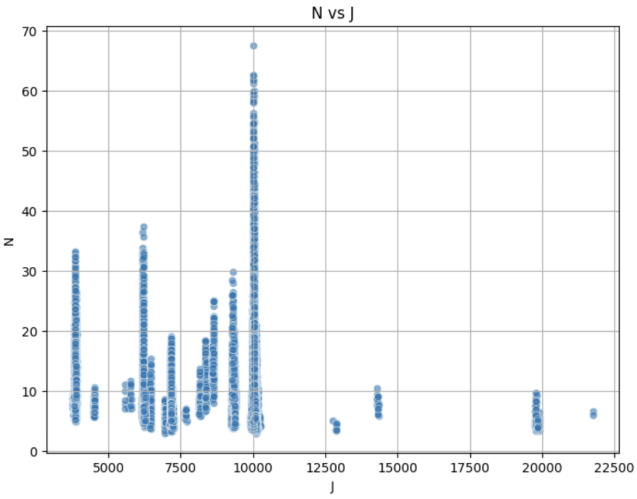

Figure 5. N vs J

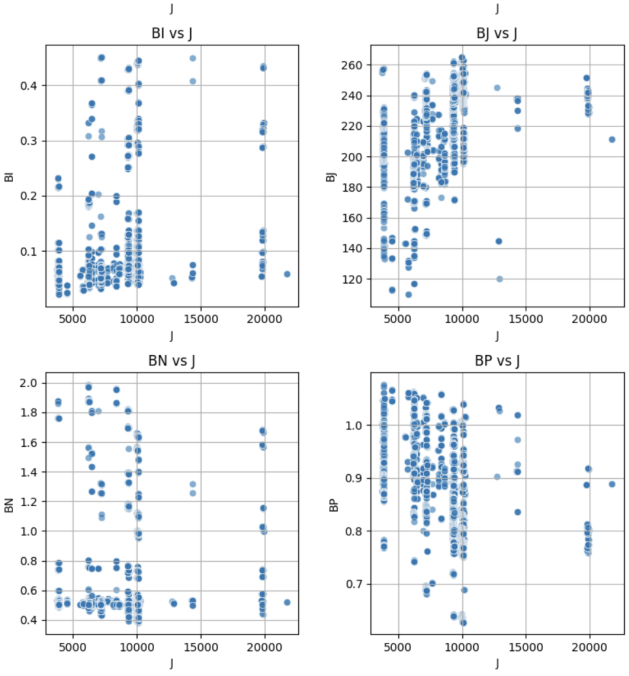

Figure 6. J vs Other features

- A Recurrent Neural Network (RNN) for data below 10,000 Hz, as sequential dependencies in this range were prominent.
- A Random Forest model for data above 10,000 Hz, where tree-based methods proved more effective.

Unlike our earlier approach, feature binning was not applied in this phase. Instead, we allowed each model to learn from the top 40 features.

3.5. Final Model Performance

After training these two specialized models, we evaluated their predictive performance on the second dataset. This hybrid approach resulted in an improvement, reducing the MSE from 16 to 13.

4. Why This Model?

We initially began with a simple neural network to train our model. However, the predictions were suboptimal, as the model could accurately predict lower values but struggled with higher values, effectively hitting a ceiling, as seen in Figure 7. With this model, we obtained a mean squared error (MSE) of 34.

To improve performance, we experimented with various models, including Recurrent Neural Network[RNN], Long Short Term Memory [LSTM](7), Bidirectional LSTM, Gated Recurrent Unit [GRU](8), and Bidirectional GRU. Unfortunately, these models did not yield significant improvements, as their results were nearly identical to those of the neural network. To further refine our approach, we performed hyperparameter tuning using Optuna on all the models mentioned above. While this led to slight improvements, the best results were achieved using an LSTM model, reducing the MSE to 28. However, the core issue persisted—the model still failed to predict higher values accurately.

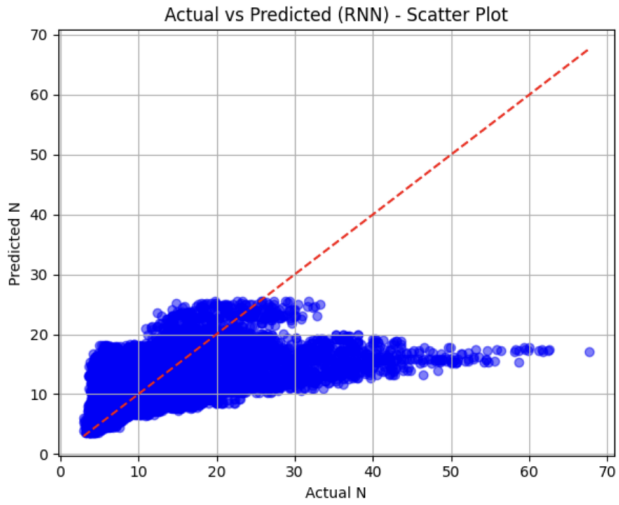

Figure 7. RNN Model

Following the feature binning strategy discussed in Section 3.2, we explored the impact of the frequency J feature on model performance. Initially, we applied feature binning using the frequency feature and trained a Random Forest-based model, with this the predictions improved significantly compared to the previous models, resulting in MSE of 16. Upon further analysis, we plotted the frequency feature against other features and discovered that the majority of data points were concentrated in the frequency range below 10,000 Hz, with very few data points above this threshold.

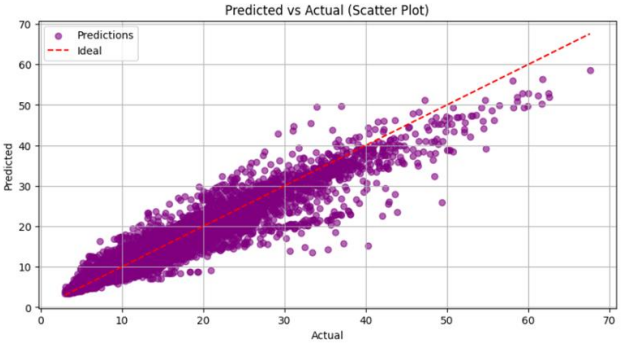

Figure 8. Random Forest using Feature binning

Based on our hybrid modelling strategy, we adopted a two-pronged approach: training an RNN model for data points in the frequency range below 10,000 Hz and a Random Forest (RF) model for values above 10,000 Hz. This hybrid model proved to be the most effective, yielding an MSE of just 13.7, making it our best-performing model.

5. Best Model

Our best model comprised of RNN and Random forest first lets see the basic concept of each -

5.1. Recurring Neural Network

A Recurrent Neural Network (RNN) is a type of deep learning model designed to process sequential data, such as time-series data, speech, and text. Unlike traditional neural networks, RNNs have a recurrent structure that allows them to retain information from previous inputs and use it to influence future predictions. This is achieved through a hidden layer that maintains a short-term memory of past data, enabling the model to understand contextual relationships within a sequence. For instance, when predicting the next word in a sentence, the RNN recalls prior words to make a more accurate prediction. RNNs are trained using a method called Backpropagation Through Time (BPTT), which adjusts the model's weights based on past errors to improve accuracy(6)

5.2. Random Forest

A Random Forest Regressor is an ensemble learning method for regression tasks that builds multiple decision trees and averages their outputs for improved accuracy. Each tree is trained on a randomly selected subset of the training data (with replacement) to enhance diversity and reduce overfitting. At each split, a random subset of features is considered to prevent dominant features from influencing the model.

The final prediction is obtained by averaging the outputs of all trees, making the model more stable and generalized. This approach mitigates overfitting and reduces sensitivity to noise. Random Forest is effective for handling high-dimensional data and capturing complex relationships.(9)

5.3. How our model is working

We segmented our dataset based on frequency, categorizing data into two groups: frequencies below 10,000 Hz and those above. The subset with frequencies below 10,000 Hz was then scaled and prepared for training using a Recurrent Neural Network (RNN). To optimize the performance of the RNN, we utilized Optuna(10) for hyperparameter tuning, ensuring the selection of the best possible parameters.

The training process was conducted based on whether the frequency $\jmath$ was below or above 10,000 Hz. After training, the RNN model achieved a training loss of 0.068 and a validation loss of 0.0701, using the L1 loss function. For the high-frequency data, we employed a Random Forest model, which resulted in a Mean Absolute Error (MAE) of 0.2055.

To evaluate model performance, we plotted both actual vs. predicted scatter plots and line graphs for both models. As observed in the figure 9, the RNN model still struggles with the ceiling effect; however, it performs better than the previous model as seen in the Figure 7 and tends to predict slightly higher values, as seen in the graphs,

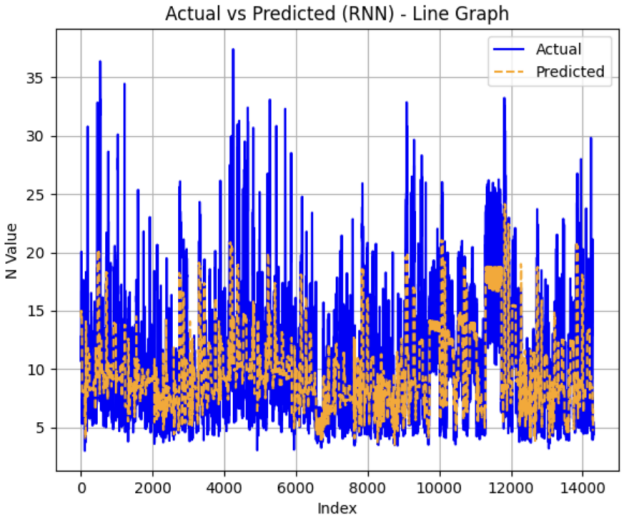

Figure 10. Line Graph RNN

while the Random Forest model demonstrates strong performance, particularly for higher-frequency data as seen in Figure 11

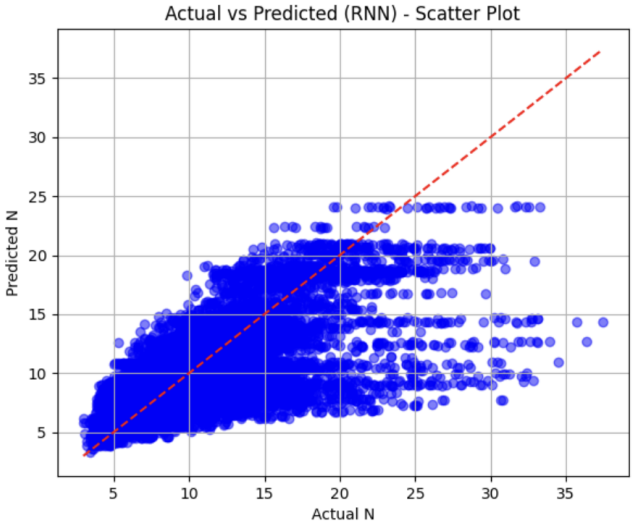

Figure 9. Scatter Plot RNN

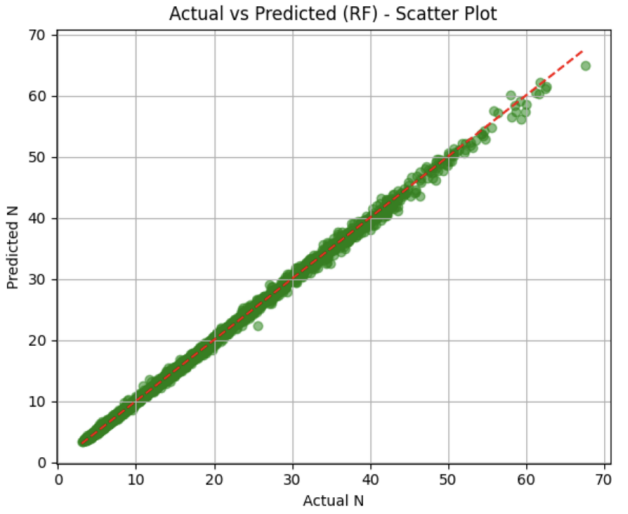

Figure 11. Scatter plot Random Forest

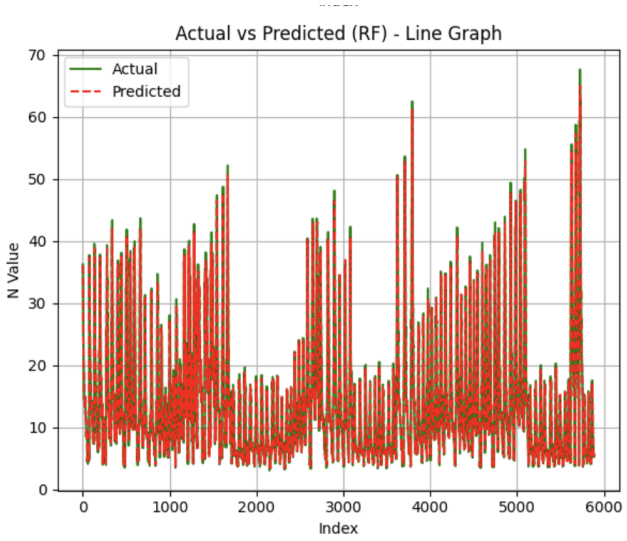

Figure 12. Line plot Random Forest

6. Final results

After performing hyperparameter optimization using Optuna, we obtained the best set of parameter for our RNN model:

- **First RNN Layer Units**: 1024
- **Second RNN Layer Units**: 256
- **Dense Layer Units**: 16
- **Learning Rate**: 0.0005
- **Batch Size**: 256

Using this optimized model, we predicted the values of N for an unseen dataset. To assess the model's performance, we calculated the Mean Squared Error (MSE), which was found to be 13.6. The graphs below illustrate the predicted values compared to the actual values, demonstrating how effectively the model generalizes to new data.

```
Number of rows in Original_N_Values.csv: 7672
Number of rows in Predicted_N_Values.csv: 7672

Model Evaluation Metrics:
Mean Squared Error (MSE): 13.6383
Mean Absolute Error (MAE): 2.5366

Sample Actual vs. Predicted Data:
   Original_N  Predicted_N
0    3.436139     4.777043
1    3.137743     4.777997
2    3.538980     4.764927
3    3.536896     4.925853
4    4.154081     4.925933
Comparison data saved to 'Comparison_N_Values.csv'
```

Figure 13. Prediction Evaluation

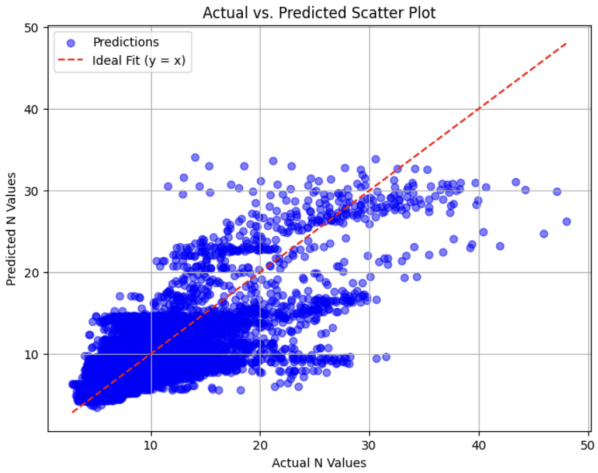

Figure 14. Scatter plot Actual vs Predicted values of N

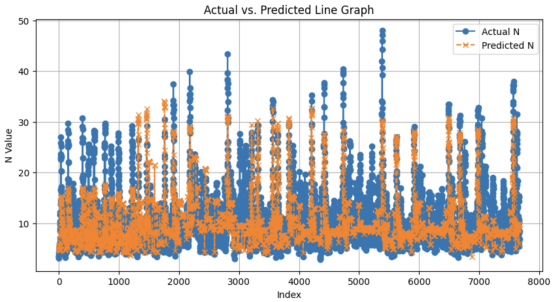

Figure 15. Line graph Actual vs Predicted values of N

7. Conclusion and Future Engineering

In this study, we explored various neural network architectures, including RNN, LSTM, and GRU, to predict N. However, due to the limited amount of available data, these complex networks struggled to effectively identify patterns, resulting in a high Mean Squared Error (MSE). Additionally, as observed in the feature engineering section 3, the data above the 10,000 Hz frequency range was sparse, which further impacted the performance of these models in predicting higher values.

To address this issue, we implemented a hybrid modeling approach, leveraging both RNN and Random Forest. While RNN was used for lower frequency ranges where sufficient data was available, Random Forest was employed for the higher frequency ranges to better capture patterns and reduce MSE. This hybrid approach ensures improved performance with the current dataset while maintaining flexibility for future improvements.

For future work, with the availability of a larger dataset, complex deep learning models such as RNN can be further refined to achieve superior predictive performance. Since

historical data influences current predictions, increasing the dataset size will enhance the learning capability of deep networks.

Additionally, integrating mathematical relationships between features and the target variable can further aid the model in understanding complex patterns, leading to improved generalization. Expanding the dataset and refining the feature engineering process will be key to enhancing model accuracy and robustness in real-world applications.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] Scikit-learn Developers, "sklearn.ensemble.IsolationForest," 2024. Accessed: Feb. 4, 2025.

[4] M. Steele, "Feature engineering examples: Binning numerical features," 2023. Accessed: 2025-02-03.

[5] Amazon Web Services, "What is a Neural Network?," 2024. Accessed: Feb. 4, 2025.

[6] T. D. Science, "Learn how recurrent neural networks work," 2023. Accessed: 2025-02-03.

[7] GeeksforGeeks, "Deep Learning - Introduction to Long Short-Term Memory," 2024. Accessed: Feb. 4, 2025.

[8] Towards Data Science, "Understanding GRU Networks," 2024. Accessed: Feb. 4, 2025.

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] Optuna Team, "Optuna: An Open-Source Hyperparameter Optimization Framework," 2024. Accessed: Feb. 4, 2025.