TIGER: A Novel Approach to Minimize Data Usage in Internet Speed Tests with RNNs

Ziv Weissman, Luke Yoffe, Justin Cao
https://github.com/lukeyoffe/cs293n

Abstract

In today's world, a reliable internet connection is increasingly important. Speed tests provide valuable insights to governments and companies about which parts of the population have poor internet connections so that investments can be made where they will have the largest impact. However, traditional speed tests are too data-intensive for individuals with poor network connections, who are often the most in need of accurate speed data.

Our goal is to develop a more cost-effective and efficient speed test. We achieve this by starting with a full-length speed test packet capture, truncating it to simulate a shorter, less resource-intensive test, and using a machine learning model to predict the speed test result. We introduce an innovative and efficient data representation for speed test packet captures, dividing packets into groups by timestamp and computing aggregated statistics for each group.

Using only half the duration of the speed tests, we can still accurately predict the results. Leveraging a larger dataset and a more refined model, our approach demonstrated significant improvements across key metrics. The enhanced model achieved a mean absolute error (MAE) of 20.51 Mbps and a median absolute error of 2.95 Mbps using only 7.5 seconds of the speed test.

Ultimately, this optimized approach reduces the cost of conducting speed tests, enabling broader data collection from underserved areas and helping to bridge the digital divide in our society.

1 Introduction

In today's interconnected world, a reliable internet connection is essential for various aspects of daily life, including education, business, healthcare, and entertainment. As internet usage continues to grow, ensuring equitable access to high-speed internet has become a priority for governments and companies. To achieve this, accurate and comprehensive data on internet speeds across different regions is necessary. Internet speed tests play a crucial role in providing this data, informing decisions on infrastructure investments and policy-making to bridge the digital divide.

However, traditional internet speed tests present a significant challenge: they consume substantial amounts of data, which can be prohibitive for individuals with already poor network connections. Ironically, these are the very individuals and regions where accurate speed test data is most urgently needed. The high data cost associated with conventional speed tests can deter participation from those with limited bandwidth, leading to an incomplete and potentially skewed understanding of internet performance in underserved areas.

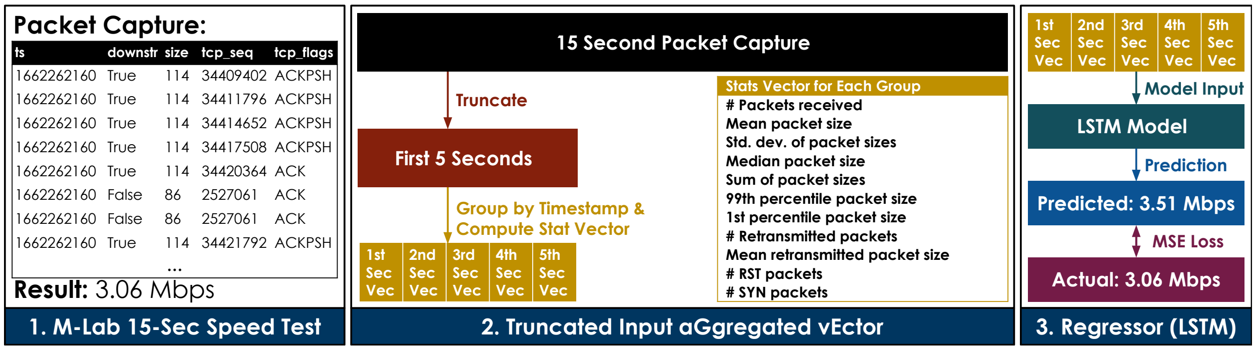
To address this issue, our approach aims to develop a more efficient and cost-effective method for conducting internet speed tests. By analyzing the elements of speed tests that are redundant, we seek to minimize the amount of data downloaded during a speed test without significantly sacrificing accuracy. Our primary focus is to determine the extent to which we can reduce the duration and data requirements of speed tests while maintaining reliable estimates of internet speed.

We propose leveraging machine learning techniques, specifically a recurrent neural network (RNN), to identify patterns early in the packet capture process. This approach will allow us to predict the final speed with reduced data, compared to conventional methods. The novelty of our research lies in its investigation of the feasibility and accuracy of predicting internet speed test outcomes using only a truncated sample of packet capture data.

Our innovative approach has the potential to revolutionize internet speed testing, particularly in environments where quick and frequent assessments are critical. It offers a path to more efficient diagnostics in network settings, allowing for rapid assessments while minimizing disruptions and resource usage. This method could be especially beneficial in rural and underserved areas, where bandwidth is limited, and the need for accurate speed data is greatest. By making speed tests more accessible and less resource-intensive, we hope to contribute to the broader goal of achieving equitable internet access for all.

In summary, our contributions are as follows:

- We introduce an innovative and efficient approach for representing speed test packet captures suitable for sequence processing models.


Packet Capture:

ts	downstr	size	tcp_seq	tcp_flags
1662262160	True	114	34409402	ACKPSH
1662262160	True	114	34411796	ACKPSH
1662262160	True	114	34414652	ACKPSH
1662262160	True	114	34417508	ACKPSH
1662262160	True	114	34420364	ACK
1662262160	False	86	2527061	ACK
1662262160	False	86	2527061	ACK
1662262160	True	114	34421792	ACKPSH
...				

Result: 3.06 Mbps

Figure 1: *The TIGER (Truncated Input aGgregated vEctor Regressor) pipeline consists of three stages: First, speed test packet captures are collected from M-Lab. Next, for each packet capture, we truncate the data, divide the packets into groups based on their timestamps, and compute statistics for each group. Finally, these statistical vectors are input into a LSTM to predict the speed test result using the truncated data.*

- We introduce a new metric, TIGER-Score, for evaluating speed test prediction models.
- We carry out comprehensive experiments, training multiple LSTM models with varying degrees of speed test truncation and different vectorization granularities.
- We show promising outcomes indicating that machine learning models can accurately predict speed test results using only a small portion of the initial data, which, if scaled, could lower the cost of speed tests and promote more equitable internet access.

2 Related Work

2.1 Data Representation

Various methods have been proposed for representing network data to facilitate more efficient and accurate analysis. FlowPic [8] formats data into images by stacking layers of features, which are then analyzed using Convolutional Neural Networks (CNNs) to identify flow characteristics. This visual representation leverages spatial patterns in the data.

NTT [1] performs data aggregation to uncover both fine and coarse patterns within network flows. By summarizing the data at different granularities, it captures a wide range of network behaviors. Similarly, ET-BERT [5] converts network flows into tokens or embeddings and applies Natural Language Processing (NLP) techniques for network classification and pattern formulation, treating network data similarly to text and leveraging NLP models' strengths in pattern recognition.

NetFound [2], an extension of ET-BERT, introduces a novel architecture with a skip connection, allowing the model to bypass certain layers. This facilitates the detection of coarser patterns, improving the model's flexibility and depth. Zapdos [7] converts network data into aggregated statistics before passing it into a model, such as a random forest, simplifying data representation and leveraging ensemble learning techniques for robust classification.

2.2 Long Short-Term Memory (LSTM) Networks

For our project, we have chosen to use Long Short-Term Memory (LSTM) [3] networks, a type of Recurrent Neural Network (RNN), due to their proven effectiveness in identifying patterns within sequential data. LSTMs are particularly well-suited for time series and sequence data because they can capture long-term dependencies and handle the vanishing gradient problem more effectively than standard RNNs. This makes them an ideal choice for analyzing packet captures and predicting internet speed outcomes with reduced data requirements.

Recurrent Neural Networks (RNNs) are a class of neural networks designed to recognize patterns in sequences of data, such as time series or natural language. Unlike traditional feedforward neural networks, RNNs have connections that form directed cycles, enabling them to maintain a memory of previous inputs in the sequence. This memory is represented through hidden states that are passed from one-time step to the next, allowing the network to capture temporal dependencies. However, standard RNNs face significant challenges, such as the vanishing or exploding gradient problem, where due to the way the gradients used for training the network are propagated back through time, the longer the input, the more the gradients tend to diminish or explode exponentially over time. This makes it difficult for the network to learn long-range dependencies and thus handle long input sequences effectively.

LSTM networks address these challenges through a more complex architecture that includes mechanisms to regulate the flow of information. Each LSTM cell contains three gates: the forget gate, the input gate, and the output gate.

The forget gate determines which information from the previous cell state should be discarded. It uses a sigmoid function to produce a value between 0 and 1 for each element in the cell state, where values closer to 0 imply forgetting the information and values closer to 1 imply retaining it.

The input gate decides which new information should be added to the cell state. It consists of two parts: the input gate layer, which uses a sigmoid function to determine which values will be updated, and a tanh layer that creates a vector of new candidate values that could be added to the state. The combination of these two results updates the cell state with new information.

The output gate determines what the next hidden state should be, incorporating the relevant parts of the cell state that will be output. The sigmoid layer decides which parts of the cell state to output, while the tanh layer scales the cell state to a value between -1 and 1.

These mechanisms allow LSTMs to effectively retain and utilize information over long sequences, making them particularly advantageous for our goal of simplifying speed tests while maintaining accuracy. Their ability to manage long-term dependencies ensures that the prediction model can effectively learn from truncated speed test data, thereby reducing the overall data requirements without sacrificing performance. This capability makes LSTMs a powerful tool for analyzing packet captures and predicting internet speed outcomes in a resource-efficient manner.

3 Method

3.1 Data Representation

Our primary objective is to train a machine learning model to predict speed test results using only a truncated portion of the test. The speed tests are represented as packet captures, containing data about each packet sent during the test. Previous approaches [1,2,5,8] have used models that take raw or lightly processed packet-level data, including header fields and sometimes the payload. These models must perform substantial processing to extract meaningful features and accomplish the prediction task, making them suitable for large, transformer-based models, which are computationally expensive and beyond our resources.

Inspired by Zapdos [7], we chose to perform manual feature engineering on the packet captures before feeding them into the model. This reduces the model's workload in feature extraction, allowing us to use a smaller, more efficient model and requiring less training data. The trade-off is that our engineered features might miss some useful information present in the raw packet captures. However, given our computational constraints, we deemed this trade-off acceptable.

Initially, we considered using FlowPic's [8] representation, which converts packet captures into 2D images with packet arrival timestamps on the x-axis and packet sizes on the y-axis. These images can be processed using image processing techniques like convolutional neural networks. However, we encountered two issues: the limited number of unique packet sizes in our captures resulted in sparse images, and FlowPic lacked other potentially useful features for speed test prediction, such as TCP flags or retransmission data.

Consequently, we developed our own data representation; a time series format seemed a natural fit for representing speed test packet captures, allowing the model to detect temporal patterns (e.g., periodic spikes in packet size indicating congestion). Our method groups packets based on their timestamps and computes a vector of aggregated statistics for each group, which is then fed into the model, as illustrated in Figure 1. For each group, we compute the statistics shown in Table 1 separately for upstream and downstream packets.

Features Computed For Gach Group
1. # Packets received
2. Mean packet size
3. Std. dev. of packet sizes
4. Median packet size
5. Sum of packet sizes
6. 99th percentile packet size
7. 1st percentile packet size
8. # Retransmitted packets
9. Mean retransmitted packet size
10. # RST packets
11. # SYN packets

Table 1: *List of features computed for each group, separately for upstream and downstream packets. Thus, each feature vector contains 22 elements.*

Before vectorizing the packet capture, we truncate the initial speed test to include only data from the beginning, simulating a shorter speed test. The hyperparameters for feature engineering are the truncated duration and the time delta (length of each time slice).

3.2 Model

After transforming each packet capture into a series of feature vectors, we divided the data into training (80%) and testing (20%) sets. We utilized the training set to train an LSTM model using PyTorch. The LSTM model comprised 3 layers with a hidden size of 64, though we increased this to 256 in a later experiment described in section 6. Following the LSTM layers, a linear layer was employed to convert the output into a scalar value, representing the predicted speed test result. We utilized the Adam optimizer [4] and optimized the model using mean squared error as the loss function.

4 Evaluation

4.1 Dataset

To assess our models' performance, we required a dataset of real-world internet speed tests. Measurement Lab's [6] Network Diagnostic Tool (NDT) provided the perfect solution. This publicly available tool allows users to measure their internet connection speed, and M-Lab generously publishes the results and network data (packet captures) from all these tests.

However, with millions of tests run daily, using the entire dataset was impractical. To ensure a manageable size while maintaining representativeness, we employed random sampling. We selected 100 days spread across 2022 and 2023. For each chosen day, we randomly sampled 30 individual speed tests, each lasting approximately 15 seconds (strictly within 0.05 seconds of 15 seconds total speed test). This resulted in a dataset of 3,000 speed tests. We address some of the constraints of using a small dataset in sub section 4.3 and section 6.

By leveraging randomly sampled real-world data, we aimed to achieve generalizability in our results. In simpler terms, the randomness ensures the data reflects a broad spectrum of internet connection experiences, increasing the likelihood that our models will perform well on unseen data.

4.2 Evaluation Metric

To evaluate our model's performance, we developed a novel metric called the TIGER-Score. This score combines the correlation coefficient and the slope of the least squares regression line to provide a comprehensive measure of the model's accuracy and reliability.

4.2.1 Components of the TIGER-Score

Correlation Coefficient (r): The correlation coefficient measures the strength and direction of the linear relationship between predicted and actual speed test results. A value of r close to 1 indicates a strong positive linear relationship, where predictions closely match actual results.

Slope of the Least Squares Regression Line: This line represents the best fit through the scatter plot of predicted versus actual values, passing through the origin. The slope indicates how well the predictions align with actual values:

- A slope of 1 implies perfect prediction accuracy.
- Deviations from a slope of 1 indicate errors, with values less than 1 suggesting underestimation and values greater than 1 suggesting overestimation.

4.2.2 TIGER-Score Formula

The TIGER-Score is computed using the following formula:

$$\text{TIGER-Score} = r \cdot e^{-|\log(\text{slope})|}$$

4.2.3 Explanation

- r: Rewards high linear correlation between predicted and actual results.
- $e^{-|\log(\text{slope})|}$: Penalizes the score based on how much the slope deviates from 1:
  - $\log(\text{slope})$: Measures the deviation of the slope on a logarithmic scale, treating underestimation and overestimation symmetrically.
  - $|\cdot|$: The absolute value ensures that the penalty is the same regardless of whether the slope is less than or greater than 1.
  - $e^{-|\log(\text{slope})|}$: Converts this deviation into a penalty factor between 0 and 1, with a perfect slope of 1 yielding no penalty (factor of 1).

4.2.4 Interpretation of the TIGER-Score

The TIGER-Score ranges between 0 and 1, with higher values indicating better model performance. A score of 1 signifies perfect correlation and accuracy, where predictions perfectly match actual values. The score effectively balances the need for strong linear correlation (r) and accurate scaling of predictions (slope), providing a robust measure of performance.

4.2.5 Example Calculation

To illustrate, consider a model with a correlation coefficient $r = 0.95$ and a slope of the regression line equal to 0.8. The TIGER-Score would be calculated as follows:

1. Calculate the deviation penalty:

$$|\log(0.8)| \approx |-0.097| = 0.097$$

$$e^{-0.097} \approx 0.91$$

2. Compute the TIGER-Score:

$$\text{TIGER-Score} = 0.95 \cdot 0.91 \approx 0.865$$

This score indicates that the model performs well but has some deviation from perfect prediction accuracy.

By detailing the components and calculation of the TIGER-Score, we provide a clear and interpretable metric that highlights both the accuracy and reliability of our model's predictions. This approach ensures that the model's performance is evaluated comprehensively, accounting for both correlation and scaling accuracy.

4.3 Metrics Excluding Top 2% of Errors

Due to the limited size of our dataset and the constraints on model complexity, our model encountered difficulties accurately predicting certain speed tests in the test dataset. These inaccuracies resulted in significant errors that disproportionately affected the overall evaluation metrics, potentially providing a skewed representation of the model's performance.

To mitigate the impact of these outliers and to provide a more representative evaluation, we recalculated the metrics after excluding the top 2% of the largest errors. This approach allows us to simulate the potential improvements in model performance that could be achieved with access to a larger dataset and a more complex model architecture.

4.3.1 Rationale for Excluding Outliers

Outliers can disproportionately influence evaluation metrics, leading to an overestimation of the model's error rate. By removing these extreme cases, we can gain a clearer understanding of the model's general performance. This approach is particularly relevant in our context, where the dataset size and model complexity are constrained, and a small number of large errors can significantly distort the overall metrics.

4.3.2 Methodology

To identify and exclude the top 2% of errors, we first calculated the absolute prediction errors for all test samples. We then sorted these errors in descending order and excluded the top 2% from the recalculation of our evaluation metrics, including the TIGER-Score. This exclusion process ensures that the remaining 98% of the data provides a more accurate reflection of the model's typical performance.

4.3.3 Adjusted Metrics

After excluding the top 2% of errors, we recalculated the following metrics:

- **Mean Absolute Error (MAE):** The average of the absolute errors between predicted and actual values.
- **Mean Squared Error (MSE):** The average of the squared errors between predicted and actual values.
- **Correlation Coefficient (r):** The measure of the strength and direction of the linear relationship between predicted and actual values.
- **Slope:** The slope of the regression line between the predicted and actual values that goes through the origin.
- **TIGER-Score:** Our custom metric that combines the correlation coefficient and the slope of the least squares regression line.

4.3.4 Discussion

Excluding the top 2% of errors provides a more optimistic yet realistic assessment of the model's performance under typical conditions. While this adjustment improves the evaluation metrics, it is important to acknowledge the presence of outliers and their potential causes. These outliers may be attributed to various factors, including anomalies in the data, limitations in the current model architecture, or inherent variability in internet speed tests.

Our analysis indicates that with access to more extensive datasets and the development of more sophisticated models, it may be possible to reduce the occurrence of such significant errors. Therefore, the adjusted metrics serve as an estimate of the potential improvements achievable through future enhancements in data collection and model development.

By presenting both the original and adjusted metrics, we provide a comprehensive evaluation of the model's performance, highlighting areas for improvement and setting a benchmark for future research.

5 Results

Due to our limited computational resources, we aimed to gain insights into how varying the total time and delta (time interval) would affect the performance of our model. We hypothesized that increasing the total time would improve the model's performance by providing more comprehensive information about the network conditions. Conversely, we predicted that decreasing the delta would initially enhance model performance by offering more granular data, but that there would be a point of diminishing returns where the model might overfit the training data and underperform on the testing data.

5.1 Delta Experimentation Results

To evaluate the impact of different delta values on the model's performance, we conducted experiments with delta values of 0.01, 0.05, 0.2, 1.0, and 5.0 seconds. The results of these experiments are summarized in Figure 2, which shows the 98th percentile TIGER Score for each delta value, with a total time segment of 10 seconds.

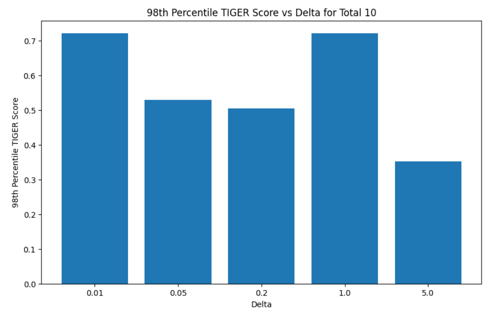


Figure 2: *98th Percentile TIGER Score vs Delta for Total 10 seconds.*

Findings: Our experimentation revealed several key insights:

- **Decreasing Delta Improves Performance:** As the delta decreases, the model's performance generally improves, as indicated by higher TIGER Scores for smaller delta values. This confirms our hypothesis that the finer granularity of data enhances model performance.

- **Performance Peaks at Small Delta Values:** The delta value of 0.01 seconds yielded the highest 98th percentile TIGER Score, suggesting that very small deltas provide the most detailed and useful information for the model.
- **Memory and Computational Constraints:** Despite the improved performance with a delta of 0.01 seconds, this value was very memory-intensive and significantly slowed down the model. This indicates practical limitations in using extremely small deltas without adequate computational resources.
- **Inconsistencies in Trends:** Some variability in performance trends across different delta values was observed, which we suspect is due to not experimenting with model complexity hyperparameters. This suggests that certain hyperparameters might be particularly well-suited for specific delta values.

Implications: The results demonstrate the importance of selecting an appropriate delta value for optimizing model performance. While smaller deltas generally lead to better performance, they also impose higher computational and memory demands. Therefore, there is a trade-off between performance and resource usage. Additionally, the observed inconsistencies highlight the potential for further optimization through hyperparameter tuning. Future research should investigate a wider array of hyperparameters, by experimenting with adjustments to learning rate, hidden size, number of layers, and loss function, to determine the optimal configurations for various delta and total time combinations.

In summary, our findings indicate that careful selection of delta values is crucial for balancing model performance and computational efficiency. These insights guide future experimentation and optimization efforts to improve the accuracy and resource efficiency of internet speed test predictions (detailed in section 6).

5.2 Total Time Experimentation Results

Given the results from our delta experiments, which indicated that smaller deltas generally perform better, we conducted further experiments to evaluate the effect of varying total time segments on model performance. We controlled the delta to 0.01 seconds and 0.05 seconds and compared the results across different total time segments. Figure 3 illustrates the 98th percentile TIGER Score for total times of 1, 2.5, 5, 7.5, and 10 seconds.

Findings: Our experimentation revealed several key insights:

- **Performance for Smaller Total Times:** For total times of 1, 2.5, and 5 seconds, a delta of 0.05 seconds surprisingly outperformed a delta of 0.01 seconds. This may seem counterintuitive, as we expected the smaller delta to consistently provide better performance due to higher data granularity.
- **Performance for Larger Total Times:** For total times of 7.5 and 10 seconds, the delta of 0.01 seconds outperformed the delta of 0.05 seconds, aligning with our initial hypothesis that smaller deltas should provide better performance.

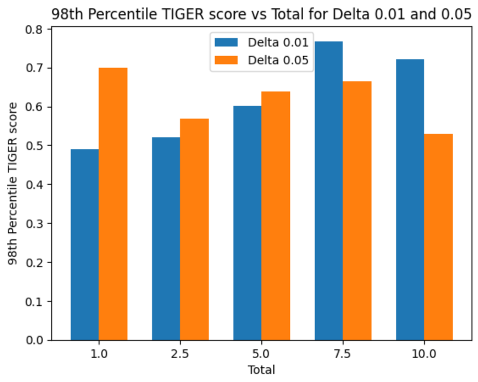


Figure 3: *98th Percentile TIGER Score vs Total for Delta 0.01 and 0.05 seconds.*

Explanation and Implications: The observed performance discrepancy for smaller total times could be attributed to several factors:

- **Randomness and Small Dataset Size:** The variance in performance might be due to the inherent randomness in the dataset and the limited size of our sample, which may not fully capture the complexity of real-world internet speed tests.
- **Hyperparameter Optimization:** The current experiments did not extensively fine-tune hyperparameters, which could mean that certain configurations were particularly well-suited to specific delta and total time combinations. Future work should include a comprehensive hyperparameter search to identify optimal settings.

Despite these anomalies, the model with a total time of 7.5 seconds and a delta of 0.01 seconds consistently performed the best across our metrics. This configuration provides a balanced approach, leveraging sufficient data granularity and total time to achieve accurate predictions. Consequently, we decided to investigate this model in greater depth and consider scaling it up for further experimentation and potential real-world applications.

In conclusion, these findings underscore the importance of carefully selecting both delta and total time segments to optimize model performance. The results also highlight the need for further experimentation with larger datasets and more extensive hyperparameter tuning to fully understand and exploit the potential of our approach.

5.3 Analysis of Best Performing Model

The top-performing model was trained on truncated input aggregated vectors from the first 7.5 seconds of a 15-second speed test, using a time series delta of 0.01 seconds. Figure 4 shows a plot of the predicted versus actual speed test values for this model on the test portion of our dataset.

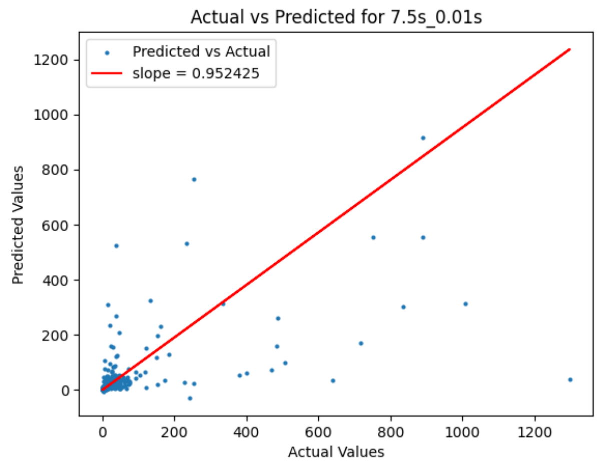


Figure 4: *Plot of predicted vs actual speed test results for the for the standard model trained on the first 7.5 seconds of a speed test with a 0.01 second time delta*

As shown, most speed test results are under 100 Mbps, with a few exceeding 1000 Mbps, reflecting the heavy-tailed distribution typical of network data. The model performs well for speed tests under 100 Mbps but struggles with outliers due to limited exposure during training. Therefore, in Table 2, we present metrics calculated using all test data points and metrics excluding the top 2% of errors (those above 326.56 Mbps), as detailed in subsection 4.3. Interestingly, excluding the top 2% of errors worsens the slope (0.800 vs 0.952), likely due to the outliers improving the slope by random chance. However, the rest of the metrics improve when excluding these extreme errors.

Metric	Value	W/out top 2%
Slope	**0.952**	0.800
Correlation Coefficient	0.607	**0.846**
TIGER-Score	0.594	**0.768**
Mean Absolute Error	23.601	**13.131**
Median Absolute Error	2.367	-
98th Percentile Abs. Err.	326.56	-

Table 2: *Metrics (with and without the top 2% of errors included in calculations) for the standard model trained on the first 7.5 seconds of a speed test with a 0.01 second time delta. Best in **bold**. Error is measured in megabits per second (Mbps).*

6 Enhanced Model Performance

In this section, we compare the performance of our initial and enhanced models to highlight the improvements from leveraging a larger dataset and refined architecture.

6.1 Introduction to the Enhanced Model

To improve the performance of our internet speed test prediction model, we utilized a more extensive dataset and a refined model architecture. The dataset comprised 22,000 speed tests sampled over 500 random days between 2021 and 2024. The LSTM model's hidden size was increased to 256 while maintaining three hidden layers. This section outlines the upscaling process and highlights the model's performance improvements.

6.2 Data Preparation and Aggregation

The new dataset was aggregated and preprocessed following our initial methodology. Each speed test's packet captures were truncated to the first 7.5 seconds with a delta of 0.01 seconds. We computed various statistical features for each group of packets, ensuring comprehensive temporal data for the model to learn from.

6.3 Model Architecture and Training

The enhanced LSTM model includes three hidden layers, each with a hidden size of 256. This configuration captures more complex patterns in the data, enhancing predictive capabilities. The model was trained for 750 epochs using the Adam optimizer and mean squared error (MSE) as the loss function. Training was conducted on high-performance computing resources to manage increased computational demands.

6.4 Evaluation Metrics

We assessed the enhanced model using several key metrics to comprehensively evaluate its performance. These metrics were compared to those of the previously described "standard" model (smaller model, less training data), both trained for a total of 7.5 seconds with 0.01-second deltas. The results, shown in Table 3, indicate that the enhanced model outperformed the standard model in all but one metric, demonstrating that a larger model trained on more data can generalize better.

6.5 Performance Analysis

The enhanced model demonstrated improvements across key metrics, though the degree of improvement varies:

Correlation Coefficient (r) The enhanced model's correlation coefficient increased from 0.6067 to 0.7101, indicating a stronger linear relationship between predicted and actual values. This improvement reflects better predictive reliability.

TIGER-Score With the TIGER-Score improving from 0.5940 to 0.6895, the model shows enhanced robustness

Metric	Enhanced	Standard
Correlation Coefficient (r)	**0.710**	0.607
TIGER-Score	**0.690**	0.594
Mean Absolute Error	**20.51**	23.60
Median Absolute Error	2.95	**2.37**
98th Percentile Abs. Error	**289.60**	326.56
Mean Squared Error	**6688.68**	8479.32

Table 3: *Comparison of metrics between enhanced and standard models trained on the first 7.5 seconds of a speed test with a time delta of 0.01 seconds. Best in **bold**. Error is measured in megabits per second (Mbps).*

and accuracy, combining higher correlation with accurate scaling of predictions.

Mean Absolute Error (MAE) and Median Absolute Error The MAE decreased from 23.60 Mbps to 20.51 Mbps, suggesting better predictive accuracy. The median absolute error increased slightly from 2.37 Mbps to 2.95 Mbps, indicating more typical prediction errors are still relatively low.

98th Percentile Absolute Error The 98th percentile absolute error decreased from 326.56 Mbps to 289.60 Mbps, showing improved handling of extreme cases. The higher correlation coefficient (98th percentile) and TIGER-Score (98th percentile) further indicate enhanced robustness against outliers.

Mean Squared Error (MSE) The MSE improved from 8479.32 to 6688.68, indicating the model's better ability to minimize larger errors.

7 Discussion and Limitations

The experiments have several limitations due to computational constraints. Firstly, the dataset size was limited. This could potentially restrict the model's ability to capture the full complexity of the problem. Secondly, a smaller, non-foundational model architecture was employed. While this choice reduced the cost of the training process, it might have limited the model's capacity to learn intricate relationships within the data. In section 6, we showed how a larger model trained on more data can perform significantly better, and we believe this trend may continue with even larger models and datasets. Additionally, an exhaustive hyperparameter search for optimal learning rates, hidden dimensions, etc., was not conducted due to computational limitations. This could have potentially led to sub-optimal model performance. Finally, only the Mean Squared Error (MSE) loss function was used to train the model. Exploring alternative loss functions might have yielded further improvements.

Another limitation is that the features that were chosen were not investigated heavily and were selected because they either carried seemingly useful data regarding the packet flow or were readily available, without thorough validation of their actual impact on the model's predictions.

As a future step, investigating feature importance would provide valuable insights into the factors that most significantly influence the model's predictions. By identifying key features, we can better understand how the model makes its decisions, which can enhance the interpretability and trustworthiness of the model. This understanding can also help in optimizing the model by focusing on the most impactful features, potentially improving performance and reducing computational complexity by eliminating less relevant data.

Moreover, analyzing feature importance will allow us to detect any shortcuts or spurious correlations between the features. This can help ensure that the model does not rely on misleading patterns that do not genuinely represent the underlying data. In specific applications, such as predicting internet speeds from packet captures, understanding which features are most critical can lead to practical improvements in network optimization and diagnostic tools. Overall, this approach ensures that our models are not only effective and efficient but also robust and reliable.

8 Conclusion

The primary objective of our project was to reduce the resource requirements for conducting internet speed tests. We approached this by feeding truncated speed test data into a Long Short-Term Memory (LSTM) network, which was tasked with predicting the final speed test result. Rather than inputting raw network data directly into the model, we preprocess the packet captures by dividing the packets into time-series buckets. For each bucket, we compute aggregated statistics for both upstream and downstream packets, such as the number of packets, mean packet size, median packet size, and additional metrics, resulting in 22 features in total. These feature vectors are then fed into the LSTM to predict the speed test result.

Our findings indicate that using only the first 7.5 seconds of a 15-second long speed test, we can achieve strong predictive performance. The enhanced model, trained on a larger dataset, demonstrated robust predictive accuracy and reliability. However, the model still shows some weaknesses in handling outliers and extreme cases, indicating areas for further refinement.

With access to more extensive datasets and the development of larger models, we anticipate further performance enhancements, making this approach viable for production use. This optimization would significantly reduce the cost of conducting speed tests, addressing current gaps in speed test data coverage, and ultimately contributing to providing more people with access to reliable internet connections, thereby bridging the digital divide in our society.

References

[1] Dietmüller, A., Ray, S., Jacob, R., and Vanbever, L. A new hope for network model generalization. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks* (November 2022), HotNets '22, ACM. (Cited on pages 2 and 3.)

[2] Guthula, S., Battula, N., Beltiukov, R., Guo, W., and Gupta, A. netfound: Foundation model for network security, 2023. (Cited on pages 2 and 3.)

[3] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780. (Cited on page 2.)

[4] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization, 2017. (Cited on page 4.)

[5] Lin, X., Xiong, G., Gou, G., Li, Z., Shi, J., and Yu, J. Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In *Proceedings of the ACM Web Conference 2022* (April 2022), WWW '22, ACM. (Cited on pages 2 and 3.)

[6] Measurement Lab. The M-Lab NDT data set. https://measurementlab.net/tests/ndt, (2009-02-11 – 2015-12-21). Bigquery table `measurement-lab.ndt.download`. (Cited on page 4.)

[7] Misa, C., Durairajan, R., Gupta, A., Rejaie, R., and Willinger, W. Leveraging prefix structure to detect volumetric ddos attack signatures with programmable switches. In *2024 IEEE Symposium on Security and Privacy (SP)* (Los Alamitos, CA, USA, may 2024), IEEE Computer Society, pp. 205–205. (Cited on pages 2 and 3.)

[8] Shapira, T., and Shavitt, Y. Flowpic: Encrypted internet traffic classification is as easy as image recognition. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2019), pp. 680–687. (Cited on pages 2 and 3.)