Development of a WebRTC-Based Video Calling Application to Predict Quality of Service Patterns Using an Artificial Neural Network

Carlos Moreno, Ezequiel Frias
Communications Department
Central University of Venezuela
Caracas, Venezuela
email: carlos.m.moreno@ucv.ve,
ezequieljfrias20@gmail.com

Vinod Kumar Verma
Computer Science and Engineering
Sant Longowal Institute of
Engineering and Technology
Punjab, India
email: vinod5881@gmail.com

Eric Gamess
MCIS Department
Jacksonville State University
Jacksonville, Alabama, USA
email: egamess@jsu.edu

***Abstract*—The development of video-calling applications using Web Real-Time Communication (WebRTC) represents an efficient and modern solution for real-time communications, enabling the direct transmission of audio, video, and data between browsers with no need for additional plugins. This research aimed to design and develop a WebRTC-based video-calling application capable of predicting Quality of Service (QoS) patterns through the implementation of an Artificial Neural Network (ANN). The proposal focused on analyzing key indicators (e.g., latency, jitter, and packet loss) that play a critical role in shaping user-perceived quality. The development of the predictive model was performed by using a Recurrent Neural Network (RNN) of the Long Short-Term Memory (LSTM) type. To validate the solution, four representative scenarios were established: acceptable quality, moderate degradation, critical quality, and extreme conditions. The results demonstrated that the LSTM model successfully captured the temporal behavior of QoS metrics and generated predictions within acceptable ranges according to standards defined by specialized organizations and industry leaders. It is concluded that the integration of LSTM neural networks into WebRTC applications constitutes a viable and effective strategy to enhance proactive QoS management and optimize the end-user experience.**

Keywords-Quality of Service; WebRTC; Video-Calling; Neural Networks; Prediction.

I. INTRODUCTION

Web Real-Time Communication [1]-[6] (WebRTC) is a set of open-source emerging technologies and APIs that enable real-time, peer-to-peer communications (audio, video, and data) directly between web browsers and mobile applications. It does not require intermediaries, plugins, or external software, making it a cornerstone of modern, decentralized communication systems. Due to its low latency, WebRTC has permitted the development of many new applications, revolutionizing how people interact online. It is now present in the majority of video conferencing systems (e.g., Google Meet), live streaming platforms, VoIP services, collaborative workspaces, online education platforms, file sharing, and multiplayer gaming. The use of WebRTC in browser-to-browser applications is expanding significantly as demand for real-time communication on the web grows, due to its standardized APIs [1] (e.g., `getUserMedia`, `RTCPeerConnection`, and `RTCDataChannel`), versatility, cross-platform compatibility, mandatory encryption for all media and data, and native integration on modern web browsers (e.g., Chrome, Firefox, Edge, and Safari).

This work proposes to develop an intuitive user interface for a WebRTC-based video call application and to analyze Quality of Service (QoS) parameters extracted from packets collected. In addition, the study seeks to design and implement a neural network model capable of predicting QoS patterns based on collected data, followed by a rigorous evaluation of its predictive performance. By combining user interface development, protocol-level traffic analysis, and advanced deep learning techniques, this research provides a systematic framework for addressing QoS prediction in real-time communication systems. The proposed approach intends to enhance both the accuracy and reliability of service quality estimation, thereby contributing to the optimization of WebRTC-based video call applications.

Recurrent Neural Networks (RNNs) of type Long Short-Term Memory (LSTM) with multiple outputs were used in this work since they are designed to handle sequential or time-series data. Unlike traditional networks, RNNs have an internal memory allowing them to use information from previous inputs to influence current outputs. Multiple outputs are used because the QoS output variables are correlated.

The rest of this paper is organized as follows. Section II discusses several peer-reviewed literature works conducted within this area of research and the problem addressed in this work. Section III describes the methodology employed, while Section IV presents and analyzes the results. Finally, Section V concludes the paper and discusses possible future work.

II. RELATED WORK

The study of real-time communication systems and QoS prediction has been widely addressed in the last two decades. Several studies have investigated the likelihood of network underperformance, anomalies, and failures, as well as the possibility of improving the QoS by applying artificial intelligence techniques.

Since WebRTC is an emerging technology, it is not considered in most of the work done in this area so far. For example, the study in [7] performed anomaly detections in network traffic using different models such as Isolation Forest, Naïve Bayes, XGBoost, LightGBM, and SVM classification. The results revealed that some of these models

exhibit impressive performance and accuracy, highlighting the strengths and limitations of each one. The authors suggested integrating deep learning techniques, such as convolutional and RNNs. Another significant contribution in the area came from Garcia and Salcedo [8], who developed a model for failure prediction in IP networks using Artificial Neural Networks (ANNs). The study focused on detecting LAN failures, such as timeouts and connection rejections, demonstrating that ANNs can significantly improve the accuracy of fault diagnosis. In [9], the authors proposed a QoS prediction model called Topology-Aware QoS-GRNN (TAQ-GRNN), which incorporates gated RNNs of LSTM type. Even if their model could be integrated into WebRTC, the authors did not consider this possibility. The authors of [10] chose six specific QoS/QoE metrics and extracted the associated values from a VoIP measurement campaign in an LTE-A environment, before employing a set of recurrent neural networks (simple RNN, LSTM, and GRU) to predict the behavior of the selected QoS/QoE metrics. Aziz, Ioannou, Lestas, Qureshi, Iqbal, and Vassiliou [11] proposed a prediction model for QoS by using an RNN to integrate a Bidirectional Long Short-Term Memory (BLSTM). It can predict the QoS-aware network traffic for over 13 hours with high accuracy. They compared the RNN-BLSTM with other algorithms (i.e., LSTM, ARIMA, SVM). Their architecture is suitable for 5G and 6G mobile networks. The work in [12] did another relevant investigation within the field of QoS and Deep Learning, with the classification of multimedia traffic by using Convolutional Neural Networks. The authors of [13] developed a model for Service QoS prediction based on feature Mapping and Inference. In [14], Gerard, Bonilla, Bentaleb, and Céspedes proposed a Machine Learning (ML) model to enhance Forward Error Correction (FEC) efficiency. According to their findings, it corrects up to 60% of errors and achieves 2.5 times better energy efficiency than standard WebRTC.

Some work has been done in the area with the use of WebRTC. For example, Google [15] has deployed ML-based Bandwidth Estimation (BWE) systems within WebRTC that utilize a combination of LSTM and dense layer architecture to process real-time statistics (e.g., RTT and packet loss). This architecture enables superior proactive congestion prediction, significantly reducing parameters such as video freezes and connection drop rates. Sakakibara, Ohzahata, and Yamamoto [16] validated the creation of highly accurate No-Reference (NR) Quality of Experience (QoE) models solely based on WebRTC client statistics (jitter and bandwidth). Their models offer computationally efficient Deep Neural Networks (DNNs) or Temporal Convolutional Networks (TCNs) suitable for client-side monitoring. A doctoral thesis from Bingol [17] studied the convergence of AI techniques and WebRTC to predict QoE indicators, as they are more representative of user satisfaction than QoS.

This work differs from other state-of-the-art approaches in several key aspects. First, our architecture is proposed to predict QoS in interactive video calls specifically, and not for streaming or other applications. Second, we initially establish a robust comparative methodology by evaluating four distinct RNNs (GRU single output, GRU multiple outputs, LSTM single output, and LSTM multiple outputs) against three crucial performance indicators (Mean Absolute Error, Mean Squared Error, and Root Mean Squared Error) to select the optimal model for implementation. Third, both the training and the subsequent operational deployment of the application rely exclusively on real-world measurements captured under diverse and varying network congestion conditions. Fourth, by utilizing a NoSQL Firebase Firestore [18] database for WebRTC metrics, this architecture provides superior scalability and high throughput with optimized performance and low latency.

Given all the aspects discussed previously, in the state-of-the-art, it is evident that in networks, QoS has become a crucial aspect to ensure an optimal user experience. This implies that the services and applications in use operate constantly. To achieve the best quality, it is necessary to invest in high-quality network infrastructure and carry out network monitoring. However, it is also important to develop applications with advanced capabilities that enable the prediction of QoS patterns. These applications might include artificial neural networks.

Based on the findings presented in the state of the art, two research questions arise:

- Question 1: Which QoS metrics can be considered to measure, analyze, and predict QoS patterns?
- Question 2: Which specific type of neural network predicts better QoS in a WebRTC-based video call?

III. METHODOLOGY

In this section, the implemented methodology is described, which includes the definition and characterization of the four study scenarios and the three indicators, the development of the WebRTC-based application and its final recurrent neural network architecture, after evaluating four possible alternatives, as well as the operation of the predictive model.

A. Establishment of Scenarios and Quality of Service Parameters

In this subsection, the QoS parameters considered in WebRTC were identified and defined, establishing criteria and metrics for the evaluation. The parameters selected were (1) latency, (2) jitter, and (3) packet loss rate. Four scenarios were chosen according to Rec. ITU-T G.1010 [19] as specified in Table I.

TABLE I. SCENARIOS SELECTED FOR STUDY

Scenario	Bandwidth	Latency	PLR	Description
(1) Acceptable Quality	50 Mbps	20 ms	0%	Ideal
(2) Moderate Degradation	2 Mbps	100 ms	3%	Congestion
(3) Critical Quality	0.8 Mbps	200 ms	10%	Deficient
(4) Extreme Conditions	0.3 Mbps	500 ms	20%	Degraded

B. Development of the User Interface

The user interface was developed using JavaScript along with the React framework, which allowed for the creation of a dynamic, modular, and scalable web application. For the implementation of real-time video calls, the PeerJS [20]

library was used. PeerJS is a JavaScript library built on top of WebRTC that simplifies Peer-to-Peer (P2P) data, audio, and video communication in web browsers.

C. Extraction, Data Processing, and Pattern Analysis

For the collection of real-time metrics related to QoS during video calls, the `getStats` [21] function provided by the WebRTC API was used. In order to evaluate the application's performance across various connectivity contexts, the Network Link Conditioner tool [22], available on macOS, was used.

Figure 1 depicts the testbed for measurements. The client with the Network Link Conditioner tool is connected to the Internet via the SimpleFibra provider, using a 400 Mbps fiber optic WAN access link. Internally, the WLAN connection is established through a Wi-Fi 5 (IEEE 802.11ac) network, operating on the 5 GHz band, channel 161, with an 80 MHz channel width. On the other hand, the remote client is connected to the Internet via the NetUno provider, also through a fiber optic link, with a bandwidth of 200 Mbps. In its WLAN, Wi-Fi 5 is also used on the 5 GHz band, channel 153, with an 80 MHz channel width.

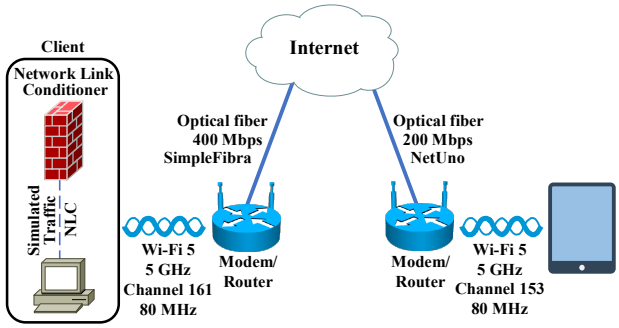

Figure 1. Testbed for Measurements

A mechanism was implemented to request a statistical report at 2-second intervals, in order to capture sudden variations in connection quality that might be overlooked with a longer interval. Then, based on the data obtained from each sample, structured JSON objects were built. Metrics collected for the object were: (1) timestamp, (2) jitterVideo, (3) jitterAudio, (4) roundTripTimeVideo, (5) roundTripTimeAudio, (6) packetsLostVideo, (7) PacketsLostAudio, (8) PacketsReceivedVideo, and (9) PacketsReceivedAudio. Using Formulas 1 and 2, the delay and packet loss rate were computed from the collected values.

$$delay = \frac{RoundTripTime}{2} \quad (1)$$

$$PacketLossRate = \frac{packetsLost}{packetsReceived + packetsLost} \times 100 \quad (2)$$

After constructing the metrics object, data were transmitted and stored in a Firebase Firestore [18] database (a cloud-based NoSQL database). To normalize the selected variables, the Python `MinMaxScaler` method from the `sklearn.preprocessing` [23] library was used.

D. Development of the Neural Network

For the analysis of patterns and the prediction of network conditions based on the collected metrics, it was decided to implement an RNN formed by four layers: one RNN input layer (receiving the six QoS metric values), one RNN layer (cell), one dense layer, and one output reshape that re-dimensioned the dense layer (outputting the six predicted QoS metric values). For the purpose of identifying the most suitable neural network model for predicting the QoS metrics, four experimental configurations were designed and evaluated. These configurations are based on the recurrent cell (first 2 layers): Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM). For each type of architecture, two output approaches were explored: one focused on predicting a single variable at a time (single output) and another capable of estimating multiple metrics simultaneously (multiple outputs). The four experimental models evaluated were: recurrent cell GRU single output (GRU-1), recurrent cell GRU multiple outputs (GRU-M), recurrent cell LSTM single output (LSTM-1), and recurrent cell LSTM multiple outputs (LSTM-M). The hyperparameters selected are shown in Table II.

TABLE II. HYPERPARAMETERS PER NEURAL NETWORK MODELS

Hyperparameter	GRU-1	GRU-M	LSTM-1	LSTM-M
Output (steps)	1	30	1	30
LSTM Layers	2	2	2	2
Neurons per Layer	128	128	128	128
Optimizer	Adam	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.001	0.001
Epochs	12	11	32	25
Batch Size	16	16	16	16

The training of the four configurations was conducted using a general single dataset comprising values obtained under the four network conditions defined in Table I, during one hour.

The initial 80% of this general dataset was used exclusively for model training (training set), while the remaining 20% (corresponding to the most recent data) was reserved for testing (testing set). Each model was trained individually, respecting its specific architecture. During the training process, the `EarlyStopping` technique was applied. In each scenario, the model that achieved the best performance during validation was saved, in order to be formally evaluated later on another test set.

E. Evaluation of the Neural Network

To compare the performance of the different models, three evaluation metrics were defined and applied to the test set: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Single-output models (for both GRU and LSTM) achieved lower error metrics. For example, LSTM achieved errors of 0.0601 and 0.0994 for MAE and RMSE, respectively, demonstrating remarkable accuracy in predicting the next immediate point. However, these models presented significant limitations for long-term predictions, such as high computational inefficiency, cumulative error propagation, and

underutilization of temporal relationships. In contrast, multiple-step (multiple-output) models were designed to address these challenges. Although their error metrics were slightly higher on average, the multi-output LSTM (MAE=0.1205, RMSE=0.2044) showed better capability for predicting extended series in a stable and coherent manner, mitigating the negative effects of error accumulation, and reducing computational cost per inference. Finally, the LSTM-M architecture was selected, consisting of one input layer (input 60 time steps and 6 features), one hidden LSTM layer (output 128 nodes), one dense layer (output 180 nodes), and one output reshape layer (output 30 time steps and 6 features), to perform predictions across the four scenarios (see Table I) without requiring retraining.

F. Measurements and Prediction of QoS per Scenario

Each call generated approximately 150 sets of samples (one every 2 seconds), capturing the following QoS parameters: audio and video jitter, audio and video round-trip time, packet loss rate, and number of packets received per channel. The model started operating from the first minute of the call, as sufficient data history was available at that point. The prediction model operated with a sliding window of historical values (60 steps) and predicted values (30 steps ahead).

IV. RESULTS AND ANALYSIS

The results of the four scenarios studied with the selected neural network are presented in the following sections, using embedded Python code within the general application built with PeerJS for WebRTC.

A. Acceptable Quality

Figure 2 shows that the model successfully estimated the video latency, closely following the actual signal trend. No significant offsets or error accumulation were observed, demonstrating the model's ability to adapt to stable network conditions.

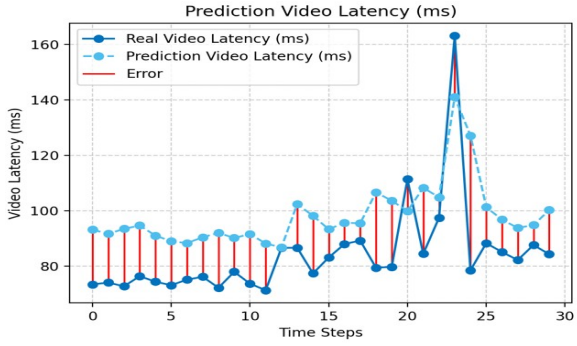

Figure 2. Actual Test Data vs. Model Prediction for Video Latency in Last-Minute Scenario 1

Figure 3 illustrates that the audio latency predictions exhibited a high level of agreement with the actual data. The model was able to maintain the trend without notable deviations, validating its ability to model this metric properly in low-variability environments.

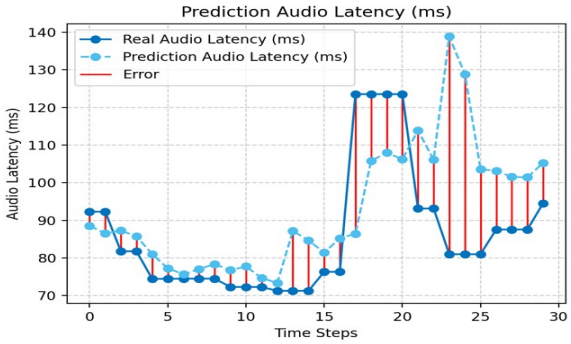

Figure 3. Actual Test Data vs. Model Prediction for Audio Latency in Last-Minute Scenario 1

Figure 4 shows that, although the model accurately predicted most video jitter values, an outlier was detected near 500 ms, indicating an anomaly in an otherwise stable network.

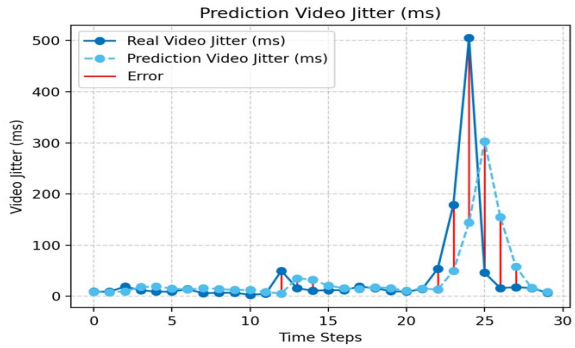

Figure 4. Actual Test Data vs. Model Prediction for Video Jitter in Last-Minute Scenario 1

As depicted in Figure 5, the audio jitter was predicted with minimal errors, showing highly stable behavior. This reinforces the idea that under ideal conditions, the model was capable of accurately capturing slight fluctuations in audio quality.

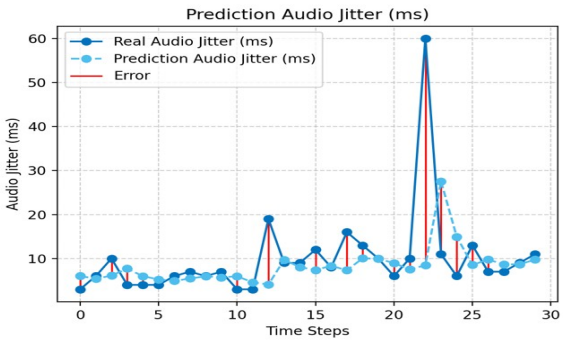

Figure 5. Actual Test Data vs. Model Prediction for Audio Jitter in Last-Minute Scenario 1

In Figure 6, it can be noted that the video packet loss rate was practically zero throughout the whole experiment, with the model predicting values close to zero.

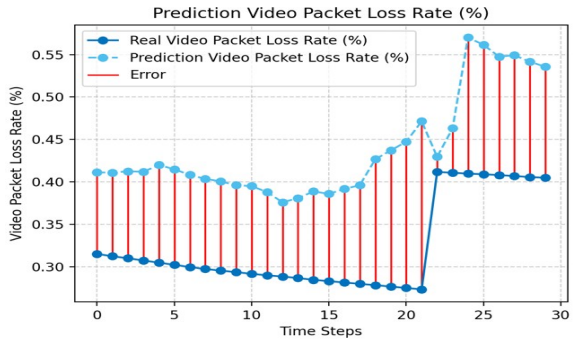

Figure 6. Actual Test Data vs. Model Prediction for Video Packet Loss Rate in Last-Minute Scenario 1

As shown in Figure 7, the audio packet loss rate prediction remained near zero, close to the measured data.

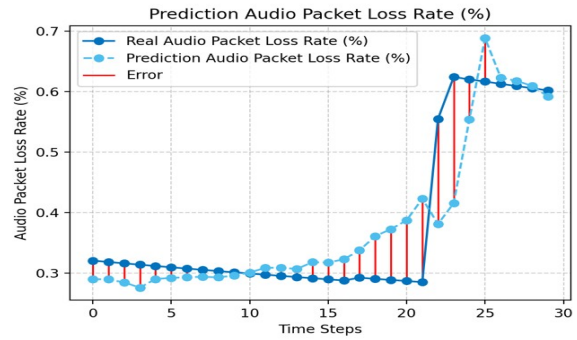

Figure 7. Actual Test Data vs. Model Prediction for Audio Packet Loss Rate in Last-Minute Scenario 1

B. *Moderate Degradation*

In Figure 8, it can be seen that the video latency showed an increase in variability compared to the acceptable quality network scenario (see Figure 2). While the model adapted well to average values, it exhibited limitations in predicting sudden latency spikes, which is expected given the less stable nature of the network in this scenario.

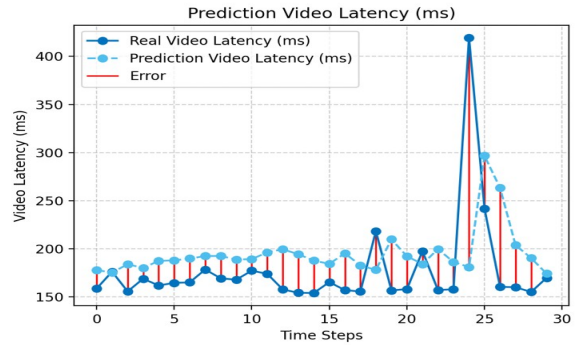

Figure 8. Actual Test Data vs. Model Prediction for Video Latency in Last-Minute Scenario 2

Figure 9 illustrates that the audio latency model effectively followed the general trend of the data, although, as with the video latency (see Figure 3), discrepancies arose when estimating extreme values. The performance is considered acceptable.

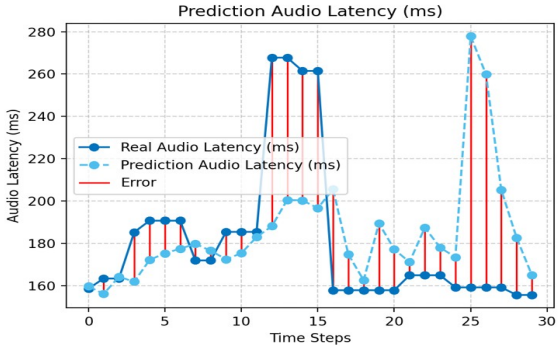

Figure 9. Actual Test Data vs. Model Prediction for Audio Latency in Last-Minute Scenario 2

According to Figure 10, the video jitter showed greater dispersion than the first scenario (see Figure 4). Nevertheless, the model could follow the overall trend, though with slightly reduced accuracy. This suggests that it can adapt to more dynamic conditions, yet with an increasing margin of error.

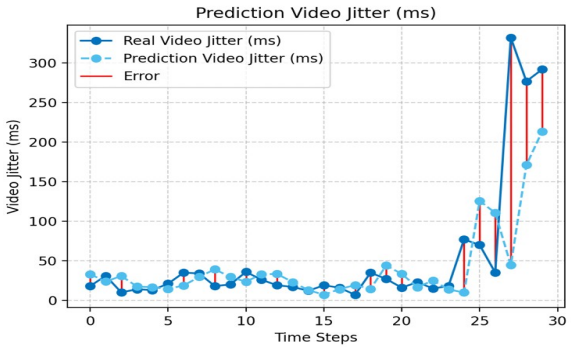

Figure 10. Actual Test Data vs. Model Prediction for Video Jitter in Last-Minute Scenario 2

As depicted in Figure 11, the behavior of the audio jitter showed wider fluctuations than in the first scenario (see Figure 5). The model maintained an acceptable ability to reflect the direction of changes compared to a stable environment (see Figure 5).

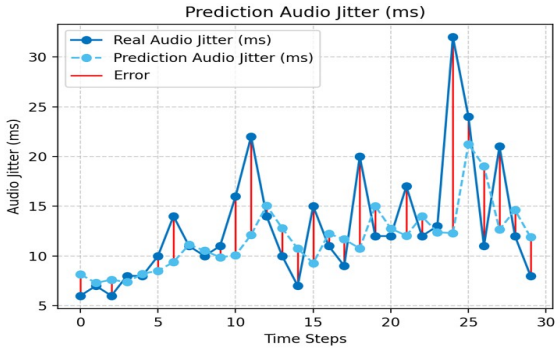

Figure 11. Actual Test Data vs. Model Prediction for Audio Jitter in Last-Minute Scenario 2

Regarding the video packet loss rate, Figure 12 indicates that the model faced greater difficulties in anticipating the actual pattern due to the intermittent and unpredictable nature of this type of traffic on a congested network. Even so, it managed to represent the overall trend of the fluctuations correctly.

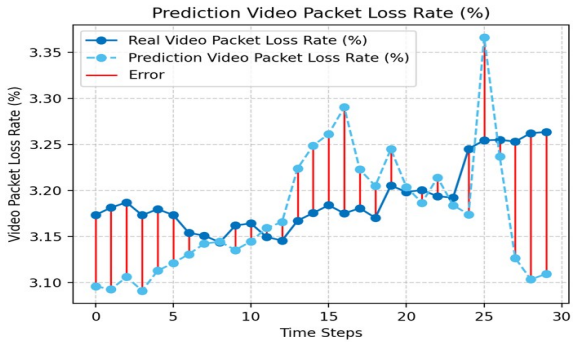

Figure 12. Actual Test Data vs. Model Prediction for Video Packet Loss Rate in Last-Minute Scenario 2

Figure 13 reveals that the audio packet loss rate exhibited variability similar to that of video (see Figure 12), although with lower intensity. The model captured the overall trend adequately, despite occasional discrepancies, demonstrating its adaptability while highlighting limitations in scenarios with irregular loss.

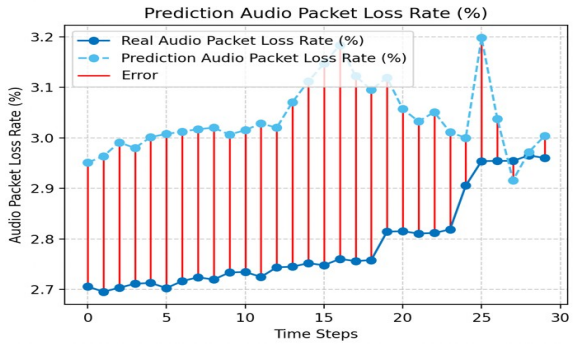

Figure 13. Actual Test Data vs. Model Prediction for Audio Packet Loss Rate in Last-Minute Scenario 2

C. *Critical Quality*

Figure 14 shows a pronounced deviation between the actual video latency values and the model's predictions. Although the model was generally able to follow the trend, differences in absolute values were evident, especially during periods of higher delay. This lack of precision can be attributed to the high baseline latency and the significant network instability.

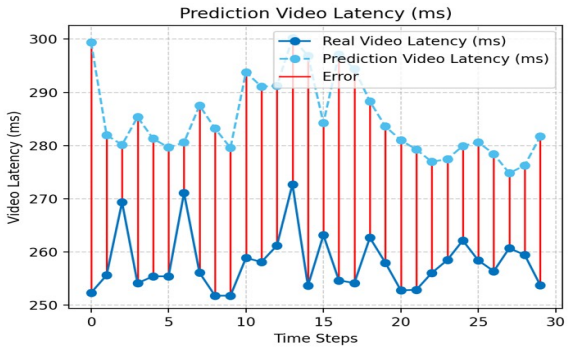

Figure 14. Actual Test Data vs. Model Prediction for Video Latency in Last-Minute Scenario 3

Similar to the video latency (see Figure 14), the audio latency also suffered discrepancies as shown in Figure 15. Although the model reasonably followed the trend, significant deviations were noted at the highest delay peaks.

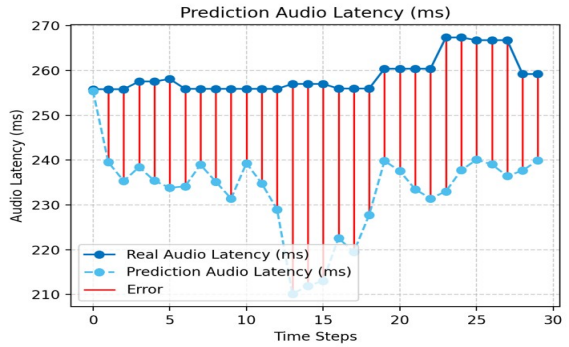

Figure 15. Actual Test Data vs. Model Prediction for Audio Latency in Last-Minute Scenario 3

In Figure 16, it can be seen that in the case of the video jitter, the model showed relatively stable performance. However, it struggled to replicate certain abrupt peaks present in the actual data. Despite this, the predictions reasonably captured the overall jitter dynamics, demonstrating the model's partial ability to adapt to rapid delay variations under critical conditions.

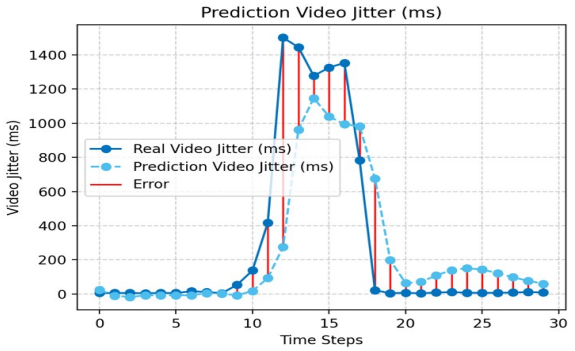

Figure 16. Actual Test Data vs. Model Prediction for Video Jitter in Last-Minute Scenario 3

As depicted in Figure 17, the audio jitter also exhibited behavior similar to that of video (see Figure 16). The model managed to follow the overall trend but faced notable difficulties during sudden changes. This illustrates that while the model can adapt to moderate fluctuations, it has limitations when faced with highly unstable events.

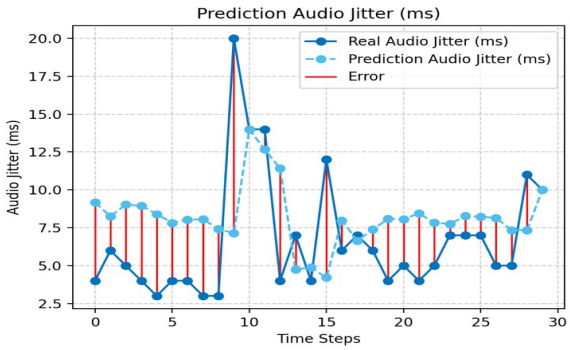

Figure 17. Actual Test Data vs. Model Prediction for Audio Jitter in Last-Minute Scenario 3

Regarding the video packet loss rate, Figure 18 indicates that the predictions generally remained close to the actual values. However, fluctuations were observed that the model was unable to predict accurately.

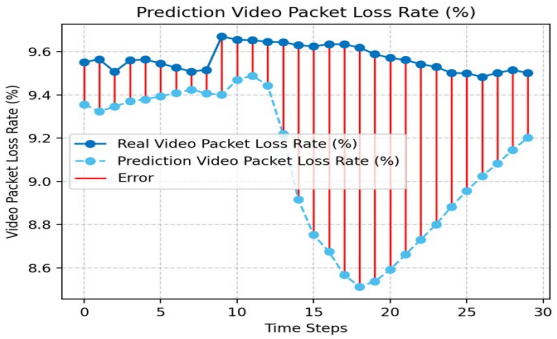

Figure 18. Actual Test Data vs. Model Prediction for Video Packet Loss Rate in Last-Minute Scenario 3

As shown in Figure 19, the audio packet loss rate exhibited patterns similar to those of video (see Figure 18). That is, while the model's predictions generally tracked the actual values, discrepancies emerged, reflecting its difficulty in anticipating abrupt changes.

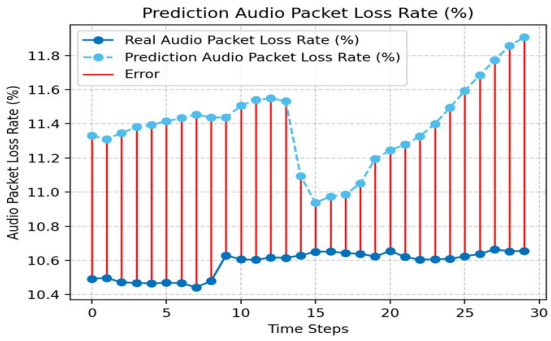

Figure 19. Actual Test Data vs. Model Prediction for Audio Packet Loss Rate in Last-Minute Scenario 3

D. *Extreme Conditions*

As can be seen in Figure 20, the model was able to reasonably follow the behavior of the video latency, adequately reproducing the most significant peaks present in the actual data. Although there are some discrepancies between the real and predicted values, the overall trend was effectively captured.

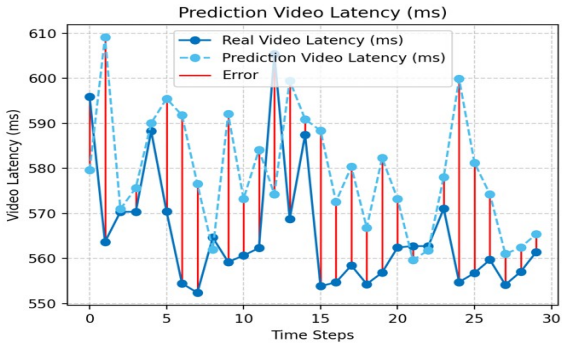

Figure 20. Actual Test Data vs. Model Prediction for Video Latency in Last-Minute Scenario 4

In contrast with the video latency (see Figure 20), the audio latency predictions exhibited greater deviations from the actual values, as shown in Figure 21. Increased dispersion and variability were observed, suggesting that the model has more difficulty adapting to rapid and erratic changes for this metric. Nevertheless, the overall trend was partially maintained, indicating that the model still achieved a coherent structural response.

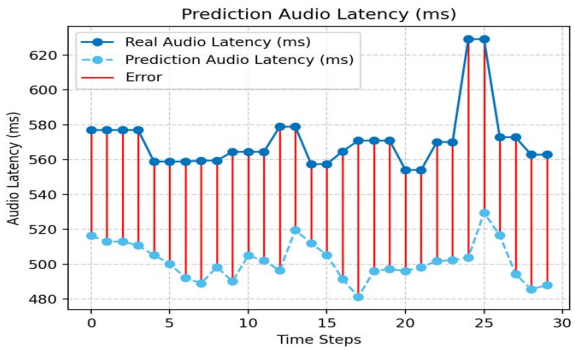

Figure 21. Actual Test Data vs. Model Prediction for Audio Latency in Last-Minute Scenario 4

In Figure 22, it can be seen that in the case of the video jitter, the model showed a reasonable ability to follow the general signal dynamics, although with specific differences in the maximum values. The predictions were consistent with the variation patterns, reflecting the model's ability to capture changes in delay instability, even if it did not achieve millimeter-level accuracy.

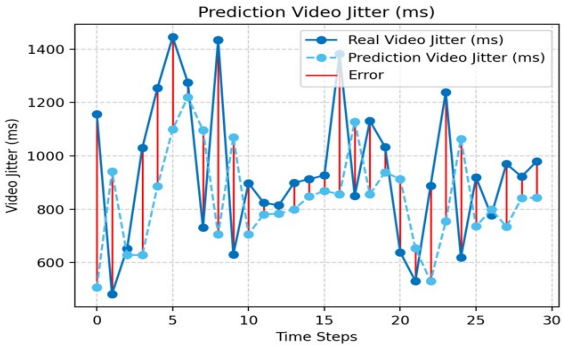

Figure 22. Actual Test Data vs. Model Prediction for Video Jitter in Last-Minute Scenario 4

As with the video jitter (see Figure 22), Figure 23 reveals that the audio jitter predictions provided an acceptable representation of the signal variations. Although discrepancies occurred at specific moments, especially during the most abrupt peaks, the model managed to represent the underlying behavior of the metric, reaffirming its partial ability to adapt to extreme fluctuations.

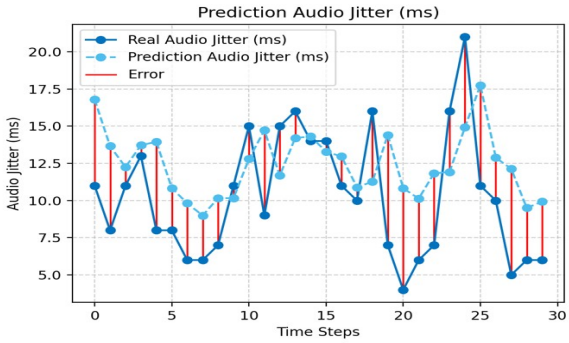

Figure 23. Actual Test Data vs. Model Prediction for Audio Jitter in Last-Minute Scenario 4

Regarding the video packet loss rate, Figure 24 indicates that the model's predictions showed an average difference of around 2 percentage points compared to the actual data.

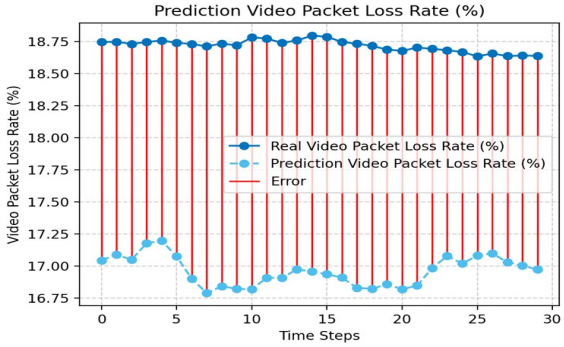

Figure 24. Actual Test Data vs. Model Prediction for Video Packet Loss Rate in Last-Minute Scenario 4

Similar to the video packet loss rate (see Figure 24), Figure 25 shows that the audio packet loss rate predictions reproduced the general structure of the actual signal, albeit with slight offsets at certain points. While an exact match was not achieved for all values, the model maintained acceptable coherence in terms of dynamics, correctly capturing the variation pattern in adverse environments.

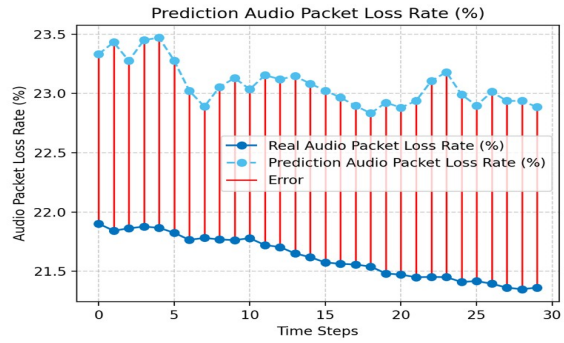

Figure 25. Actual Test Data vs. Model Prediction for Audio Packet Loss Rate in Last-Minute Scenario 4

V. CONCLUSIONS AND FUTURE WORK

After completing each of the phases outlined in this research project, it can be concluded that the integration of technologies such as WebRTC and RNNs represents a viable and modern alternative for addressing the problem of QoS prediction in video-call applications.

The research demonstrated that WebRTC, as a base technology, facilitates the creation of real-time communication environments with measurement and adaptation capabilities, removing previous technological barriers. The versatility of WebRTC, combined with a robust simulation infrastructure, made it possible to collect real metrics under different network conditions, thereby enriching the training of the predictive models.

During the system development, it was evidenced that LSTM-type neural networks are capable of capturing the temporal behavior of the evaluated metrics (latency, jitter, and packet loss rate), allowing the anticipation of their future evolution with an acceptable level of accuracy, especially under stable or moderately degraded conditions. In more

extreme scenarios, with packet losses of up to 80% or abrupt variations in delay, the model showed limitations in absolute accuracy, although it was still able to reflect the general trends of network behavior. This characteristic is particularly useful for implementing early warning mechanisms or dynamic adaptation that can be activated before communication quality noticeably degrades for the user.

One of the most significant contributions of this work was demonstrating that a deep-learning-based model can be fed with the first few minutes of a call to generate reliable predictions of its future behavior.

The adopted predictive approach demonstrated robustness when trained across multiple network scenarios, which allowed the neural network to learn diverse patterns and therefore generalize better under new conditions.

The following recommendations are proposed to strengthen the developed solution and encourage future research: expansion of the dataset, inclusion of new QoS and QoE metrics, implementation of the model in real production environments, exploration of more complex architectures, and design of autonomous network management systems.

ACKNOWLEDGMENT

We acknowledge the Central University of Venezuela (UCV), Sant Longowal Institute of Engineering and Technology (SLIET), and Jacksonville State University (JSU) for partially funding this project.

REFERENCES

[1] World Wide Web Consortium (W3C), “WebRTC 1.0: Real-Time Communication Between Browsers”, March 2025, https://www.w3.org/TR/webrtc/

[2] H. Alvestrand, Overview: Real-Time Protocols for Browser-Based Applications, RFC 8825, January 2021, doi: 10.17487/RFC8825, https://www.rfc-editor. org/info/rfc8825

[3] E. Rescorla, “Security Considerations for WebRTC”, RFC 8826, January 2021, doi: 10.17487/RFC8826, https://www. rfc-editor.org/info/rfc8826

[4] E. Rescorla, “WebRTC Security Architecture”, RFC 8827, January 2021, doi: 10.17487/RFC8827, https://www.rfc-editor.org/info/rfc8827

[5] C. Perkins, M. Westerlund, and J. Ott, “Media Transport and Use of RTP in WebRTC”, RFC 8834, January 2021, doi: 10.17487/RFC8834, https://www.rfc-editor.org/info/rfc8834

[6] H. Alvestrand, “Transports for WebRTC”, RFC 8835, January 2021, doi: 10.17487/RFC8835, https://www.rfc-editor.org/info/rfc8835

[7] S. Ness, V. Eswarakrishnan, H. Sridharan, V. Shinde, N. Venkata Prasad Janapareddy, and V. Dhanawat, “Anomaly Detection in Network Traffic Using Advanced Machine Learning Techniques”, IEEE Access, vol. 13, pp. 16133–16145, August 2025, doi: 10.1109/ACCESS.2025.3526988.

[8] G. Garcia and O. Salcedo, “Predicción de Fallos en Redes IP Empleando Redes Neuronales Artificiales”, Trabajo de Grado para Magister en Ciencias de la Información y las Comunicaciones, Universidad Distrital Francisco José de Caldas, Colombia, 2017.

[9] Y. Wang, Z. Jia, X. Zhang, B. Shao, H. Wang, and X. Xing, “TAQ-GRNN: A Topology-Aware QoS Prediction Model Based on Gated Recurrent Neural Networks”, 2024 IEEE 13th Data Driven Control and Learning Systems Conference (DDCLS 2024), August. 2024, pp. 303-308, doi:10.1109/DDCLS61622.2024.10606915.

[10] M. Di Mauro, G. Galatro, F. Postiglione, W. Song, and A. Liotta, “Evaluating Recurrent Neural Networks for Prediction of Multi-Variate Time Series VoIP Metrics”, 2024 22nd Mediterranean Communication and Computer Networking Conference (MedComNet 2024), Nice, France, 2024, pp. 1-8, doi: 10.1109/MedComNet62012.2024.10578296.

[11] W. A. Aziz, I. I. Ioannou, M. Lestas, H. K. Qureshi, A. Iqbal, and V. Vassiliou, “Content-Aware Network Traffic Prediction Framework for Quality of Service-Aware Dynamic Network Resource Management”, IEEE Access, vol. 11, pp. 99716–99733, August 2023, doi: 10.1109/ACCESS.2023.3309002.

[12] Z. Wu, Y.-N. Dong, X. Qiu, and J. Jin, “Online Multimedia Traffic Classification from the QoS Perspective Using Deep Learning”, Computer Network, vol. 204, pp. 1–13, Elsevier, January 2022, doi: 10.1016/j.comnet.2021.108716.

[13] P. Zhang, J. Ren, W. Huang, Y. Chen, Q. Zhao, and H. Zhu, “A Deep-Learning Model for Service QoS Prediction Based on Feature Mapping and Inference”, IEEE Transactions on Services Computing, vol. 17, no. 4, pp. 1311–1325, August 2024, doi: 10.1109/TSC.2023.3326208.

[14] J. Gerard, D. C. Bonilla, A. Bentaleb, and S. Céspedes, “Optimizing Quality and Energy Efficiency in WebRTC with ML-Powered Adaptative FEC”, 2024 IEEE International Conference on Multimedia and Expo Worlshops (ICMEW 2024), July 2024, pp. 57-64, doi: 10.1109/ICMEW63481. 2024.10645390.

[15] “Optimizing RTC Bandwidth Estimation with Machine Learning”, https://engineering.fb.com/2024/03/20/networking-traffic/optimizing-rtc-bandwidth-estimation-machine-learning/

[16] K. Sakakibara, S. Ohzahata, and R. Yamamoto, “Deep Learning-Based No-Reference Video Streaming QoE Estimation Using WebRTC Statistics”, 2024 IEEE International Conference on Artificial Intelligence in Information and Communication (ICAIIC 2024), February 2024, pp.1-7, doi: 10.1109/ICAIIC60209.2024.10463278

[17] G. Bingol, “Advancing Video Communication: From WebRTC Quality Prediction to Green Appplications”, Ph.D. Dissertation, Department of Electrical and Electronical Engineering, University of Cagliari, Cagliari, Italia, 2025.

[18] Google Firebase Firestore, “Cloud Firestore”, https://firebase.google.com/docs/firestore

[19] ITU-International Telecommunications Union, “G.1010: End-User Multimedia QoS Categories”, https://www.itu.int/rec/T-REC-G.1010-200111-I

[20] PeerJS, “PeerJS Simplifies WebRTC Peer-to-Peer Data, Video, and Audio Calls”, https://peerjs.com

[21] WebRTC for Developers, “Breaking Changes in getStats”, https://www.webrtc-developers.com/breaking-changes-in-getstats/

[22] Apple, “Network Link Conditioner”, https://nshipster.com/network-link-conditioner

[23] 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.