NEURAL NETWORK BASED PROTOCOL OPTIMIZER

[1]Ashwini Deokate, [2]Aditya Karale, [3]Anurag Nasare, [4]Chaitnaya Kale

[1]Professor, [2,3,4]Students

[1]Department of AI Engineering,

[1,2,3,4] Priyadarshini Bhagwati College of Engineering, Nagpur, Maharashtra, India

Abstract : This paper presents an AI-driven adaptive network protocol optimization framework using neural networks to improve transmission performance under dynamic network conditions. The proposed model predicts optimal Window Size and Retransmission Timeout (RTO) using bandwidth, latency, packet loss, and congestion as input parameters. A synthetic dataset is generated to simulate realistic communication scenarios, and a Multi-Layer Perceptron (MLPRegressor) is trained with hyperparameter optimization for accurate prediction. Experimental results achieved an R^2 score of 0.983, demonstrating high predictive accuracy. The proposed approach enhances throughput, reduces latency, and enables intelligent real-time protocol parameter tuning beyond conventional rule-based network mechanisms.

IndexTerms **- Network Optimization, Neural Networks, Congestion Control, MLPRegressor**

1. INTRODUCTION

1.1 Background

Traditional network protocols rely on fixed rule-based configurations and heuristic algorithms for data transmission **[1]**. Although effective in conventional environments, they often perform poorly in dynamic networks such as cloud computing, 5G/6G, and IoT due to variations in bandwidth, latency, congestion, and packet loss. These rapidly changing conditions make static protocol configurations inadequate.

Advancements in Artificial Intelligence (AI) and Machine Learning (ML) have introduced adaptive mechanisms for network optimization **[2]**. Through data-driven learning, AI-enabled systems can model relationships between network states and optimal protocol parameters, improving adaptability, throughput, and reliability. Neural networks are particularly effective in capturing non-linear dependencies, making them suitable for adaptive protocol optimization.

1.2 Problem Statement

Conventional network protocols with fixed transmission parameters often fail to maintain optimal performance under unpredictable network conditions **[3]**. Parameters such as Transmission Window Size and Retransmission Timeout (RTO) are commonly configured using predefined mechanisms that may become suboptimal as network dynamics change. This can lead to increased latency, reduced throughput, and inefficient congestion control.

Therefore, there is a need for an intelligent adaptive framework capable of predicting and optimizing transmission parameters in real time based on network conditions **[4]**.

1.3 Objectives

The primary objective of this work is to develop an AI-driven adaptive network protocol optimization system using neural networks. The specific objectives are as follows:

1. To predict optimal transmission parameters, including Window Size and Retransmission Timeout (RTO), using real-time network conditions.
2. To model network behavior using input parameters such as bandwidth, latency, packet loss, and congestion levels.
3. To design, train, and evaluate a Multi-Layer Perceptron Regressor (MLPRegressor) for learning non-linear relationships between network inputs and optimized outputs.
4. To improve network efficiency through intelligent and dynamic protocol adaptation.

1.4 Scope of the Study

This study focuses on developing and evaluating an adaptive network optimization framework using simulated communication scenarios generated through a synthetic dataset. The model considers four network input parameters and predicts two protocol output parameters.

The scope includes data preprocessing, neural network training, performance evaluation using metrics such as R-squared (R^2) score and Mean Squared Error (MSE), and storage of trained model parameters for real-time inference. The study is limited to simulation-based analysis and excludes live network deployment.

2. NEED OF THE STUDY

The rapid advancement of cloud computing, IoT, and 5G/6G communication systems has significantly increased the complexity of modern network environments. Traditional transport protocols based on static heuristics and predefined congestion control mechanisms often fail to adapt efficiently under dynamic conditions such as fluctuating bandwidth, latency variation, packet loss, and congestion. Consequently, intelligent and adaptive optimization techniques have become an active research area. This section presents a review of related work, identifies research gaps, and highlights the contribution of the proposed study.

2.1 Literature Review

Several researchers have contributed toward adaptive network optimization and intelligent congestion control. Postel [1] introduced the Transmission Control Protocol (TCP), establishing fundamental mechanisms for flow control and retransmission management. Although TCP laid the foundation for reliable communication, its dependence on static heuristic rules limits adaptability in dynamic network conditions.

Jacobson [2] proposed congestion avoidance and control mechanisms that significantly improved TCP performance through additive increase and multiplicative decrease strategies. While effective, the approach remains reactive and threshold-driven, lacking predictive intelligence required for modern networks.

Goodfellow *et al.* [5] demonstrated the capability of deep learning models to capture complex non-linear relationships across diverse domains. Their work established neural networks as powerful predictive tools, suggesting their potential applicability in adaptive protocol optimization. However, direct implementation for transport-layer parameter tuning remained limited.

Cui *et al.* [6] explored machine learning-based congestion control methods and reported improved adaptability over conventional approaches. Their work showed that learning-based systems can outperform static algorithms, though optimization was primarily focused on congestion response rather than simultaneous tuning of multiple protocol parameters.

Mao *et al.* [7] investigated neural network-driven network optimization for resource allocation and traffic management. Their results demonstrated improved network efficiency; however, limited emphasis was placed on adaptive prediction of transport-layer parameters such as Window Size and Retransmission Timeout.

Xu *et al.* [8] presented an AI-driven adaptive communication protocol optimization approach that highlighted the potential of intelligent parameter tuning. Although promising results were reported, challenges related to lightweight deployment, synthetic data-driven modeling, and low-latency inference remain open for further study.

From the reviewed literature, it is evident that significant progress has been made in intelligent networking; however, comprehensive adaptive optimization of transmission parameters using lightweight neural models remains insufficiently explored.

2.2 Research Gap

Based on the literature survey, the following research gaps have been identified:

1. Existing transport protocols rely largely on static heuristics and lack intelligent adaptive behavior.
2. Most machine learning-based studies focus mainly on congestion control rather than simultaneous optimization of multiple protocol parameters such as Window Size and RTO.
3. Limited studies address modeling the non-linear relationships among bandwidth, latency, packet loss, congestion, and protocol tuning through neural networks.
4. Real-time, lightweight AI inference mechanisms for protocol optimization remain underexplored.
5. Simulation-driven frameworks for generating diverse network scenarios for model training are limited in existing studies.

2.3 Contribution of the Proposed Work

To address these gaps, the present work makes the following contributions:

1. Proposes an AI-driven adaptive network protocol optimization framework using neural networks.
2. Develops a predictive model for simultaneous optimization of Window Size and Retransmission Timeout (RTO).
3. Utilizes multiple network state parameters to model adaptive protocol behavior.
4. Implements a Multi-Layer Perceptron (MLPRegressor) with hyperparameter optimization for accurate prediction.
5. Introduces a synthetic simulation framework for generating realistic communication scenarios.
6. Demonstrates the feasibility of intelligent real-time protocol parameter tuning using machine learning.

2.4 Summary

This chapter presented the need for the study through analysis of existing literature, identification of research gaps, and discussion of the contributions of the proposed work. The review establishes the necessity of developing intelligent adaptive optimization techniques for next-generation communication networks.

3. METHODOLOGY AND IMPLEMENTATION

This chapter presents the methodology used to develop the AI-driven network optimization model, including the proposed architecture, simulation environment, neural network formulation, and hyperparameter optimization process.

3.1 Proposed System Architecture

Traditional network protocol configuration relies on static heuristics and threshold-based responses [3]. The proposed framework replaces this approach with a predictive machine learning model. The system consists of three main modules:

1. **Network Telemetry Collector:** Simulates continuous monitoring of network state variables.
2. **AI Inference Engine:** A trained neural network that processes scaled telemetry data and predicts optimal protocol parameters in real time.
3. **Protocol Actuator:** Converts model outputs such as Window Size and RTO into protocol adjustments.

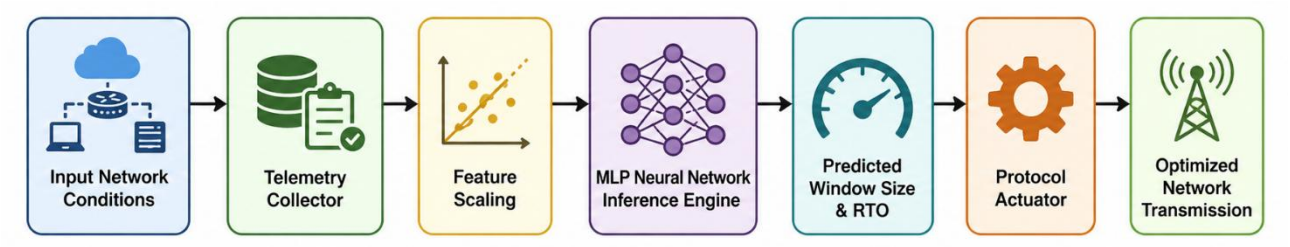

Figure 3.1 Proposed System Working Architecture

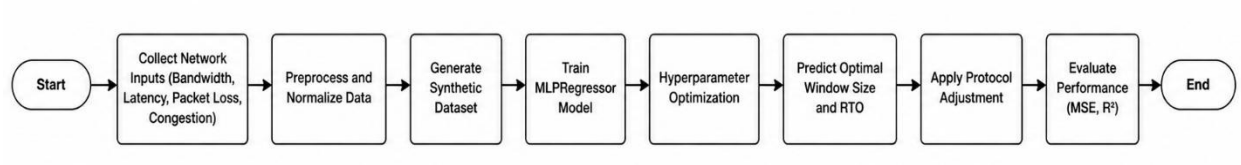

Figure 3.2 Flowchart of Proposed Methodology

3.2 Simulation Environment and Data Generation

Due to limited availability of labeled datasets for dynamic network conditions, a stochastic simulation environment was developed in Python to generate realistic communication scenarios [8].

3.2.1 Network State Variables

The input feature vector is represented as:

$$X = [b, l, p, c]$$

where:

- b= Available bandwidth (Mbps)
- l= End-to-end latency (ms)
- p= Packet loss probability (%)
- c= Normalized congestion metric $(0 \leq c \leq 1)$

3.2.2 Target Variable Generation

The target output vector is represented as:

$$Y = [W, RTO]$$

where:

- **Optimal Window Size ((W))** increases with bandwidth but is reduced by packet loss and congestion.
- **Optimal Retransmission Timeout ((RTO))** increases with latency and congestion to reduce unnecessary retransmissions.

Target outputs were generated using deterministic functions combined with Gaussian noise to represent real-world variability. To avoid scale imbalance and improve training stability, Z-score normalization was applied using StandardScaler [9].

3.2.3 Feature Scaling

To avoid scale imbalance and improve training stability, Z-score normalization was applied using StandardScaler:

$$Z_i = \frac{x_i - \mu_i}{\sigma_i}$$

3.3 Neural Network Formulation

The optimization engine uses a Multi-Layer Perceptron Regressor (MLPRegressor) from the scikit-learn framework to map a 4-dimensional input space to a 2-dimensional output space [10].

3.3.1 Activation and Optimization

The Rectified Linear Unit (ReLU) activation function is used in hidden layers to address vanishing gradient issues, while model weights are optimized using the Adam solver [11]., which provides adaptive learning rates. The maximum convergence limit was set to 400 epochs.

3.3.2 Loss Function

The network is trained by minimizing Mean Squared Error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

where $N$is the batch size, $Y_i$is the actual output, and $\hat{Y}_i$is the predicted output.

3.4 Implementation and Hyperparameter Optimization

An automated hyperparameter grid search was implemented to identify the optimal neural network configuration [12].

3.4.1 Architecture Search Space

Multiple hidden layer structures were evaluated through cross-validation, including configurations such as ((64,32)), ((32,32,16)), and ((128,64)).

3.4.2 Regularization

The L2 regularization parameter $(\alpha)$was varied over:

$$\{0.01, 0.001, 0.0005, 0.0001\}$$

to balance bias-variance tradeoff and reduce overfitting.

3.4.3 Experiment Tracking

A custom SQLite logging system was implemented to record model configurations, timestamps, alpha values, and test MSE for each experiment, tagged as **MLPRegressor_AUTO**.

The configuration producing the lowest MSE was selected, and the trained model was serialized using **joblib** for low-latency deployment [13].

3.5 Chapter Summary

This chapter presented the methodology and implementation of the proposed AI-driven adaptive network optimization system. It described the system architecture, simulation-based dataset generation process, neural network formulation, and hyperparameter optimization strategy. The inclusion of feature scaling, automated tuning, and experiment tracking ensured robust model development. The optimized trained model forms the foundation for performance evaluation and experimental analysis presented in the next chapter.

4. RESULTS AND DISCUSSION

This chapter presents the experimental results, performance evaluation, and discussion of the proposed AI-driven adaptive network optimization system. It analyzes the predictive capability of the neural network using statistical metrics and graphical interpretation.

4.1 Experimental Setup

The experiments were conducted using a synthetic dataset consisting of 1,500 samples. The dataset was divided into 80% training data and 20% testing data. The MLPRegressor model was trained to predict optimal Window Size and Retransmission Timeout (RTO) based on network conditions. Model performance was evaluated using standard regression metrics.

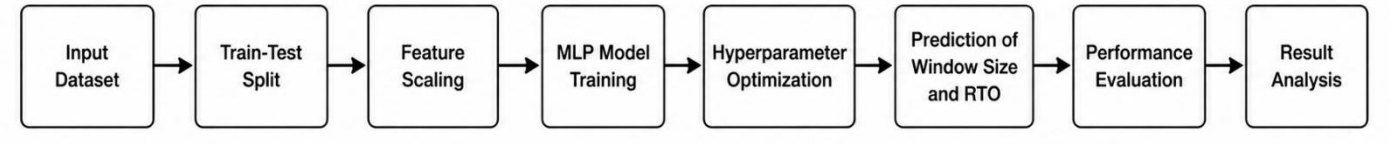

Figure 4.1 Experimental Workflow

4.2 Model Performance Metrics

The accuracy and reliability of the trained model were evaluated using the following metrics:

1. **Coefficient of Determination (R^2 Score):** The model achieved an R^2 score of **0.983**, indicating excellent predictive capability and strong fitting of the model to the data.
2. **Mean Squared Error (MSE):** The model produced an MSE of approximately **319.76**, which is within acceptable limits considering the output range.

Table 4.1 Performance Metrics

Metric	Value
R^2 Score	0.983
MSE	319.76

4.3 Exploratory Data Analysis

4.3.1 Feature Influence on Window Size

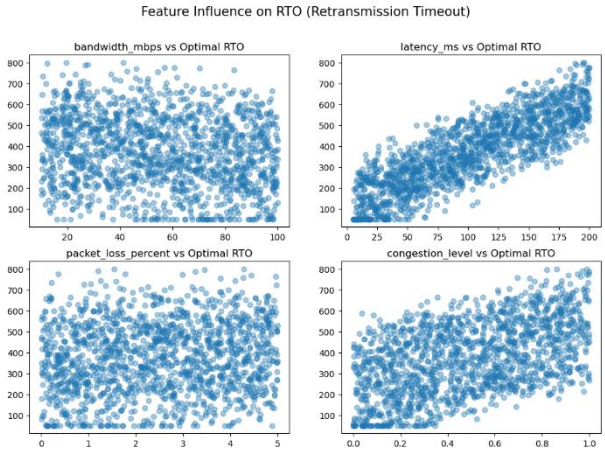

Figure 4.2 Feature Influence on Optimal Window Size

The analysis shows that bandwidth has a strong positive relationship with predicted window size. As bandwidth increases, the model recommends larger transmission windows to maximize throughput. In contrast, packet loss and congestion negatively affect window size, reducing transmission aggressiveness to avoid network flooding.

4.3.2 Feature Influence on Retransmission Timeout

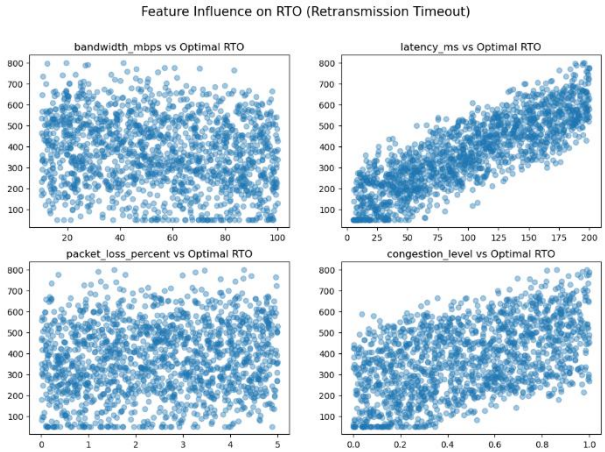

Figure 4.3 Feature Influence on Optimal RTO

The analysis indicates that increasing latency and congestion lead to higher predicted RTO values. This adaptive behavior helps reduce premature retransmissions and improves protocol stability.

Key Observations

- **Bandwidth vs Window Size:** Strong positive correlation observed.
- **Latency and Congestion vs RTO:** Higher latency and congestion increase RTO.
- **Packet Loss Impact:** Higher packet loss causes reduction in window size.

4.4 Model Evaluation and Convergence

4.4.1 Training Convergence Analysis

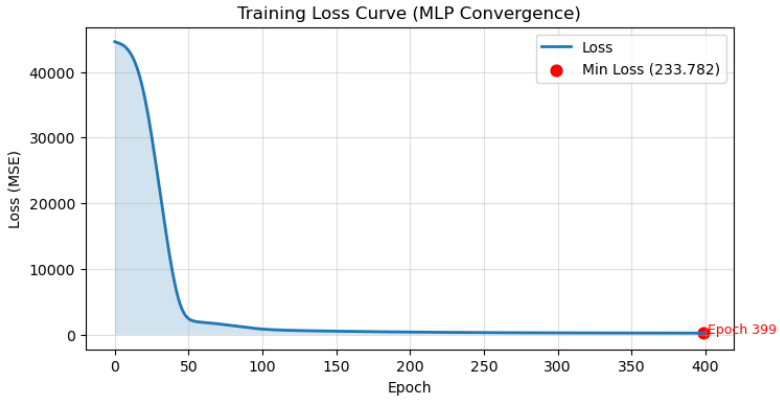

Figure 4.4 Training Loss Curve

The training loss curve shows a steady reduction in Mean Squared Error over successive epochs, indicating stable convergence of the neural network. Rapid error reduction in early epochs confirms efficient learning without major overfitting.

4.4.2 Predicted vs Actual Performance

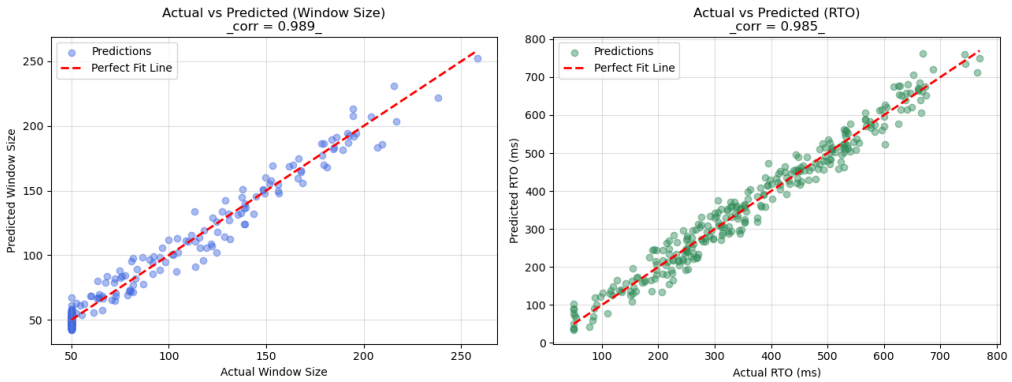

Figure 4.5 Predicted vs Actual Values

Comparison of predicted and actual values shows strong alignment along the identity line, validating the model's high predictive accuracy. The clustering of points supports the achieved R^2 score of 0.983.

4.5 Discussion

The results confirm that the neural network successfully captured complex non-linear relationships between network conditions and protocol parameters. Compared to traditional static protocols, the proposed model demonstrates superior adaptability.

Adaptive Window Sizing

Instead of fixed reactions such as reducing the window size after packet loss, the model dynamically adjusts the transmission window based on bandwidth, congestion, and packet loss severity.

Smart RTO Prediction

The proposed model replaces conventional moving-average RTO estimation with multidimensional prediction, making retransmission timing more robust under fluctuating latency conditions.

Performance Advantages

- Improved throughput through adaptive window optimization.
- Reduced latency through intelligent RTO adjustment.
- Better congestion handling compared to traditional heuristic methods.
- Real-time protocol parameter tuning using machine learning.

Overall, the results demonstrate that machine learning can effectively optimize network protocol parameters in dynamic environments.

4.6 Chapter Summary

This chapter presented the experimental setup, performance metrics, exploratory data analysis, and model evaluation results of the proposed system. The neural network achieved high predictive accuracy with an R^2 score of 0.983 and low error, validating the effectiveness of the proposed approach. The analysis confirms that AI-driven adaptive optimization can significantly improve transmission parameter tuning under varying network conditions.

5. CONCLUSION

This chapter presents the summary of findings, advantages of the proposed AI-driven optimization system, future research directions, and overall conclusion of the study.

5.1 Summary of Findings

This work successfully demonstrated the feasibility and effectiveness of an **AI-Driven Adaptive Network Protocol Optimization** system using neural networks. By employing a Multi-Layer Perceptron (MLPRegressor), the system predicted optimal transmission parameters, namely Window Size and Retransmission Timeout (RTO), based on dynamic network conditions such as bandwidth, latency, packet loss, and congestion.

The developed model achieved high predictive performance with an R^2 score of approximately **0.983**, confirming its ability to capture complex non-linear relationships governing network behavior. The results showed that machine learning can effectively optimize protocol parameters and improve adaptive transmission control under varying network conditions.

Key Findings:

- Accurate prediction of optimal Window Size and RTO.
- Successful modeling of non-linear network behavior using neural networks.
- High predictive accuracy with low error metrics.
- Demonstration of intelligent real-time protocol parameter adaptation.

5.2 Advantages of AI-Driven Optimization

Compared to traditional fixed-rule protocols, the proposed AI-based approach offers several advantages:

1. **Dynamic Adaptability:** The system continuously adapts to changing network conditions, improving responsiveness and preventing congestion.
2. **Improved Throughput and Reduced Latency:** Optimized transmission window and RTO prediction improve throughput while minimizing unnecessary retransmissions.
3. **Data-Driven Precision:** The model eliminates dependence on manually designed heuristics and uses learned patterns for protocol optimization.
4. **Intelligent Congestion Handling:** The system responds proportionally to network conditions rather than using rigid threshold-based control.

5.3 Future Work

Although the proposed system was evaluated using simulated data, several directions exist for future enhancement:

1. **Real-World Network Data Integration:** Future work can train and validate the model using real network traffic traces from cloud, IoT, or 5G/6G environments.
2. **Reinforcement Learning-Based Optimization:** The framework can be extended using Reinforcement Learning to enable autonomous learning of congestion control policies.
3. **Low-Latency Deployment:** Deploying the trained model in kernel-space using technologies such as eBPF can support ultra-low latency inference for packet-level decisions.
4. **Integration with Existing Protocols:** The proposed model can be evaluated as an adaptive congestion control mechanism within protocols such as TCP and QUIC.

5.4 Overall Conclusion

The proposed AI-driven adaptive network optimization framework demonstrates that neural networks can effectively tune transmission parameters in dynamic communication environments. Unlike conventional static protocols, the proposed system provides intelligent, real-time, and data-driven optimization of network behavior. The strong performance metrics and adaptive characteristics validate the potential of machine learning as a next-generation approach for intelligent network protocol optimization.

5.5 Chapter Summary

This chapter summarized the major findings of the study, highlighted the advantages of AI-driven protocol optimization, and presented possible future research directions. The results confirm that intelligent adaptive models can significantly improve network protocol performance and provide a strong foundation for future AI-enabled communication systems.

I. Acknowledgment

The authors express their sincere gratitude to **Prof. Ashwini Deokate**, Department of AI Engineering, Priyadarshini Bhagwati College of Engineering, Nagpur, Maharashtra, India, for her invaluable guidance, constant encouragement, and technical support throughout this research work. Her expert suggestions and mentorship played a significant role in the successful completion of this study. The authors also thank the Department of AI Engineering and Priyadarshini Bhagwati College of Engineering for providing the necessary resources and academic environment for carrying out this research, and acknowledge everyone who directly or indirectly contributed to this work.

References

[1] J. Postel, "Transmission Control Protocol," *RFC 793*, Internet Engineering Task Force, Sep. 1981. doi: 10.17487/RFC0793.
[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. doi: 10.5555/3086952.
[3] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, Stanford, CA, USA, 1988, pp. 314–329. doi: 10.1145/52324.52356.
[4] W. Cui, Y. Liu, X. Wang, and J. Zhang, "Machine learning-based congestion control for intelligent communication networks," *IEEE Access*, vol. 8, pp. 168045–168057, 2020. doi: 10.1109/ACCESS.2020.3023954.
[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. doi: 10.5555/3086952.
[6] W. Cui, Y. Liu, X. Wang, and J. Zhang, "Machine learning-based congestion control for intelligent communication networks," *IEEE Access*, vol. 8, pp. 168045–168057, 2020. doi: 10.1109/ACCESS.2020.3023954.
[7] Y. Mao, J. Zhang, and K. B. Letaief, "Neural network based intelligent network optimization for resource allocation," *Future Generation Computer Systems*, vol. 115, pp. 486–497, Feb. 2021. doi: 10.1016/j.future.2020.09.041.
[8] Z. Xu, L. Chen, H. Wang, and M. Guizani, "AI-driven adaptive communication protocol optimization for next-generation networks," *IEEE Trans. Network Science and Engineering*, vol. 10, no. 4, pp. 2456–2468, Jul.–Aug. 2023. doi: 10.1109/TNSE.2023.3267815.
[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980
[10] Scikit-learn Developers, "StandardScaler," *Scikit-learn Documentation*, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[11] Scikit-learn Developers, "MLPRegressor," *Scikit-learn Documentation*, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 315–323.
[13] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Machine Learning Research*, vol. 13, pp. 281–305, 2012.
[14] Joblib Development Team, "Joblib: Running Python functions as pipeline jobs," 2024. [Online]. Available: https://joblib.readthedocs.io/en/latest/