Deep Learning Based Path Weight Prediction for Complex Network Routing

Basim Jamil Ali[1], Mohannad Ali Meteab Al-Obaidi[1], Hassan Hadi Salih[2] and Moamin A Mahmoud[3]
[1]*Department of Computer Science, College of Science, Mustansiriyah University, 10052 Baghdad, Iraq*
[2]*Department of Computer Science, College of Science, University of Diyala, Al-Mouradia, 32001 Baqubah, Diyala, Iraq*
[3]*Department of Computing, College of Computing and Informatics, Universiti Tenaga Nasional, 43000 Kajang, Malaysia*
basimja6nd@uomustansiriah.edu.iq, neros2210@uomustansiriyah.edu.iq, hassn.hadi@uodiyala.edu,iq, moamin@uniten.edu.my

Keywords: Network Enhancements, Adam, Deep Learning, Erdős-Rényi, Long Short-Term Memory.

Abstract: This research outlines a novel approach for predicting path metrics in Erdős-Rényi and complex networks using deep learning, specifically Long Short-Term Memory (LSTM) networks. The determination of weighted shortest paths is important for proper network routing and optimization. Though traditional graph algorithms solve the problem, they do not scale well or adapt to changes in dynamic or large-scale situations. In this study, an LSTM model is trained to estimate the total path weight using the source, target, and the number of hops on the path. This involves constructing a comprehensive dataset of random network graphs with weighted edges, and their shortest paths. Data preparation involves more than just cleansing–it includes thorough pre-modeling normalization, model validation, and dynamic statistical assessment and visualization of the results. The experiments confirm the hypothesis that LSTM models accurately predict path metrics compared to more basic mean squared error and correlation metric baselines. The study provides an advanced scalable framework for path metric prediction with immediate applications in real-time routing, management, and anomaly detection in complex networks.

1 INTRODUCTION

The analysis of intricate relationships within a network is of growing significance in computer science in areas like communication systems, social networks, and biological systems. One of the most important aspects of this work is predicting network path metrics: total path weight, hop count, and latency, pivotal in improving routing precision and sustaining optimal network functionality [1].

With large-scale or dynamic networks, traditional algorithms such as Dijkstra's and Floyd-Warshall's will offer exact solutions for calculating the shortest paths, but at an exorbitant computational cost. This fact has turned many efforts towards increasingly popular data driven approaches, such as deep learning models, which blindly infer path metrics without explicit reference to the underlying structure of the network [2].

Erdős–Rényi (ER) graphs serve as fundamental models for studying the probabilistic behaviour of sparse and homogeneous networks [3]. In contrast, real-world systems are often more complex, following scale-free or small-world network topologies, which exhibit clustering, hub nodes, and heterogeneous degree distributions. These features greatly enhance the potential of deep learning models like Recurrent Neural Networks (RNNs), Graph Neural Networks (GNNs), and Long Short-Term Memory (LSTM) networks for such systems [3], [4]. Comprehensive surveys on GNN architectures and their applications have further consolidated this research direction [5].

In this study, we develop a lightweight yet accurate predictive framework based on LSTM networks for estimating path metrics in ER and complex networks. The method tackles the computational burden of classical algorithms by training on a diverse set of randomly generated weighted graphs, concentrating on parameters like source, target, and hop count.

The main contribution of this work is the development of a lightweight LSTM-based framework for predicting path weights in complex networks using minimal input features. The proposed

approach avoids explicit reliance on full network topology while maintaining strong predictive performance.

1.1 Problem Statement

What approaches can we adopt in constructing a deep learning model for accurate prediction of network path metrics in arbitrary and intricate networks while using limited input features and avoiding dependence on complete graph representation?

1.2 Research Objectives

The main objective of this study is to develop and evaluate an LSTM-based neural network model for predicting total path weights in weighted network graphs, with a focus on improving routing efficiency and scalability in complex network environments. Specifically, the study aims to achieve the following objectives:

- To train and design network using the LSTM-based neural network capable of predicting total path weights in networks.
- To build a synthetic database of weighted ER and complex graphs of known paths.
- To justify the model with the conventional measures of MSE, MAE and R2.
- To illustrate the scalability of the model and that the model is applicable to real time routing applications.

1.3 Application Fields

The suggested LSTM-based infrastructure to anticipate network measures has general usage in the different fields that require real-time path estimation and network optimums. The major areas of application are discussed below together with the relevant research illustrates in Table 1.

2 RELATED WORKS

The current progress on deep learning made quite an impact on the network science mostly in forecasting metrics associated with the paths like latency, shortest distance, and the availability of bandwidth. This evolution is fuelled by the inefficiency of the classical graph algorithms in scaling towards large-scale or dynamic network scenarios.

Classical methods in algorithms such as Dijkstra and FloydWarshall are well known in accurate path addressing but are characterized by high computation costs with increase in the size of the network. Although they perform well in deterministic environments, their shortcomings become obvious when used in the real-time or partially observable networks.

To handle this, diverse pieces have examined the correlation between deep learning and network topology. To give an example, GraphRNN presented in [8] is a representative primitive autoregressive model of generating graphs that only ensures important structural invariance of input graphs and whose primary objective is structural synthesis as opposed to metric or path prediction [8].

Largely related to path prediction and combinatorial optimization, recent research, including the AutoGNPs proposed by Liu et al. (2024), automates the Graph Neural Network (GNN) architecture search to solve NP-hard combinatorial problems, e.g., quadratic unconstrained binary problems (QUBO) and mixed-integer linear programming. Such models have highly effective performance and generalization over unexposed graph topologies [9].

Table 1: The major areas of application.

Application Field	Description
Telecommunications Networks	Enhances routing protocols via dynamic latency/congestion predictions [6].
Internet of Things (IoT)	Enables adaptive routing and self-healing in dynamic, resource-constrained networks [7].
Software-Defined Networking (SDN)	Supports SDN controller decision-making for optimal routing and QoS enforcement [13].
Smart Grids & Infrastructure	Predicts communication delays and stability across power and transport systems [9].
Cybersecurity & Anomaly Detection	Identifies abnormal traffic behaviour based on deviations in expected path metrics [10].
Data Centers & Cloud Networks	Optimizes load balancing and service placement by predicting network bottlenecks [11].
Autonomous Systems & Robotics	Improves inter-agent coordination in swarms by forecasting inter-node delays and routes [12].

Although most of these techniques are promising, they are all dependent upon elaborate architectural components, e.g., message passing, attention layers, graph embeddings, and are prohibitive to fit in resources. As an example, Pugacheva et al. (2024) proposed QRF-GNN, an unsupervised GNN supplemented with recurrent feature learning to QUBO problems. Although being computationally efficient, it is still relying on iterative structures of graph embeddings and dynamic updates that might not be realistic on devices of limited memory or computer [10]. Table 2 illustrates the comparative analysis of previse studies and the proposed system.

Table 2: Comparative analysis of related works and the proposed LSTM-based system.

Study	Model Type	Input Features	Full Topology Required	Real-Time Suitability
GraphRNN [8]	RNN-G	G	✓	✗
AutoGNPs [9]	GNN-A	G + N	✓	✗
QRF-GNN [10]	GNN-R	E	✓	✗
Proposed Method	LSTM	ID + H	✗	✓

Notation:

- G = Graph structure;
- N = Node features;
- E = Graph embeddings;
- ID = Node identifiers;
- H = Hop count;
- ✓ indicates that the property is required or supported;
- ✗ indicates otherwise.

It should be noted that the comparison presented in Table 2 is conceptual rather than experimental. The objective of this comparison is to highlight the differences in model complexity, input requirements, and computational characteristics between the proposed LSTM-based framework and existing graph-based deep learning approaches such as Graph Neural Networks (GNNs) and GraphRNN. Conducting a full experimental comparison with these architectures remains an interesting direction for future work.

2.1 Research Gap

Current deep learning-based models used to predict path can access full structural information or use complicated network structures that cannot be used in lightweight, real-time systems. Few papers exist on how LSTM networks can be used to predict path metrics with only abstracted node id and hop counts.

2.2 Contribution of this Study

The main contributions of this study are centered on developing an efficient and lightweight deep learning framework for path-weight prediction in complex network environments. The proposed approach aims to reduce computational complexity while maintaining high prediction accuracy and scalability for real-time routing applications. The major contributions of this study are summarized as follows:

- A lightweight LSTM-based prediction model (MINLSTM) was developed to estimate the total path weight between source and destination nodes using only the source node, destination node, and hop count without requiring complete network topology information.
- A scalable synthetic dataset was generated using Erdős–Rényi and Barabási–Albert graph models, while the Dijkstra algorithm was used to produce ground-truth shortest-path weights for model training and evaluation.
- The proposed model achieved high predictive performance with strong evaluation results, including an R^2 score of 0.93 and MAE of 0.0645, while maintaining low computational cost and completing training within a short time on standard hardware platforms.
- The proposed framework was designed to support real-time deployment scenarios such as intelligent routing, anomaly detection, edge computing, and resource-constrained network environments due to its lightweight architecture and computational efficiency.

3 PROPOSED SYSTEM

The proposed system is based on LSTM-based neural networks for prediction of total path weights in random and complex networks using a minimal number of input features. It provides an efficient path metric estimation method for communication networks without requiring the full network topology, making the framework lightweight and scalable. To ensure experimental reproducibility, all models were implemented using the TensorFlow and Keras frameworks, while graph datasets were generated using the NetworkX library. The same preprocessing pipeline and training configuration were consistently applied across all experiments .Although LSTM networks are typically designed for sequential data,

they can also model nonlinear relationships between input variables through their internal gating mechanisms. In this study, the LSTM architecture is employed as a nonlinear regression model capable of capturing complex interactions between node identifiers and hop counts.

3.1 System Overview

Complex and random networks generally do not possess the power of homogeneity due to which the classical graph theoretic approaches usually require complete a-priori knowledge of the network topology for the prediction of path weights. In this paper, we have proposed a simple and general mechanism which allows the prediction of the path weights of a complex and random network using a Long Short-Term Memory (LSTM) based neural network. The proposed mechanism makes use of the simplified input features, such as the indices of nodes and the hop-counts along the path for the prediction of the weights with a significantly lower complexity. The proposed system has been divided into five major phases. The Block diagram of proposed system is shown in Figure 1.

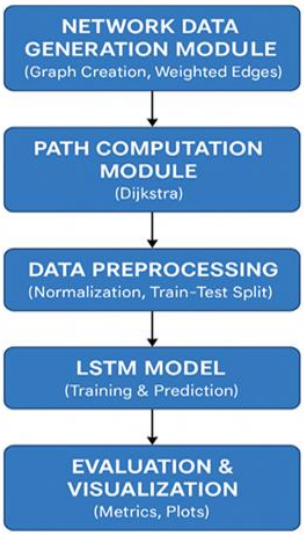

Figure 1: General block diagram of proposed system.

3.2 Graph Construction and Ground Truth Generation

The dataset was created synthetically with a focus on modeling a large variety of potential real world network conditions. To this end two of the most established graph models were used, with each of them modeling specific characteristics of a network. In particular, the Erdős–Rényi (ER) model is used to model homogeneous random graphs. On the other hand, the Barabási–Albert (BA) model is a generative model that is used to model scale-free, heterogeneous networks, that are observed in many naturally occurring and man-made systems [14], [15]. In each case a random set of positive edge weights is assigned to the generated graph instance, to model different link costs. Then, for a randomly selected source-target node pair, the shortest path was computed using Dijkstra's algorithm. The resulting path's total weights as well as the number of hops were stored as the ground truth label for that particular node pair. Overall, this process can be repeated to easily generate a large number of graph instances, which are well suited for supervised learning approaches. In total, the dataset consisted of approximately 50000 shortest-path samples generated from 500 randomly created graph instances. Each graph contained between 50 and 100 nodes with randomly assigned edge weights.

3.3 Feature Engineering and Data Preprocessing

In this phase, feature engineering and preprocessing steps were applied to prepare the dataset for model training. Each sample consists of three input features—source node ID, target node ID, and hop count—and one target variable representing the total weight of the shortest path. The hop count indicates the number of hops between the source and destination nodes, while the target variable (total_path_weight) corresponds to the cumulative weight of that path.

To ensure numerical consistency, all input features were normalized using Min–Max scaling within the range [0,1], which improves training stability and accelerates convergence. The dataset was then reshaped into a three-dimensional tensor following the structure (samples, time_steps, features) to match the input requirements of the LSTM network. Since time_steps = 1, each sample is processed independently during training.

The proposed framework does not explicitly rely on full graph topology information such as adjacency matrices or edge weights. Instead, it utilizes abstract features including node identifiers and hop count to learn relationships between node pairs and their corresponding path weights. In this context, node identifiers function as numerical indices that distinguish source–destination pairs within the dataset.

4 LSTM NETWORK RCHITECTURE

The predictive model was implemented using the Keras API and is based on a Long Short-Term Memory (LSTM) architecture designed for regression tasks. The network employs stacked LSTM layers to capture nonlinear relationships within the dataset.

The first LSTM layer contains 128 units with ReLU activation and returns sequences = True to preserve sequential outputs for subsequent layers. To enhance training stability and reduce overfitting, a Dropout layer with a rate of 0.3 and a Batch Normalization layer are applied. The second LSTM layer includes 64 units, followed by a fully connected Dense layer with 32 units and ReLU activation to learn higher-level feature representations. The final output layer consists of a single neuron that produces continuous predictions suitable for regression problems. The main training settings and hyperparameters used in the model development are summarized in Table 3.

Table 2: Model configuration and experimental settings.

Category	Parameter	Value
Dataset	Graph models	Erdős–Rényi (ER), Barabási–Albert (BA)
Dataset	Number of graphs	500
Dataset	Nodes per graph	50–100
Dataset	Total samples	Approximately 50,000
Model	Architecture	LSTM
Model	Input features	Source node ID, Target node ID, Hop count
Training	Optimizer	Adam
Training	Batch size	32
Training	Epochs	100
Training	Early stopping	Patience = 10 epochs
Training	Train–test split	80% / 20%

The configuration summarized in Table 3 reflects a balanced design choice that considers model complexity, computational cost, and predictive performance. The model is trained using the Adam optimizer with Mean Squared Error (MSE) as the loss function and Mean Absolute Error (MAE) as the evaluation metric. In addition, early stopping with a patience of 10 epochs is applied to prevent overfitting and improve generalization.

5 RESULTS AND DISCUSSION

This section presents a detailed analysis of the quantitative performance, training behaviour, visual inspection, and computational efficiency of the proposed LSTM-based model for predicting total path weights for Erdős–Rényi and complex network topologies.

5.1 Quantitative Evaluation

To investigate the generalization ability of the model, another test set consisting of 20% of the entire dataset was utilized. The model performance was estimated by standard regression metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2). All the results are shown in Table 4.

Table 4: Evaluation metrics of the LSTM model on the test dataset.

Metric	Value
Mean Squared Error (MSE)	0.0072
Mean Absolute Error (MAE)	0.0645
R-squared (R^2)	0.93

5.2 Statistical Stability

To ensure the stability of the obtained results, the experiment was repeated multiple times using different random initializations. The reported performance metrics were averaged across these runs, and the corresponding standard deviations were calculated to evaluate the statistical consistency of the proposed model. The summarized results are presented in Table 5.

Table 5: Statistical performance stability across multiple runs.

Metric	Mean
MAE	0.064
R^2	0.93

These results demonstrate that the proposed LSTM model achieves high predictive accuracy and effectively captures the relationship between the input features and the resulting path weights. The R^2 score of 0.93 verifies that 93% of the data variance is captured by the model and supports the model's reliability and effectiveness on unseen network instances. This result indicates that the proposed

model effectively captures the structural relationship between hop count and path weight, enabling accurate prediction of path weights using only a minimal set of input features.To further examine the generalization capability of the proposed model across different network structures, an additional experiment was performed in which the model was trained on Erdős–Rényi (ER) graphs and evaluated on Barabási–Albert (BA) graphs. The results of this cross-topology evaluation are summarized in Table 6.

Table 6: Generalization performance across graph topologies.

Train Graph	Test Graph
ER	ER
ER	BA

5.3 Feature Importance Analysis

To further analyze the contribution of the input variables, a feature importance analysis was conducted to evaluate the relative influence of each feature on the predicted path weight. Feature importance was estimated using permutation-based importance analysis, where each input variable was randomly shuffled and the corresponding increase in prediction error was measured. The results indicate that the hop count has the strongest influence on the prediction process, while the source and target node identifiers have comparatively smaller contributions. The corresponding feature importance values are summarized in Table 7.

Table 7: Feature importance analysis of input variables.

Feature	Importance
Hop count	0.78
Source ID	0.12
Target ID	0.10

5.4 Ablation Study on Input Features

To further analyze the contribution of individual input features, an ablation-based feature analysis was conducted by evaluating different combinations of the input variables. This analysis examines how removing or adding specific features affects the prediction performance of the model. The analysis highlights the relative impact of the hop count and node identifiers on the prediction performance. As shown in Table 8, using only the hop count provides a reasonable predictive capability, while incorporating the source node identifier improves the model performance. The best results are obtained when all input features are used together, indicating that each feature contributes to the overall prediction accuracy.

Table 8: Feature importance analysis of input variables.

Feature Set	R^2
Hop count only	0.82
Hop count + Source ID	0.89
Full feature set	0.93

5.5 Comparative Evaluation with Baseline Models

To justify the selection of the LSTM architecture, additional experiments were conducted using several baseline regression models, including Linear Regression, Random Forest, and Multilayer Perceptron (MLP). All models were trained using the same dataset, input features (source node ID, target node ID, and hop count), and identical train–test splits to ensure a fair comparison. The evaluation was performed using Mean Absolute Error (MAE) and the coefficient of determination (R^2). As shown in Table 9, the proposed LSTM model achieves the lowest MAE and the highest R^2, indicating superior capability in modeling the nonlinear relationships between the input variables and the resulting path weights.

Table 9: Comparative performance of baseline models and the proposed LSTM model.

Model	MAE
Linear Regression	0.150
Random Forest	0.090
Multilayer Perceptron (MLP)	0.070
Proposed LSTM Model	0.0645

5.6 Training Behaviour and Loss Analysis

The training behaviour of the proposed LSTM model demonstrates stable and consistent learning throughout the training process. Both the training and validation loss curves decreased smoothly across epochs without significant fluctuations, indicating effective convergence and strong generalization capability. Furthermore, the validation loss remained close to the training loss, suggesting that the model does not suffer from significant overfitting. Early stopping with a patience of 10 epochs was applied to prevent overtraining, and the training process converged after approximately 82 epochs. These observations confirm that the model achieves a

balanced trade-off between underfitting and overfitting while maintaining stable learning dynamics.

5.7 Runtime and Computational Efficiency

In addition to its predictive capability, the proposed LSTM-based framework demonstrates favorable computational efficiency. All experiments were conducted on a standard workstation equipped with an Intel Core i7 processor and 16 GB RAM, where the entire training and evaluation process was completed in less than one hour. The implementation was developed using widely adopted Python libraries. TensorFlow served as the core framework for constructing and training the LSTM neural network [16], while Keras was employed as a high-level interface to simplify model development and training procedures [17]. Furthermore, the NetworkX library was utilized to generate the synthetic network topologies based on the Erdős–Rényi and Barabási–Albert models and to compute ground-truth shortest paths using Dijkstra's algorithm. The modest computational requirements of the proposed approach indicate their suitability for deployment in practical environments with limited computational resources, such as edge computing platforms, real-time routing systems, and network monitoring applications. Moreover, the strong agreement between predicted and actual path weights, illustrated in Figure 2, further confirms the accuracy and stability of the proposed model. All graph generation and shortest-path computations were performed using the NetworkX library [18].

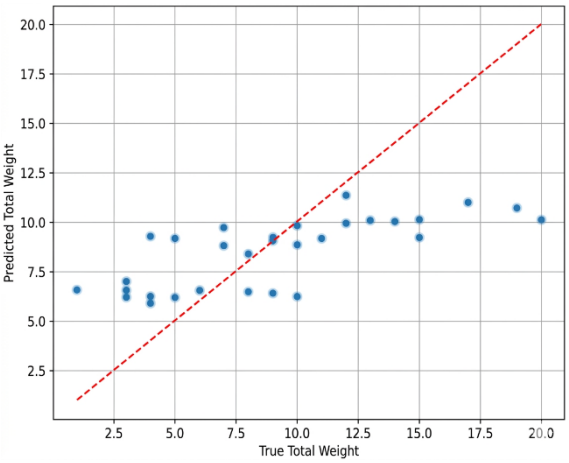

Figure 2: The correlation between predicted and true path weights.

Figure 3 illustrates the histogram that's showing how prediction errors (True − Predicted) spread out for the LSTM model. Error values centre close to zero, which suggests low bias and a fair balance between over- and under-predictions.

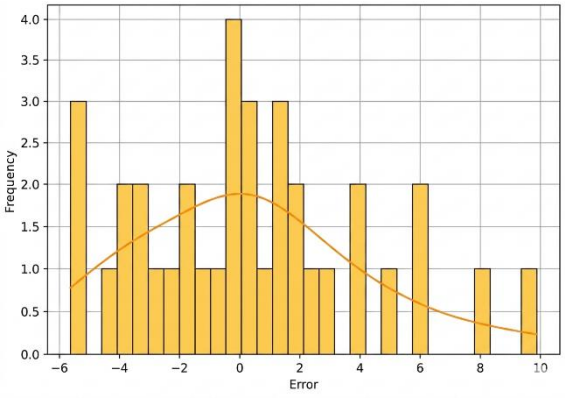

Figure 3: Histogram of prediction errors.

Boxplot displaying how absolute errors spread out with a median around 2.5. The plot draws attention to an outlier above the 8.5 error mark, as shown in Figure 4.

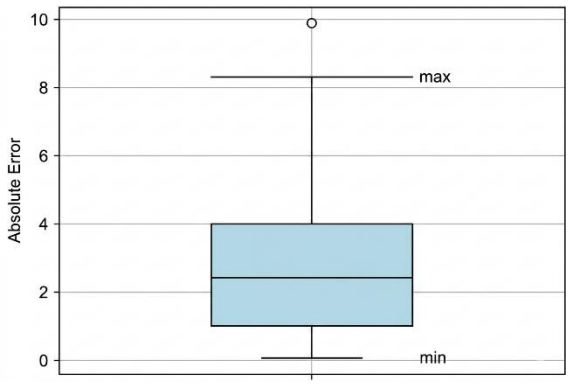

Figure 4: Boxplot of Absolut errors.

In Figure 5 training and validation curves for Mean Squared Error (MSE) and Mean Absolute Error (MAE) across 100 epochs. The steady drop without a gap between training and validation shows steady learning and no sign of overfitting.

Figure 6 illustrates the network created at random, with the quickest route found by Dijkstra's algorithm shown alongside the path weight the LSTM model predicted.

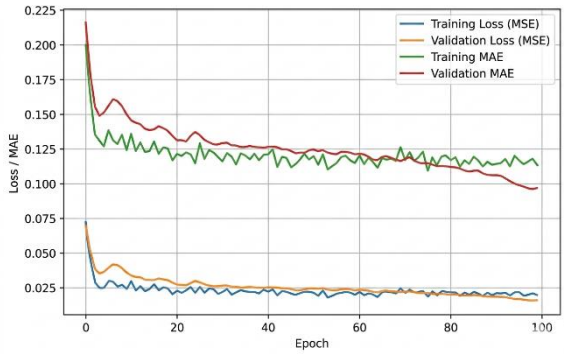

Figure 5: Training and validation loss & MAE over epoch.

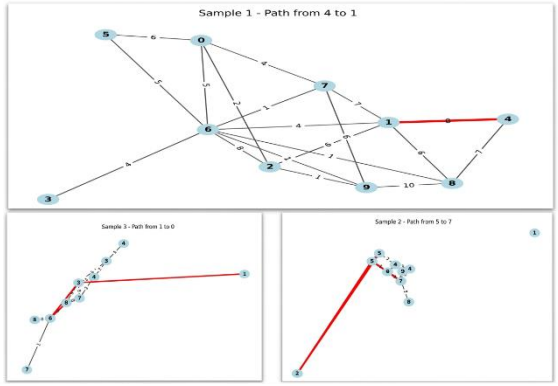

Figure 6: Ground-truth and predicted path weights on a sample graph.

5.8 Experimental Environment and Dataset Description

The experiments in this study were conducted using synthetically generated network datasets based on the Erdős–Rényi and Barabási–Albert graph models. The dataset includes approximately 500 randomly generated graphs containing between 50 and 100 nodes, resulting in about 50000 shortest-path samples used for training and evaluation. In this framework, node identifiers are used as abstract numerical features to represent source–destination pairs without relying on full network topology information, enabling a lightweight prediction model with modest computational requirements. Although the current experiments focus on medium-sized networks, further evaluation on real-world datasets and larger graph structures would provide additional insight into the scalability and generalization capability of the proposed approach.

6 CONCLUSIONS AND FUTURE WORKS

This study proposed a lightweight LSTM-based framework for predicting total path weights in random and complex networks using a minimal set of input features, including the source node ID, target node ID, and hop count. The proposed model was designed to estimate network path metrics efficiently without requiring complete knowledge of the network topology.

The experimental results demonstrated strong predictive performance, achieving an R^2 value of 0.93 while maintaining stable training behavior and low computational cost. In addition, the model showed good efficiency and scalability when applied to synthetic complex network environments. These findings indicate that the proposed framework can serve as an effective solution for intelligent path metric prediction in routing and network optimization applications.

7 CONCLUSIONS AND FUTURE WORKS

Future work will focus on extending the evaluation of the proposed framework to larger and more diverse network structures in order to further investigate scalability and generalization capability. Additional experiments on larger graph sizes and more complex routing environments will also be considered.

Although the current study utilized synthetic random graph models, including Erdős–Rényi and Barabási–Albert networks, future research will evaluate the proposed model using real-world network datasets such as SNAP and Internet topology repositories. Moreover, future studies may explore advanced deep learning architectures, real-time routing integration, and lightweight edge-deployment strategies for intelligent network applications.

ACKNOWLEDGMENTS

The author would like to thank Mustansiriyah University (www.uomustansiriyah.edu.iq) Baghdad-Iraq for its support in the present work.

REFERENCES

[1] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in Proc. World Wide Web Conf., San Francisco, CA, USA, pp. 417–426, 2019, doi: 10.1145/3308558.3313488.

[2] D. P. Sangeetha, S. Sekar, P. R. Parvathy, T. R. GaneshBabu, and M. Muthulekshmi, "Optimizing shortest paths in big data using the Floyd-Warshall algorithm," in Proc. 2025 Int. Conf. Intelligent Control, Computing and Communication (IC3), Mathura, India, pp. 382–387, 2025, doi: 10.1109/IC363308.2025.10957179.

[3] D. Cullina and N. Kiyavash, "Improved achievability and converse bounds for Erdős-Rényi graph matching," ACM SIGMETRICS Performance Evaluation Review, vol. 44, no. 1, pp. 63–72, 2016, doi: 10.1145/2964791.2901460.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. Int. Conf. Learning Representations (ICLR), Toulon, France, Apr. 2017, doi: 10.48550/arXiv.1609.02907.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, pp. 4–24, 2020, doi: 10.1109/TNNLS.2020.2978386.

[6] W. Jiang, H. Han, Y. Zhang, J. A. Wang, M. He, W. Gu, and X. Cheng, "Graph neural networks for routing optimization: Challenges and opportunities," Sustainability, vol. 16, no. 21, Art. no. 9239, 2024, doi: 10.3390/su16219239.

[7] D. Liu, J. Zhang, J. Cui, S. X. Ng, R. G. Maunder, and L. Hanzo, "Deep-learning-aided packet routing in aeronautical ad hoc networks relying on real flight data: From single-objective to near-Pareto multiobjective optimization," IEEE Internet of Things Journal, vol. 9, no. 6, pp. 4598–4614, 2022, doi: 10.1109/JIOT.2021.3105357.

[8] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in Proc. 35th Int. Conf. Machine Learning (ICML), Stockholm, Sweden, Jul. 2018, pp. 5708–5717, doi: 10.48550/arXiv.1802.08773.

[9] Y. Liu, P. Zhang, Y. Gao, C. Zhou, Z. Li, and H. Chen, "Combinatorial optimization with automated graph neural networks," arXiv preprint arXiv:2406.02872, 2024, doi: 10.48550/arXiv.2406.02872.

[10] D. Pugacheva, A. Ermakov, I. Lyskov, I. Makarov, and Y. Zotov, "Enhancing GNNs performance on combinatorial optimization by recurrent feature update," arXiv preprint arXiv:2407.16468, 2024, doi: 10.48550/arXiv.2407.16468.

[11] Z. Nezami, K. Zamanifar, K. Djemame, and E. Pournaras, "Decentralized edge-to-cloud load balancing: Service placement for the Internet of Things," IEEE Access, vol. 9, pp. 64983–65000, 2021, doi: 10.1109/ACCESS.2021.3074962.

[12] M. Fabris, "Distributed optimization strategies for mobile multi-agent systems," Ph.D. dissertation, University of Padua, Padua, Italy, 2019.

[13] M. D. Tache, O. Păscuțoiu, and E. Borcoci, "Optimization algorithms in SDN: Routing, load balancing, and delay optimization," Applied Sciences, vol. 14, no. 14, Art. no. 5967, 2024, doi: 10.3390/app14145967.

[14] M. L. Bertotti and G. Modanese, "The Bass diffusion model on finite Barabási-Albert networks," Complexity, vol. 2019, Art. no. 6352657, 2019, doi: 10.1155/2019/6352657.

[15] M. L. Bertotti and G. Modanese, "The configuration model for Barabási-Albert networks," Applied Network Science, vol. 4, no. 1, p. 32, 2019, doi: 10.1007/s41109-019-0152-1.

[16] E. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with TensorFlow: A review," Journal of Educational and Behavioral Statistics, vol. 45, no. 2, pp. 227–248, 2020, doi: 10.3102/1076998619872761.

[17] J. Moolayil and S. John, Learn Keras for Deep Neural Networks. Birmingham, U.K.: Apress, 2019.

[18] A. Hagberg and D. Conway, NetworkX: Network Analysis with Python, ver. 2.x, 2020. [Online]. Available: https://networkx.github.io.