Predictive Modeling of Loss Ratio for Congestion Control in IoT Networks Using Deep Learning

Hanane Benadji, Lynda Zitoune, Véronique Vèque

Predictive Modeling of Loss Ratio for Congestion Control in IoT Networks Using Deep Learning

Hanane Benadji*, Lynda Zitoune*, Véronique Vèque*
* Université Paris-Saclay,
CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France.
Email: {hanane.benadji, lynda.zitoune, veronique.veque}@centralesupelec.fr

***Abstract*—Congestion in the Internet of Things (IoT) networks arises when multiple flows share the same network, which can significantly impede the performance of IoT networks. This problem is exacerbated by the limitations of low-power lossy networks (LLNs), resulting in increased latency, high packet losses, reduced goodput, and other capacity-related issues. To ensure a high quality of service (QoS), and network reliability, it is crucial to implement effective congestion control mechanisms in IoT networks. Congestion in a network leads to an increase in packet losses. The loss ratio represents the proportion of lost packets to the total number of transmitted packets and is a critical metric for assessing the network's traffic load and congestion levels. This paper emphasizes the significance of studying IoT application-generated traffic to predict the loss ratio accurately. For instance, reliable data transfer is essential for IoT applications such as health monitoring, which are highly susceptible to performance degradation due to congested traffic and packet loss. This study proposes a novel approach that uses time series data and Deep Learning (DL) models to predict loss ratio in IoT networks. Our approach involves the implementation of a sliding window technique, as well as the validation and comparison of various DL models using data generated by the Cooja/Contiki framework.**

***Index Terms*—IoT Networks, Congestion Control, Network Traffic Analysis, Loss Ratio Prediction, Time Series Data, Sliding Window, and Deep Learning.**

I. Introduction

The growth of the Internet of Things (IoT) has led to a substantial increase in the number of connected devices, with projections of over more than 75 billion objects joining the Internet by 2030 [1], [2]. However, several factors, including application-specific traffic characteristics, limited processing, storage, and power capacity of IoT devices, can affect the operation of IoT networks [3]. Consequently, congestion has become a complex and unavoidable problem in these networks [4], resulting in a degradation of the quality of service (QoS): increased latency, packet loss, and reduced goodput [5]. Effectively managing congestion in IoT networks is crucial for ensuring efficient operation and achieving the desired QoS.

To address congestion in IoT networks, two transmission models have been developed: one uses TCP, such as Message Queuing Telemetry Transport (MQTT) [6], while the other as the Constrained Application Protocol (CoAP) uses User Datagram Protocol (UDP) [7]. These models are reactive congestion indicators composed of static rules [8] based on loss-based, delay-based, and hybrid-based [6]. For example, each time a loss occurs, the throughput is reduced. However, we argue that proactive resolution is more efficient and suggest the use of lightweight and adapted DL techniques to predict congestion indicators accurately [9]–[11].

To implement a reactive congestion control scheme for IoT networks. This study proposes using DL models based on time series data to predict congestion in IoT networks, using loss ratio variation as the indicator [5]. The aim is to assess the accuracy and precision of DL models to select the more appropriate model. This paper makes five significant contributions. Firstly, we collect a new dataset for the IoT networks using the Cooja/Contiki framework [12], and we preprocess the data to extract relevant features. Secondly, we design and train six DL models, namely including Recurrent Neural Networks (RNN), Gated Recurrent Units (GRU), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (Bi-LSTM), and Encoder-Decoder Long Short-Term Memory (ED-LSTM) for time series forecasting [13]. Thirdly, we configure the hyperparameters of the DL models. Fourthly, we investigate the influence of different window sizes on prediction accuracy and precision to enhance learning and prediction efficiency. Fifth, we evaluate the models using three standard performance metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE), by comparing the actual and predicted loss ratio values in the training and testing sets.

Outline: Section II summarizes some related work on applying DL approaches for network traffic management. Section III describes Time Series Forecasting for the loss ratio. Section IV sums up most DL models for Time Series Prediction. The following Section presents the experimental design and evaluates the DL models using a Sliding Window. Finally, section VI concludes the paper and gives some future work.

II. RELATED WORK

DL has gained widespread adoption in various fields due to its exceptional ability to handle complex patterns and relationships in sequential data [13], [14]. This makes it a preferred choice for modeling time series data, particularly in the network domain. In [15], the authors compared and classified Linear and nonlinear prediction models. The study aims to predict Traffic Matrix in Software Defined Networks (SDN) to manage the network. They proposed an LSTM-based framework called NeuTM after finding the hyperparameters of LSTM, which was shown a good prediction performance and converged quickly. In [16], the authors compared LSTM, ED-LSTM, and LSTM with the "Attention model" for predicting throughput in Long Term Evolution (LTE) networks to be used in time-critical applications. The results demonstrated that LSTM with Attention outperformed the other models, as RMSE and nRMSE metrics indicated. In [17], the authors compare the accuracy of two models, K-Nearest Neighbour (KNN) and LSTM techniques, to predict the Received Signal Strength (RSSI) and Packet Delivery Ratio (PDR) metrics to classify the link quality in IoT networks. However, the results show an overfitting, which could lead to erroneous prediction results that degrade the network's performance.

Recently, several research studies have investigated machine learning and DL to improve the TCP congestion control mechanism at the transport layer. These studies rely on congestion indicators that are grouped into three categories:

Loss-based: DL-TCP [9] is a modified congestion control algorithm designed for uplink traffic in disaster zones with mmWave signals. The algorithm uses LSTM to predict the blockage duration by considering factors such as mobility, location, and received Signal-to-Interference-plus-Noise Ratio (SINR) values of the TCP sender, achieving an approximate 90% probability of accuracy. Based on the blockage duration, proposed TCP congestion control algorithm effectively differentiates between temporary link disconnections and actual congestion.

Delay-based: In [10], the authors used The fixed-share Experts algorithm, which is a machine learning (ML) algorithm, to adjust the sending rate in wireless networks. The approach uses the Round Trip Time (RTT) values as an input to predict the next RTT in congested networks and update weights based on the difference between the estimated and actual RTTs. The results demonstrate that the model rapidly adapts to changes in Round-Trip Time, resulting in a significant reduction in the number of retransmitted packets and a notable improvement in goodput, particularly in highly congested scenarios.

Hybrid-based: Combine loss and delay techniques to predict the congestion status of 5G access networks, particularly mmWave links. In [11], the authors compare classification ML techniques to predict congestion status and investigate the impact of mmWave physical issues on transport layer metrics, including delay and inter-arrival time (IaT) at the receiver endpoint's transport layer and RLC (radio link control) buffer size. The authors categorized the comparison based on binary or multi-level classification. The simulation results showed the k-means algorithm is preferred for ML-based congestion control for mmwaves.

Overall, the LSTM model is the most cited [9], [14], [15], [17] in work aimed at improving the performance of networks due to its ability to retain information over extended periods. This study provides a comprehensive analysis and comparison of accuracy-based DL models to identify the most appropriate model for accurately predicting the loss ratio in IoT networks. This aims to implement a proactive congestion control protocol in IoT networks, making IoT devices intelligent with efficient and reliable communication adaptable to various environments and applications [8].

III. TIME SERIES FORECASTING

This section describes the steps to create Time Series Loss Ratio Forecasting in IoT networks.

A. Problem statement

In developing an efficient and accurate model in DL, the availability and sufficiency of training data play a crucial role. However, due to the scarcity of publicly accessible datasets for IoT networks, this study uses data generated from the realistic simulator Cooja/Contiki framework. Cooja/Contiki is widely used for evaluating and designing IoT networks [12]. This environment provides a realistic emulation of constrained devices with varying processing capabilities. In this work, two types of IoT devices, Wismote and Sky Mote, are considered to emulate various communication scenarios and derive the dataset. The motes embedded MSPSim [12], an accurate hardware emulator, and run Contiki, the open-source operating system for IoT devices. Table I summarizes the hardware specification embedded on the motes.

TABLE I: Cooja/ Contiki: hardware specifications of Wismote and Sky motes

Wismote specifications	
RAM	16KB
ROM	256KB
Micro-Controller	MSP430F5437
CPU Clock Speed	25Mhz
RADIO	CC2520 2.4GHz / 250Kbps data rate
Sky mote specifications	
RAM	10KB
ROM	48KB
Micro-Controller	MSP430F1611
CPU Clock Speed	8MHz
RADIO	CC2420 2.4GHz / 250Kbps data rate

B. Network Model

We simulate a network of 100 motes organized in a grid topology, including 98 CoAP senders, one server on

the Wismote devices, and an RPL border router on Sky Mote devices. Each sender generates periodic traffic of a message of size 112 bytes every ten seconds to the server, and it receives back from the server of a message size 86 bytes; this traffic represents a realistic traffic profile of a smart healthcare application, particularly the patient surveillance [3]. The CoAP protocol [7] is used as the congestion control protocol in the application layer. It is a lightweight protocol designed for IoT devices with limited resources and bandwidth, making it suitable for our simulation. Table II summarizes the protocol stack we consider. All the transactions between CoAP senders and the server are recorded in the log file that we parse to extract the loss ratio every 10 seconds and form the dataset. This log file is useful for examining and evaluating the network's performance regarding the CoAP parameters as in [5].

TABLE II: Cooja/Contiki: simulation parameters

Simulation parameters	
Application Layer	CoAP
Routing Layer	RPL (Routing Protocol for LLNs)
Transport Layer	UDP
Network Layer	6LoWPAN/ IPV6
MAC Layer	CSMA/ CA
Link/Physical Layer	IEEE 802.15.4
Radio Duty Cycling	ON ContikiMAC driver
PDR (Packet Delivery Ratio)	100%

C. Sliding Window Technique

In this study, the time series data represents the loss ratio, denoted as $X_T = \{X_1, X_2, X_3, ..., X_T\}$, which is a series of positives values collected at equally spaced time slots $T = \{1, 2, 3, ..., T\}$. This study aims to predict the loss ratio at time $t \in T$ using information from prior times, known as lag times or lags [13]. For example, the loss ratio at the previous time is X_{t-1}, and at the time before that is X_{t-2}. To achieve this goal, we start with preprocessing the obtained dataset [18], to prepare data for effective analysis or use. We perform data normalization to scale all dataset features to a comparable scale, enhancing DL models' performance and convergence. After that, we divide the time series data into two sets: the training and test sets are denoted as $N_k = \{X_1, X_2, X_3, ..., X_k\}$ and $M_k = \{X_{k+1}, X_{k+2}, X_{k+3}, ..., X_T\}$, respectively, where $k \in T$. The training set is used to train the DL models and find the optimal hyperparameters [19] for each model, while the test set is used to evaluate the model's accuracy and precision.

Then, we transformed the time series data, which consisted of observed loss ratios $\{X_1, X_2, X_3, \ldots, X_N\}$, into a supervised learning problem $f(X) = Y$, where X represents input variables and Y represents output variables. To achieve this, we employed the sliding window technique to transform the data into fixed-length sequences of size m, where m is less than the total number of samples (N). Specifically, we divided the data into fixed-length input sequences representing a sequence of loss ratio values and an output value representing the loss ratio to predict. We used time lags $T = 1$ to slide the window m along the time series. We repeated this process until we structured all the data into $S = N - m$ subsamples. Finally, we used the new data to train and evaluate DL models.

IV. Deep Learning Models

Given the remarkable capacity of DL techniques in effectively handling and extracting insights from extensive sequential data, significant research efforts have been directed towards harnessing DL to enhance network performance, as discussed in [14], [15], [17]. This paper proposes using DL models to predict loss ratios in IoT networks. Specifically, we introduce three distinct categories of DL models:

The first category is Basic RNN. RNNs have recurrent connections that enable the processing of new inputs while incorporating previous ones. This inherent ability for dynamic sequence modeling facilitates the detection of long-term patterns. In RNN, we employ the hyperbolic tangent (tanh) activation function. To stabilize the learning process in RNNs, we use the Root Mean Square Propagation (RMSprop) optimizer. Extending our exploration, we delve into LSTM, an extension of RNNs designed to mitigate short-term memory limitations. LSTM employs gates, including the forget gate, input gate, and output gate, to identify which data in a sequence is relevant. In LSTM, we employ the Rectified Linear Unit (ReLU) as an activation function and Adam optimizer. Another type of RNN used in this study is GRU, which offers a streamlined alternative to LSTM with fewer parameters. GRU architecture includes two fundamental gates, the update gate and the reset gate, making it less complex than LSTM. Employing the tanh activation function within the context of GRU, we optimize the learning process through the RMSprop optimizer.

The second category is Advanced RNN. Which includes Bi-LSTM and ED-LSTM. Bi-LSTM shares the same architecture as LSTM and introduces bidirectional processing by incorporating information from both forward and backward directions. This dynamic approach, implemented through two hidden layers, enables information propagation throughout the sequence at every step. Employing the tanh activation function, Bi-LSTM's optimization relies on the RMSprop optimizer. On the other hand, ED-LSTM emerges as an extension of the LSTM model, tailored to accommodate variable-length inputs and outputs. ED-LSTM comprises an encoder tasked with compressing time series data into compact vectors and a decoder responsible for restoring the original time series data from the compressed vectors. The activation function chosen for ED-LSTM is also the tanh, and the optimization strategy remains the RMSprop optimizer.

The third category is CNN. CNN comprises convolution and pooling layers, followed by fully connected

layers. Convolution and pooling layers extract features from the input sequence while reducing dimensions. Subsequently, fully connected layers employ these extracted features to generate predictions. In our study, ReLU is the chosen activation function within the CNN architecture. We employ the RMSprop optimizer for optimization purposes.

V. Experiments and Results

A. Experimental design

The generated dataset of 10000 observations, representing the loss ratio computed every 10 seconds, is preprocessed to obtain more accurate and consistent results when training the considered models. The dataset is scaled to the range of [0, 1] and divided into a training set to fit the models and a testing set to evaluate the models, representing a 60/40 split of the dataset [13]. Afterward, we transformed each training and test set into a state-space representation with varying W = 5, 10, 15, and 20 window sizes for each experiment, using a time lag of T = 1 for 1-step-ahead prediction purposes. We conducted a thorough hyperparameter tuning process for each DL model employed in our experiments. This involved evaluating different values for key hyperparameters, including the number of layers, hidden units, and activation functions, using the Grid Search technique [19]. Through this process, we identified the optimal parameter combinations that yielded the highest performance in loss ratio prediction for each DL model. Table III summarizes the crucial parameters for the DL models considered in our study. The term 'input' refers to the data fed into the considered learning models; a 'hidden unit' refers to a single node within a hidden layer in a neural network, and the term 'output' refers to the predicted value of a single size.

TABLE III: Hyperpamaters of the considered DL models

Models	Input	Hidden Layers	Hidden units	Output
LSTM	5-10-15-20	1	20	1
Bi-LSTM	5-10-15-20	1	10	1
ED-LSTM	5-10-15-20	2	20 for each layer	1
CNN	5-10-15-20	2	Layer 1: 64; Layer 2: 20	1
GRU	5-10-15-20	1	128	1
RNN	5-10-15-20	2	Layer 1: 64; Layer 2: 20	1

The models were constructed and trained in this work using the $Keras$ library [20]. The training was conducted on an Intel Core $i9$ machine with $32GB$ of memory.

B. Results and discussion

The results of the prediction models are based on ten experimental runs for both the training and testing datasets. We varied the window size to evaluate the impact of the amount of data used in each experiment. The mean and 95% confidence interval of the RMSE is displayed for each prediction model to assess robustness. A lower confidence interval indicates higher confidence in the prediction, as shown in Fig. 1. Table IV presents the results for RMSE, MAE, and MSE, described by the following formulas:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

y_i and $\hat{y}_i$ are the observed and the predicted values, respectively, and n represents the total number of predictions.

Table IV shows the values for all measurements. MAE values do not vary from model to model and remain stable because MAE is not very sensitive to large errors. In contrast, MSE penalizes large errors more severely than MAE, and RMSE rescales the errors computed by MSE, making it easier to read and classify models according to their accuracy.

Fig. 1 presents the results for different window sizes, including 5, 10, 15, and 20. For a window size of 5 (Fig. 1a), the best training and test set performance is observed for ED-LSTM and RNN, with an RMSE of around 0.09 for both training and test sets. Increasing the window size to 10 (Fig. 1b) maintains a similar performance to a window size of 5 for all models, except for ED-LSTM, which shows an improvement in the training set with an RMSE of around 0.08. The performance of the LSTM, RNN, and GRU models remains consistent with a window size of 15 (Fig. 1c), while the Bi-LSTM model's performance is slightly better with a window size of 10. The CNN model's performance does not improve with an increase in the window size. On the other hand, ED-LSTM model performance slightly improves with a window size of 15, with an RMSE of approximately 0.08 for the training set. For a window size of 20 (Fig. 1d), the performance of LSTM, Bi-LSTM, CNN, RNN, and GRU models remains the same as with a window size of 15. ED-LSTM continues to demonstrate the best overall performance, exhibiting an RMSE of less than 0.08 in training and less than 0.1 in testing sets.

The experimental results demonstrate that all DL models perform similarly regarding TrainRMSE and TestRMSE. This finding confirms that our dataset's split dataset ratio of 60/40 is sufficient for training and testing the DL models.

Moreover, the results show that the ED-LSTM model is a robust and effective tool for predicting loss ratio, followed by RNN, GRU, Bi-LSTM, and LSTM models. To assess the accuracy of the trained ED-LSTM model,

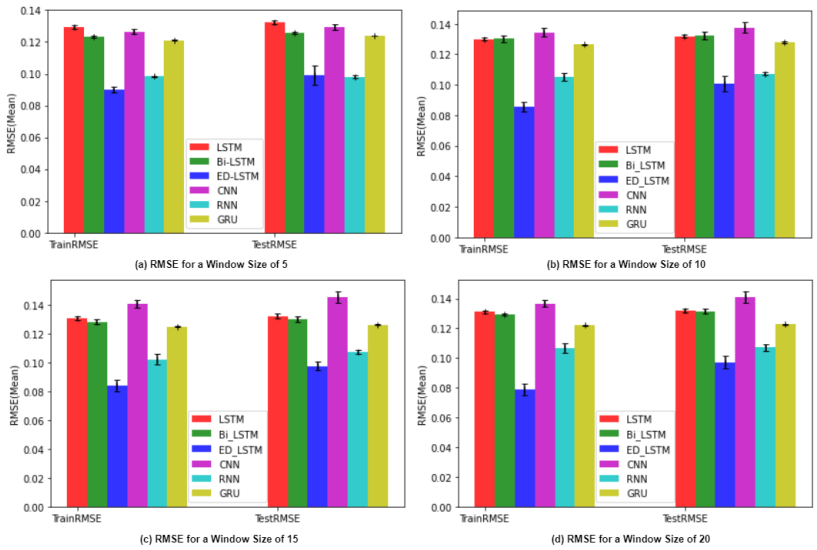

Fig. 1: RMSE evalution: mean and 95% confidence interval.

TABLE IV: RMSE, MAE, and MSE on training and test sets Vs window size.

Window size of 5						
Metrics	LSTM	Bi-LSTM	ED-LSTM	CNN	RNN	GRU
TrainRMSE	0.129584	0.123411	**0.090034**	0.126525	**0.09822**	0.12116
TestRMSE	0.132458	0.125601	**0.099007**	0.129314	**0.098189**	0.123887
TrainMAE	0.052748	0.053687	0.053865	0.052185	0.055990	0.067429
TestMAE	0.054163	0.054801	0.055128	0.054658	0.056765	0.069833
TrainMSE	0.008285	0.007950	0.008036	0.008311	0.008114	0.011634
TestMSE	0.008165	0.007950	0.007887	0.008599	0.007785	0.011851
Window size of 10						
TrainRMSE	0.130003	0.13022	**0.085395**	0.134392	0.105121	0.126332
TestRMSE	0.131772	0.132065	**0.100905**	0.137622	0.107236	0.127948
TrainMAE	0.052577	0.053127	0.053724	0.054022	0.061215	0.068961
TestMAE	0.053420	0.052594	0.055002	0.056246	0.061757	0.070966
TrainMSE	0.008335	0.007913	0.008517	0.008598	0.010168	0.013009
TestMSE	0.007948	0.007474	0.008294	0.008650	0.009751	0.013077
Window size of 15						
TrainRMSE	0.130708	0.128324	**0.083739**	0.140897	0.102301	0.124868
TestRMSE	0.132282	0.12999	**0.097578**	0.145435	0.107454	0.126255
TrainMAE	0.052085	0.052945	0.053392	0.053706	0.058417	0.072884
TestMAE	0.053104	0.053756	0.054461	0.058958	0.058997	0.074609
TrainMSE	0.008085	0.008306	0.008453	0.008284	0.009768	0.012656
TestMSE	0.007933	0.007966	0.008191	0.009752	0.009350	0.012750
Window size of 20						
TrainRMSE	0.131071	0.129455	**0.078683**	0.136817	0.106735	0.122068
TestRMSE	0.131997	0.131587	**0.097312**	0.14127	0.10719	0.122597
TrainMAE	0.051156	0.052159	0.054183	0.051836	0.055597	0.076215
TestMAE	0.051592	0.051731	0.054753	0.056189	0.055516	0.077172
TrainMSE	0.007980	0.007970	0.008193	0.007568	0.008315	0.011992
TestMSE	0.007630	0.007318	0.007844	0.008450	0.007665	0.011896

we compared it to the LSTM model using new loss ratio data of 2000 observations that we collected by simulating a network of 14 motes organized in a ring topology, as in [5], using the Cooja/Contiki framework. We employed a window size of 5 for making predictions and compared the actual loss ratio values with the values predicted by the ED-LSTM and LSTM models in Figs. 2 and 3. The results demonstrate that both models exhibit strong performance; however, the ED-LSTM model outperforms the LSTM model, achieving a lower RMSE value of 0.06 compared to 0.07. Furthermore, the ED-LSTM model accurately predicts loss ratio values ranging from 0.2 to 0.6 (Fig. 2).

To further evaluate the accuracy of ED-LSTM and LSTM models, we conducted an in-depth analysis by generating error histograms, as shown in Figs. 4 and 5. The error histogram calculates the absolute difference between the predicted and actual values. Analyzing the error histograms reveals that both models exhibit a significant proportion of observations with 0 errors, accounting for 0.6 of the total. This finding suggests that both models accurately predict most observations. Approximately a 0.1 proportion of observations with errors falling within 0 to 0.2. This indicates that some observations exhibit rela-

tively minor errors within this range. However, a smaller proportion of observations with errors are 0.2 to 0.6. This suggests that errors within this range are infrequent in the predictions generated by both models. Additionally, the error histogram reveals a predominant proportion of observations with 0 errors, and only a limited number of observations exhibit errors exceeding 0. For both models, the distribution of errors follows an exponential pattern. Based on these results, we can conclude that the close correspondence between the predicted and actual values suggests the absence of overfitting or underfitting. This indicates that the ED-LSTM model is a dependable and precise tool for accurately predicting loss ratio in IoT networks.

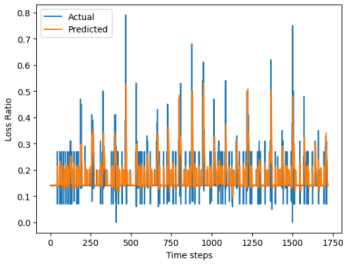

Fig. 2: Actual vs. predicted values: ED-LSTM Model.

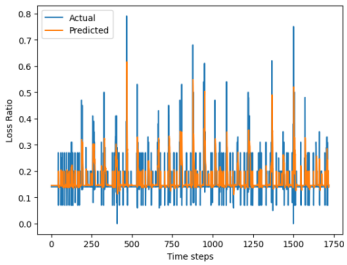

Fig. 3: Actual vs. predicted values: LSTM Model.

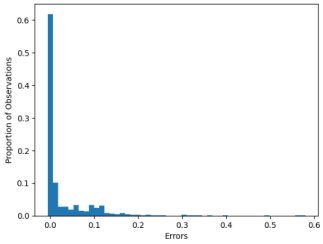

Fig. 4: Histogram of Errors: ED-LSTM Model.

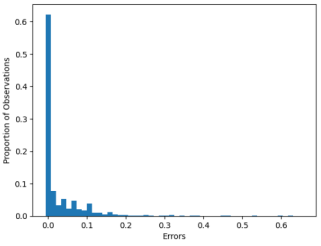

Fig. 5: Histogram of Errors: LSTM Model.

VI. Conclusion and Future Work

In the context of this study, our investigation into the efficacy of DL models for loss ratio prediction in IoT networks using time series data and the sliding window approach. Our findings underscore the ED-LSTM model's consistent superiority in accurately forecasting loss ratio within IoT networks. The ED-LSTM's remarkable performance is attributed to its specific architectural composition, notably its adeptness at capturing intricate temporal dependencies within sequences of variable lengths. This proficiency enables it to grasp and adapt to the intricate dynamics intrinsic to IoT data. This competitive edge firmly distinguishes the ED-LSTM model, rendering it a robust instrument for loss ratio prediction.

In future work, we aim to design a proactive congestion control solution based on our ED-LSTM model. This solution significantly improves network performance and is adaptable to various environments and IoT applications.

References

[1] H. Tahaei, *et al.*, "The rise of traffic classification in iot networks: A survey," *Journal of Network and Computer Applications*, vol. 154, p. 102538, 2020.

[2] L. Columbus, "Roundup of internet of things forecasts and market estimates, 2016," *Forbes. com, November 27th*, 2016.

[3] J. Mocnej, *et al.*, *Network Traffic Characteristics of the IoT Application Use Cases*, ser. Technical report series. School of Engineering and Computer Science, Victoria University of Wellington, 2018.

[4] N. Mishra, *et al.*, "An analysis of IoT congestion control policies," *Procedia Computer Science*, vol. 132, pp. 444–450, 2018.

[5] H. Benadji, *et al.*, "Performance evaluation and congestion analysis in iot network," in *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2022, pp. 232–237.

[6] B. H. Çorak, *et al.*, "Comparative analysis of IoT communication protocols," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, 2018, pp. 1–6.

[7] Z. Shelby, *et al.*, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014.

[8] P. K. Donta and S. Dustdar, "Towards intelligent data protocols for the edge," in *IEEE International Conference on Edge Computing and Communications (EDGE)*. IEEE, 2023, pp. 1–9.

[9] W. Na, *et al.*, "DL-TCP: Deep learning-based transmission control protocol for disaster 5g mmWave networks," *IEEE Access*, vol. 7, pp. 145 134–145 144.

[10] A. Nunes, *et al.*, "A machine learning framework for TCP round-trip time estimation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, p. 47, 2014.

[11] L. Diez, *et al.*, "Can we exploit machine learning to predict congestion over mmWave 5g channels?" *Applied Sciences*, vol. 10, no. 18, p. 6164, 2020.

[12] A. Kurniawan, *Practical Contiki-NG*. Apress.

[13] R. Chandra, *et al.*, "Evaluation of deep learning models for multi-step ahead time series prediction," vol. 9, pp. 83 105–83 123, 2021.

[14] O. Aouedi, *et al.*, "Intelligent traffic management in next-generation networks," *Future Internet*, vol. 14, no. 2, p. 44, 2022.

[15] A. Azzouni and G. Pujolle, "NeuTM: A neural network-based framework for traffic matrix prediction in sdn," in *2018 IFIP Network Operations and Management Symposium (NOMS)*, 2018, pp. 1–5.

[16] H. Na, *et al.*, "LSTM-based throughput prediction for LTE networks," *ICT Express*, pp. 1–6, 2021.

[17] C. Boucetta, *et al.*, "QoS in IoT networks based on link quality prediction," in *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6.

[18] S. García, *et al.*, *Data Preprocessing in Data Mining*, ser. Intelligent Systems Reference Library. Springer International Publishing, 2015, vol. 72.

[19] J. Brownlee, *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.

[20] https://keras.io/.