LSTM-Based Proactive Congestion Management for Internet of Vehicle Networks

Aly Sabri Abdalla[1], Ahmad Al-Kabbany[2], Ehab F. Badran[2], and Vuk Marojevic[1]
[1] Department of Electrical and Computer Engineering, Mississippi State University, MS 39762, USA
[2] Department of Electronics and Communications Engineering, Arab Academy for Science, Technology, and Maritime Transport, Alexandria, Egypt

***Abstract*—Vehicle-to-everything (V2X) networks support a variety of safety, entertainment, and commercial applications. This is realized by applying the principles of the Internet of Vehicles (IoV) to facilitate connectivity among vehicles and between vehicles and roadside units (RSUs). Network congestion management is essential for IoVs and it represents a significant concern due to its impact on improving the efficiency of transportation systems and providing reliable communication among vehicles for the timely delivery of safety-critical packets. This paper introduces a framework for proactive congestion management for IoV networks. We generate congestion scenarios and a data set to predict the congestion using LSTM. We present the framework and the packet congestion dataset. Simulation results using SUMO with NS3 demonstrate the effectiveness of the framework for forecasting IoV network congestion and clustering/prioritizing packets employing recurrent neural networks.**

***Index Terms*—V2X, IoV, Safety, Congestion Management, Machine Learning, NS3, SUMO, LSTM.**

I. Introduction

Vehicle-to-everything (V2X) communication is a new generation of wireless technology that enables vehicles to interact with the surrounding infrastructure and other mobile units on the road. The vehicles can communicate under the V2X umbrella in four modes: vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-network (V2N), and vehicle-to-pedestrian (V2P) [1]. These modes of communication offer cooperative awareness to provide intelligent transportation services. There are two major V2X standards: Dedicated Short Range Communications (DSRC) and cellular-based V2X (C-V2X). DSRC is standardized by the IEEE, whereas C-V2X is defined by the Third Generation Partnership Project (3GPP) in the context of 4G and 5G networks [2].

The Internet of Vehicles (IoV) emerged for linking vehicles together through networks of interconnected units that communicate with one another and with the surrounding infrastructure. The IoV has the potential to reduce congestion, improve traffic flow, reduce road accidents, and improve overall transportation safety and efficiency through connectivity enabling real-time situational awareness. This can be done by exchanging safety-related information through the basic safety message (BSM). The BSM is broadcast by a DSRC or C-V2X device to inform other nearby vehicles about vehicle position, speed, and so forth, as well as any circumstance that has been detected on the road. These messages are short, single-message transmissions that are sent regularly by all vehicles [3].

Mechanisms to detect, avoid, and manage packets congestion to alleviate vehicle congestion is of critical importance as mostly the road traffic congestion is leading to packet congestion via the increased communication demand within close proximity of vehicles. V2X congestion management mechanisms can be categorized into five classes: rate-based, power-based, carrier sense multiple access (CSMA)/collision avoidance (CA)-based, priority and scheduling-based, and hybrid [4]. Several work items have discussed this problem in the literature [5]–[9]. In [5], the authors aim to anticipate road hazards for V2X via Long Short-Term Memory (LSTM) to enhance road safety. The authors of [6] proposed stacked LSTM for improving the traffic conditions of vehicular networks. The work presented in [7] introduces an accumulative approach to time series prediction based on LSTM, aiming to improve the accuracy of predictions. The study in [8] integrates convolutional neural network (CNN) and LSTM models to classify various Transmission Control Protocol (TCP) congestion control algorithms, enhancing the understanding and management of V2X congestion control. In [9], the authors studied a hybrid congestion control for allocating traffic by employing a fusion model for traffic prediction.

While these work items have explored the use of LSTM for congestion control for vehicular networks, the integration of LSTM for congestion protection with the packet prioritization for proactive congestion management to improve delay and minimize dropped safety packets has not been well addressed. Therefore, this work introduces a proactive open-loop prioritized scheduling scheme that predicts IoV channel congestion to avoid such conditions. Using a LSTM for real-time forecasting and a historical congestion database with selective features, the proposed method ensures accurate prediction results. It classifies packets into two categories, ensuring high-priority safety messages are transmitted without delay even in congested scenarios. The key contributions of this paper are detailed further

- We design and implement a real-time framework for proactive network congestion management for IoV. Congestion events are predicted before they occur to facilitate packet scheduling actions that enhance the delivery ratio of the most important packets.
- Through the course of this study, an enormous dataset is generated using Network Simulator Version 3 (NS3), which contains the congestion channel parameters to be

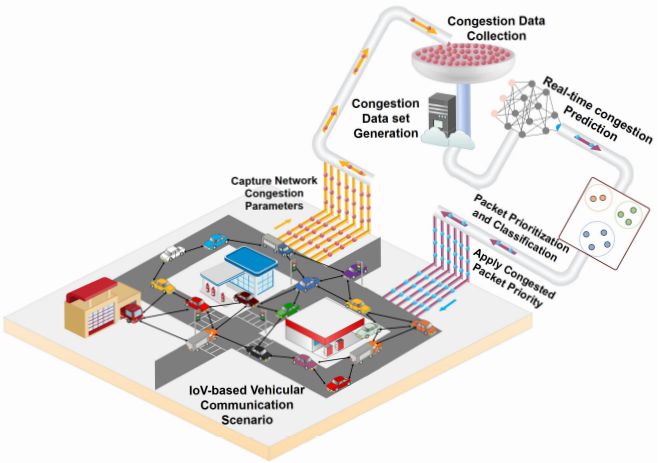

Fig. 1: IoV-based vehicular networks congestion scenario with the proposed solution components.

used as historical entries for the prediction phase.

- A LSTM architecture is designed to maximize the accuracy of the prediction.
- The K-means clustering technique is employed to cluster packets into two classes and minimize transmission delay of safety-critical packets in congested environments.

The structure of the paper is given as: Section II discusses the proposed multi-tier strategy for congestion prediction and packet classification. Section III numerically analyzes the performance. Section IV summarizes the main takeaways and Section V provides the concluding remarks.

II. PROPOSED SOLUTION

The increasing population and expanding metropolitan areas has led to increased road congestion. Vehicular traffic congestion can lead to network congestion which severely impacts the network performance. Furthermore, the degradation of link reliability and packet delivery ratio due to a congested channel can have dire road safety consequences. As a result, it is critical to be able to handle congested situations in vehicular networks and provide the highest safety levels for vehicles and pedestrians. The problem focuses on developing an effective framework for channel congestion prediction enabling real-time adjustments that are protocol agnostic and scalable.

We introduce a framework to address real-time congestion prediction and packet classification in IoV networks. We utilize LSTM, known for its capability to capture temporal dependencies in traffic data, to predict network congestion. We implement stacked LSTM allowing the model to learn hierarchical traffic patterns through multiple LSTM layers to improve prediction accuracy. Additionally, we introduce K-means clustering for packet classification based on key features extracted from simulation data, enabling dynamic prioritization and timely delivery of safety-critical messages.

Fig. 1 illustrates the main components of the proposed solution. The congestion management framework consists of three phases. Fig. 2 depicts the flow. First, the IoV-based vehicular communication network scenario is practically implemented and the congestion parameters are captured under different conditions to create data set. The generated data set is used as inputs to a neural network that predicts future crowded events in real time. Finally, the system assigns distinguished priorities depending on the content of the messages, which will assist the packet clustering during network congestion.

A. Congestion data set generation

The aim of this stage is to detect and store the congestion parameters of vehicles in the IoV-based vehicular network in a dataset. The generated dataset contains the number of dropped packets and the time of occurrence of these packet drops. We are utilizing the NS3 as it incorporates realistic traffic models and mobility patterns that closely mimic real-world conditions. NS3 is widely recognized for testing V2X protocols because of its accuracy in emulating vehicular networks [10], [11]. The process is composed of three sub-processes: We first create an IoV-based vehicular communication scenario where the vehicles start transmitting packets under neutral congestion conditions. The IoV-based communication environment will be implemented through via multi-objective functions, such as initialization of grid parameters, creation of vehicles, and implementation of agents and traffic via a traffic simulatoras Simulation of Urban MObility (SUMO) tool.

The NS3 software applies a variable congestion rate which affects the transmission delay and packet delivery ratio of the system. The variable congestion rate is employed in our model by using a Markov chain, which can be referred to as the multi-state Markov chain error model. It is implemented to show the transition probabilities of congestion rates in matrix format where the rows reflect the current state and columns the next state of congestion, where these transitions are probabilistically determined, affecting transmission delay and packet delivery ratio. Finally, for documentation purposes, the dataset measurements captured over the simulation time that is divided into a number of time slots within each time slot there will be a different congestion rate, also packet delivery ratio will be monitored and the number of drop packets will be counted, which makes the dataset more representative of real-world situations. Fig. 1 shows the key components of the proposed solution including developing the IoV-based vehicular congestion dataset, real-time congestion prediction, and packet prioritization and classification. NS3 includes some functions to terminate and avoid congestion in the network, such as adaptive congestion window size and back-off timer which controls the retransmission timer period.

This paper's focus is on designing multiple functions for monitoring several factors, which give the direct indication on the number of dropped packets during the simulation time. The main modified function to generate the dataset is the Congestion Action Difference (CongActDiff) counter. CongActDiff counts how many congestion flags were activated during a predefined interval within the overall simulation scenario. The congestion flag is enabled or activated when NS3 starts the process of changing the congestion window phase from a slow start to a fast retransmission phase. Moreover, modification of the congestion window phase occurs when the size of the congestion window becomes higher than a threshold

as an indicator of the increase of the number of retransmitted packets. Algorithm 1 presents the congestion action function scenario. The simulation scenario is generated by executing the tool command language (TCL) script, which does the node initialization, variable congestion rate implementation, and congestion dataset entry storage. The observation of the final results can be done graphically by plotting CongActDiff over the simulation time or by checking the saved dataset. This dataset consists of 25 records where each record is captured over 18 hours and covers 12,961,249 observations.

Algorithm 1 CongActDiff Algorithm

Precondition: *Allocate Packet P at transmitter side*
Initialize:
  $CongAct,\ CongInit,\ Packs,$
  $CongActTime,\ and\ CongDiff = 0$
  $PackTH = 350$
for *each P in Packets* **do**
  if *congestion* $flag[P] \leftarrow$ true &&
    $P \neq\ re-transmitting\ packet$ **then**
    $CongAct \leftarrow CongAct\ +\ 1$
    $CongActTime \leftarrow instance()\ .\ clock()$
    congestion $flag \leftarrow$ false
  end if
  $Packs \leftarrow Packs\ +\ 1$
  if $Packs\ counter =\ PackTH$ **then**
    $CongDiff = CongAct - CongInit$
    $CongInit\ =\ CongAct$
  end if
end for
return $CongDiff\ ,\ CongActTime$

B. Real-Time Congestion Prediction

The prediction phase reiles on two elements to enable proactive actions in real-time: Firstly, the previous crowdedness log from networked vehicles, which contains the CongActDiff counter and the occurrence time during the day, is obtained. Secondly, the historical congestion dataset is fitted to the LSTM [12]. In this paper, the aim is to reach the optimum structure of the LSTM network to achieve accurate real-time prediction readings from many perspectives. LSTMs are particularly well-suited for handling time-series data generated by vehicular networks, as they can capture long-term dependencies and complex temporal patterns. The LSTMs, with forget, input, and output gates, address the vanishing and exploding gradient problems, ensuring stable training. Additionally, stacked LSTM layers enhance the model's capacity to learn hierarchical features, leading to improved real-time prediction accuracy. This capability is essential for proactive congestion management in IoV environments. Recent studies have demonstrated the effectiveness of LSTMs in similar applications [5]–[9]

LSTM models can be classified as a function of the batch size, the number of epochs, and the hidden layers. An LSTM with multiple hidden layers is referred to as a stacked LSTM. The stacked LSTM model can be seen as a representational optimization, where the added layers are used to defragment the learned features of prior layers and generate high-level representations. The second concern of the LSTM model structure is the size of the batch, which controls the update of the model weights. Because real-time congestion forecasting needs an enormous historical log of previous crowded events, which cannot be processed by the LSTM model at once, the dataset is divided into small groups called batches. The batch size controls the tradeoff between the training speed and the learning process. The third feature of the LSTM model is the effect of the number of epochs on the real-time prediction process. In particular, when the model is exposed to the same random samples of the dataset many times, the model may memorize it. As a result, the LSTM network becomes over-fitted. Too few numbers of epochs, on the other hand, may lead to under-fitting. The appropriate choices for the epoch and batch sizes is presented in the next section. This solution for the congestion prediction problem is classified as an LSTM sequence-to-sequence $(seq2seq)$ prediction, where the input and output are time series sequences.

Fig. 3 shows the internal structure of LSTM *Memory Block*. We denote C_t and h_t as the cell state and final hidden state at time t, respectively. We consider x_t the vector of the input sequence at time step t. Eq. 1 to Eq. 6 shows how the single LSTM works, while Eq. 7 generalizes this to the stacked LSTM [6]. First, the computation of the forget and input gate vectors are given as follows:

$$f_t = \sigma(W_1^f.x_t + W_h^f.h_{t-1} + b_f), \tag{1}$$

$$i_t = \sigma(W_1^i.x_t + W_h^i.h_{t-1} + b_i), \tag{2}$$

Then, we compute a vector of new candidate values that can be added to the cell state:

$$\tilde{C}_t = tanh(W_1^C.x_t + W_h^C.h_{t-1} + b_C), \tag{3}$$

The next step is to update the new cell status C_t with the new information and the calculated gate vectors:

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1}, \tag{4}$$

Then, the output gate determines the required portion of the cell state will be at the output:

$$o_t = \sigma(W_1^o.x_t + W_h^o.h_{t-1} + b_o), \tag{5}$$

Finally, we compute the hidden state at the output using the output gate vector and the cell state:

$$h_t = o_t * tanh(C_t) \tag{6}$$

The connection between the input x_t and three LSTM gates and cell input is controlled by weight matrices W_1^i, W_1^f, W_1^o, and W_1^C, while W_h^i, W_h^f, W_h^o, and W_h^C are weight matrices adjusting the link between previous hidden layer state h_{t-1} and three gates plus cell input. The b_i, b_f, b_o, and b_C are bias factors of input, forget, output, and cell input respectively, a sigmoid activation function used in LSTM calculation and $tanh$ expresses the hyperbolic tangent function which used to overcome the vanishing gradient problem. In a stacked LSTM, these operations are repeated for each l layer:

$$h_t^l, C_t^l = \text{LSTM}(x_t, h_{t-1}^{l-1}, C_t^{l-1}) \tag{7}$$

C. Packet Prioritization and Classification

Congestion scenarios and channels can effectively decrease the reliability of the system. The previous crowdedness disadvantage has a damageable weight on the packet delivery ratio. The third procedure proposed in our system is crystallizing the importance of delivering safety message in congestion conditions. The predicted congestion facilitates system flexibility by giving higher priority to safety critical messages for improving their delivery ratio in congested conditions.

Prioritization of packets is based on the content of the message by adding a priority field to the TCP header within the creation of packets in the Medium Access Control (MAC)

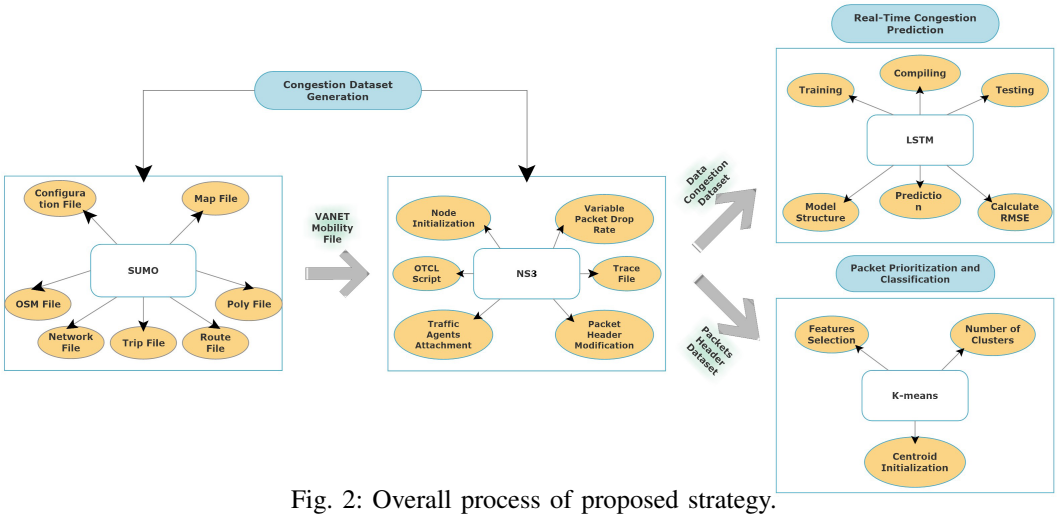

Fig. 2: Overall process of proposed strategy.

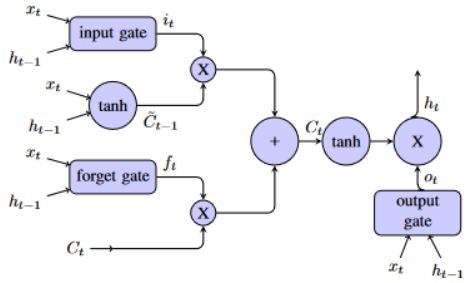

Fig. 3: LSTM *Memory Block* structure.

layer before transmission. A classification technique is then used to classify the flow of packets that will face congestion in the channel based on the priority and time to live (TTL) fields in the packet header. The TTL field defines the remaining hops before the intermediate node. If any node at any time receives a packet with TTL equaling zero then this packet will be dropped to avoid formulating loops inside the network. By finding the almost expired and highest priority safety message, this packet is transmitted with minimum hops and delays.

The K-means unsupervised machine learning algorithm is implemented for clustering packets [13]. It is a widely used clustering solution because of its speed of convergence, efficient handling of large dimension features, and simplicity of implementation [13]. K-means classifies packets upon training with the large dataset generated for header fields of packets during the simulation time. NS3 by default does not generate an output file containing any information about the packet header. While the TTL field is already defined as one of the packet header fields, the other classification feature (priority) needs to be added to the header structure of NS3. The dataset containing the modified packet header fields after the NS3 modifications can then be fed to the K-means.

The K-means algorithm is utilized for classifying packets based on features extracted from simulation data, such as packet size, TTL, priority, generation time, and hop count. This algorithm is selected for its efficiency and simplicity in handling large datasets. Initially, centroids are chosen either randomly or using the K-means++ method to ensure optimal starting points. Each packet is then assigned to the nearest centroid based on Euclidean distance, forming clusters. The centroids are iteratively updated by calculating the mean of the packets in each cluster until convergence is achieved. The performance of the clustering is validated using the Silhouette Score, ensuring well-defined and distinct clusters. This approach enables dynamic prioritization and timely delivery of safety-critical messages, enhancing the overall effectiveness of the congestion management framework.

The new packet header trace contains header fields, such as source IP, destination IP, TTL, packet size, and recently added field priority. Algorithm 2 indicates the steps needed to add the new priority field to the IP header of the generated packets at the transmitter. Without loss of generality, the condition that define safety packets used in our algorithm is that the size is below 100 bytes. Furthermore, the defined safety message will take random priority between 1 to 10, on the other hand, normal packets will have priority from 11 to 20.

Algorithm 2 Priority Field Algorithm

Precondition: *Allocate Packet P at transmitter side*
for *each P in Packets* **do**
  $PH \leftarrow hdr\_access(p)$
  if $PH \rightarrow ptype(P) \leftarrow Routing$ **then**
    if $PH \rightarrow size(P) <= 100$ **then**
      $PH \rightarrow prio(P) = random(1\ 10)$
    else
      $PH \rightarrow prio(P) = random(11\ 20)$
    end if
  end if
end for

III. PERFORMANCE EVALUATION

Fig. 2 shows the stages of the proposed solution and the operations that occur in each of the respective stages. This framework is implemented employing different software tools including SUMO, NS3, LSTM, and K-Means. The proposed framework executes in a centralized node in an IoV network and can be integrated into radio access network (RAN) intelligent controllers of future Open-RAN systems [14] or in the form of O-RAN distributed learning [15].

A. Simulation Parameters

SUMO is used to generate mobility patterns of vehicles [16]. The scenario of this paper is based on a real map extracted from OpenStreetMap (OSM) for the Arab Academy for Science and Technology (AAST) campus in Alexandria, Egypt. The simulation area is two dimensional 1530 m length

and 1052 m width with 93 junctions and 134 single lane route. The simulation scenario employs the two-ray ground radio propagation model with omnidirectional antennas and the 802.11p MAC protocol for IoV communications. The packets are routed between vehicles using the ad hoc on-demand distance vector (AODV) routing protocol. The number of vehicles simulated in this scenario is 150 driving at a fixed speed of 25 kilometers per hour. The simulation time is 450 hours. We evaluate our framework for four congestion stages of 0.1, 0.35, 0.2 and 0.3 packet drop rates. The next stage is selected using the Markov chain matrix mechanism.

B. LSTM Network Structure and Results

The LSTM is fed with the generated dataset from the previous steps. The data set is partitioned as 70% and 30% for training and testing. We consider different LSTM structures based on the size of the batch, number of epochs and the number of hidden layers. Table I shows the obtained training and testing accuracy and the overall root mean square error (RMSE) of the different LSTM models based on the previously mentioned categories. The results show that the performance of the LSTM model greatly improves by decreasing the size of the batch from 128 to 32. The small batch size is capable of effectively capturing the generalization pattern of the dataset. The second parameter used to investigate the LSTM model structure is the number of epochs, where more epochs enhance the accuracy of predictions as shown in Table 1.

The depth of the the LSTM model refers to the number of hidden layers and the number of neurons in each layer. Table I includes the results obtained for different depths of the network. Where we can notice that using two or more layers provides an accurate understanding of the complex representations in the dataset. Also, the use of a small number of neurons in the hidden layer can lead to under-fitting. On

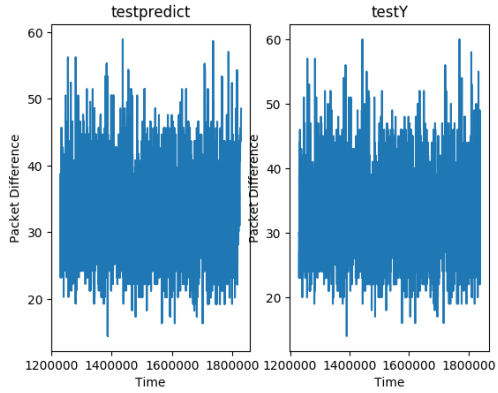

Fig. 4: Predicted test verus test data.

the other hand, a large number of neurons per hidden layer will be the reason for an over-fitted model. After many trials the proposed model is composed of two hidden layers with 64 and 16 neurons respectively, a batch size of 64 samples, and 1000 epochs. The RMSE is added to evaluate the accuracy of results. The proposed model structure achieves a test accuracy equivalent to 99.3% with 2147 RMSE.

Fig. 4 contains samples taken during the simulation from the predicted and the actual test data to illustrate the efficiency of the LSTM model. Table 1 shows that the two-layer stacked LSTM significantly outperforms the single LSTM. Specifically, the two-layer stacked LSTM achieves an RMSE that is less than 50% of the RMSE of the single LSTM, with a prediction accuracy of 98.53%. This high level of accuracy, combined with efficient training, highlights the stacked LSTM's ability to capture long-term dependencies, making it ideal f rks.

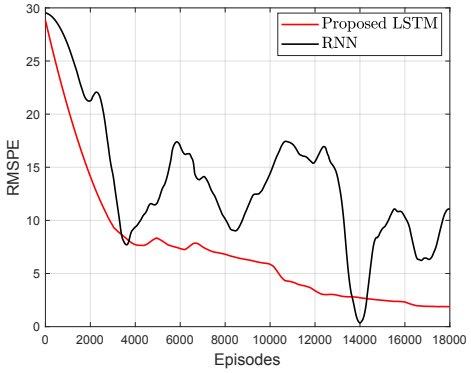

Fig. 5: The RMSPE performance of the proposed LSTM solution versus RNN benchmark technique.

Fig. 5 illustrates the Root Mean Square Percentage Error (RMSPE) comparison between the proposed LSTM-based method and a standard Recurrent Neural Network (RNN) over the training episodes. The RMSPE, a metric used to evaluate the accuracy of predictive models, is defined as: $RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2} \times 100$, where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of observations. The proposed LSTM method demonstrates superior performance, rapidly converging to a lower RMSPE within the initial episodes and maintaining a stable error rate of approximately 3-5%. In contrast, the RNN shows significant fluctuations with a higher RMSPE, stabilizing around 8-12%. This outcome highlights the LSTM's ability to effectively capture long-term dependencies and overcome the vanishing gradient problem that degrades the RNN performance, resulting in more accurate and stable predictions.

C. Packet Prioritization and Classification Model

The K-means classification model is fed by a packet header dataset during the simulation time, which contains $1,048,576$ entries. This dataset contains all previously mentioned packet header fields despite that the effective fields which are used to categorize the packet are the TTL and the priority. Therefore, based on the provided classification features the aim of this step is to classify all created packets into two classes, the most critical and the least critical packets. The most critical packets have a small TTL and high priority and will be sent first and immediately. The least critical packets deal with the channel's normal conditions, which are sent in case of free congestion or dropped channel. For the K-means++ implementation presented in Fig. 6, the candidate center must minimize the sum squared distances from each case point being clustered to the first randomly chosen cluster center.

IV. Takeaways

Building a rich dataset for our IoV research in this paper is challenging for these reasons: the need to collect data from

		Batch Size	No.of Epochs	Hidden layers Size (Nuerons)	Training Accuracy	Test Accuracy	Overall RMSE
LSTM Model Structure	Based on Batch Size	**32**	**1000**	**1 layer (64)**	**98.23**	**99.16**	**2454.07**
		64	1000	1 layer (64)	98.20	98.59	4325.29
		128	1000	1 layer (64)	98.51	98.57	4388.65
	Based on No.of Epochs	32	350	1 layer(64)	97.15	95.26	13712.17
		32	750	1 layer (64)	98.56	96.62	10368.44
		32	**1000**	**1 layer (64)**	**98.23**	**99.16**	**2454.07**
	Based on Hidden Layers Size	64	1000	1 layer(64)	98.20	98.59	4325.29
		64	1000	2 layers(64-32)	98.40	98.86	3497.03
		64	**1000**	**2 layers(64-16)**	**98.53**	**99.26**	**2147.31**

TABLE I: Categorical LSTM model structure results.

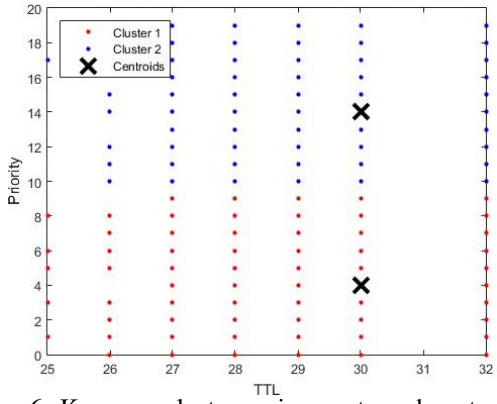

Fig. 6: K-means cluster assignments and centroids.

multiple sources, the need to capture data at different levels of granularity, and the sheer amount of data needed. The quality of the data is a fundamental aspect of data-driven research and has an effect on the accuracy of the models and inferences. Data quality is defined in terms of validity, reliability, and representativeness. The quality level can be measured using the fidelity of the data via how closely the data mimics the desired characteristics of the real world by measuring the accuracy, completeness, timeliness, relevance, and authenticity of the data. Defining accurate and consistent data acquisition procedures is essential and we have performed several validation checks, including ① checking for dataset completeness by ensuring the availability of necessary data points and the lack of significant gaps or missing values, ② checking the data for errors and inconsistencies to verify its accuracy, and ③ checking for any potential biases in the data that could impact the results or conclusions drawn from the data. In addition, we aim to do more verification of the dataset in the future by comparing the data points with actual data measured in real-world scenarios performed over V2X testbeds/industrial test tracks or in production IoV environments during trials similar to what we have done for aerial vehicle research [17].

V. Conclusion

In this research, we are concerned with the problem of network congestion management for the IoV. Different from previous techniques in the literature, we have proposed a novel pipeline that features a proactive congestion prediction using LSTM to mitigate packet loss and ensure reliable safety message delivery. We adopt K-means clustering to classify packets based on priority and TTL parameters. Performance evaluation for different LSTM structures and training hyperparamters are presented. The adopted LSTM architecture performance exceeds 98% prediction accuracy, enhancing delivery ratio and reducing delays. We were able to cluster the packets effectively into two groups of high and low message priority levels.

Acknowledgment

The work performed by Aly S. Abdalla and Vuk Marojevic was in part supported by NSF award CNS-2120442.

References

[1] A. Alalewi *et al.*, "On 5G-V2X Use Cases and Enabling Technologies: A Comprehensive Survey," *IEEE Access*, vol. 9, pp. 10–37, 2021.

[2] A. Nabil, K. Kaur, C. Dietrich, and V. Marojevic, "Performance analysis of sensing-based semi-persistent scheduling in c-v2x networks," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–5.

[3] K. Z. Ghafoor *et al.*, "Enabling Efficient Coexistence of DSRC and C-V2X in Vehicular Networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 134–140, 2020.

[4] A. Balador *et al.*, "Survey on decentralized congestion control methods for vehicular communication," *Vehicular Communications*, vol. 33, p. 100394, 2022.

[5] F. A. Al-Yarimi, "Enhancing road safety through advanced predictive analytics in V2X communication networks," *Computers and Electrical Engineering*, vol. 115, p. 109134, 2024.

[6] X. Du *et al.*, "Stacked LSTM deep learning model for traffic prediction in vehicle-to-vehicle communication," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2017, pp. 1–5.

[7] V. Elangovan *et al.*, "An Accumulative Method to Time Series Prediction for Vehicle Communication," in *2023 IEEE Vehicle Power and Propulsion Conference (VPPC)*. IEEE, 2023, pp. 1–6.

[8] B. Nithya *et al.*, "A CNN-LSTM Approach for Classification of Major TCP Congestion Control Algorithms," in *Intelligent Sustainable Systems: Selected Papers of WorldS4 2021, Volume 2*. Springer, 2022, pp. 579–591.

[9] T. Zhang *et al.*, "A hybrid method of traffic congestion prediction and control," *IEEE Access*, vol. 11, pp. 36 471–36 491, 2023.

[10] Z. Ali *et al.*, "3GPP NR V2X mode 2: Overview, models and system-level evaluation," *IEEE Access*, vol. 9, pp. 89 554–89 579, 2021.

[11] M. Drago *et al.*, "MilliCar: An ns-3 module for mmWave NR V2X networks," in *Proceedings of the 2020 Workshop on ns-3*, 2020, pp. 9–16.

[12] F. Karim *et al.*, "LSTM Fully Convolutional Networks for Time Series Classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

[13] K. P. Sinaga and M.-S. Yang, "Unsupervised K-Means Clustering Algorithm," *IEEE Access*, vol. 8, pp. 80 716–80 727, 2020.

[14] A. S. Abdalla *et al.*, "Toward Next Generation Open Radio Access Networks–What O-RAN Can and Cannot Do!" *IEEE Network*, pp. 1–8, 2022.

[15] M. Kouchaki *et al.*, "OpenAI dApp: An Open AI Platform for Distributed Federated Reinforcement Learning Apps in O-RAN," in *2023 IEEE Future Networks World Forum (FNWF)*, 2023, pp. 1–6.

[16] M. Behrisch *et al.*, "Sumo - simulation of urban mobility: An overview," in *in SIMUL 2011, The Third International Conference on Advances in System Simulation*, 2011, pp. 63–68.

[17] A. S. Abdalla *et al.*, "Open-source software radio performance for cellular communications research with uav users," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021, pp. 1–6.