Received 22 March 2023, accepted 20 April 2023, date of publication 27 April 2023, date of current version 3 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3270889

RESEARCH ARTICLE

Transformer Based Traffic Flow Forecasting in SDN-VANET

ALI ABIR SHUVRO, MOHAMMAD SHIAN KHAN, MONZUR RAHMAN, FAISAL HUSSAIN, MD. MONIRUZZAMAN, AND MD. SAKHAWAT HOSSEN

Department of Computer Science and Engineering, Islamic University of Technology, Gazipur 1704, Bangladesh

Corresponding author: Md. Sakhawat Hossen (sakhawat@iut-dhaka.edu)

ABSTRACT Intelligent Transportation System (ITS) provides services for proper traffic assistance. ITS helps in creating a transportation system that is smart, safe and efficient. Vehicular Ad-hoc Network supplies internet connectivity to vehicles and helps in traffic guidance. This paper uses a modified transformer architecture for time-series vehicular data to predict traffic flow. Time-series sequences are generated from the dataset for capturing temporal dependencies. Our proposed transformer-based model has been engineered to capture inter-feature correlations along with inter-sample correlations. The 2D-Transformers model has a significant decrease in error compared with Transformers and LSTM-based models. The prediction generated from the model can be transmitted throughout a network of vehicles. So, a holistic networking model is proposed where the vehicles will be connected to Road-side Units (RSUs) and the backbone network will be Software Defined Network (SDN). The traditional design principles, that incorporate data, control and management planes together in a network device, are incapable to adapt with this much data growth, bandwidth, speed, security, and scalability compared to SDN as it provides with centralized programmable mechanism reliably. The trained parameters learned using the transformer model can be passed throughout the network for traffic guidance.

INDEX TERMS Vehicular ad-hoc network, transformers, sequence length, encoders, attention, traffic flow, software-defined network.

I. INTRODUCTION

THE Intelligent Transportation Systems (ITS) technologies provide services to better traffic and transportation information. It makes transportation smarter by providing meaningful insights into traffic data which are captured by highly advanced traffic detection sensors, and in essence, makes the roads safer and more coordinated. Traffic data is captured via detectors/sensors, so traffic forecasting has been critical to the development of ITS.

Traffic data can be spatial-temporal data, which means it contains information about the time and space of the vehicles at a given time. Additional data might be present as the average speed, occupancy, etc. which will help us forecast the traffic flow. LSTM models have a higher performance in comparison to models which incorporate algorithms like HTM [1]. Short-term prediction of traffic conditions is necessary for overcoming traffic congestion [2].

Time-series traffic data is the new and improved way of predicting traffic as it can utilize sequential data to provide a proper prediction. Perfect use of sequential machine learning models can also be implemented. Transformers have been seen to produce remarkable results in this regard.

Vehicular Ad-hoc Networks or VANETs play a key role in ITS. Safety and roadside equipment communication of vehicles is a key factor when it comes to VANET. It ensures vehicle-to-vehicle and vehicle-to-infrastructure communication which makes it easier for ITS to communicate and collect more information. VANET can be incorporated with Software Defined Network or SDN as in the architecture there are SDN components (SDN controller, SDN wireless nodes, SDN RSU). Software-Defined VANET or SDVN can operate differently based on the degree of control of the SDN controller. SDVN also provides some benefits which

The associate editor coordinating the review of this manuscript and approving it for publication was Qingchun Chen.

can be utilized to provide many services [3]. Sadio et al. have proposed a design of a complete SDVN prototype [4]. Such designs can be used with our work to incorporate the trained weights of the models into the SDN-enabled network.

Traffic prediction is quite a challenge to tackle as spatial-temporal data is needed to be handled [5]. Traffic prediction data can be represented with a graph and Graph Neural Network along with RNN can be used for the prediction of data [6].

However, there are some limitations to it due to the continuous nature of time and the adjacent behaviour of space. The use of RNNs is limited due to temporal dependencies for exploding and vanishing node problems. GRU and LSTM overcome this shortcoming of RNN. Therefore, several algorithms combining Traffic Transformer have emerged as it works well with sequential data [6]. A combination of GCN and transformer has outperformed several other models mainly consisting of RNN in the past [5].

Traffic data is in time-series format, so through a Transformer-based learning model, traffic flow can be forecasted with better performance and a computationally efficient model. The memory of the Transformer has no bound with available resources. Multidimensional time-series data is necessary for more accurate prediction. Incorporating multidimensional data can open a door for handling a different kind of situation in case of prediction. Our Transformer is capable of handling such multi-featured time-series data efficiently. Furthermore, a holistic model can be suggested which incorporates gathering data from the RSUs, learning using the SDN controllers and finally passing those learned weights to the RSUs and eventually to the vehicles.

Training a time-series dataset on a learning model is especially challenging as it involves making some adjustments to predict long-term dependencies in case of time, and variable spatial dependencies. The core contributions of the work can be summarized as follows:

1) Using a state-of-the-art transformer model which provides more accurate results for the time series data.
2) Introducing the 2D-Transformers model with 2D Multi-head Attention Mechanism and using the multivariate model, inter-feature dependencies were ensured.
3) Proposed a holistic model for traffic guidance in an up-and-coming SDN-based VANET.

II. RELATED WORKS

Vehicular Ad Hoc Networks are a potential field of research and have caught a lot of interest. It is this eye-catching because of its potential to provide vehicle road safety, increase the efficiency of traffic and provide convenience and comfort for passengers and drivers. Mobile Vehicular Cloud and review cloud applications can be considered as significant technology to develop intelligent transportation [7]. Vehicular ad hoc networks (VANET) refers to the creation of a mobile ad-hoc network created in the domain of vehicles which supports communications among vehicles, roadside units and base stations to provide safe and efficient transportation. There are three types of architecture in VANET and those are pure cellular wireless local area networks, pure ad-hoc networks and hybrid networks [8]. There are basic characteristics of Vehicular networks. There are many applications associated with this field. There are requirements which come with many challenges. Solutions are also there which can be considered to overcome challenges [9]. With the advancement of technology and the establishment of smart cities, there is an increasing need for Intelligent Transport Systems (ITS). Safety, communication and traffic-related issues are one of the major concerns of ITS. There are machine learning techniques which can address these issues in a feasible manner and overcome such challenges [10]. Security is a concern in the case of routing of the VANET nodes. Enhancing network robustness, defence mechanisms with low-security overhead can be introduced [11]. As per security is concerned, blockchain technology has been incorporated with VANET to secure ride-sharing applications. The Proof of Driving (PoD) in a blockchain-based ride-sharing service in VANET, which consumes fewer resources than Proof of Work (PoW), maintains a fair selection than Proof Of Stake (PoS) as well as the randomness of consensus nodes in a publicly distributed network of vehicles [12].

The widespread IP network is a very complex and difficult case of management, to configure according to predefined policies as well as reconfigure according to faults, loads and changes. These networks are vertically integrated which means the control and data plane are bundled together. Software Defined Networking is such a field that breaks this vertical integration separating the control logic from routers, and switches and provides flexibility, dynamic, optimal and centralized logical control over the system [13], [14]. Due to design limitations, the traditional network is unable to make users interact with the traffic or shape their own traffic policies. Once the flow management (forwarding policy) has been determined, the only way to adjust will be by changing the configuration of the devices which has restricted the network operator who wants to scale their networks on traffic demands. This leads to SDN as it provides with centralized programmable mechanism [15]. Northbound API in the Control layer is used to program the network and request services from the Application layer by the applications and overall management system. An OpenFlow protocol which is often used as a southbound API is used between the Control layer and Infrastructure layer [16]. Wireless sensor networks are low-rate networks with few resources and short communication ranges and when it expands it faces network management and heterogeneous-node networks challenges. Here SDN can be incorporated with wireless sensor networks to bring efficiency and sustainability and it is called Software Defined Wireless Sensor Networks (SDWSN). This model can also play a critical role in incorporating SDN with the Internet of Things (IoT) [17]. IoT is a network which is open, geographically distributed, and consists of heterogeneous networking infrastructures. Managing these in dynamic situations is an important challenge. This challenge can be overcome by using SDN with IoT. SDN can be designed for IoT for dynamically achieving quality to different IoT tasks in heterogeneous wireless networking [18]. For smart cities an IoT-SDN structure can be incorporated to bring benefits like energy savings, network scalability and load balancing [19]. The rapid growth of the internet and mobile communication technology has led us to more complexity. Now, these need to be efficiently organized, managed and optimally maintained. This is why more intelligence is needed to be incorporated and traditional networks failed to do so. SDN brings the chance to use machine and deep learning and provide this much-needed intelligence in the networks in this Big-Data world for its logically centralized control, global view of the network, software-based traffic analysis and dynamic updating of forwarding rules [20], [21]. SDN can be therefore used to tackle network management challenges which include erratic network conditions and states, high-level language support for network setup, improved visibility, network diagnostics, troubleshooting and flow classification [22], [23], [24].

For the rapid growth of roadside accidents, to ensure passengers' safety, an ad-hoc network that is vehicular ad-hoc network is being encouraged. Managing the whole network from a single remote controller, SDN-VANET helps to overcome the drawbacks and complexity of standard VANET architecture [25]. The adaptability, controllability and versatility of controlling the network as a whole of SDN help to build an effective and secure VANET with simplified network control. When VANET is implemented with SDN, in the architecture there are SDN components (SDN controller, SDN wireless nodes, SDN RSU) which are incorporated with VANET and so Software Defined VANET can operate differently based on the degree of control of the SDN controller [3], [26]. SDN is used in Multi-access Edge Computing for the effective management of networks and services. Integration of SDN can leverage several advantages which are also useful for the incorporation of 5G [27], [28]. A global optimal route can be found to efficiently propagate messages from source to destination with dynamic network density in SDVN. Centralized Routing Protocol is the SDN-based routing framework that uses modified Dijkstra's algorithm for efficient message propagation in VANET and outperforms some other traditional protocols [29].

SDVN can be used to detect and predict traffic collaborating with machine learning. K-means clustering can be used to detect and LSTM with RNN can be used to then predict the traffic condition [30]. HTM can be used alongside LSTM for short-term arterial traffic prediction. Several Machine learning techniques have been deployed to predict traffic state and also vehicles [31]. LSTM models have a higher performance in comparison to models which incorporate algorithms like HTM [1]. Short-term prediction of traffic conditions is necessary for overcoming traffic congestion [2]. Results by applying a self-adjusted Neural Network (NN) have been satisfactory as well [32]. Vacant parking spots have been predicted using different models like Support Vector Regression (SVR), LSTM and dense convolutional Neural Networks with LSTM which showed significant improvements in the domain of time-series prediction [33], [34], [35]. Several models revolving around graph convolutional network has also been effective in predicting short-term traffic data. Temporal Graph Convolutional network yields decent performance when it comes to solving this particular problem of traffic prediction [36]. A long short-term memory (LSTM) based regression model can be used to predict 24-hour traffic counts data. The algorithm was not discussed along with the dataset used [37]. Newer and better models can be used in this sector to improve performance and proper analysis would help understand the internals of the prediction models.

Traffic prediction is quite a challenge to tackle as spatial-temporal data is needed to be handled [5]. Traffic prediction data can be represented with a graph and Graph Neural Network along with RNN can be used for the prediction of data [6]. However, there are limitations to it due to the continuous nature of time and the adjacent behaviour of space. The use of RNNs is limited due to temporal dependencies for exploding and vanishing node problems. GRU and LSTM overcome this shortcoming of RNN. Therefore, several algorithms combining Traffic Transformer have emerged as it works well with sequential data [6]. A combination of GCN and transformer has outperformed several other models mainly consisting of RNN in the past [5].

Time series data with proper sequencing can be used to provide a suitable solution to some of the existing problems. The short-term traffic condition will be addressed in a more efficient manner if time series data is used to train the model. Again, since multiple features are naturally present in these time series data, using these features to find a prediction altogether would yield a much better result. Modification of the classic transformer architecture to incorporate multiple variables in each of the time sequences is necessary.

III. METHODOLOGY

The aim of this study is to forecast the *average occupancy* of a station at a certain time given an already observed time-series sensor-generated vehicular dataset. Basically, given the station, the number of samples, percent observed, total flow, average speed and some other parameters, the prediction of average occupancy can be obtained.

Time-series analysis on sequential data was conducted for effective prediction of traffic flow. By training a model on time-series data, accurate predictions of the condition of traffic can be produced in the future. Again, predictions at specific time intervals are possible using this methodology. After preprocessing the collected time-series data, the obtained sequential data can be used for accurate predictions. Different deep learning models were used which have been

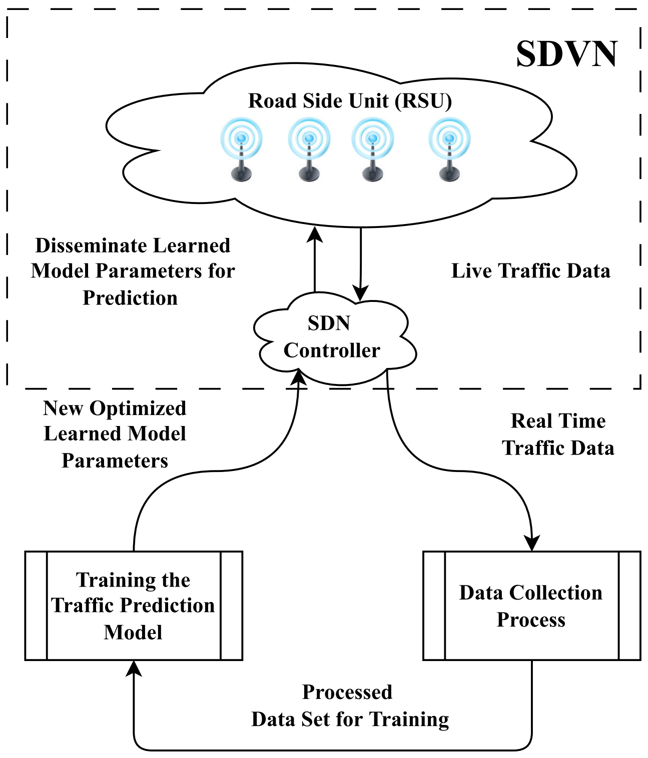

SDVN
Road Side Unit (RSU)
Disseminate Learned Model Parameters for Prediction
Live Traffic Data
SDN Controller
New Optimized Learned Model Parameters
Real Time Traffic Data
Training the Traffic Prediction Model
Data Collection Process
Processed Data Set for Training

FIGURE 1. Architecture overview.

proved in the past to have worked phenomenally well in terms of time-series mode of data, especially in the case of traffic flow [37].

A. MODEL ARCHITECTURE

As shown in Figure 1, the proposed system consists of three segments. They are *Data Collection*, *Traffic Flow Prediction Model* and *underlying SDVN Architecture*.

After the collection of the data, it was preprocessed; the steps involved cleaning, reduction, and transformation of data. For transforming the traffic time-series data into sequential data, the data had to undergo sequence generation. The final sequential data is used in the training of the model. A separate controller or server is used to train the model.

The trained weights obtained from the model are fed to the SDVN architecture. The SDN controller propagates these weights or model parameters to the RSUs which these units finally use to find predictions. The predictions are transferred to the vehicles. The vehicles in turn also provide the live traffic data which is further sent down by the RSUs to the controller for newer optimized model parameters.

IV. IMPLEMENTATION

The segments mentioned in section III-A that need to be implemented are described in this section. Along with that, we also talk about the dataset that we used for our experiment. Let us look at the environment used for conducting the experiment.

TABLE 1. Variables along with their unique frequencies existing in the dataset.

Feature Name	Number of unique feature values
station number	194
station length	115
samples	886
percent observed	90
total flow	10410
average occupancy	5679
average speed	788

A. DATASET

The dataset we used was obtained from Caltrans Performance Measurement System (PeMS) [38]. This dataset is used as a standard as it contains real-life data with a lot of unforeseen outcomes. The dataset seems to behave similarly compared to many other datasets [5]. Nearly 40,000 traffic sensors are deployed in almost all notable streets in the state of California and traffic data is collected everyday. Hourly data for the whole year of 2021 from PeMS district 7 was taken and finally, an 80-20 train-test split was performed.

Multivariate time-series data sequences are used. For that reason, the variables or features which was selected are:

B. PREPROCESSING

At first, raw hourly data was collected from the PeMS website. This included selecting a specific year and district where there are adequate busy streets for variation. Some of the additional features were trimmed for making the experiment computationally feasible. Next, mean normalization was used on the dataset so that the values are in the range of [0, 100]. This ensures that the model is not biased towards only a few features.

C. GENERATING SEQUENCE

In order to convert a time-series dataset into sequential data, it is necessary to make a sequential copy of a portion of the data and each time introduce 1 more sample. Sequence generation is important so that the model gets an idea of past and present situations in different spatial and temporal perspectives in that area which has an impact on future conditions. Each time, the model gets the whole sequence of data and predicts just a single sample.

While creating sequences, the data is arranged in such a way that data of (n-1)th sample of all other variables will be along with nth data of the target variable. This is made like this so that when any prediction algorithm is used, it genuinely picks the future target variable based on the present values of the other variables. The process of creating sequences takes just a few minutes. The sequence number that is picked determines the time required for training. For a large sequence length, the training time becomes large as well, and

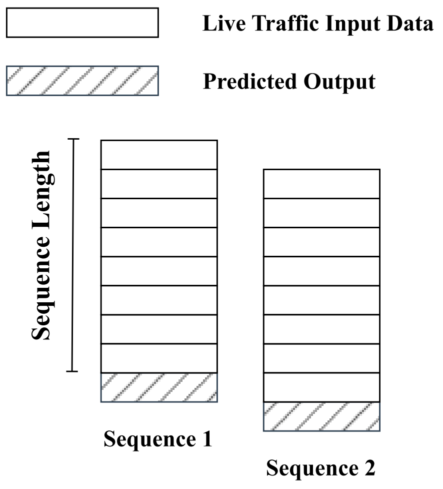


FIGURE 2. Processing sequences in the dataset.

TABLE 2. Shape of train and test dataset with respect to sequence number.

	Sequence Length	
	4	8
Train dataset shape	1529305 ×36	1529305 ×72
Test dataset shape	169924 ×36	169924 ×72

vice-versa. In fact, the dataset size will be exactly the multiple of the sequence number. The shapes of the datasets obtained are shown in the table below:

D. TRAINING

The Transformer Architecture which is mainly used in NLP for word generation and translation is similar to category prediction or logistic regression. For the regression of continuous data, the basic transformer architecture has been made aligned to the regression model which can fit the regression data properly.

After setting every module of the architecture properly, the training data is passed to the Data Generator which generates batch-wise data. This batch-wise data is at first fed to the Embedding layer and then to the Positional Encoding layer. The encoded data carries information about the relative order of a sequence sample in the input sequence. The encoded data makes it possible for the Transformer to use the feature and positional information about the order of the sequence sample. The encoded data generated from this module is passed to the Encoder. The exact same process is used to represent the input to Decoder as well.

1) THE ENCODER

The Encoder consists of N Encoder Layers which are identical and stacked upon each other. Each of the encoder layers consists of 2 sub-layers namely Multi-Head Attention and Position-Wise Feed Forward Network. The Multi-Head Attention layer is based on the self-attention mechanism in

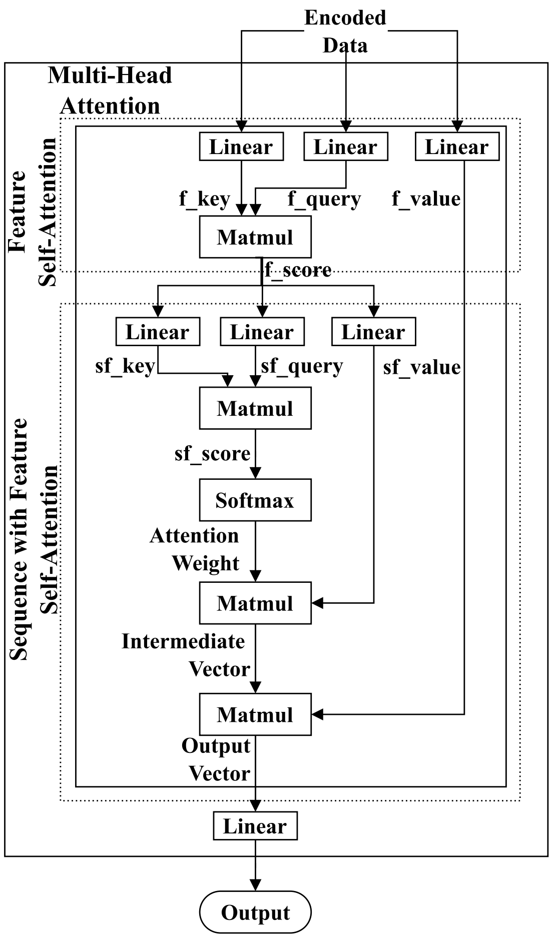


FIGURE 3. 2D multi-head attention mechanism.

multiple heads and the Positional Feed Forward networks that consist of fully connected Neural Networks which are applied to each of the positions.

2) 2D MULTI-HEAD ATTENTION LAYER

In the case of traffic flow, there are several factors which need to be considered. These factors are responsible for traffic conditions in a given time step. This is why sequences of multiple features are needed in case of such time series traffic flow forecasting.

There are several attention heads inside the multi-head attention mechanism and each has a self-attention mechanism. The theory behind it is to give the model more representational power. That means it will be able to learn from different points of view generalizing more forms of representation, thinking that each head will learn something different. As multiple features are included in such situations, the standard self-attention mechanism of the transformer is not enough to focus on more than one feature along with focusing on a sequence of samples because transformers are built to work with word sequences which consist of a single word at a specific time instead of a group of features.

As per Figure 3, this self-attention mechanism needs two kinds or dimensions of six linear layers. The first dimension of the linear layer is focused on features and the second is on the sequence. The encoded data from the previous layer is used to calculate *f_query*, *f_key* and *f_value*. These are calculated after the input is fed to feature focused linear layer.

$$f\_query = Linear_{Q_F}(s) \tag{1}$$

$$f\_key = Linear_{F_K}(s) \tag{2}$$

$$f\_value = Linear_{F_V}(s) \tag{3}$$

The *f_query* and *f_key* are multiplied to calculate the *f_score* matrix. It determines how much focus a feature should put on every other feature. This is a feature-to-feature calculation. Each feature will have a score that corresponds to other features in that time step. It is divided by the square root of d_k which represents the number of features per head to ensure stability mitigating exploding effect.

The *f_score* matrix is used to generate a new query, key and value with the help of a sequence-focused linear layer. The *f_score* matrix is fed to sequence focused linear layer resulting in *sf_query*, *sf_key*, *sf_value*. These three outputs will represent the sequence as well as implicitly keep attention information among the features.

$$sf\_query = Linear_{Q_{SF}}(f\_score) \tag{4}$$

$$sf\_key = Linear_{K_{SF}}(f\_score) \tag{5}$$

$$sf\_value = Linear_{V_{SF}}(f\_score) \tag{6}$$

Then the *sf_query* and *sf_key* will be multiplied to calculate the *sf_score* matrix which will undergo a softmax operation generating the attention weights. These weights denote the self-attention among each and every sequence as well as features.

$$Attention(Q_F, K_F, V_F) = (softmax(\frac{Q_{SF} \times K_{SF}^T}{\sqrt{d_k}})V_{SF})V_F \tag{7}$$

Then the attention weights are used to do a weighted sum of the *sf_value* followed by another weighted sum operation using *f_value* to generate the output vector. Thus important values will get priority. Then the output vector is fed to the final linear layer with a residual connection to produce an output which is ready to go to the next layer.

There are several attention heads inside the multi-head attention mechanism which has the self-attention mechanism individually. The theory behind it is to give the model more representational power thinking that each head will learn something different.

3) POSITION-WISE FEED FORWARD NETWORK

This sub-layer of an encoder layer is nothing but a fully connected feed-forward network with a residual connection. If the input is x then:

$$PFFN(x) = ReLU(xW_1 + b_1)W_2 + b_2 \tag{8}$$

The fully-connected layer is applied which has weights W_1 and biases b_1. ReLU non-linearity, max with zero, is applied to it followed by another fully-connected layer with weights W_2 and biases b_2.

4) THE DECODER

The concept is similar to the encoder. The decoder consists of N Decoder Layers which are identical and stacked upon each other. Each of the decoder layers consists of 3 sub-layers namely Masked Multi-Head Attention, Encoder-Decoder Multi-Head Attention and Position-Wise Feed Forward Network.

5) GENERATING FEATURE WEIGHTS

The output from the last decoder layer is passed on to the Generator. The Generator basically has a Linear layer and a Softmax layer. It produces the impact factor of each feature to produce the target "*Average Occupancy*". This denotes the contribution of each feature to the target value. The weighted sum of the features will be the target output value. In the training phase, the prediction is compared with the actual value and loss are computed and the loss is backpropagated. There, the parametric weights of the model are updated which are used later for inference and finally to calculate the predicted value.

V. RESULT AND DISCUSSION

A. ENVIRONMENT AND CONFIGURATION

Google Colab and Google Colab Pro have been used for the experiment. As the experiment is resource-heavy, in most cases, Google Colab Pro has been the go-to notebook environment. Google Colab Pro proves a heavy RAM option which gives an average of 25 GB RAM to its users. A Tesla P100 GPU with 16 GB memory and a storage capacity of around 200 GB.

B. EVALUATION METRICS

1) MSE

MSE or Mean Squared Error is an error calculating method. It is mainly used to show how close a regression line is to a set of predictable points. If Y be the actual output and $\hat{Y}$ be the predicted output, then we get:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{9}$$

2) RMSE

RMSE or Root Mean Squared Error is an error calculating method. It is more popular in its use as it is measured in the same unit as the response variable. RMSE basically tells us about the average deviation between the predicted points and the actual points. If Y be the actual output and $\hat{Y}$ be the predicted output, then we get:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}} \tag{10}$$

TABLE 3. Hyper-parameters used for the experiment.

Hyper-parameter	Value
Batch Size	200
Learning rate	0.001
Number of Encoders	5
Dropout	0.2

TABLE 4. MSE, RMSE and MAE values obtained from the predictions.

Sequence length	Model	MSE	RMSE	MAE
8	**2D-Transformers**	23.2914	4.8261	3.1736
	Transformers	37.3473	6.1113	4.0098
	LSTM	29.6576	5.4459	3.7024
4	**2D-Transformers**	27.3781	5.2324	3.6056
	Transformers	48.8498	6.9830	5.5894
2	**2D-Transformers**	38.2698	6.1863	4.1891
	Transformers	73.2579	8.5591	6.9661

3) MAE

MAE or Mean Absolute Error is another error calculating method. It is the absolute difference between the paired observations. It is used to measure the accuracy of continuous variables. If Y be the actual output and $\hat{Y}$ be the predicted output, then we get:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid Y_i - \hat{Y}_i \mid \tag{11}$$

C. EXPERIMENTAL RESULTS

The model was trained on the Google Colab environment. The following hyper-parameters which were empirically observed through several experiments were selected:

The batch size refers to the number of training examples used by the model in each iteration. Generally, larger batch size yields better results. The batch size here has been picked to be 200 considering the memory constraint. The learning rate is the step size of each iteration. Basically, the percentage of change in the weight due to the error in the training sample is determined by the learning rate. The learning rate has been picked to be 0.001 by a trial-and-error method which properly works for our dataset. Iterative processing of the input is done in each layer of the encoder. By increasing the number of encoders, the model will be able to catch more deviations and feature contributions to the result. By keeping the memory constraint in mind, the number of encoders has been chosen to be a maximum of 5.

We have used the target variable as *Average Occupancy* and with different sequence lengths. The model was measured in terms of MSE, RMSE and MAE. After the convergence of the training, the results that were obtained are as follows:

This shows the variation of different types of errors on the basis of the change in sequence lengths. The transformer architecture considering time series data along a specific variable has been considered in [5] and [39]. The transformer model across a specific variable failed to capture deviations which are related to other features in the data. This is what is depicted in the table 4. Both the sequence lengths 4 and 8 have produced better results compared to the Transformer and LSTM model of sequence length = 8. Our model was trained for around 30 minutes for each of the variables of encoders. We ensured a proper learning period for the Transformer and LSTM models by giving them each more than an hour of training time. We have kept the batch size at 200 and a learning rate of 0.001. The number of encoders was a maximum of 5 which achieved minimum error.

TABLE 5. MSE values for varying the number of encoders.

Number of Encoders	Sequence length			
	4	8	4	8
	2-D Transformers		Transformers	
2	93.6362	67.1748	97.0214	78.3611
3	30.6794	27.0518	82.1477	66.5763
4	26.8774	24.8331	67.1285	66.3342
5	27.3781	23.2914	48.8498	37.3473

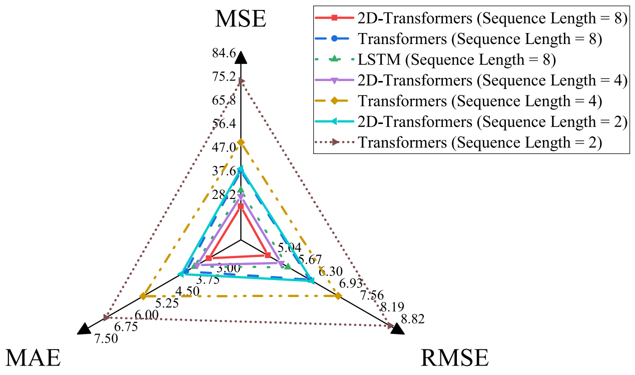


FIGURE 4. Comparison between 2D-Transformers, Transformers and LSTM.

Varying the number of encoders, the errors also fluctuate. Now, let us look at how encoders have impacted learning by observing the variation in MSE for different numbers of encoders:

It is evident that the number of encoders has a direct impact on the model's learning. A similar pattern is observed for both Transformers architecture and the proposed 2-D Transformer architecture. With the increase in the number of encoders, the model learns better and gives a lesser error.

D. RESULT ANALYSIS

1) IMPACT OF SEQUENCE LENGTH

It is seen that with the increase in sequence length, all the errors (MSE, RMSE and MAE) seem to decrease. This is because the sequence length necessarily defines how much of the time-series data are taken together for correlation at a time. So, if the sequence length increases, more data is taken together as a sequence and correlation is much more evident by the model. Similar results are also shown by our 2D-Transformer model where increasing the sequence length

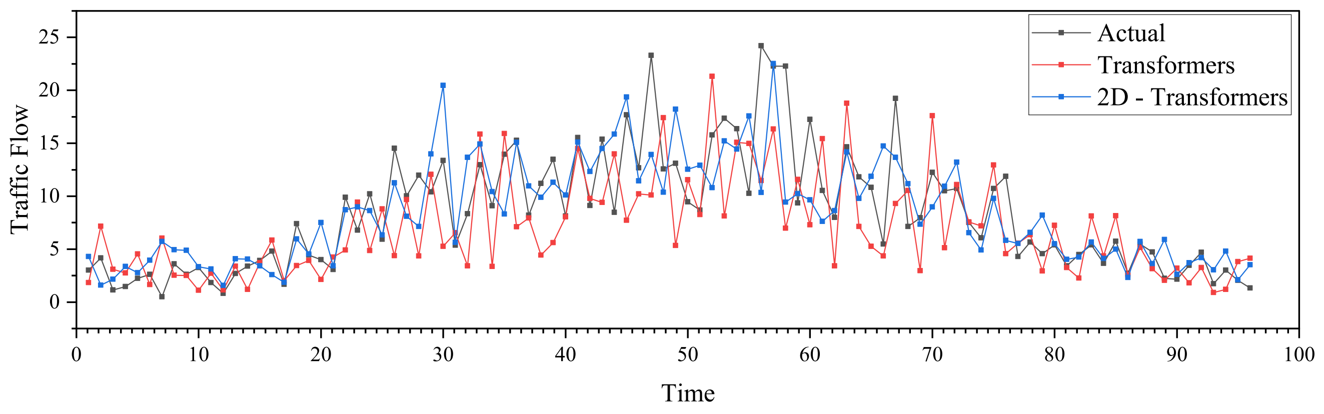


FIGURE 5. Performance of the model over four days.

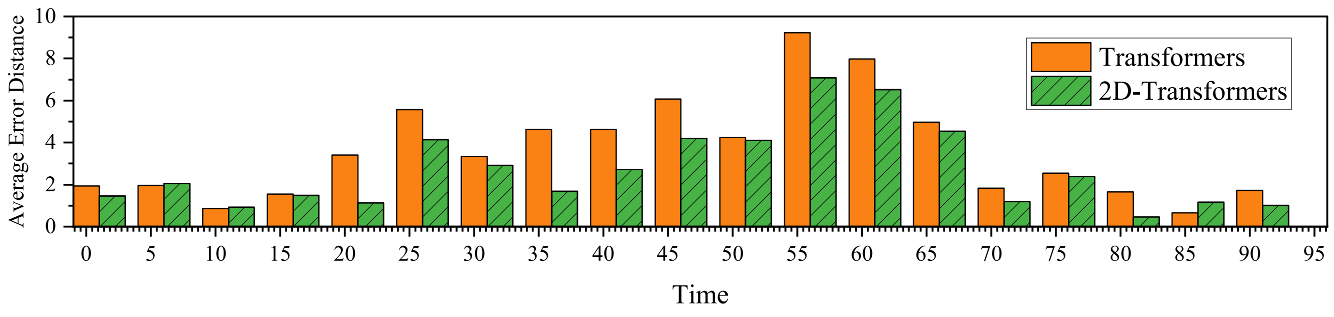


FIGURE 6. Average error distance of transformers and 2D-transformers over four days.

decreased the error up to a certain extent. We have taken sequence lengths 2, 4 and 8 where 4 has given a better result than 2, and 8, has given a better result than 4.

If we look at the radar chart in Figure 7, we will see that our model of sequence lengths 4 and 8 has outperformed the LSTM model of sequence length = 8 and Transformers model of sequence lengths 4 and 8 in all three metrics.

2) TEMPORAL PERFORMANCE OF THE MODEL

The model makes near-accurate predictions of the traffic flow which can be observed from Figure 5. The figure shows the performance of the model over a continuous time frame of 4 days in an hourly manner. A similar pattern can be observed over 5-minutes data as well. One of the critical problems that can be observed for the basic Transformer model is that the sudden changes in traffic flow were not captured. This prediction is of utmost importance as unusual traffic flow prediction is one of its main practical uses. These abrupt changes were successfully captured by our model. The model can be used for traffic guidance and management systems and make positive improvements in ITS.

Figure 6 shows the Average Error Distance which is the average error of each of the models with respect to the actual value over the 4 days of time. Each of the error distances is taken by averaging over the 5 consecutive hours of hourly errors caused by both 2D-Transformers and Transformers models. This gives a comprehensive idea about the improvements offered by 2D-Transformers over a long period of time. It is evident that 2D-Transformers have outperformed the Transformer model in almost each of the hours.

3) IMPACT OF NUMBER OF ENCODERS

The number of encoders directly helps in improving the performance of the model. As the number of encoders increases, the model is able to learn much more correlation among the features. Let us see how the number of encoders has made the model improve in performance:

The basic Transformer model starts to improve with the increase in the number of encoders as expected. But the error is still larger than that of 2D-Transformers. Even with 5 encoders, the model has a higher error than 2D-Transformers with only 3 encoders.

Our experiment shows that our 2D-Transformer increasing the number of encoders from 2 to 5 has decreased the MSE by around 70%, which is for sequence length = 4 as shown in Figure 4. Similar results are seen in the case of sequence length = 8. The MSE has decreased by around 66%. This graph shows that the 2D-Transformer model has converged and has a steady error rate moving forward.

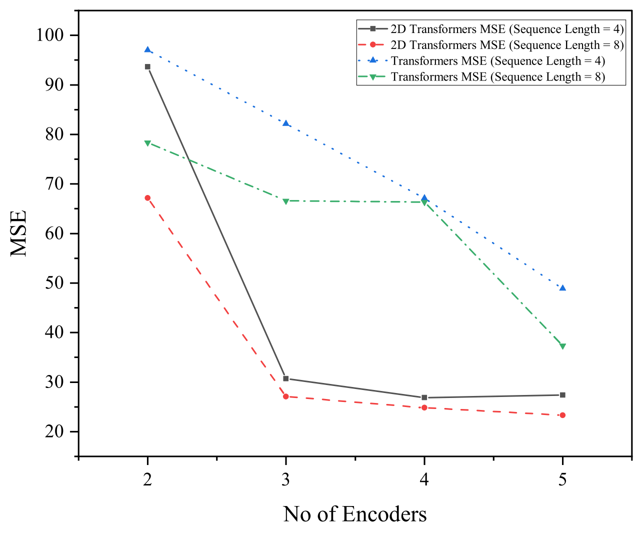


FIGURE 7. Impact of number of encoders in MSE.

E. INTEGRATING INTO AN SDN ENVIRONMENT

Software-Defined Network (SDN) architecture is being chosen for deploying our traffic prediction model and for controlling and managing vehicular network communication. It can be deduced that SDN architecture ensures better connectivity and performance for vehicular communication over traditional networks, and upon experimentation, similar results have been achieved. In this section, we will put some light on why SDN is being chosen over traditional networks for integrating vehicular communication, the experimental setup that is being conducted on the Mininet emulator, and will discuss the simulation results.

One of the key concepts of SDN is the decoupling of data and the control plane. In the case of traditional networks, routing devices perform the duties of both the control plane and the data plane. In SDN, a controller-switch architecture is observed for the decoupling of the control and data plane. Controllers act as the control layer and switch just forward the data.

1) SDN EXPERIMENTAL SETUP

The experiment was conducted in Mininet Emulator with Floodlight Controller. The Mininet emulator resided in a Ubuntu 14.04 OS. The floodlight controller is used as the remote controller of the network. A custom topology that has been built to emulate a real-world scenario consists of 4 switches and 4 hosts.

Through the experimentation in the simulation, we have measured on how much time it usually takes for the model data to be transmitted to the other devices. Various samples of data are being gathered as to how much time it takes for the full data to go from the devices, in the SDN network.

2) SDN SIMULATION AND RESULTS

We have found the results of the experimentation in terms of the time needed for transmitting the data to the other devices. We have sample data from 100kb to 500kb and transmitted them over the network. The time that is required for the data transmission is presented in the table below. We also have been able to showcase the topology in the floodlight controller as shown in the figure.

TABLE 6. Time needed for data transmission in the topology based on file size.

Data size (in KB)	Time required (in seconds)
100	0.001
200	0.001
300	0.005
400	0.007
500	0.008

VI. FUTURE WORKS

A. DATASET MODIFICATION

Each of the stations in the dataset has a specific coverage area. This coverage varies from station to station. Because of this reason, some of the stations which have a smaller coverage might have less traffic present even though it is closer to some heavy traffic. This sort of case makes it hard for the model to recognize correlations among stations with their positions and neighbouring stations.

In order to make the model perform better, these smaller stations can be merged with their nearest larger stations. This modification in the dataset will yield a much better result as the model will be able to understand the spatial correlations.

B. RESOURCE ALLOCATION

Larger sequence lengths and batch sizes would generally yield a better result. More encoders would also help in improving performance. To sum up, bigger sequences, batches and more encoders would yield a much better result.

Again, a larger dataset would mean that the model is learning from more samples, which generally also produces much better results and avoid bias.

VII. CONCLUSION

It is clear from the experiments that our transformer model has performed amazingly on the dataset. This demonstrates how well-suited transformer-based models are for estimating traffic flow. The transformer design can tolerate dependencies across vast distances and protracted timespan since it accommodates massive volumes of data.

VANET's ad-hoc nature meshes seamlessly with the SDN architecture. The centralized decision-making process of SDN can guarantee a quicker transmission rate and appropriate control mechanisms that are needed for fast-moving vehicles. The research work has some limitations. Rigorous data analysis could be done to find dependencies across stations which could create an avenue for feature engineering. The model has not been simulated in a real-life SDN environment which could also provide challenges in its behaviour in a practical setting with dynamic constraints.

It can be concluded a comprehensive model for traffic flow prediction and propagation inside ITS might be produced by combining VANET and SDN with the precise prediction made possible by a transformer model.

REFERENCES

[1] J. Mackenzie, J. F. Roddick, and R. Zito, "An evaluation of HTM and LSTM for short-term arterial traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1847–1857, May 2019.

[2] X. Yu, S. Xiong, Y. He, W. E. Wong, and Y. Zhao, "Research on campus traffic congestion detection using BP neural network and Markov model," *J. Inf. Secur. Appl.*, vol. 31, pp. 54–60, Dec. 2016.

[3] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined VANET: Architecture and services," in *Proc. 13th Annu. Medit. Ad Hoc Netw. Workshop (MED-HOC-NET)*, Jun. 2014, pp. 103–110.

[4] O. Sadio, I. Ngom, and C. Lishou, "Design and prototyping of a software defined vehicular networking," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 842–850, Jan. 2020.

[5] L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu, "Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting," *Trans. GIS*, vol. 24, no. 3, pp. 736–755, Jun. 2020.

[6] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong, "Spatial-temporal transformer networks for traffic flow forecasting," 2020, *arXiv:2001.02908*.

[7] M. Gerla, "Vehicular cloud computing," in *Proc. 11th Annu. Medit. Ad Hoc Netw. Workshop (Med-Hoc-Net)*, Jun. 2012, pp. 152–155.

[8] I. Abbasi and A. S. Khan, "A review of vehicle to vehicle communication protocols for VANETs in the urban environment," *Future Internet*, vol. 10, no. 2, p. 14, Jan. 2018.

[9] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 4, pp. 584–616, 4th Quart., 2011.

[10] S. Khatri, H. Vachhani, S. Shah, J. Bhatia, M. Chaturvedi, S. Tanwar, and N. Kumar, "Machine learning models and techniques for VANET based traffic management: Implementation issues and challenges," *Peer Peer Netw. Appl.*, vol. 14, no. 3, pp. 1778–1805, May 2021.

[11] C. Harsch, A. Festag, and P. Papadimitratos, "Secure position-based routing for VANETs," in *Proc. IEEE 66th Veh. Technol. Conf.*, Sep. 2007, pp. 26–30.

[12] S. Kudva, S. Badsha, S. Sengupta, I. Khalil, and A. Zomaya, "Towards secure and practical consensus for blockchain based VANET," *Inf. Sci.*, vol. 545, pp. 170–187, Feb. 2021.

[13] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, vol. 2, nos. 2–6, p. 11, 2012.

[14] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[15] F. Alam, I. Katib, and A. S. Alzahrani, "New networking era: Software defined networking," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 11, 2013.

[16] K. Kirkpatrick, "Software-defined networking," *Commun. ACM*, vol. 56, no. 9, pp. 16–19, Sep. 2013.

[17] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[18] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet-of-Things," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.

[19] A. Rahman, M. K. Nasir, Z. Rahman, A. Mosavi, S. S, and B. Minaei-Bidgoli, "DistBlockBuilding: A distributed blockchain-based SDN-IoT network for smart building management," *IEEE Access*, vol. 8, pp. 140008–140018, 2020.

[20] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2019.

[21] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.

[22] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[23] C. Bernardos, A. de la Oliva, P. Serrano, A. Banchs, L. Contreras, H. Jin, and J. Zuniga, "An architecture for software defined wireless networking," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 52–61, Jun. 2014.

[24] M. Abbasi, H. Rezaei, V. G. Menon, L. Qi, and M. R. Khosravi, "Enhancing the performance of flow classification in SDN-based intelligent vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4141–4150, Jul. 2021.

[25] G. A. Qadir, "Software defined network based VANET," *Int. J. Sci. Bus.*, vol. 5, no. 3, pp. 83–91, 2021.

[26] M. O. Kalinin, V. M. Krundyshev, and P. V. Semianov, "Architectures for building secure vehicular networks based on SDN technology," *Autom. Control Comput. Sci.*, vol. 51, no. 8, pp. 907–914, Dec. 2017.

[27] S. D. A. Shah, M. A. Gregory, S. Li, R. D. R. Fontes, and L. Hou, "SDN-based service mobility management in MEC-enabled 5G and beyond vehicular networks," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13425–13442, Aug. 2022.

[28] S. D. A. Shah, M. A. Gregory, S. Li, and R. D. R. Fontes, "SDN enhanced multi-access edge computing (MEC) for E2E mobility and QoS management," *IEEE Access*, vol. 8, pp. 77459–77469, 2020.

[29] M. Zhu, J. Cao, D. Pang, Z. He, and M. Xu, "SDN-based routing for efficient message propagation in VANET," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.* Cham, Switzerland: Springer, 2015, pp. 788–797.

[30] J. Bhatia, R. Dave, H. Bhayani, S. Tanwar, and A. Nayyar, "SDN-based real-time urban traffic analysis in VANET environment," *Comput. Commun.*, vol. 149, pp. 162–175, Jan. 2020.

[31] R. M. Alamgir, A. A. Shuvro, M. Al Mushabbir, M. A. Raiyan, N. J. Rani, M. M. Rahman, M. H. Kabir, and S. Ahmed, "Performance analysis of YOLO-based architectures for vehicle detection from traffic images in Bangladesh," in *Proc. 25th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2022, pp. 982–987.

[32] S. Rahimipour, R. Moeinfar, and S. M. Hashemi, "Traffic prediction using a self-adjusted evolutionary neural network," *J. Modern Transp.*, vol. 27, no. 4, pp. 306–316, Dec. 2019.

[33] J. Fan, Q. Hu, and Z. Tang, "Predicting vacant parking space availability: An SVR method with fruit fly optimisation," *IET Intell. Transp. Syst.*, vol. 12, no. 10, pp. 1414–1420, Dec. 2018.

[34] J. Fan, Q. Hu, Y. Xu, and Z. Tang, "Predicting vacant parking space availability: A long short-term memory approach," *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 2, pp. 129–143, Mar. 2022.

[35] Y. Feng, Y. Xu, Q. Hu, S. Krishnamoorthy, and Z. Tang, "Predicting vacant parking space availability zone-wisely: A hybrid deep learning approach," *Complex Intell. Syst.*, vol. 8, no. 5, pp. 4145–4161, Oct. 2022.

[36] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.

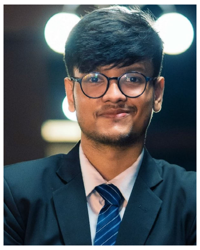
[37] X. Du, H. Zhang, H. V. Nguyen, and Z. Han, "Stacked LSTM deep learning model for traffic prediction in vehicle-to-vehicle communication," in *Proc. IEEE 86th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2017, pp. 1–5.

[38] C. Chen, *Freeway Performance Measurement System (PeMS)*. Berkeley, CA, USA: Univ. California, Berkeley, 2002.

[39] Y. Jin, L. Hou, and Y. Chen, "A time series transformer based method for the rotating machinery fault diagnosis," *Neurocomputing*, vol. 494, pp. 379–395, Jul. 2022.

ALI ABIR SHUVRO was born in Dhaka, Bangladesh, in 1999. He received the B.Sc. degree in computer science and engineering from the Islamic University of Technology (IUT), Gazipur, Bangladesh.

He is currently a Lecturer with the Department of Computer Science and Engineering, IUT. He is enthusiastic about different aspects of computer science. His research interests include computer networks, software-defined networks, machine learning, deep learning, and computer vision. He was awarded the Prestigious IUT Gold Medal, in 2022, for his outstanding academic excellence.



MOHAMMAD SHIAN KHAN received the B.Sc. degree in computer science and engineering (CSE) from the Islamic University of Technology (IUT), Gazipur, Bangladesh, in 2022.

His research interests include computer vision, machine learning, deep learning, and networking. He has been actively involved in research projects during his undergraduate studies. He is looking forward to continuing his research and making significant contributions in these fields.

MONZUR RAHMAN received the B.Sc. degree in computer science and engineering from the Islamic University of Technology, in 2022. His educational background in computer science, along with his passion for technology and innovation, has led him to pursue a career in the field of computer science research. He is particularly interested in studying the latest advancements in software development and artificial intelligence and has a strong desire to contribute to the ongoing research efforts in these areas. He is eager to continue his education and expand his knowledge and skills in the field of computer science in order to become a leading researcher in the field.

FAISAL HUSSAIN received the B.Sc. and M.Sc. degrees in computer science and engineering from the Islamic University of Technology, Bangladesh. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Islamic University of Technology. His research interests include D2D communication, SDN, VANET, network optimization, and network security.

MD. MONIRUZZAMAN received the bachelor's and master's degrees from the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), and the Ph.D. degree from Federation University Australia, in 2021. He is currently an Assistant Professor with the Department of CSE, IUT.

MD. SAKHAWAT HOSSEN received the B.Sc. and Ph.D. degrees in computer science and engineering (CSE) from the Islamic University of Technology (IUT), Gazipur, Bangladesh, and the M.Sc. degree in internetworking from the Royal Institute of Technology (KTH), Sweden. He is currently an Associate Professor with the Department of CSE, IUT. His research interests include D2D communication, SDN, VANET, the Internet of Things, RFID, cellular automata, combinatorial and evolutionary optimization problems, internet security, wireless sensor networks, IP networks, and VoIP.