Islamic University of Technology (IUT)

Department of Computer Science and Engineering (CSE)

SDN-based Time Series Traffic Flow Forecasting in VANET

Authors

Ali Abir Shuvro, 170041003

Mohammad Shian Khan, 170041046

Monzur Rahman, 170041047

Supervisor

Md. Sakhawat Hossen

Assistant Professor, Department of CSE,

Islamic University of Technology (IUT)

A thesis submitted to the Department of CSE

in partial fulfillment of the requirements for the degree of B.Sc.

Engineering in CSE

Academic Year: 2020-2021

9th May, 2022

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by *Ali Abir Shuvro*, *Mohammad Shian Khan* and *Monzur Rahman* under the supervision of *Md. Sakhawat Hossen*, Assistant Professor of the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh.

We are very grateful to our supervisor *Md. Sakhawat Hossen*, Department of Computer Science and Engineering, Islamic University of Technology (IUT), for his supervision, knowledge and support, which has been invaluable for us.

It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Ali Abir Shuvro

Student ID: 170041003

Mohammad Shian Khan

Student ID: 170041046

Monzur Rahman

Student ID: 170041047

Supervisor:

Md. Sakhawat Hossen

Assistant Professor

Department of Computer Science and Engineering

Islamic University of Technology

Acknowledgement

We would like express our gratitude towards IUT authority for granting us the fund and providing assistance required to implement our proposed system. We are indebted to our supervisor, Md. Sakhawat Hossen for providing us with insightful knowledge and guiding us at every stage of our journey, Finally, we would like to express our heartiest appreciation towards our family members for their continuous support, motivation, suggestions and help, without which we could not have achieved this progress in our work.

Abstract

Intelligent Transportation Systems(ITS) provides services for proper traffic assistance. Vehicular Ad-hoc Network(VANET) provides internet connectivity to vehicles and helps in traffic guidance. In this paper, traffic flow prediction is done using a modified transformer architecture for time-series vehicular data. Sequences are generated from the dataset for capturing temporal dependencies. The transformer model has been engineered to capture inter-feature correlations along with inter-sample correlations. Our transformer model has performed much better than other models like LSTM. We also propose a holistic networking model 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 incorporates data, control and management planes together in a network device, are incapable to adapt with this much data growth, bandwidth, speed, security, scalability compared to SDN as it provides with centralized programmable mechanism reliably. The trained parameters learned using the transformer model will be passed throughout the network for traffic guidance. Similar sized packets are passed using a simulator to demonstrate the time required for the propagation of the parameters.

Keywords: *Vehicular Ad-hoc Network, Transformer, Sequence length, Encoders, Attention mechanism, Traffic flow, Software-defined Network*

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Intelligent Transportation Systems (ITS) technologies provide services to better traffic and transportation information. It makes transportation smarter by providing meaningful insights to 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 in forecasting the traffic flow. LSTM models have a higher performance in comparison to models which incorporate algorithms like HTM [3]. Short-term prediction of traffic conditions is necessary for overcoming traffic congestion. [4]

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

VANETs play a key role in ITS. VANET stands for vehicular ad hoc network. Safety and road-side 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. 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. Software-Defined VANET can operate differently based on the degree of control of the SDN controller. SDVN also provides some benefits which can be utilized to provide many services. [5]

SDN-based VANETs overcome the problems in simple VANET as it decreases the load on the overall system by the separation of the data plane and the control plane. Software-Defined Networking is such a field that breaks this vertical integration separating the control logic from routers, switches and providing flexibility, dynamicity, optimality, and centralized logical control over the system . [1]

SDN enables better network configuration and monitoring as it focuses on bringing all the network configurations under a remote controller which is connected with a range of switches that form the data layer. It is in the control layer where all sorts of network configuration and programming take place. SDN makes the configuration of a VANET structure much more efficient and makes network monitoring easier. So SDN can improve network management issues like

frequent changes in network conditions and states, providing support for network configuration in a high-level language, providing better visibility, and performing network diagnosis and troubleshooting. [6] Bringing SDN architecture to wireless mobile networks for leveraging mobile improvements even better than existing technology [7]

Traffic prediction is quite a challenge to tackle as spatial-temporal data is needed to be handled. [8] . Traffic prediction data can be represented with graph and Graph Neural Network along with RNN can be used for prediction of data.[9].

However, there are some limitations to it due to the continuous nature of time and the adjacent behavior of space. The use of RNN 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 has emerged as it works well with sequential data. [9]. A combination of GCN and transformer has outperformed several other models mainly consisting of RNN in the past. [8].

Traffic data is in time-series format, so through a transformer-based learning model, we have tried to forecast traffic data with better performance, and a computationally efficient model. Furthermore, we suggest a holistic model 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.

1.2 Research Challenges

We faced several challenges during the research work. There is not much data available globally on traffic. The dataset that we collected provided periodic data which was large in size as it was collected for over months. So training our model on such a big dataset was a challenge. We ran into a GPU capacity shortage as the dataset was too big to be trained on some limited GPU that google collab provided. The model that we used was transformer-based but as a matter of fact, transformers are designed mostly for processing strings, words of natural language. So we had to tweak the transformer model to take time-series data as input. Meanwhile simulating the VANET based traffic condition over Mininet software was a challenge as virtual networks had to be created and optimal controller was needed to chosen.

1.3 Contribution

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 not fixed spatial dependencies. Following are the contributions we made through our thesis work:

1. We improved the performance of traffic flow prediction.
2. Using multivariate models, we ensured inter-feature dependencies by modifying transformer attention mechanism
3. We simulated weight propagation in an SDN-enabled environment.
4. We are providing a holistic model for traffic guidance in an up-and-coming SDN-based VANET.

2 Background Study

2.1 VANET

Vehicular Ad Hoc Networks is a potential field of research and has caught lot of interest. It is this much eye catching because of its potential to provide vehicle road safety, to increase efficiency of traffic and to provide convenience and comfort to for passengers and drivers. Mobile Vehicular Cloud and review cloud applications can be considered as significant technology to develop intelligent transportation [10]. Vehicular ad hoc networks(VANET) refers to the creation of mobile ad-hoc network created in the domain of vehicle which support communications among vehicles, road side unit and base station to provide safe and efficient transportation. Every single node in VANET is responsible for participating and acting as router of the network. They communicate through intermediate nodes which are within their own transmission range. There are three types of architecture in VANET and those are pure cellular wireless local area network, pure ad-hoc networks and hybrid networks [11]. There are basic characteristics of Vehicular networks. There are many applications associated to this field. There are requirements which comes with many challenges. Solutions are also there which can be considered to overcome theses challenges. [12] With the advancement of technology and the establishment of smart cities, there is an increasing need for Intelligent Transport system (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 feasible manner and overcome such challenges [13]. Security is a concern in case of routing of the VANET nodes. In this paper they provide a secure position-based routing scheme of which security overhead is low and the scheme is deployable. They considered enhancing network robustness, defense mechanisms [14]. 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), maintain fair selection than Proof

Of Stake (PoS) as well as the randomness of consensus nodes in a public distributed network of vehicles. They also introduced a filtering technique that is based on Service Standard Score to detect and eliminate malicious nodes and efficiently choose nodes. There is another consensus protocol beside PoW and PoS which is called Practical Byzantine Fault Tolerant (PBFT) which scales to only tens of nodes. For this their goal of such design of their work is to make PBFT usable in public blockchain network by introducing PoD in VANET [15].

Table 1: Related works on VANET

Title of the paper	Published year	Main Contribution
Machine learning models and techniques for VANET based traffic management - Implementation issues and challenges - [13]	2021	Discusses various Machine Learning techniques which can facilitate and work well with VANET architecture.
Towards secure and practical consensus for blockchain based VANET - [15]	2021	Proof of Driving (PoD) proof variant was introduced in a blockchain-based ride-sharing service in VANET.
A Review of Vehicle to Vehicle Communication Protocols for VANETs in the Urban Environment - [11]	2018	Discusses VANET along with the pure cellular, pure ad-hoc, and hybrid structures.
Vehicular Cloud Computing- [10]	2012	Mobile Vehicular Cloud and cloud applications can be considered significant technologies to develop intelligent transportation.
Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions - [12]	2011	Talks about basic characteristics, applications, requirements, challenges, and relevant solutions regarding vehicular networking.
Secure Position-Based Routing for VANETs - [14]	2007	Provides a secure position-based routing scheme where security overhead is low and is deployable.

2.2 SDN

The widespread IP network is very complex and difficult in case of management, to configure according to predefined policies as well as reconfiguring 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 field that breaks this vertical integration separating the control logic from routers, switches and provides flexibility, dynamic, optimal and centralized logical control over the system [1]. There are variety of high quality services such

as mobile devices, virtualization, high automation and security, efficient Big-Data management. There are other advance user requirements which the current traditional networks can not handle efficiently and provide such services as it is not dynamic compared to SDN. SDN is software-based whereas traditional networking is hardware-based. Due to design limitations, 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 adjustment will be by changing the configuration of the devices which has restricted the network operator who wants to scale their networks on traffic demands. The traditional design principles, that incorporates data, control and management planes together in a network device, are incapable to adapt with this much data growth, bandwidth, speed, security, scalability compared to SDN as it provides with centralized programmable mechanism [16]. SDN decouples the switch or router and its data-routing instructions by adding application programming interface. The centralized control abstracts the underlying network devices from applications and network services. In this ONF whitepaper they have introduced three layers which depicts the structure of SDN. These layers are: Infrastructure layer, Control layer and Application layer [17]. 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 southbound API is used between Control layer and Infrastructure layer. It uses a set of commands that allows forwarding of data and determines the behaviour and condition based on application requests sent via northbound APIs [18]. Wireless sensor networks is a low rate network with little 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 Internet of Things (IoT) [19]. IoT is a network which is open, geographically distributed, and consists of heterogeneous networking infrastructures. Managing these in dynamic situation is an important challenge. This challenge can be overcome by using SDN with IoT. In this paper, they have designed SDN for IoT for dynamically achieving quality to different IoT tasks in heterogeneous wireless networking [20]. For smart cities an IoT-SDN structure can be incorporated with Black SDN for higher security. In this paper, they also introduced Network Function Virtualization (NFV) to bring benefits like energy savings, network scalability and load balancing [21]. SDN faces many challenges in its way and to achieve the goal what it shows, the challenges must be resolved. In this study, they discussed about these challenges in the area of performance, scalability, security, and interoperability [22]. The rapid growth of

internet and mobile communication technlogy has led us to more complexity. Now these needs to be efficiently organized, managed and optimally maintained. This is why more intelligence are needed to be incorporated and traditional networks failed to do so. For its inherently distributed features, machine learning is very difficult to apply. But 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 [23]. So SDN can improve network management issues like frequent changes in network conditions and states, providing support for network configuration in high level language, providing better visibility and performing network diagnosis and troubleshooting [6]. Bringing SDN architecture to wireless mobile network for leveraging mobile improvements even better than existing technology [7].

Table 2: Related works on SDN

Title of the paper	Published year	Main Contribution
Software-Defined Networking: The New Norm for Networks - [17]	2014	This ONF whitepaper has introduced three layers that depict the structure of SDN. These layers are: Infrastructure layer, Control layer, and application layer.
DistBlockBuilding: A Distributed Blockchain-Based SDN-IoT Network for Smart Building Management - [21]	2020	1. SDN, IoT and Blockchain is used together. 2. Smart building scenario is considered.
A Software Defined Networking Architecture for the Internet-of-Things - [20]	2014	This paper has designed SDN for IoT for dynamically achieving quality for different IoT tasks in heterogeneous wireless networking.
DistBlackNet: A Distributed Secure Black SDN-IoT Architecture with NFV Implementation for Smart Cities - [24]	2019	1. They incorporated an IoT-SDN structure for smart cities with Black SDN for higher security for smart cities. 2.They also introduced Network Function Virtualization(NFV) to bring benefits like energy savings, network scalability and load balancing.

Are We Ready for SDN? Implementation Challenges for Software-Defined Networks - [22]	2013	1. Discussed the architecture of SDN 2. Benefits and necessity of SDN (performance, flexibility, scalability, security) 3. Challenges faced by SDN
Software-Defined Networking: A Comprehensive Survey - [1]	2014	This survey paper tells how SDN has more facilities than traditional network and details about basic SDN architecture.
A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements - [19]	2017	This survey paper tells about how SDN can be incorporated with Wireless Sensor Network to provide efficiency, sustainability which is called SDWSN.
A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges - [23]	2018	The work focused on implementing machine learning knowledge and technologies for improving the performance, security, scalability, and efficiency of the SDN.
Improving Network Management with Software Defined Network - [6]	2013	Tells about how SDN can improve network management issues like frequent changes in network conditions, providing support for network configuration in high level language, which provides better visibility and performs network diagnosis and troubleshooting.
Software Defined Networking - [18]	2013	This survey paper describes working procedure inside SDN architecture.
New Networking Era - Software Defined Networking - [16]	2013	This survey paper describes why SDN is better than traditional networking in detailed manner.
An Architecture for Software Defined Wireless Networking - [7]	2014	This paper worked for bringing SDN architecture to wireless mobile network for leveraging mobile improvements even better than existing technology.

2.3 SDN-based VANET or SDVN

For rapid growth of road side accident, to ensure passengers' safety, an ad hoc network that is vehicular ad hoc network is being encouraged. To prevent from the limitations and complexities of basic VANET structures by handling the whole network from a single remote controller, Software Defined Networking (SDN) is used along with VANET which is called SDN-VANET [25]. Common VANET has some problems that can be overcome by incorporating the technology of software-defined networks. The adaptability, controllability, versatility of controlling the network as a whole of SDN helps to build an effective and secure VANET with simplified network control [26] 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. Software Defined VANET can operate differently based on the degree of control of the SDN controller. SDVN also provides some benefits which can be utilized to provide many services [5]. A global optimal route can be found to efficienty propagate message 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 efficiently message propagation in VANET and outperforms some other traditional protocols [27].

Table 3: Related works on SDN-based VANET or SDVN

Title of the paper	Published year	Main Contribution
Software defined network based VANET. [25]	2021	Depicts SDN-based VANET, its working, benefits, challenges and services, applications, and security attacks.
Architectures for building secure vehicular networks based on SDN technology. [26]	2017	Proposes secure architecture of VANET utilizing the adaptability, controllability, versatility of controlling the network as a whole using SDN.
Towards software-defined VANET: Architecture and services. [5]	2014	Proposes an SDN-based architecture of VANET and its operational modes, benefits and services.

SDN-based routing for efficient message propagation in VANET. [27]	2015	Proposes Centralized Routing Protocol (CRP) which is the SDN-based routing framework that uses modified Dijkstra's algorithm for efficiently message propagation in VANET and outperforms some other traditional protocols.

2.4 Traffic Flow Forecasting

SDVN can be used to detect and predict traffic collaborating with machine learning. From machine learning, K-means clustering can be used to detect and LSTM with RNN can be used to then predict the traffic condition [28]. HTM can be used alongside LSTM for short-term arterial traffic prediction.Several Machine learning techniques have been deployed to predict traffic state. LSTM models have a higher performance with comparison to models which incorporate algorithms like HTM [3].Short term prediction of traffic condition is necessary for overcoming traffic congestion [4]. Results by applying self adjusted Neural Network(NN) has been satisfactory as well [29]. 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 [30]. In this paper, they propose a long short-term memory (LSTM) based regression model to predict 24-hour traffic counts data. They didn't talk about the algo or any technical differences they made in details. They didn't show any calculation or what type of data they took or what are the features and what is the traffic output. They just told they had dataset, they trained, tested etc etc and compared with logistic regression (Scikit). Steps: Collecting data, constructing Stacked LSTM Model to implement regression, found efficiency comparing with logistic models [31]

Table 4: Related works on traffic flow forecasting

Title of the paper	Published year	Algorithms used	Datasets used
An evaluation of htm and lstm for short-term arterial traffic flow prediction. [3]	2015	HTM	South Australian Department of Planning, Transport and Infrastructure (DPTI) has provided the data.

SDN-based real-time urban traffic analysis in VANET environment [28]	2020	LSTM	Obtained by simulation using NS-3 and SUMO
Research on campus traffic congestion detection using BP neural network and Markov model - [4]	2016	Markov, NN	Collected in Wuhan University of Technology from 7:00 am to 6:30 pm by cameras. Video data is collected where speed and densities are measured and stored.
Traffic prediction using a self-adjusted evolutionary neural network -[29]	2019	NN (self adjusted)	For three days, 1500m length highway road of six-lane sections are used to collect the data
Stacked LSTM Deep Learning Model for Traffic Prediction in Vehicle-to-Vehicle Communication - [31]	2017	LSTM, vehicle count prediction	In this paper, they propose a long short-term memory (LSTM) based regression model to predict 24-hour traffic counts data.
Traffic Flow Prediction With Big Data - A Deep Learning Approach - [32]	2014	Traffic flow prediction, NN	Caltrans Performance Measurement System (PeMS)
T-GCN; A Temporal Graph Convolutional Network for Traffic Prediction - [30]	2019	T-GCN	SZ-taxi dataset and Los-loop dataset
Traffic flow forecasting with Particle Swarm Optimization and Support Vector Regression - [33]	2014	SVR, PSO	6 days of traffic flow data in 2nd Ring Road of Beijing.

Traffic prediction is quite a challenge to tackle as spatial-temporal data is needed to be handled [8]. Traffic prediction data can be represented with graph and Graph Neural Network along with RNN can be used for prediction of data [9]. However, there are some limitations to it due to the continuous nature of time and adjacent behaviour of space. Use of RNN is

limited due to temporal dependencies for exploding and vanishing node problems. GRU and LSTM overcomes this shortcoming of RNN. Therefore, several algorithms combining Traffic Transformer has emerged as it works well with sequential data [9].
A combination of GCN and transformer has outperformed several other models mainly consisting of RNN in the past [8].

Table 5: Related works on traffic prediction using Transformer

Title of the paper	Published year	Weaknesses	Datasets used
Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting-[34]	2019	Additional factors like weather and social events has not been taken into account for traffic prediction.	PeMSD4 and PeMSD8
Spatial-Temporal Transformer Networks for Traffic Flow Forecasting - [9]	2020	Does not perform good in compared to other algorithms while trying to predict short-term traffic.	PeMSD7(M), PEMS-BAY
Traffic transformer - Capturing the continuity and periodicity of time series for traffic forecasting - [8]	2020	A drawback of this paper is that it only accounts for temporal attention mechanism.	METR-LA, PEMS-BAY

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.

2.5 Some notable work done on traffic prediction

As the attention mechanism has emerged to provide significantly better result [35], the use of this is seen in many sectors. There has been a few notable work done on traffic prediction and forecasting using transformers. In the following sub-sections, we will analyze some of the related works done on this field where notable results are seen.

2.5.1 ASTGCN: Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting

Guo et al. suggested a transformer-based model for traffic flow detection in VANET. Previous models cannot capture spatial-temporal correlation inside the data. Long-term dependencies were not properly addressed as well. Correlation between vehicle traffic in hourly, weekly, and daily manner was not addressed which resulted in various problems during forecasting. They used an attention mechanism to capture the spatial-temporal correlation.

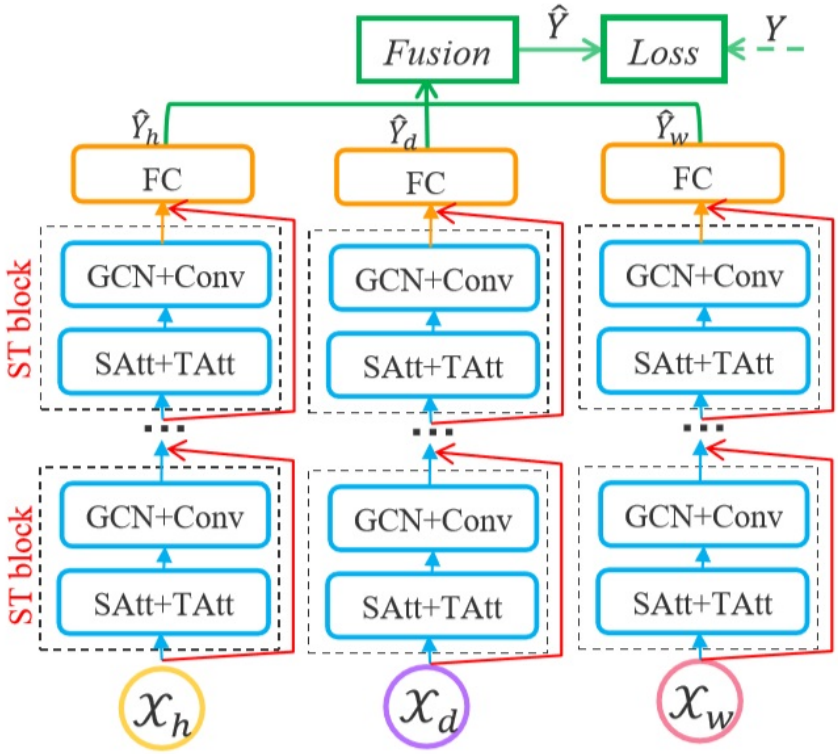


Figure 1: The framework for ASTGCN

Spatial attention captures all the special spatial relations while temporal attention captures all the timed relations. Again, Graph Convolution Network is used to achieve an idea about neighboring areas. Finally, hourly, daily, and weekly models are fused to finally find out its loss.
But there are some limitations in the work. A transformer model with time series data can be used which would result in the model understand the temporal correlations much better. Furthermore, an encoder-based model can also be used to increase efficiency and reduce computation. Simulated analysis in the SDN environment can be explored. Dynamic changes in the VANET can be easily utilized using SDN.

2.5.2 Spatial-Temporal Transformer Networks for Traffic Flow Forecasting

Xu et al. talked about the Dynamic process inside the Graph Convolutional Network. By introducing transformers with GCN, they have achieved such results. Traffic anomalies have been resolved. Bigger datasets with more features were inefficient in training which their model was able to cope with while the model has been developed for bidirectional traffic. Dynamic Graph Convolutional Network is used to capture unprecedented incidents. Fixed Graph Convolution captures all the static data relations. Again, Spatial and Temporal transformers are merged to create a spatial-temporal block. Finally, a prediction layer is introduced to get the required output.

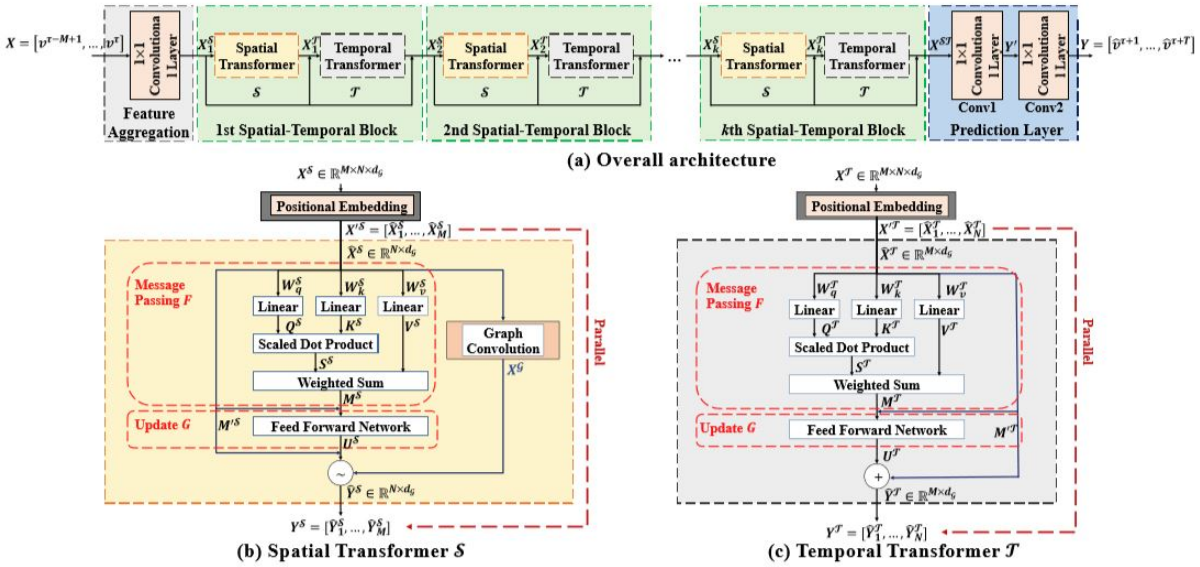


Figure 2: The framework for STTN

Their research work also has some limitations and gaps. Dependencies across specific periodic intervals can be addressed intuitively using transformers and converting the dataset into time series data. Simulated analysis in the SDN environment can be explored. Analysis of the impact of attention mechanisms for their model can be done.

3 Problem Formulation

Our goal is to forecast the *average occupancy* of a station at a certain time given an already observed time series dataset. Basically, given the station, number of samples, percent observed, total flow, average speed and some other parameters, we need to find the average occupancy of that region.

At a particular time t, Given $X_t[\ x_t^1,\ x_t^2,\ x_t^3,\ x_t^4,\ x_t^5,\ x_t^6,\ x_t^7]$ we need to predict x_{t+1}^6 or Y_{t+1} which contains the *average occupancy* of time t+1.

Here, x^1, x^2, x^3, x^4, x^5, x^6 and x^7 are station number, station length, number of samples, percent observed, total flow, average occupancy and average speed respectively.

4 Methodology

After analyzing the problem at hand, we came to the conclusion that time-series data is needed to be used for formulating a proper solution. By training a model on this data, we will be able to produce near to accurate predictions of the condition of traffic in the future. We will also be able to produce predictions at specific time intervals through this methodology. We needed to preprocess the data so that, we obtain the desired time series data using which we might pass down accurate predictions. Different deep learning models were used which have been proved in the past to have worked phenominally well in terms of time-series mode of data.

4.1 Model Architecture

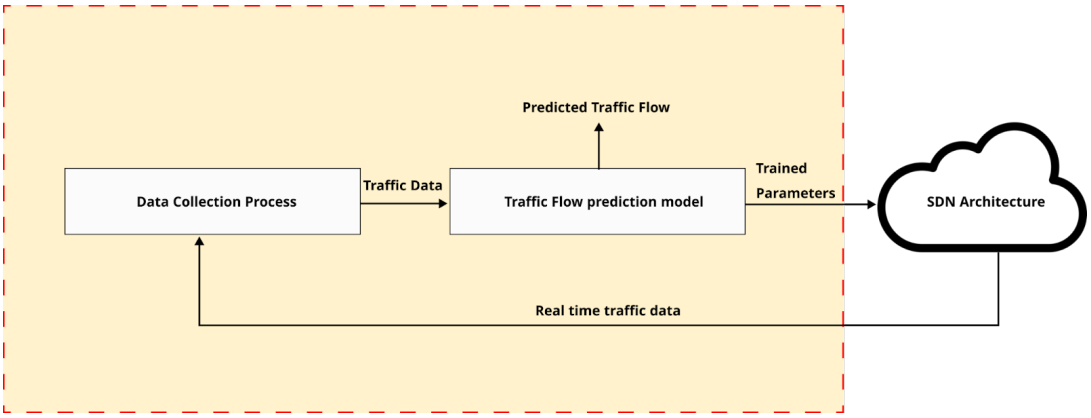


Figure 3: Architecture Overview

As shown in figure 3, the proposed system is consisted of three segments. They are *Data Collection*, *Traffic Flow Prediction Model* and *SDN Architecture*.

Now, the traffic flow prediction model can be expanded to show its data flow including the sequence generation.

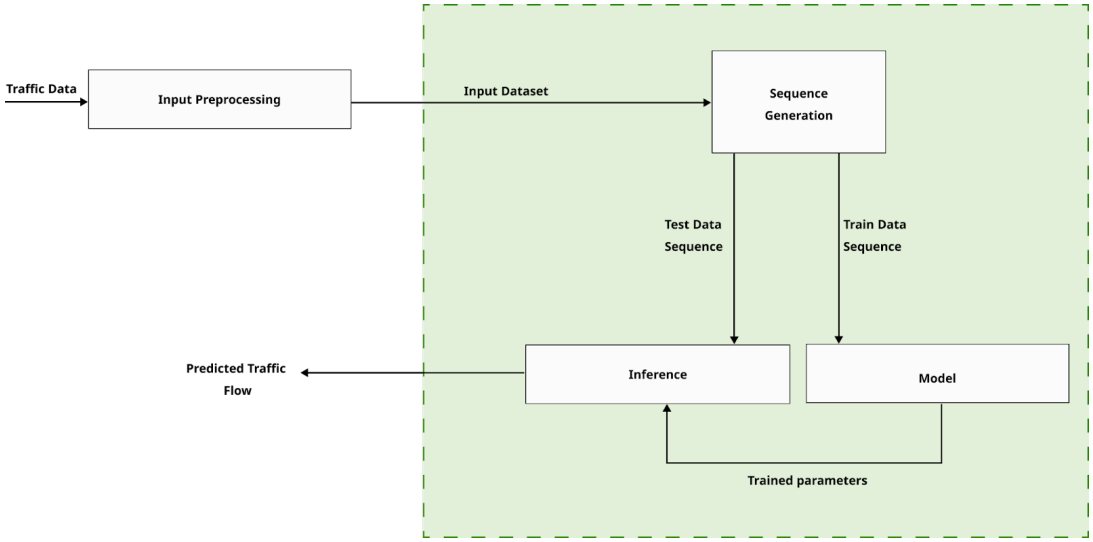


Figure 4: Data flow of the traffic flow prediction model

We collected data from the Caltrans Performance Measurement System (PeMS)[36]. The data is collected from nearly 40,000 detectors across the freeway system of several major metropolitan areas of California. After collection of the data, we pre-processed the data for enhancing the performance of the model, the steps involve cleaning, reduction, and transformation of data. Through the data collection process, we have obtained the necessary traffic data for training our model. We needed to generate time-series data from the pre-processed data.

For transforming the traffic data into time-series data, the data had to undergo sequence generation. We generate this sequence using the sequence generator. We split the data into two parts, the train set and the test set. Train set helps the prediction model to make proper parameters along with parametric weights which are later evaluated by the test set. This sequential data enters the model, which starts the next phase of our architecture. This traffic data undergoes through state-of-the-art deep learning model for future prediction of traffic data. During training, the model gradually starts to get probabilistic and holistic view of the situation and keep tuning its parametric weights accordingly.

The model block in Figure 4 can be further expanded for proper visualization and understanding. The data flow of this figure will be discussed further in Section 5.4.

After training is done, we enter the next phase of our architecture which is feeding the SDN architecture with the trained weights obtained from the prediction model. SDN consists of the control plane and the data plane, we convey the weights to the base stations in such a way that the vehicles have the proper real time prediction available based on the current state of the traffic.

Therefore through our model, the vehicle receives real-time traffic guidance regarding important

factors like congestion, speed of traveling via that street etc though the prediction model. After that, it gets delivered rapidly in real-time through the road side base station that are connected via an SDN architecture, in this way through the constant feedback loop the vehicles receive better traffic guidance.

5 Implementation

The segments mentioned in section 4.1 that needs to be implemented are described in this section. Along with that, we also talk about the dataset that we have used for our experiment. Lets look at the environment used for conducting the experiment.

5.1 Dataset

The dataset we used was obtained from Caltrans Performance Measurement System(PeMS) [36]. Nearly 40,000 traffic sensors are deployed in almost all notable streets in the state of California and traffic data is collected everyday. We have used hourly data from PeMS from district 7 . We have taken the hourly data for the whole year of 2021.

We will be using multivariate time-series data. For that reason we have taken the following variables from the collection:

Table 6: 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

5.2 Preprocessing

At first, we collected the raw data from the PeMS website. The data was in an hourly fashion. We selected a year and collected that year's full data. We had to pick a district which had busy streets so that the stations will be able to give us data that varies.

After analyzing the PeMS website and the stations along with the positions and traffic load, we decided on the dataset that we want to work on. We removed some of the parameters that will

not impact on the outcome of our experiment. This ensured that the experiment remained light and we would not be creating a bloated model to fit the data.
Next, we use mean normalization on the dataset so that the values are in the range of $[0, 100]$. This ensures that none of the parameters are ignored by the model.

5.3 Creating Sequence

In order to convert regular dataset into time-series data, sequences are needed to be created. This essentially means making a sequential copy of a portion of the data and each time introducing 1 more sample. The sequence generation is important so that the model gets an idea of past and present situation in different spatial and temporal perspective in that area which has impact on future condition. The model gets the whole sequence of data and predicts just the single sample while the next time, again a sequence will be fed and the next sample will be predicted by the model.

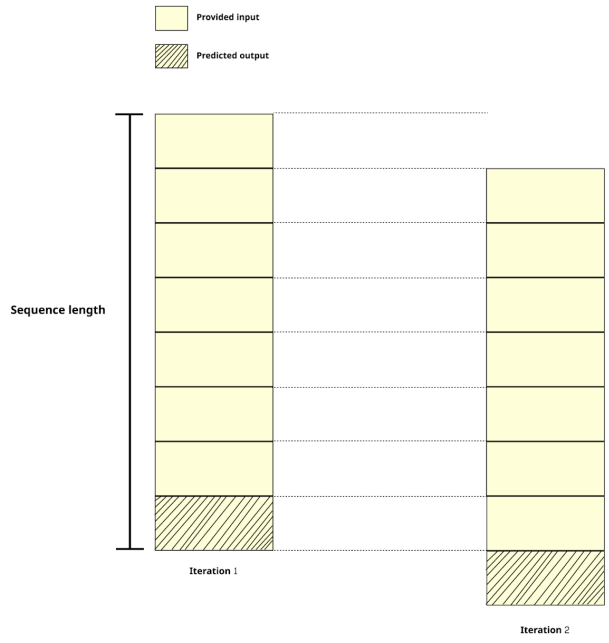


Figure 5: Processing sequences in the dataset

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 sequence number that is picked determines the final size of the dataset. For a large sequence

length, the dataset will become large as well and vice-versa. In fact, the dataset size will be exactly the multiple of the sequence number.
After doing an 80-20 train-test split the shape of the datasets obtained:

Table 7: 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

The process of creating sequences took around 2-5 minutes on average.

5.4 Training

The training part is the heart and soul of the experiment and it comprises mostly of how the data is being interpreted by the model. We have used one of the latest architectures which is the Transformer Architecture which has opened a new era in Natural Language Processing field. Transformer Architecture is mainly used in NLP and used for word generation/translation which is similar to category prediction or logistic regression. For regression of a continuous data, the basic transformer architecture has been made aligned to 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 generated batch-wise data. This batch-wise data is at first fed to Embedding layer and then to 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 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.

5.4.1 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. Multi-Head Attention layer is based on self-attention mechanism in multiple head and Positional Feed Forward network that consists of fully connected Neural

Networks which is applied to each of the position identically. The output of the first encoder layer is feed to next encoder layer and so on. Finally the output from the last encoder layer goes to decoder layer for further attention mechanism.

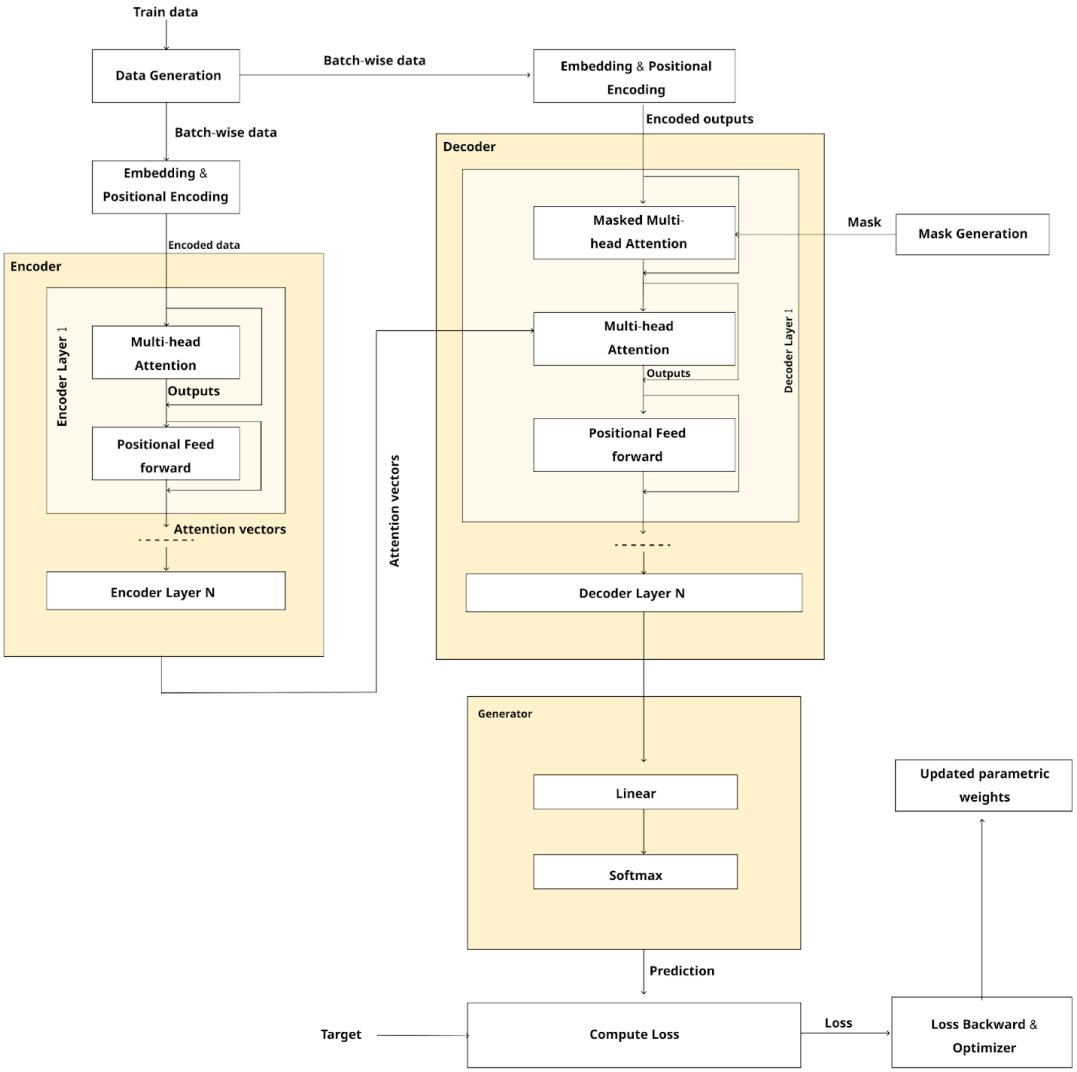


Figure 6: Data flow through the internal architecture of the model

5.4.2 Multi-Head Attention Layer of Transformer Architecture

As per figure 7, the encoded data from previous layer is used to calculate *query*, *key* and *value*. These are calculated after the input is being fed to linear layer. The weights, calculated using backpropagation, of the linear layer are used to calculate *query*, *key* and *value*.

$$query = Linear_Q(s) \tag{1}$$

$$key = Linear_K(s) \tag{2}$$

$$value = Linear_V(s) \tag{3}$$

This query represents each of the sequence vectors whereas the key and value represents all the sequence vectors.The *query* and *key* value is multiplied to calculate the *score* matrix. This multiplication finds out the connection of a sequence vector with all other sequence vectors. So it will produce a single vector per sequence vector. So for sequence length, the output vector will have row of sequence length. It determines how much focus a sequence should put on other sequences. Each sequence will have a score that corresponds to other sequence in that time step. It is divided by square root of d_k which represents number of features per head. It is to allow more stable gradients so that multiplying values does not put any exploding effect in the calculation. After that a softmax operation is applied to the score matrix which generates the attention weights.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{4}$$

Then the attention weights are being used to do weighted sum of the *value* to generate the output vector. The higher the attention weights the more the value will persist thus drowning out the irrelevant sequence samples. Then the output vector is being fed to final linear layer with a residual connection to produce an output which is ready to go to next layer.

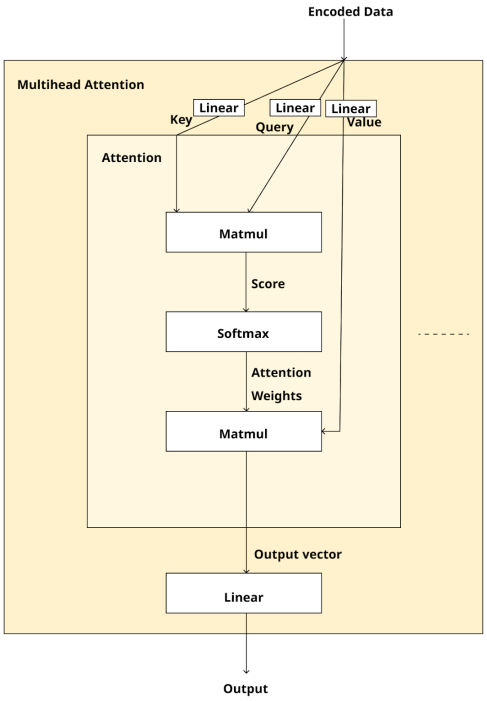


Figure 7: Multi-Head Attention Mechanism

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. That means it will be able to learn from different point view generalizing more form of representation thinking that each head will learn something different.

5.4.3 Multi-Head Attention Layer of Our Architecture

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

As multiple features are included in such situation, the standard self-attention mechanism of transformer is not enough to focus on each and every possible features along with focusing on sequence of samples.

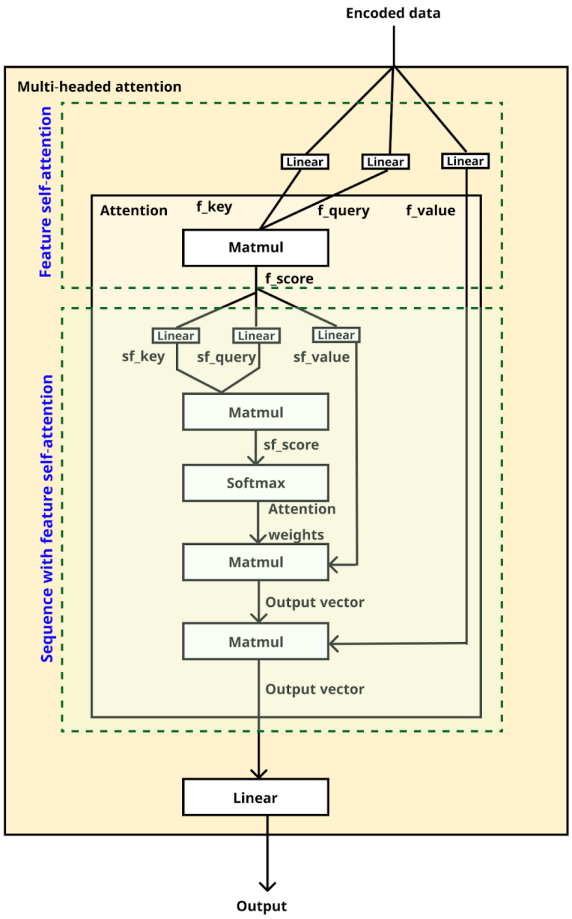


Figure 8: Extended Multi-Head Attention Mechanism

As per figure 8, this self-attention mechanism needs two kinds of six linear layers. First type of linear layer is focused on features and the other is on sequence. The encoded data from previous layer is used to calculate f_query, f_key and f_value. These are calculated after the input is being fed to feature focused linear layer.

$$f\_query = Linear_{Q_F}(s) \tag{5}$$

$$f\_key = Linear_{F_K}(s) \tag{6}$$

$$f\_value = Linear_{F_V}(s) \tag{7}$$

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

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

$$sf\_query = Linear_{Q_{SF}}(f\_score) \tag{8}$$

$$sf\_key = Linear_{K_{SF}}(f\_score) \tag{9}$$

$$sf\_value = Linear_{V_{SF}}(f\_score) \tag{10}$$

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.

$$Attention(Q_F, K_F, V_F) = (softmax(\frac{Q_{SF} * K_{SF}^T}{\sqrt{d_k}})V_{SF})V_F \tag{11}$$

Then the attention weights are being used to do 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 the priority. Then the output vector is being fed to final linear layer with a residual connection to produce an output which is ready to go to 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.

5.4.4 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 input is x then:

$$PFFN(x) = ReLU(xW_1 + b_1)W_2 + b_2 \tag{12}$$

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

5.4.5 The Decoder

The concept is similar to 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. There are three differences between the encoder and the decoder.

1. There is a linear layer and softmax layer inside which the output of the decoder layer is passed which produces the impact factor of every features on the predicted value.
2. Decoder has a sub-layer which is called Masked Multi-Head Attention which prevents seeing into the future.
3. Decoder has a sub-layer which is called Encoder-Decoder Multi-Head Attention.

First the data is passed through the Embedding and Positional Encoding block to first decoder layer. In case of training the decoder uses "$teacher\ forcing$" method. The encoded data then passed to Masked Multi-Head Attention mechanism where decoder layer uses mask to prevent "*cheating*" by masking the input passed to decoder layer so that it cannot see the values which it is not supposed to see while teacher forcing method is used. Here the self-attention among the outputs of decoder is measured. Then the output of this layer goes to next sub-layer which is Encoder-Decoder Multi-Head Attention.
The calculation is same as encoder layer's one but the input to this layer is from encoder layer,

contains information about self-attention of the original input, and the output of first sub-layer of the decoder layer. So the calculation here denotes how each sequence is connected to other sequence between the input and output of the model. The calculation finds out the holistic attention weights between original input and output. Then the output is forwarded to Position-Wise Feed Forward Network followed by the next decoder layer if not the last.

5.4.6 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 on the target value. The weighted sum of the features will be the target output value. In training phase the prediction is compared with the actual value and loss is being computed. After that, the loss is backpropagated using Loss Backward and Optimizer. There, the parametric weights of the model are updated which are used later for inference and finally to calculate the predicted value.

6 Experiment

6.1 Environment and Configuration

Table 8: Environment and Configuration for the experiment

Environment	Google Colab Pro
RAM	25GB
GPU	16GB
Storage	200GB

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

The reason for using Google Colab Pro is that while creating sequences, the size of the dataset multiplies with the sequence number. So, even a small dataset after generating sequences gets quite large.

6.2 Evaluation Metrics

6.2.1 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:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{13}$$

6.2.2 RMSE

RMSE or Root Mean Squared Error is an error calculating method. It is more popular in its use at 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:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}} \tag{14}$$

6.2.3 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 accuracy for continuous variables. If Y be the actual output and $\hat{Y}$ be the predicted output then we get:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid Y_i - \hat{Y}_i \mid \tag{15}$$

6.3 Experimental Results

The model was trained for around 30 minutes on the Google Colab Pro environment and the following hyper-parameters were provided:

Table 9: Hyperparameters used for the experiment

Hyperparameter	**Value**
Batch Size	200
Learning rate	0.001
Number of Encoders	5
Dropout	0.2

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. 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 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 contraint in mind, the number of encoders have been chosen to be 5.

We have used the target variable as *Average Occupancy* and with different sequence lengths. We measured the model in terms of MSE, RMSE and MAE. Each of the models were trained for around 30 minutes. The results that were obtained:

Table 10: MSE, RMSE and MAE values obtained from the predictions (Normalization range [0,100])

	MSE	**RMSE**	**MAE**
Our Model (Sequence length = 8)	**23.29139**	**4.82612**	**3.17355**
Our Model (Sequence length = 4)	27.37812	5.23241	3.60564
Our Model (Sequence length = 2)	38.26981	6.18625	4.18914
LSTM (Sequence length = 8)	29.65757	5.44588	3.70243

This shows the variation of different types of errors on the basis of the change of sequence lengths. Both the sequence length 4 and 8 have produced better results compared to LSTM model of sequence length = 8. In the case of our models, we have kept the training time to about 30 mins each while the LSTM model was trained for around 2 hours. We have kept the batch size as 200 for this portion and learning rate of 0.001. The number of encoders have been kept the highest value possible i.e. 5 to achieve minimum error.

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

		Sequence Length	
		4	**8**
4* **No. of Encoders**	**2**	93.6362	67.17477
	3	30.67938	27.05175
	4	26.87737	24.83311
	5	27.37812	23.29139

It is evident that the number of encoders have a direct impact on the model's learning. With the increase in the number of encoders, the model seems to learn better and give a lesser error.

6.4 Result Analysis

6.4.1 Impact of sequence length

It is seen that with the increase of 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 showed by our transformer model where increasing the sequence length decreased the error upto 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.

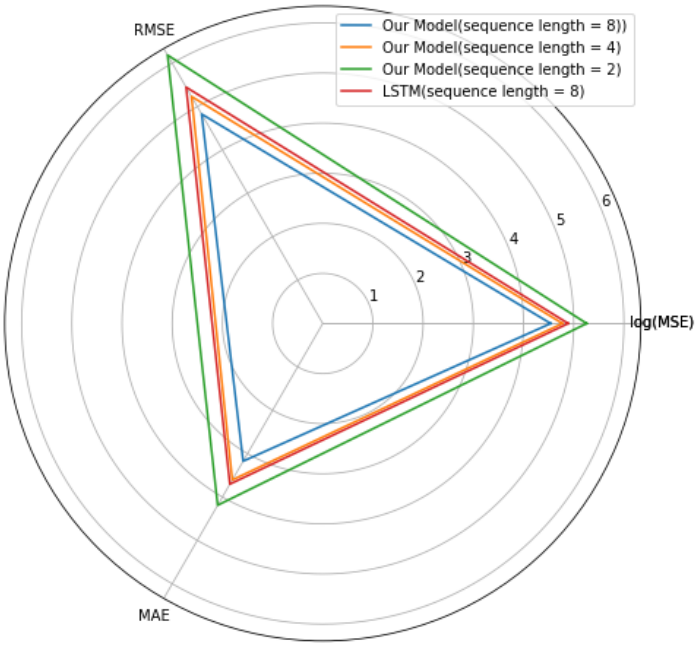


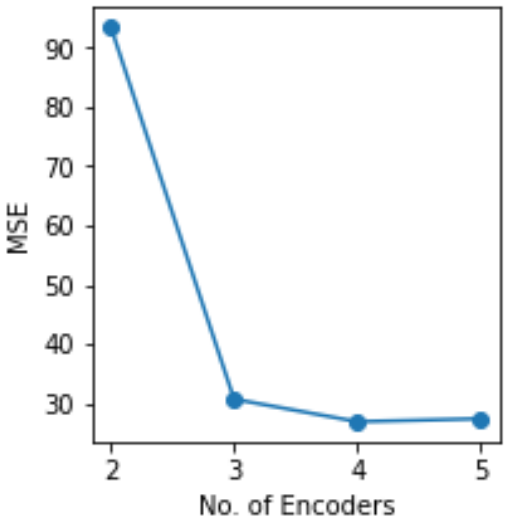
Figure 9: Comparison between different sequence lengths of our model and LSTM

If we look at the radar chart in Figure 9, we will see that our model of sequence lengths 4 and 8 has outperformed the LSTM model of sequence length = 8 in all three metrics.

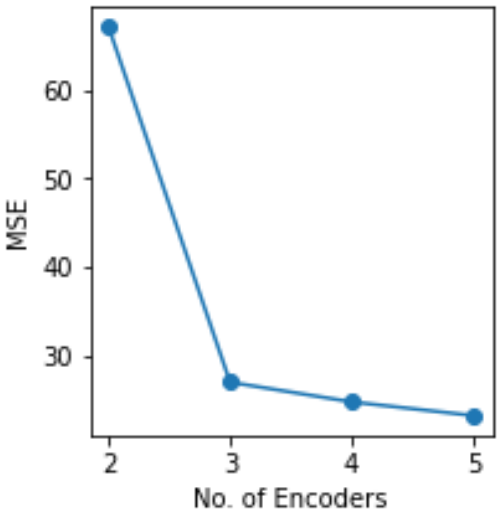
6.4.2 Impact of number of encoders

The number of encoders directly help 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 have made the model improve in performance:

Figure 10: Impact of number of encoders in MSE



(a) MSE for sequence number = 4



(b) MSE for sequence number = 8

Our experiment shows that increasing the number of encoders from 2 to 5 have decreased the MSE by around 70% which is for sequence length = 4 as shown in figure 10a. Similar results are seen in the case of sequence length = 8. The MSE has decreased by around 66% as shown in figure 10b.

6.5 Integrating into an SDN environment

Software-Defined Network (SDN) architecture is being chosen for deploying our traffic prediction model and for controlling and managing the vehicular network communication. SDN is the new norm of networking where the control plane and data plane is separated, therefore ensuring that the control of the network is separated from data forwarding.

It is an emerging networking paradigm that has principles focusing on centralized view of network, better management and programmability of the network. Moreover, vehicular communication can be more efficient if it is implemented in a SDN environment [25]. Therefore it can be deduced that, SDN architecture ensures better connectivity and performance for vehicular communication over traditional network, and upon experimentation, similar results have been achieved. In this section we will put some light upon why SDN is being chosen over traditional network for integrating the vehicular communication, the experimental setup that is being conducted on Mininet emulator, and we will discuss the simulation results.

6.5.1 SDN over traditional network

One of the key concepts of SDN is the decoupling of data and control plane. In the case of traditional network, routing devices perform duties of both control plane and data plane. In SDN as shown in the figure 11, a controller-switch architecture is observed for the decoupling of the control and data plane. Controllers act as the control layer and switches just forward the data.

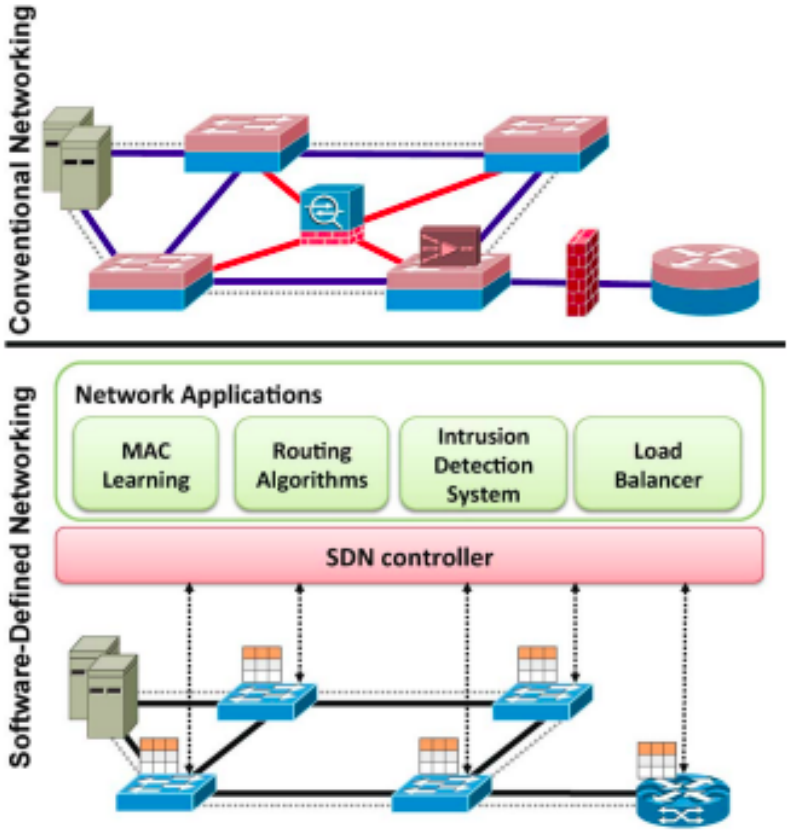


Figure 11: Traditional Networking vs SDN [1]

Traditional networks can't provide flexibility and efficiency in data transmission and network topology changes due to the coupling of control and data layer. Data layer performs forwarding of data packets and control layer builds the network topology and routing traffic. Therefore, each router has their own routing protocols that decides the route that the packets has to take for successful data transmission. Configuration of such a network is difficult especially if the network is very dynamic. Moreover the routing protocols in traditional network scenario are based on destination based routing. That means, the route for packets to travel are based on the destination rather than parameters like, QoS (Quality of Service) etc. Therefore it becomes very difficult to configure the network or change important components of how the network should work. In other words, it becomes difficult to change the current state of the network in the traditional way of network architecture. Moreover the networking devices in traditional network are proprietary. Therefore, the scope of development of networking devices and protocols with new features is limited. When it comes to big data, for example, data in the data centers, the traditional network cannot fully utilize the true potential of the network infrastructure due to its limitations, therefore many of the features that, we want from our network are not available in a traditional network. The old way of managing a network is getting cumbersome, as there is

a lot of manual debugging in it and addition of any new device in the existing network requires people to work on it. In other words as the traditional network is vertically integrated, configuration and managing change is very complex and the complexity increases with the scale of the network. [1]

When it comes to vehicular communication, a common issue is the lack of balance in the seemingly multi-path topology and poor network usage of the traditional network. [10] The network asks for an increased bandwidth as well, due to the dynamic nature of the networking nodes. Accuracy of transmission of data is also a critical factor in vehicular communication, therefore reliability is considered to be important because this communication is so dynamic in nature. Reliability is therefore, an important parameter to judge the effectiveness of the network used in such communication. As said earlier, human input is needed to configure the network for handling new changes to it, but the vehicular network is so dynamic that the system has to configure itself and a centralized view of the system is needed, therefore a centralized network can do well regarding to such communication as it is open and dynamic in nature.

SDN, that stands for Software-Defined Network, meets the requirements which are necessary for the network. It is a new paradigm that is both open-source and highly configurable, so this paradigm enables the network to be open and flexible. It overcomes the vertical integration of the traditional network through the separation of the control plane and the data plane. The control plane is responsible for network control which is achieved through an SDN enabled controller, and the data plane comprises of switches which are nothing but mere data forwarding devices. Therefore the brain of the network is being separated from the hands and legs of the network. The controller is said to be the brain of the network and the switches and other data forwarding devices are said to be the working body of the network. Above the controller we have a application layer which can be user defined networking apps that can improve the network capability and enable new features, the data layer resided below the controller. Between the controller and the switches are the south bound APIs and the north bound APIs connect the application layer with the control layer. Additionally, there might be east-west protocols that connect controllers with other controllers as well. So, this is a brief description of how the SDN architecture looks like.

There is a multitude of reasons behind choosing SDN over traditional network. The first reason is to have the network control and management separated from the data transmission, which ensures a centralized view of the network. It leads to better network management and ensures dynamic configuration of the network. Again, new features can be added to the network through designing networking applications and binding them to the controller through the north bound

protocols. Network reliability is greatly increased in the SDN paradigm as we have the overall view of the network through the controller, granular control over the network is also ensured in SDN. Additionally, user experience is also enhanced in the SDN paradigm due to better reliability of data transmission in the network.

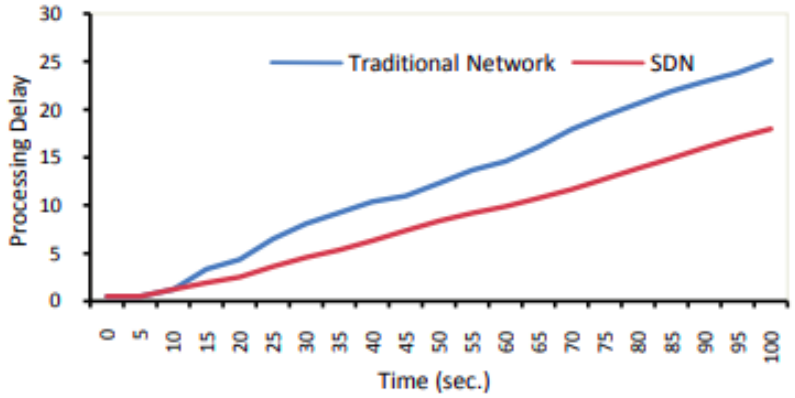


Figure 12: Comparison of performance between traditional and SDN network [2]

Finally when it comes down to network performance, there are a number of parameters that are considered. Latency, Packet Delivery Ratio, CPU utilization, packet switching etc. are the key indicators of measuring network performance. SDN enabled network has been found to outperform traditional network in terms of the above mentioned criteria [2]. In the figure 12, a comparison between processing delay between traditional network and SDN shows that, SDN clearly outperforms traditional network after the initialization of the network.

6.5.2 SDN experimental setup

The experiment is conducted in Mininet Emulator with Floodlight Controller. The Mininet emulator resided in a Ubuntu 14.04 OS. Mininet creates virtual networks that can emulate a lot of devices without the need of actual hardware. It takes advantage of process-based virtualization and network namespaces. Process based virtualization refers to the encapsulation of the OS to enable multiple virtual networks to run on the same system. Again if the system fails in the case, the workload will fail as well. Furthermore, network namespace refers to the logical copy of the underlying network stack from the host system. It is therefore a Linux Kernel feature that enables to isolate networks through the process of virtualization. Mininet mainly deals with hosts, switches and controllers. There is a powerful CLI that enables mininet to perform a number of features. For example, the Linux kernel commands can run on the virtual hosts

as though they were the real hosts. The switches in mininet are Openflow enabled. They are software based switches namely, Open vSwitch and Openflow enabled switches. The choice of controller has to be made based on the network requirements. There is a default controller provided by Mininet, but we might also use remote controllers to simulate, visualize and also run a number of networking apps over the network. Beside the CLI they provide, a powerful python library is also being provided, using this library we can configure custom topology and use other features as well.

The floodlight controller is being used as the remote controller of the network. It is an open-source, Java openflow controller. The core advantage of using this controller is the ability to use REST API. It is a centrally multi-threaded controller. As it is developed in Java, so the advantages of multi-threading is being taken. The north bound API is a RESTful API and the controller is licensed under Apache Licensing. There is no consistency and fault detection mechanism in this controller. The primary reason for choosing floodlight is, it has been designed for highly concurrent systems so to achieve the higher throughput needed for the network and it is one of the earlier prominent version of SDN controllers. Through multi-threaded design it can make use of CPU's parallel computation.

The custom topology that has been built to emulate real world scenario consists of 4 switches and 4 hosts. They are being viewed by the controller, the network brain is therefore, the Floodlight controller. The custom topology has been built by the code using the mininet library in python. python """Custom topology MyTopo is passed as the topology of 4 switches and 4 hosts, one host connected to each of the switches """

from mininet.topo import Topo

class MyTopo(Topo): "Custom Topology."

def build(self): "Create custom topo."

Add hosts and switches switches = []

for i in range(1,4): switches.append(self.addSwitch('s

hosts = []

for i in range(1,9): hosts.append(self.addHost('h'+str(i)))

Add links self.addLink(switches[0], switches[1]) self.addLink(switches[0], switches[2]) self.addLink(switches[1], switches[3]) self.addLink(switches[1], switches[2])

for i in range(0,4): for j in range(0,2): self.addLink(switches[i], hosts[i])

topos = 'mytopo': (lambda: MyTopo()) After booting the controller, the network is being connected with the floodlight controller. So the controller has the visualization of the underlying network. The hosts represent the RSU of the vehicular network. 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. That is we have transmitted the model data output to the hosts and measured the time needed for the other devices to receive the data. Various sample of data is being gathered as to how much time it takes for the full data to go from the devices, in the SDN network.

6.5.3 SDN simulation and results

Finally we have found the results for the experimentation in terms of 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 being presented in the table below. We also have been able to showcase the topology in the floodlight controller as shown in the figure.

Table 11: 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

Though this experiment, we are being able to show that, how fast and reliable data transmission is in the SDN topology. Therefore, we have been able to experiment to produce such a network which simulates the vehicular data in an SDN environment.

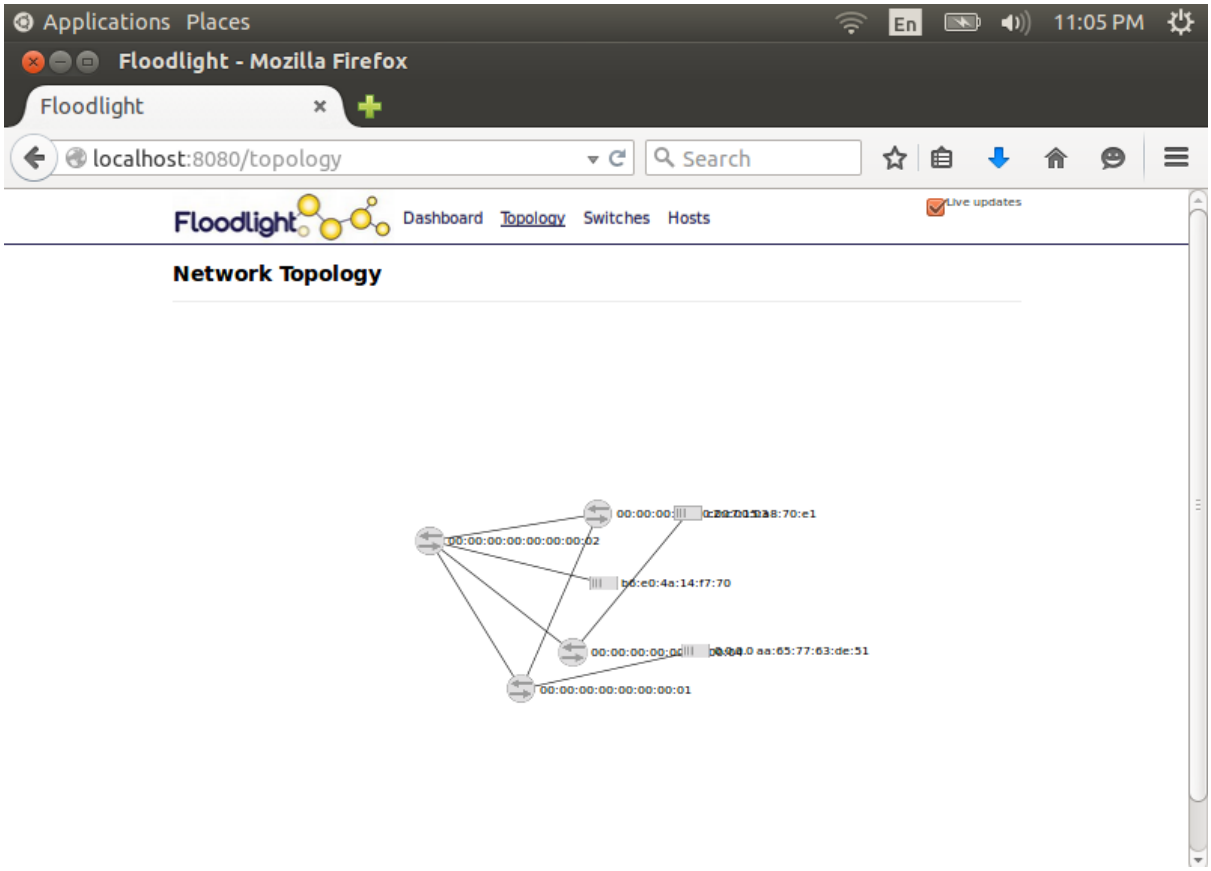


Figure 13: Network Topology Visualization in Floodlight Controller web interface

Additionally it was being made possible to simulate the network and the floodlight controller was able to generate a visualization of the network topology. It is being made possible through RESTful API which is present in the floodlight controller. Through this REST API, the controller was being able to show us the network in the web.

7 Future Works

7.1 Dataset modification

Each of the stations in the dataset have a specific area coverage. 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 cases make the model fail to recognize correlations among the stations with its positions and neighboring stations.

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

7.2 Resource allocation

Larger sequence lengths and batch sizes would generally yield a better result. Even though sequence length of 16 performed worse in our experiment, but that can be alleviated with large number of encoders. To sum up, bigger sequences, batches and more encoders would yield a much better result. This could be ensured if there was a larger RAM available.
Again, a larger dataset would mean that the model is learning from more samples which generally also produces a much better results and avoid bias. A larger storage could be used and better GPU with computational power would help in this regard.

8 Conclusion

It is seen from the experiments that our transformer model has performed extraordinarily on the dataset. This proves that the transformer based models will be very effective in predicting traffic flow. Since the transformer architecture supports large amount of data, so the dependencies across distant areas and lengthy time periods can also be accomodated.
The ad-hoc nature of VANET blends perfectly with the SDN structure. Fast moving vehicles would require a faster transmission rate and also proper control mechanism which can be ensured by the centralized decision making process of SDN.
It can be concluded that the combination of VANET and SDN along with accurate prediction using a transformer model would perform better and be more efficient in predicting traffic flow.

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[2] M. I. Lali, R. Mustafa, F. Ahsan, M. Nawaz, and W. Aslam, "Performance evaluation of software defined networking vs. traditional networks," *The Nucleus*, vol. 54, no. 1, pp. 16–22, 2017.

[3] J. Mackenzie, J. F. Roddick, and R. Zito, "An evaluation of htm and lstm for short-term arterial traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1847–1857, 2018.

[4] 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," *Journal of information security and applications*, vol. 31, pp. 54–60, 2016.

[5] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined vanet: Architecture and services," in *2014 13th annual Mediterranean ad hoc networking workshop (MED-HOC-NET)*, pp. 103–110, IEEE, 2014.

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

[7] C. J. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, and J. C. Zúñiga, "An architecture for software defined wireless networking," *IEEE wireless communications*, vol. 21, no. 3, pp. 52–61, 2014.

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

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

[10] M. Gerla, "Vehicular cloud computing," in *2012 The 11th annual mediterranean ad hoc networking workshop (Med-Hoc-Net)*, pp. 152–155, IEEE, 2012.

[11] I. A. Abbasi and A. Shahid Khan, "A review of vehicle to vehicle communication protocols for vanets in the urban environment," *future internet*, vol. 10, no. 2, p. 14, 2018.

[12] 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 communications surveys & tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[13] 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-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1778–1805, 2021.

[14] C. Harsch, A. Festag, and P. Papadimitratos, "Secure position-based routing for vanets," in *2007 IEEE 66th Vehicular Technology Conference*, pp. 26–30, IEEE, 2007.

[15] S. Kudva, S. Badsha, S. Sengupta, I. Khalil, and A. Zomaya, "Towards secure and practical consensus for blockchain based vanet," *Information Sciences*, vol. 545, pp. 170–187, 2021.

[16] F. Alam, I. Katib, and A. S. Alzahrani, "New networking era: Software defined networking," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 11, 2013.

[17] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, vol. 2, no. 2-6, p. 11, 2012.

[18] K. Kirkpatrick, "Software-defined networking," *Communications of the ACM*, vol. 56, no. 9, pp. 16–19, 2013.

[19] 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.

[20] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *2014 IEEE network operations and management symposium (NOMS)*, pp. 1–9, IEEE, 2014.

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

[22] S. Sezer, S. Scott-Hayward, P. K. 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 Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[23] 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 Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2018.

[24] M. J. Islam, M. Mahin, S. Roy, B. C. Debnath, and A. Khatun, "Distblacknet: A distributed secure black sdn-iot architecture with nfv implementation for smart cities," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1–6, IEEE, 2019.

[25] G. A. Qadir, S. Askar, *et al.*, "Software defined network based vanet," *International Journal of Science and Business*, vol. 5, no. 3, pp. 83–91, 2021.

[26] M. O. Kalinin, V. Krundyshev, and P. Semianov, "Architectures for building secure vehicular networks based on sdn technology," *Automatic Control and Computer Sciences*, vol. 51, no. 8, pp. 907–914, 2017.

[27] M. Zhu, J. Cao, D. Pang, Z. He, and M. Xu, "Sdn-based routing for efficient message propagation in vanet," in *International Conference on Wireless Algorithms, Systems, and Applications*, pp. 788–797, Springer, 2015.

[28] J. Bhatia, R. Dave, H. Bhayani, S. Tanwar, and A. Nayyar, "Sdn-based real-time urban traffic analysis in vanet environment," *Computer Communications*, vol. 149, pp. 162–175, 2020.

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

[30] 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 Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.

[31] X. Du, H. Zhang, H. Van Nguyen, and Z. Han, "Stacked lstm deep learning model for traffic prediction in vehicle-to-vehicle communication," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, IEEE, 2017.

[32] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2014.

[33] J. Hu, P. Gao, Y. Yao, and X. Xie, "Traffic flow forecasting with particle swarm optimization and support vector regression," in *17th international ieee conference on intelligent transportation systems (itsc)*, pp. 2267–2268, IEEE, 2014.

[34] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 922–929, 2019.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[36] C. Chen, *Freeway performance measurement system (PeMS)*. University of California, Berkeley, 2002.