Traffic Flow Prediction Using Deep Learning Models

Satya Pranavi Manthena
Computer Science Department
San Jose State University
San Jose, USA
satyapranavi.manthena@sjsu.edu

Saurabh Kale
Computer Science Department
San Jose State University
San Jose, USA
saurabh.kale@sjsu.edu

Maithili Vinayak Kulkarni
Computer Science Department
San Jose State University
San Jose, USA
maithilivinayak.kulkarni@sjsu.edu

***Abstract*—For the effective implementation of Intelligent Transportation Systems (ITS), accurate and timely traffic flow information is critical. Traffic data has exploded in recent years, ushering in the era of big data. The main issue of a traffic flow prediction system is determining how to build an adaptive model based on past data. Existing traffic flow forecast approaches rely on shallow learning models, which are unsatisfactory for many real-world applications. This situation prompts us to reconsider traffic flow prediction using deep learning models and large amounts of traffic data. In this paper, we present the Long Short-Term Memory (LSTM) model, Gated Recurrent Unit (GRU) model, Stacked Auto Encoder (SAE), and an Auto-Regressive Integrated Moving Average (ARIMA) model, all of which are deep learning methods. These techniques are applied to real-world traffic big data collected by performance measurement systems. Compared to other deep learning and shallow machine learning prediction networks, experimental results demonstrate that the gated recurrent unit model is more applicable and performs better.**

***Index Terms*—auto-regressive moving average, big data, deep learning, gated recurrent unit, long short term memory, prediction, ,shallow learning, and stacked auto encoder.**

I. Introduction

With the advancement of urbanization and the development of automobiles, transportation challenges are becoming increasingly complex: the traffic flow has become crowded, and the traffic environment is degrading. The World Health Organization reported in 2020 that the number of road traffic-related deaths continues to rise, with 1.68 million deaths documented in 2018, encouraging research into traffic forecasting. Accurate and efficient traffic flow information is strongly needed for individual travelers, business sectors, and government agencies. It can assist road users in making better travel decisions, reduce traffic congestion, lower carbon emissions, and increase traffic operations efficiency. The objective of traffic flow prediction is to provide such traffic flow information.

Traffic flow was previously predicted based on past trends and patterns. Currently, with the widespread use of classic traffic sensors and new sensor technologies, the data collected is enormous, ushering in the big data era and making it difficult to represent the data in patterns. Furthermore, the existing methods to predict big data-driven traffic flow use shallow learning; therefore, many practical applications are unsatisfactory, motivating us to reconsider the big data-based traffic flow prediction challenge. Furthermore, as the number of factors affecting the data grows, several layers are required to understand the complex and abstract relationships within the data and how each parameter involved is correlated. Hence, there is a need to develop a multi-layered framework that can help model the traffic patterns without historical data and be more efficient in computation results and accuracy. Deep learning, a form of machine learning technique, has recently sparked educational and industry interest. It has been used successfully in applications such as classification, natural language processing, dimensionality reduction, object identification, motion modeling, etc. Deep learning techniques use multiple-layer designs or deep architectures to uncover inherent qualities in information from the smallest to the highest level. As a result, they may discover massive amounts of structure in the data. Due to the complexity of traffic flow processes, deep learning algorithms can describe traffic characteristics without previous information, resulting in accurate traffic flow forecast performance.

This study uses LSTM NN, GRU NN, and SAE approaches to estimate traffic flow. We apply the Adam optimizer with adaptive learning rates to optimize this LSTM NN. On the Caltrans Performance Management System (PeMS) dataset, we examined the performance of ARIMA, SAE, LSTM NN, and GRU NN models. We discovered that LSTM and GRU NNs outperform ARIMA and SAEs. In addition, GRU NNs outperform LSTM NNs in performance and often converge faster.

II. Overview

A. Chapters Overview

The first chapter will address the paper's introduction, why the subject matter is essential, the purpose of this effort, and the problem the paper seeks to tackle. The second chapter gives an overview of the paper and the significant contributions provided. Chapter 3 discusses similar work in the field and other current research publications that have sought to solve

the same issue. Chapter 4 discusses the dataset and how it was collected. The deep learning models used to estimate traffic flow: LSTM, GRU, SAE, and ARIMA, are discussed in Chapter 5. Chapters 6 and 7 discuss the experimental assessment and the obtained outcomes. Finally, chapter 8 concludes the study with a conclusion and review of the findings and the planned future work that the authors hope to do in the future.

B. Contributions

- Crawling traffic flow data from the CalTrans website and preparing and cleaning the dataset.
- Producing the best outcomes by utilizing four machine learning methods: LSTM, GRU, SAE, ARIMA.
- Comparing the results from these experiments in a clear and graphical manner.

III. Related Work

Traffic flow prediction has been studied since the 1970s due to its importance in urban traffic planning. As a result, there have been several techniques for traffic flow prediction developed over several decades. Traffic flow prediction models are classified into two types: parameter models and non-parameter models.

A. Parameter Models

Models with a predefined structure based on some assumptions and parameters that may be estimated from experimental observations are called parametric models.

According to Smith and Demetsky [2], several statistical parametric procedures were evaluated, and smoothing techniques were found to be effective. In [3], Ahmed and cook first described the usage of ARMA to forecast short-term highway traffic data, which has then become the most often used parameter model. Influenced by Ahmed and Cook [3], many time series prediction models based on ARMA were presented, which include Kohonen-ARMA (KARIMA) [4], Subset ARMA [5], Seasonal ARMA [6]. These models were built on the assumption that the variance and mean of the time series are stationary.

Parameter models have a significant advantage. For starters, they are usually straightforward models [2]. Second, they are typically simpler to solve and take less time to complete than non-parameter models. All the authors mentioned above [4,5,6] confirm that parameter models may not sufficiently capture the nonlinear and stochastic nature of traffic flow, resulting in larger prediction errors compared to non-parameter models.

B. Non-parameter Models

Models with no defined structure and no fixed parameters are called non-parameter models. RNN, SVM, K-Nearest Neighbors (KNN), and Artificial Neural Network (ANN) are examples of non-parameter models.

Non-parameter models, such as Neural Network(NN) models, are more expressive and fit nearly any function reasonably. Non-parameter model optimization issues, on the other hand, frequently fall into non-convex optimization problems, which may be prone to local minimums. Non-parameter model optimization may be hard, and overfitting is a difficult problem to overcome. With Krizhevsky et al. [7] claiming enormous success in deep learning, many researchers are attempting to adapt deep NN algorithms to traffic prediction. In [8], Lv et al. were the first to use SAEs to estimate traffic flow and found it performed better than SVM, a feed-forward neural network (FFNN). Ma et al. [9] employed LSTM NN for traffic speed prediction and found it successful compared to most non-parametric models and the ARMA model. Tian et al. [10] introduced the LSTM NN for traffic flow prediction. They demonstrated that it surpasses most non-parameter models. Because of its improved ability to recall long-term dependencies, LSTM NNs have a distinct edge in traffic flow prediction.

In 1997, LSTM was introduced for language models and was well-known for its capacity to memorize long-term dependencies [11]. However, given their intricate structure, LSTM NNs often require a long time to solve. Due to its simplified structure and solution, GRU was recommended as a refinement for LSTM to conduct machine translation in 2014 to speed up training [12]. Whereas GRU NNs have been employed very few times for traffic flow prediction and have demonstrated to have performed better than ARIMA and LSTM.

IV. Dataset

The data utilized in this article came from the PeMS dataset, which comprises over 40,000 sensors installed across California.

A. Data Source

The PeMS collects traffic data in real-time from over 40,000 individual detectors. These sensors cover the interstate system in California's central urban regions. PeMS is also an Archived Data User Service (ADUS), with over ten years of data available for historical research. It incorporates data from Caltrans and other local agency systems such as traffic detectors, incidents, lane closures, toll tags, census, traffic counts, vehicle classification, weight-in-motion, and roadway inventory. The data used for the experiment is collected from the fourth district of PeMS dataset, which lies in Alameda, Oakland, part of the U.S's bay area. On the CalTrans website, the user can download data for a specific day on a specific lane for a specific sensor and specify the parameter that can

be flow, speed, or occupancy over a specific time as shown in Fig. 1. The original data on the website is collected every 5 mins and has parameters of :

- Traffic flow: the number of vehicles that pass over the detector in a 30-second period
- Traffic speed: the speed of the vehicle passing over the detector
- Traffic occupancy, which is the fraction of time that a vehicle has been over the detector

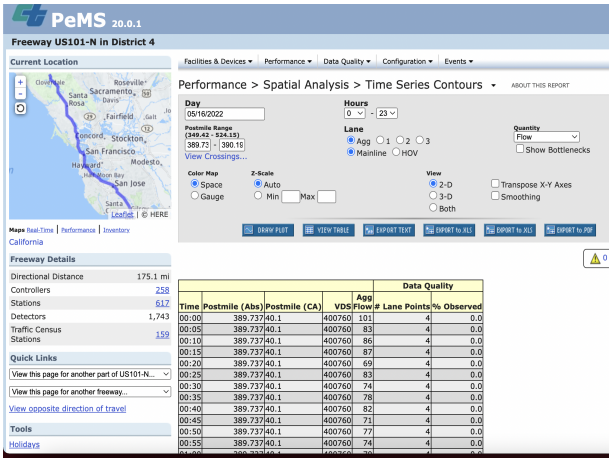

Fig. 1. Data Source: the CalTrans website

B. Data Retrieval

The CalTrans website provides data for each day in a 5-minute interval. To retrieve data for one whole year to perform our experimentation would become a tedious manual task. So, we have written a web crawler, a computer program that searches web pages for specific information in an automated and methodical manner. Beautiful Soup, a Python module that retrieves data from HTML and XML files, is used by the web crawler we created to obtain data. It interacts with the desired parser to provide idiomatic techniques for browsing, locating, and modifying the parse tree.

The website of CalTrans has a login page, and data cannot be accessed unless logged into the website. To get past the login screen and handle robots, we have used the Mechanize library of python. The website URL is provided as input with date and month as variables since we need to scrape the data for the whole year of 2021. The Traffic Flow, Speed, and Occupancy data were aggregated over all the lanes instead of a specific lane. The data is collected between sensors 389.73 - 390.19 over the Guadalupe Freeway.

The time series data is collected for 12 months, from January 2021 to December 2021. The data is divided into two parts: the first contained data from January 2021 to September 2021, which is used as our training data, and the second contained data from October 2021 to December 2021, which is used to test the model prediction accuracy.

C. Data Pre-Processing

Out of the seven columns of data published on the website, only four relevant columns were retained, which are: time, aggregated flow, lane points, and observed percentage. The original data does not contain a date column in the table. So, to make the data more meaningful, the date and time of each row were concatenated to create a timestamp. As part of the data pre-processing, we have also handled null values and ignored the summation rows, which contain the record for the total number of entries for a day.

D. Exploratory Data Analysis

On the pre-processed data, we have conducted an exploratory data analysis using a heatmap to graphically represent the data, as shown in Fig. 2. The heatmap is a data visualization technique that shows the magnitude of a phenomenon as color in two dimensions. The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space. The heatmap is a data visualization approach that displays the magnitude of a phenomenon in two dimensions as color. The color change may be via hue or intensity, providing the reader with apparent visual indications regarding how the occurrence is clustered or fluctuates across space. We can see that the data is not skewed and has days where the traffic flow is maximum, minimum, and also mediocre. This kind of training data would help the model learn efficiently and predict traffic flow for all kinds of scenarios.

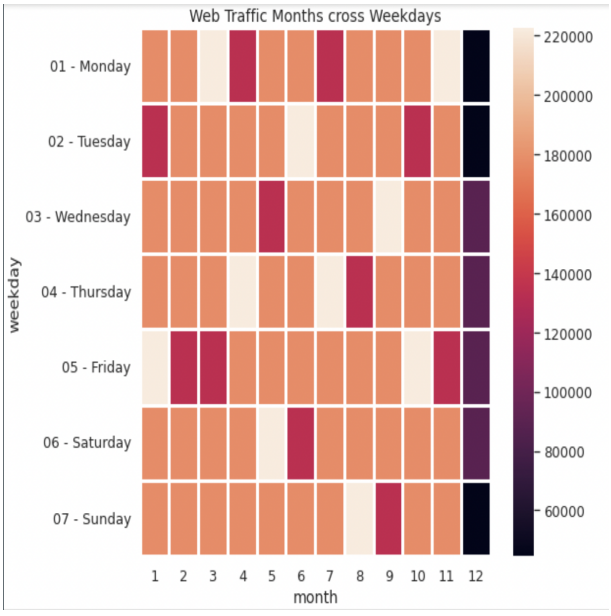

Fig. 2. Heatmap of the training data

V. Deep Learning Models

In machine learning, deep learning models are a part of a new learning curve along with artificial intelligence (AI). Big data are its applications are currently being widely popular due to the boost of machine learning models in the fields of image analysis and recognition of speech and large textual files and represent the possibility of further integration. However, the mathematics involved with machine learning models and the computational complexity make it difficult to understand for scientists and researchers who want to implement and utilize them in their projects. Various deep learning models such as Autoencoders (AEs), Long Short-Term Memory (LSTM), Feedforward Neural Networks, and Convoluted Neural Networks (CNN) are being used in big data applications. These models provide the basis of deep learning and set it apart from conventional machine learning models due to their complex architecture and their usage in various high-level research.

Also, these models and their structures need to be understood by scientists as there are modeled like a Lego set, one layer on top of another, to present a brain network-like structure. Therefore, it's important to understand these models to get a clear idea about their functioning and modeling.

A. Neural Network Architectures

Artificial Neural Networks (ANNs) are mathematical models that are inspired by how the brain operates. The following models, on the other hand, are not intended to be biologically realistic. Instead, these models are designed to examine data.

1) Recurrent Neural Networks: Dependencies that are smaller in length and complexity can be captured using RNN. RNNs can be further distinguished into full networks or partial network structures. Williams and Zipser created the first RNN in the late 1980s when there was a surge in the establishment and creation of neural network topologies and various important research. Several RNN discoveries aim to uncover and model relationships as well as retrieve important features and data from time series. Numerous RNN extensions are created and built in recent years to address a range of challenges based on this basis. A discrete wavelet modification is used to copy the functioning of the regular activation functions in an RNN. A distinction is made between the conventional activation functions such as sigmoid and tangens hyperbolicus, and it is stressed that enhanced control of the information flow could lead to improved modeling accuracy. However, RNNs have the drawback of disappearing gradients. As a result, RNN is less effective at capturing non-stationary relationships that arise over time. Some researchers describe a method for addressing the problem of long-term dependencies that tries to construct a memory within conventional RNNs. Dilated recurrent skip connections are used to extend RNNs in order to capture complicated relationships between time-series data which are continuous in nature and provide a set of storage capabilities. The skip connections allow you to process data from previous time steps without having to send it through the complete network. As demonstrated by Fig 3.

As a result, the average length of skip links has a significant impact on whether short-term or long-term dependencies are prioritized. RNNs, on the other hand, are best used to predict highly dynamic, time-variant systems with stationary or nonstationary short-term dependencies.

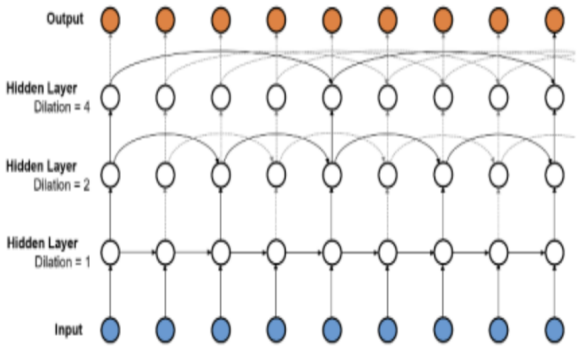

Fig. 3. Architecture of cells in Recurrent Neural Network

2) Cell Architecture: The vanishing gradient effect prevents RNN from adequately capturing various features and their associations with distinct features, as well as long-term dependencies. As a result, gating techniques have been created to replace traditional activation functions. The input, forget, and output gates in LSTM cells allow changes to a vector created and stored in the memory and are recursively iterated to compute the weights and newly created associations. The network can memorize several time dependencies with distinct properties thanks to the well-defined transition of data in the cells. LSTM was developed to model the long-term dependencies and to resolve the vanishing gradient problem.

The Gated Recurrent Unit (GRU) is created as an improvement of the LSTM architecture of cell states and models the network with a smaller number of cell states and a smaller number of computations. GRU was used to represent time series having the aim of developing a system that combines the power to forecast long-term dependencies with better short-term aggregation. The goal is to allow for adaptive modeling of interdependence across temporal ranges. GRU, in comparison to LSTM, has a simpler cell layout that uses a gating system but just contains a smaller number of gates. The key distinction from LSTM is that the cell state can be entirely altered and updated with short-term information via the reset gate at each repetition. However, LSTM has a procedure that restricts the amount of change gradient that may be achieved at each iteration. As a result, unlike GRU, LSTM does not allow prior knowledge to be totally erased. Cells with gating mechanisms generate much higher prediction outcomes than standard RNN techniques, according to empirical tests on cell

layouts. Furthermore, researchers determined the advantage of LSTM over GRU in the context of a large-scale study on various network design variants. Knowing the fact that GRU cells have a smaller number of parameters, no meaningful time savings could be demonstrated. Furthermore, it was discovered that the LSTM gating mechanism aids in the filtering of unnecessary input data and improves the correctness of time-variant behavior modeling.

3) Long Short-Term Memory: The LSTM network (Long Short-Term Memory) is a sort of RNN (Recurrent Neural Network) that is commonly used to learn sequential data prediction issues. LSTM, like any other neural network, includes layers that enable it to model and classify patterns for improved performance. The fundamental and conventional operation of an LSTM can be seen as holding the required data and discarding the data that isn't needed or beneficial for subsequent prediction.

There are many different forms of LSTM networks, however, they can be divided into three categories.

- Forward pass LSTM
- Backward pass LSTM
- Bidirectional LSTM (Bi-LSTM)

The LSTM RNN architecture makes use of the calculation capability provided by normal RNN architecture as a hidden layer gate and takes the output generated to further process the data. It passes on to the gates and stores the dependencies in memory. The computations performed within the cells are highly complex and weights are updated at each iteration with the help of previous weights and the interactions with the current values. The dynamic equations for these calculations are following:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$\tilde{c}_t = g(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \odot g(c_t)$$

A simple LSTM network consists of the following components.

- Forget gate
- Input gate
- Cell state
- Output gate

a) Forget Gate: As previously stated, the primary aspect of the LSTM is its ability to remember and determine data flowing into the machine learning model while also discarding data that is unnecessary to know and using the same for further process. The part of remembering the weights and the important information regarding the data and utilizing them later in LSTM is done with the help of cell states and gates.

This helps in recognizing whether the information passed to the machine learning models can pass through each gate and each layer of the LSTM. The cells require two types of inputs to compute the predictions which are the currently passed data from the previous cell of the LSTM and the data from the passed backwards for better weight computations. The data is passed into the forget gate which then determines whether the data is useful and can be used ahead with the help of sigmoid functions are to be discarded so as to not affect the network negatively.

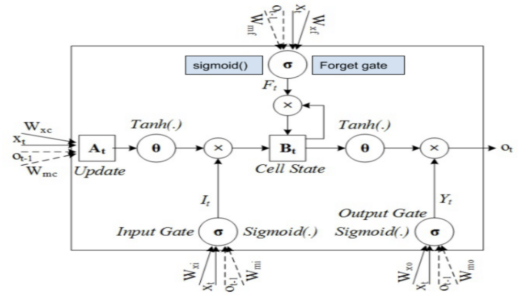

Fig. 4. LSTM Forget gate

b) Input Gate: It helps in finding the significance of the current passed data from the forget gate which removes the unessential part and could be used for further transformation. This gate is very important to compute the predictions and the results as the data which is actually been carried forward and its weightage are determined by the input gate with the help of sigmoid and tanh functions which also helps in reducing the bias in the data. Forget gate with input gate provides the basis for good predictions and the network is modeled very accurately.

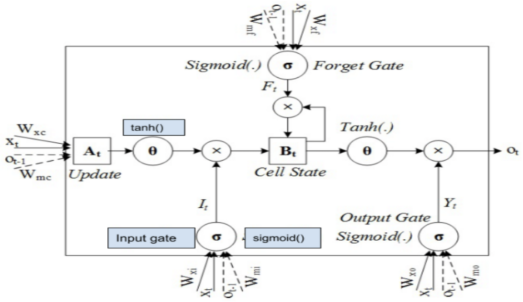

Fig. 5. LSTM Input gate

c) Cell state: The data passed forward from the forget gate and input gate is processed at the cell state. The values and weight are already processed and multiplied using the tanh and sigmoid functions in the previous gates and are now combined together to generate valuable numeric data which can be lost due to the exponential multiplication performed on the near-zero values. The summation of values from the input gate and forget gate is performed to maintain the relevance of the data and increase its integrity.

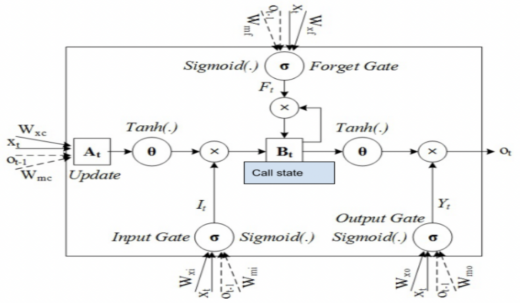

Fig. 6. LSTM Cell State

d) Output gate: The significance of the data in the layer is determined by the usage of the input gate which can help in computing the weightage of the data. The sigmoid function is used in finding the output from the last gate. The tanh function gets the new value of the weights and is multiplied with the output generated by the sigmoid function and is further passed on to the next cell state. The output gate is also the last gate of the whole cell and the computations and is helpful in transferring the data ahead in the network.

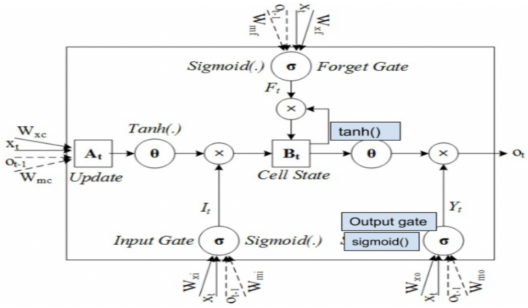

Fig. 7. LSTM Output gate

4) Gated Recurrent Network: GRU networks are a type of Recurrent Neural network (RNNs) having a minimum of one loop or cycle of iterations within the network and the layered architecture. They were developed in 1997 and have been updated during the course of the machine learning era. The problems raised by RNNs such as vanishing and exploding gradients are resolved due to the convoluted nature and complex structure of cells and gates within the GRU. There are multiple neurons in the Gated Recurrent Network which are recursively used to compute the updated weights using backpropagation and auto-differentiation. The neurons in the input layer map the input data and the output layer consists of the same number of neurons as the features to be classified or predicted. The main function of the GRU is the hidden layers between the input and the output layers. The main challenge of updating the cell states and managing is carried by the reset and the update gate of the GRU. Fig 8 shows the structure of a memory cell as a circuit diagram. Both the gates in the GRU have two similar input values coming into the architecture. One is the output generated by the memory cells from an older time period, say t-1, and the latter is the

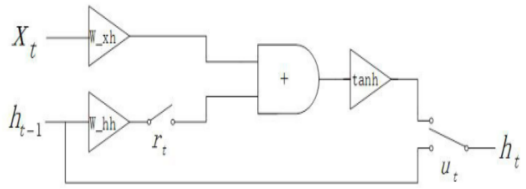

Fig. 8. Architecture of GRU cells

input values presented to the GRU at the current time period, say t. Each of the gates is used to perform different functions such as:

- The forget or the reset cell is utilized to manage the impact of ht-1 on the current time period data xt. If ht-1 is unimportant to xt, rt can be used to forget the value, and ht-1 will not affect the value xt.
- The update gate determines whether present time period data xt should be removed or updated. The data can be ignored as the present information xt when ut is turned on the under branch, by deleting the link from ht-1 to ht.

This successfully solves the gradient vanishing problem by causing the gradient to reverse propagate. The formulas presented below showcase the actual working of the gates inside the cells of GRU at every time period. The reset gate's equation is:

$$u_t = s(W_u \cdot [h_{t-1}, x_t])$$

The equation of the update gate:

$$r_t = s(W_r \cdot [h_{t-1}, x_t])$$

The equations of the output:

$$\widetilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - u_t) * h_{t-1} + u_t * \widetilde{h}_t$$

5) ARIMA: The acronym for Auto-Regressive Integrated Moving Average is ARIMA. It's a time-series data model that can establish a wide range of similar and important properties.

These models are a type of probabilistic and mathematical models used in the analysis and predictions of time-series data perfectly with the help of its Moving Average and Auto Regression parts. By favoring only, a specific set of common features in the time series data, ARIMA is very useful to generate important and essential data outputs related to time series.

Auto-Regressive Integrated Moving Average (ARIMA) is an acronym for Moving Average. Due to its aggregated nature, ARIMA can give powerful results and evaluations. This acronym is descriptive, expressing the main characteristics of the model. In a general, they are:

- AR means Auto Regression which is a model which

makes use of the association and relationship between the lagged data presented using the time series and the currently presented observation.

- Integration is the main part of the ARIMA model." Subtracting previously used data for the old-time steps with the currently calculated data for the present time step is performed by the Integrated part of the ARIMA.
- Moving Average is an acronym for moving average. This model uses the relationship between errors generated due to the previous time steps and the average of lagged dataset computed at every time step.

The ARIMA model is represented with the values p, d, and q which stand for the three parts of the ARIMA. Each part is associated with a value based on its numeric calculations. ARIMA model is widely used in the context of time series data

$$X_t = \alpha + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q}$$

$$X_t = \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \beta_0 X_0 + \epsilon_t$$
$$X_{t-1} = \beta_1 X_{t-2} + \beta_2 X_{t-3} + \cdots + \beta_0 X_0 + \epsilon_{t-1}$$

$$X_t = \alpha + \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \beta_p X_{t-p} + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q}$$

due to its nature of moving average and integration along with autoregression parts. The banking and financial sectors can hugely benefit from the usage of ARIMA as well as medical domains can make use of the easy, scalable, and powerful dependency of time series presented by ARIMA.

6) Auto Encoders:

a) Simple Auto Encoders: An autoencoder is a neural network that employs backpropagation to generate output values that are almost identical to the input values. The autoencoder takes the input in the format of a vector with multiple dimensions and then passes it forward to a neural network architecture which will remove the curse of dimensionality and reduce it further to only a few dimensions and features. This can be done with the help of Principal Component Analysis (PCA) but is already integrated with the SAE architecture. The remaining two dimensions are encoder and decoder. The encoder is a network of layers that take up the input data and reduces the dimension further with the help of a process called bottlenecking. This process reduces the dimensions as well as due to the connected representation of encoder data is reduced exponentially which makes it easy to use for fitting. The decoder picks the data from the bottleneck and again generates more features using feature extractions and used for the output of the model.

b) Stacked Auto Encoders: A stacked autoencoder is a collection and combination of numerous sparse auto encoders which creates a mesh-like structure in which the output of each layer is connected to the input of the other layer.

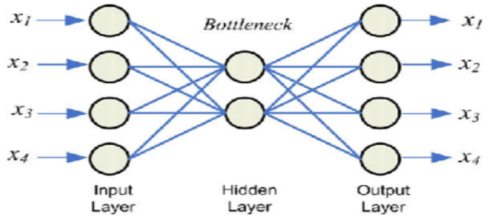

Fig. 9. Layered Architecture of Simple Auto Encoder

Unsupervised learning is combined with a supervised learning method to train the hidden layers of the stacked autoencoders. SAE is composed of three layers which are mainly the input layer, the encoding layer, and the output layer. The first layer, the input layer, takes in the data from the input space and fits the model representing the autoencoder. The weights and biases generated at each layer of the SAE are then passed on to the next layer for further modifications. As all the layers in the SAE are done computing the weights and the training phase is over, backpropagation is utilized to determine the weights to reduce the loss function, as well as accurately, determine the values for predictions. Due to the improvements in SAEs accuracy of the autoencoders has increased exponentially. Medical researchers could use SAEs to determine the long-term dependencies between the illness and patients' medical history which can be time-series data. This can help in classifying and determining diseases accurately and efficiently well beforehand.

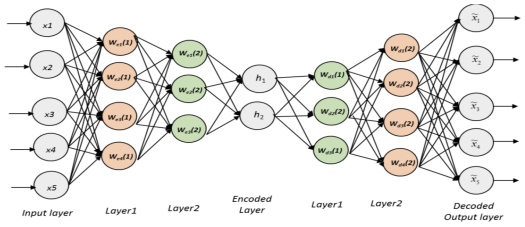

Fig. 10. Layered Architecture of Stacked Auto Encoder

VI. Proposed Method

The project's initial approach was to make use of the PeMS Bay 4 dataset which consists of traffic flow from October 2015 to January 2016. The data cleaning and preprocessing were done using the pandas library as we split the timestamp into the week, day, and year columns. These features will be later used for training along with the labeled traffic flow count. The hourly and weekly traffic flow counts were predicted using the matplotlib library's pyplot which gave a visual explanation of the traffic flow data for the 4 months. The heat graph in Fig 11 showcased the scarcity of data and pointed out the traffic flow for each day in the week for the following months. Train and test splits of 80% and 20% respectively were used to split the data for the training and testing process. Initially, the Adaboost

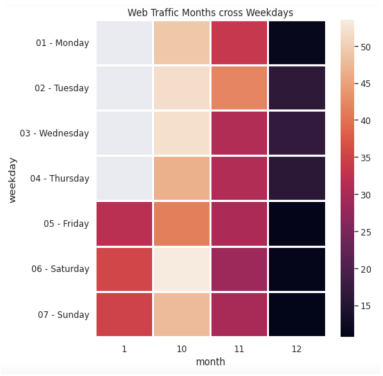

Fig. 11. Heat Map of month, weekday, and traffic flow

Regressor model with 5000 estimators and 42 random states along with a 0.01 learning rate was used to fit the model on training data. It could not predict the traffic flow successfully and switched to Long Short-Term Memory (LSTM) model. Min Max Scaler was used to scale the values between [0,1] so as to get a common range between the values for the ease of model fitting. LSTM model with 1 input layer with 50 neurons and 1 output layer was used with Adam optimizer. After 50 epochs, the model converged with an RMSE score of 27.098 which is very poor.

Due to the initial approach, the realization of the lack of data leads to the use of web-crawling on the PeMS Caltrain's website to crawl the traffic flow, speed, and occupancy values using the BeautifulSoup4 library of Python for the US 101-N freeway of District 4 with co-ordinates as 389.73 and 390.19 sensors. The Guadalupe Freeway sensors data was scarped from January 2021 to December 2021 which is around 100,000 entries. The aggregate flow, aggregate speed, and aggregate occupancy was extracted as features and split into 5-minute intervals.

1) Data Split: The crawler dataset was distributed into 2 groups for training and testing. From January 2021 till September 2021 was used for training and October 2021 till December 2021 was provided for testing purposes. The 'data.csv' was loaded into the Google Colab notebook and was differentiated and processed using data frames for the training and testing.

2) Data Preprocessing: After the splitting of the dataset, lagging was used to create useful and synchronous data. A lagging provides varied data to be collected and processed to derive some conclusion with better understanding. Forecast the future in time series is done with the help of lagging which collects the independent data variables and features and regression models are used to predict the future. The data from time period t-1 is used to find out the data in time period t. Therefore, lags of 12 timestamps were used to generate multidimensional arrays in each row with 12 elements of lags. As well, Min-Max Scaler is used to scale the data of variable ranges into to fixed range of [0,1] which provides ease in model training.

3) Data Modeling: Machine learning models such as LSTM, GRU, SAEs, and ARIMA were used to predict the traffic flow, speed, and occupancy of the US 101-N freeway for the testing data as well as for the next 24 hours. For LSTM, 4-layer architecture was used which provided the best results which include 1 input layer with 12 neurons to match the lag of 12 of the time-series datasets. Two hidden layers with 64 neurons each were used to provide in-depth backpropagation dependency for better predictions and 1 dropout layer with 20 percent dropout turns off 20 percent of the neurons from the last output layer which reduces the risk of overfitting in LSTM. At last, a single dense output layer with single neurons provides the numeric value for the traffic flow. Sigmoid activation functions with 256 batch sizes and 50 epochs were used to fit the model. Similarly, GRU followed the same architecture as proposed by LSTM with 4-layers and one dropout layer. Stacked Autoencoders (SAE) used a 5-layer architecture which consists of 12 neurons in the input layer, 400 neurons in each of the 3 hidden layers, and a single dense output layer. ARIMA model with p, d, q values of 5, 1, and 0 was best suited for the time-series data with high AIC score and was run for model fitting.

4) Model Training: Training the model with various hyper-parameters helped to determine the best values with the help of validation and training loss metrics used during the model fitting. LSTM and GRU were retrained on the aggregate speed and aggregate occupancy features which gave similar results as the aggregate traffic flow. SAE with epochs of 50, 100, and 200 was used to determine the best parameters and ARIMA with various combinations of p, d, and q values to get the highest accuracy.

5) Model Predictions: All the models predicted the traffic flow for the month of October 2021 to December 2021 and for 1st October 2021 to get the 24-hour 5-minute interval traffic flow predictions.

VII. Experimental Evaluation

In this project, four different machine learning algorithms were used along with parameterized and non-parameterized. The predicted scores were generated and then compared with the help of four evaluation metrics which are Root Mean Square Error (RMSE), Mean Absolute Error (MAE), R-squared (R2), and Mean Absolute Percentage Error (MAPE). All of these metrics determine the validity of the results obtained by comparing the predicted values against the labeled test values and providing numeric values.

A. LSTM

Long Short-Term Memory is used with the lagged dataset to predict the aggregate traffic flow, aggregate speed, and the

occupancy of the road during the test period. Various epochs such as 200, 100, and 50 were utilized of which 50 gave decent results as running the LSTM model for 200 epochs took more than an hour. Hyperparameters such as optimizer, batch size, number of epochs, number of hidden layers, dropout layer size, and number of neurons in each hidden layer were tried and tested.

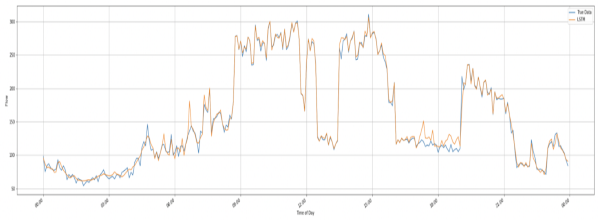

Fig. 12. The model predicted the traffic flow over a 24-hour period accurately

Also, speed was used as a feature for prediction purposes which also gives good results similar to the traffic flow.

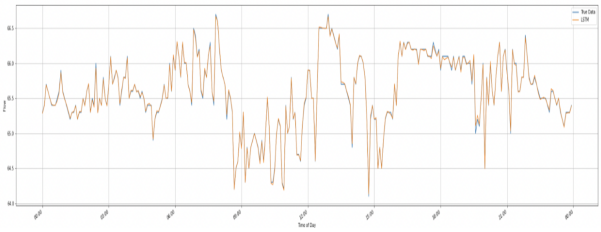

Fig. 13. The aggregate speed prediction over a 24-hour period

B. GRU

Similar to LSTM, GRU was also trained with 50, 100, and 200 epochs from which again 50 epochs were the perfectly balanced parameter. In the case of this project, GRU worked better and provided less error compared to LSTM when ran alongside LSTM to predict the traffic flow, speed and accuracy. And speed is also correctly predicted and more accurately by

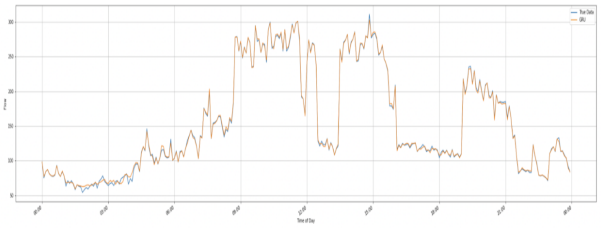

Fig. 14. The GRU model predicts flow perfectly over a 24-hour period

GRU when compared to LSTM. The better performance of GRU over LSTM may be due to the less complex nature of the GRU as the cell structure is very simple with less computation to determine the long-term dependencies as the traffic flow data is straightforward and the model which does it better and efficiently can produce accurate results.

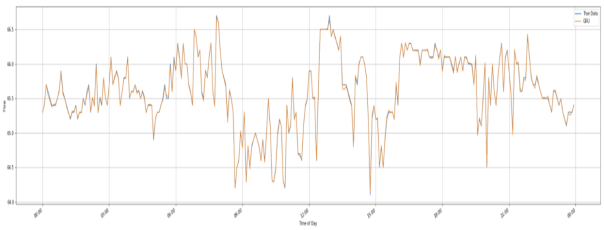

Fig. 15. The GRU model predicts speed over a 24-hour period

C. ARIMA

ARIMA model with numerous (p, d, q) parameters was tried and tested and determined using the AIC score. The lag size was also an important factor to determine the values as time series data is based on the relationship between each time step and lagging. ARIMA models the time series data well so this project tried to explore the various lag and visualized the relationship. The ARIMA model is better suited for time

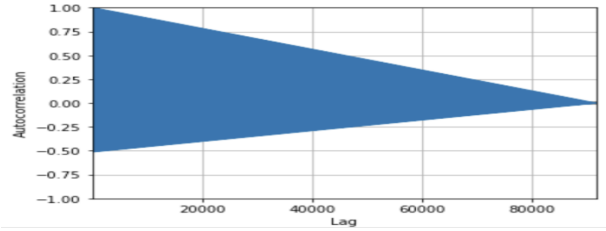

Fig. 16. The graph of Lag vs Autocorrelation

series data than any other type of dataset. However, it does not help in predicting the traffic flow in the scope of the project. The results were a bit off compared to GRU and LSTM which are classical deep learning models.

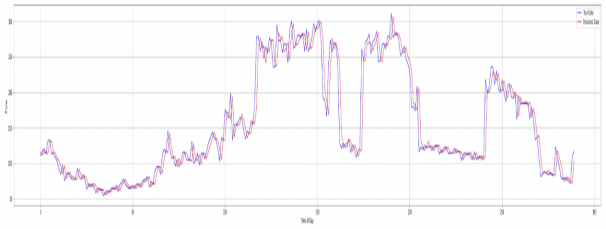

Fig. 17. The time-series graph of traffic flow with the time period

D. SAE

The fastest and most efficient model of all was the Stack Auto Encoder which takes fewer computations and provides better results. Various epochs were used along with a number of neurons in each layer from 100, 200, and 400 experimented among which 400 was the best epochs amount. Results were not great but better in terms of the simplistic nature of the SAE model.

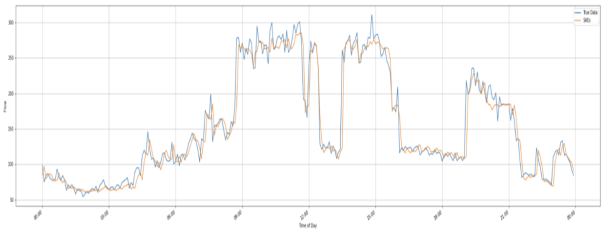

Fig. 18. The time-series graph of traffic flow with the time period by SAE

VIII. Discussion

In this project, GRU outperforms LSTM using the same layered architecture and similar hyperparameters. On average, GRU has reduced the MAE by about 10% to ARIMA and 5% to the LSTM model. ARIMA was supposed to be outperforming SAE as it is well suited for time-series data but SAE being a deep learning model works better. The deep learning approach to the traffic prediction problem is best suited as the error rate is comparatively less than the machine learning models and GRU is the best of the models.

	LSTM	GRU	SAE	ARIMA
MAPE	3.828851	1.532124	7.554953	8.790843
MAE	4.444571	1.721191	11.040941	13.263835
MSE	53.820511	5.828635	346.121024	493.682717
RMSE	7.336246	2.411257	18.604328	22.218972
R2	0.99025	0.998944	0.9373	0.91056

Fig. 19. Table: Model accuracy comparison between all models

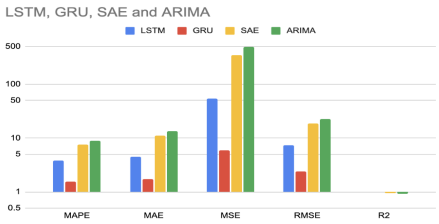

Fig. 20. Graph: Model accuracy comparison between all models

IX. Conclusion

This project performs traffic flow predictions using deep learning methods. Traffic management measures, such as route guiding and traffic control, rely heavily on traffic flow forecasting. With the growth of capabilities in automobiles, such as autopilot mode, the importance of traffic flow forecasting is escalating. Much traffic flow data is now available because of the advancement of information technology (IT) and big data. This study explored crawling traffic flow data from the CalTrans website, prepping and cleaning the dataset, and experimenting with several deep learning models such as LSTM, SAE, GRU, and ARIMA. Google Colab was used as a platform to implement machine learning algorithms.

The experimental results proved that feature engineering and data cleansing is of utmost importance in the machine learning workflow. We performed a univariate analysis for the ARIMA model. SAE was tested with epochs 100 and 200. LSTM and GRU were used to predict traffic speed and occupancy. GRU outperforms LSTM and SAEs irrespective of the epoch size. GRU has reduced the MAE by about 10The model generated with GRU has more accuracy than traditional models, demonstrating that deep learning models are more effective learners of training data than linear models such as ARIMA. They are more capable of recognizing long-term dependencies. Hence, the machine learning approach (Gated Recurrent Unit) is best suited for traffic flow prediction because the error is comparatively less than the other models tested. The study is based on traffic flow prediction across one sensor on the US 101 freeway, which prompts the study to look into the model's predictability utilizing the data from multiple sensors across the 101 freeway. Hopefully, this project will be a valuable starting point for other researchers to work on and improve upon the establishments of this project that GRU has performed better.

Acknowledgment

We would like to thank Prof. Mike Wu for his ongoing assistance and support, as well as the enormous amounts of information he offered to everyone during the course.

References

[1] R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2016, pp. 324-328, doi: 10.1109/YAC.2016.7804912.

[2] B. L. Smith, and M. J. Demetsky, "Short-term traffic flow prediction models-a comparison of neural network and nonparametric regression approaches," Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 1994, pp. 1706-1709 vol.2, doi: 10.1109/ICSMC.1994.400094.

[3] M. S. Ahmed, and A. R. Cook, "Analysis of freeway traffic time-series data by using box-jenkins techniques," Transportation Research Record, Vol. 722, pp. 1-9, Jan. 1997.

[4] X. Luo, L. Niu, and S. Zhang, "An algorithm for traffic flow prediction based on improved SARIMA and GA," KSCE Journal Civil Engineering, vol. 22, no. 10, pp. 4107–4115, Oct. 2018.

[5] H. Wang, L. Liu, S. Dong, Z. Qian, and H. Wei, "A novel work zone short-term vehicle-type specific traffic speed prediction model through the hybrid EMD–ARIMA framework," Transportmetrica B: Transport Dynamics, vol. 4, no. 3, pp. 159–186, Sep. 2016.

[6] B. Moghimi, A. Safikhani, C. Kamga, and W. Hao, "Cycle-length prediction in actuated traffic-signal control using ARIMA model,"Journal of Computing in Civil Engineering, vol. 32, no. 2, Mar. 2018, Art. no. 04017083.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in NIPS, 2012.

[8] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: a deep learning approach," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 2, pp. 865-873, Apr. 2015.

[9] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," Transportation Research Part C: Emerging Technologies, vol. 54, pp. 187-197, May 2015.
[10] Y. Tian, and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in 2015 IEEE International Conference on Smart City, Chengdu, pp. 153-158, 2015.
[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.
[12] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, D.Bahdanau, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.