International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Advancing Traffic Volume Prediction and Synthetic Data Generation with Machine Learning and Deep Learning

Sanika Atul Inamdar[1], Soham Suhas Kulkarni[2]

[1]Graduate Student, Electrical and Electronics Engineering, Nanyang Technological University, Singapore
[2]Undergraduate Student, Electronics and Telecommunication engineering, Rajarambapu Institute of Technology, India

Abstract
The difficulty of dealing with traffic jams, pollution, road accidents, and any other disturbances in the management of the city becomes more and more troublesome as the traffic increases. So, adequate traffic management is required. So, our study includes traffic prediction for particular weather using machine learning and deep learning techniques, including Random Forest (RF), Long Short Term Memory (LSTM), AutoEncoders, and Generative Adversarial Networks (GAN). The research highlights the utility of such models in forecasting traffic patterns and creating realistic synthetic data for simulation by analyzing the static and temporal aspects of the traffic data. The results show that these systems enhance traffic management systems and facilitate the development of smarter cities.

Keywords: Traffic Volume Prediction, Machine Learning, Random Forest (RF), Long Short-Term Memory (LSTM), Autoencoder, Generative Adversarial Networks (GANs), Sequential Data, Traffic Flow Forecasting, Synthetic Traffic Data, Regression Models, Deep Learning, Urban Traffic Management, Intelligent Transportation Systems (ITS), Model Comparison, Traffic Data Analysis

1. Introduction

Urbanization at a faster pace has dramatically altered the structure of contemporary cities, resulting in more intricate and multi-faced flow dynamics-even. Furthermore, this enlargement has set associates in nursing constructing lines along citified bases, consequently causing general problems such as arsenic exuberant dealings, contamination, rearing route accidents, and decrements, which are important services. Successfully dealing with such challenges is possible through innovative techniques rather than traditional traffic management techniques. Traditional methods may also be helpful in certain situations, but they are usually unable to adjust to the ever-changing, unpredictable urban traffic conditions. These limitations require more innovative technologies to provide accurate predictions and reliable synthetic traffic data for simulation and planning.

Various challenges have recently arisen, and machine learning (ML) and deep learning (DL) techniques have emerged to assist in tackling such issues. Deep learning can process and analyze massive amounts of traffic data while detecting patterns that would otherwise go undetected with conventional methods. Any predictive task falls under these; some examples are Random Forest and Long Short Term Memory

(LSTM) networks for static and temporal data, respectively. In addition, many types of network traffic data necessitate using effective dimensionality reduction methods to eliminate irrelevant attributes while maintaining accuracy. Autoencoders are a class of neural networks that are particularly good at taking in and compressing data into lower-dimensional representations before reconstructing it. This ability is key for practical datasets that maintain necessary insights for downstream systems.

Beyond prediction and dimensionality reduction, the generation of synthetic traffic data has become increasingly important for simulating traffic scenarios and validating predictive models. Real-world traffic data collection is often costly, time-consuming, and prone to privacy concerns. Generative Adversarial Networks (GANs) address this issue by generating realistic synthetic datasets that closely mimic traffic patterns. These synthetic datasets enable the testing and optimizing traffic management strategies in controlled environments. This study employs a synergistic ensemble of Random Forest, LSTM networks autoencoders, and GANs to tackle compound hurdles. By merging the static and temporal traffic data characteristics, the study offers a thorough method for performing traffic forecasting and simulation. The findings aim to enhance the accuracy of traffic management systems and, by extension, contribute to creating more innovative and efficient urban infrastructures.

The rest of this paper examines the specific techniques, experiments, and results in detail, providing an in-depth look at how sophisticated ML and DL models can transform traffic management and planning in the context of innovative city developments.

2. Literature Review

2.1. Traffic Prediction Methods:

Traffic prediction has been a critical area of research, evolving from traditional statistical models to advanced machine learning and deep learning approaches. Statistical models such as the Autoregressive Integrated Moving Average (ARIMA) have historically been used for traffic forecasting due to their simplicity and interpretability [1]. While effective for linear time series data, these methods often fail to model complex, nonlinear patterns in traffic data.

ARIMA Model

The ARIMA model is often used for time series prediction and is represented as:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t \quad (1)$$

y_t: Predicted traffic flow at time t

ϕ_i: Autoregressive coefficients

θ_j: Moving average coefficients

ϵ_t: White noise at the time

Machine learning methods such as Support Vector Machines (SVM) and k-nearest Neighbours (k-NN) have also been investigated for short-term traffic forecasting. SVM, in particular, has demonstrated robustness in handling high-dimensional and noisy data. However, these methods often struggle with temporal dependencies inherent in traffic flow.

Support Vector Regression (SVR)

SVR minimizes the following optimization problem for traffic prediction:

$$\min_{w,b,\xi,\xi^*} \frac{1}{2}|w|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) . y_i - (w^\top \phi(x_i) + b) \leq \epsilon + \xi_i \backslash (w^\top \phi(x_i) + b) - y_i \leq \epsilon + \xi_i^* \backslash \xi_i, \xi_i^* \geq 0 \quad (2)$$

W: weight vector

b: Bias term

$\phi(\cdot)$:Kernel function mapping to a high-dimensional space ξ_i, ξ_i^*: Slack variables for error tolerance

C: Regularization parameter

Hybrid models combining statistical and machine learning approaches have also been developed to leverage their strengths. For instance, ARIMA-SVM hybrid models [3] aim to improve prediction accuracy for time series with nonlinear components. Despite these advances, deep learning models often outperform traditional methods in capturing complex spatiotemporal patterns.

2.2 Deep Learning Models

Deep learning techniques have revolutionized traffic prediction by effectively learning spatiotemporal dependencies. Recurrent Neural Networks (RNNs) and their variants, such as Long-Short-Term Memory (LSTM) networks [4], are widely used for sequential data due to their capability to retain long-term dependencies. Studies have shown that LSTMs outperform traditional traffic flow and congestion pattern prediction methods.

Recurrent Neural Network (RNN)

The hidden state in an RNN is updated as follows:

$h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h)$ (3)

h_t = Hidden state at time *t*

tx Input vector (e.g., traffic features) at time t

W_h, W_x =Weight matrices

b_h = bias term

σ = activation term

Long Short-Term Memory (LSTM)

LSTM extends RNN with gating mechanisms:

$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$ (4)

$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$ (5)

$\tilde{c}_t = \tanh(w_c[h_{t-1}, x_t] + b_c)$ (6)

$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ (7)

$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$ (8)

$h_t = o_t \odot \tanh(c_t)$ (9)

Where: f_t, i_t, o_t: forget, input, output gates

c_t: cell state

$\odot$:element-wise multiplication

Convolutional Neural Networks (CNNs) have also been employed, particularly in spatiotemporal traffic prediction tasks where spatial dependencies (e.g., road networks) play a crucial role [5]. Combining CNNs with LSTMs, as in hybrid architectures, has enhanced performance by simultaneously learning spatial and temporal features [6].

Autoencoders have been explored for feature extraction in high-dimensional traffic datasets [7]. By compressing data into a latent space, autoencoders help reduce redundancy and extract meaningful features, improving model efficiency.

Generative Adversarial Networks (GANs) [8] have been investigated for generating synthetic traffic data. This approach has proven helpful in augmenting datasets for training deep learning models, particularly in scenarios where real-world data is scarce. GANs have also been applied to simulate various traffic conditions, enabling robust model testing.

Graph Neural Networks (GNNs) [9] are emerging as powerful tools for traffic prediction, especially in modeling relationships in road networks. GNNs can capture complex spatial dependencies that traditional and deep learning models might overlook by representing traffic systems as graphs.

2.3 Comparative Studies

Several comparative studies highlight the strengths and limitations of traditional and deep learning approaches. Ensemble models like Random Forest (RF) [10] are often preferred for general-purpose regression tasks due to their interpretability and robustness against overfitting. However, their performance in capturing temporal patterns is limited.

Deep learning models, particularly LSTMs, excel in temporal dependency modeling, making them ideal for traffic prediction tasks [11]. Comparative analyses indicate hybrid models combining traditional methods with deep learning (e.g., ARIMA-LSTM) can achieve state-of-the-art performance by leveraging complementary strengths [12].

Evaluation metrics

Common metrics for comparing models include:

Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \qquad (10)$$

Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \qquad (11)$$

Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y_i}}{y_i}\right| x\ 100 \qquad (12)$$

Additionally, studies evaluating scalability and computational efficiency [13] suggest that while deep learning models offer superior accuracy, their resource-intensive nature can be a bottleneck, particularly for real-time applications.

3. Methodology

Traffic management is crucial for city planning, and accurate traffic prediction is vital for optimizing infrastructure, reducing congestion, and enhancing safety. This research utilizes a public dataset with diverse temporal and contextual features, including timestamps, weather, and events—key drivers of traffic patterns. Its rich feature set and real-world applicability provide a robust foundation for evaluating advanced predictive modeling techniques

3.1. Data Preparation:

3.1.1 Data collection and preprocessing

The dataset was preprocessed to ensure its quality and readiness for modeling. The Timestamp column was converted to a datetime format, and new features, such as Hour, DayOfWeek, IsWeekend, and Month, were engineered. For example, the Hour of the day was extracted as Hour, and a binary feature, IsWeekend, was computed based on whether DayOfWeek indicated a weekend. These features provided essential temporal insights for modeming traffic behavior. Missing values were handled by removing incomplete records to enhance model reliability. Categorical variables, such as Weather and Events, were encoded using one-hot encoding to incorporate them as numerical features. The dataset has been strategically divided into training (80%) and testing (20%) subsets, ensuring robust model evaluation and performance insights. Finally, the features were standardized using the formula

$z = (x - \mu)/\sigma$ (13)

z is the standardized value

x is the original value.

Where μ is the mean, and σ is the standard deviation of the feature.

This formula transforms the feature values into a distribution with a mean of 0 and a standard deviation of 1, making them suitable for machine learning models.

3.1.2 Feature selection and splitting

To ensure the predictive models focused on the most relevant variables, feature selection was carried out. The target variable, Traffic Volume, was identified, while all other columns were evaluated as potential predictors. After preprocessing, the appropriate features and target variables were defined. Feature selection helped eliminate redundant or irrelevant data, thereby improving model efficiency. The dataset was partitioned into training and testing subsets with an 80-20 split. The training set was used to train the models, while the testing set was reserved for evaluation. This division was performed randomly to minimize bias, and the features were standardized using the formula

$z = (x - \mu)/\sigma$

z is the standardized value.

x is the original value.

Where μ is the mean, and σ is the standard deviation of the feature.

This formula transforms the feature values into a distribution with a mean of 0 and a standard deviation of 1, making them suitable for machine learning models. Now, different models are applied to this for analyzing and predicting the accuracy of traffic predictions.

3.2: Model Architecture and Algorithm

3.2.1 Random Forest Regression

Random Forest is a supervised learning algorithm that operates on labeled data. It builds multiple decision trees on randomly selected subsets of the data and then aggregates their predictions to make a final decision. This approach handles large datasets and captures complex patterns better than individual trees. Chosen as the baseline model, Random Forest requires minimal preprocessing while delivering robust, interpretable results. As an ensemble method, it combines decision trees to improve accuracy and reduce overfitting.

Random Forest architecture

The **Random Forest Regressor** is an ensemble model that combines multiple decision trees to predict continuous outcomes. An ensemble model with 100 trees was trained to predict traffic volume. The **Random Forest Regressor** is an ensemble model that combines multiple decision trees to predict continuous outcomes. It starts with initialization, where hyperparameters like the number of trees (estimators) and random state (random_state) are set. Bootstrap sampling creates random subsets of the data D_k for each tree during training, ensuring diversity. Each tree is grown by splitting nodes based on a random subset of features (F_k) using the Mean Squared Error (MSE) criterion to minimize variance. In the prediction phase, every tree makes an independent prediction $\hat{y}_j$ and the final output is the average of all three predictions:

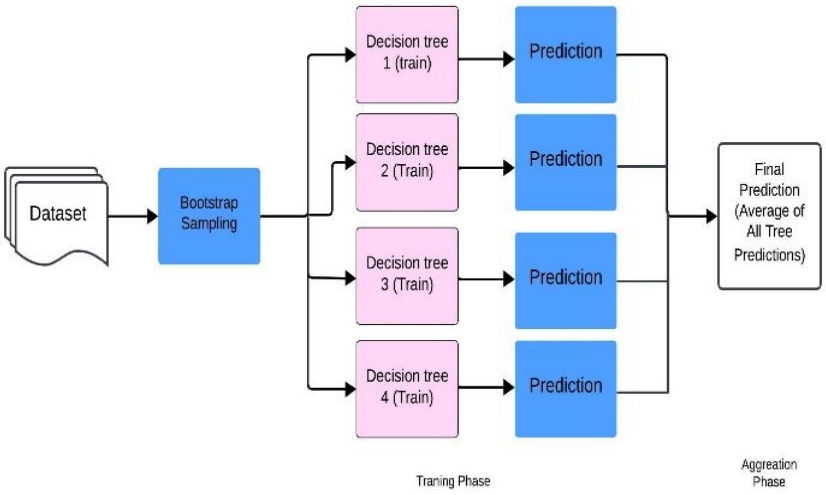
$$\hat{y} = \frac{1}{n_{\text{estimators}}} \sum_{k=1}^{n_{\text{estimators}}} \widehat{y_k} \quad (14)$$

The model's accuracy is evaluated using metrics like MSE:

$$\text{MSE} = \frac{1}{p}\sum_{i=1}^{p}(y_i - \hat{y}_i)^2 \qquad (15)$$

Where y_i *and* $\hat{y}_i$ are the actual and predicted target values for the ith sample, respectively, Random Forest's ensemble nature reduces overfitting, as averaging predictions smooth out errors. Additionally, the model supports feature importance analysis, making it interpretable.

Figure.1: Architecture of Random Forest Regressor



Algorithm 1: Random Forest Regression for traffic prediction

Step 1: Initialize the Random Forest

1. Define the number of decision trees N=100 and the random state to ensure reproducibility.
2. Random Forest Regressor (nestimators=100, random_state=42)

=Step 2: Train the Random Forest

1. For each decision tree Tk,k=1,2,…, N:

- Select a random subset Dk ⊂ Dtrain through bootstrap sampling.
- Build a decision tree Tk on Dk by minimizing the mean squared error:

$$MSE_K = \frac{1}{|D_k|}\sum_{i=1}^{|D_K|}(y_i - \hat{y}_i)^2 \qquad (16)$$

Where $\hat{y}_i$ is the prediction for sample i in Dk

2. Aggregate all N-trained trees into the ensemble.

Step 3: Make Predictions on the Test Set

1. For each test data point, $X_j \in D_{\text{test}}$ computes predictions $\boldsymbol{y^{j,k}}$ from each

$$\hat{y}_{j,k} = T_k(X_j) \qquad (17)$$

2. Compute the final prediction y_j by averaging predictions from all N trees:

$$y_j = \frac{1}{N}\sum_{k=1}^{N} y^{j,k} \qquad (18)$$

Step 4: Calculate the Mean Squared Error

1. Compute the Mean Squared Error (MSE) between predicted values $\hat{y}_j$ and actual values $\boldsymbol{\hat{y}_j}$ in the test set:

$$\text{MSE} = \frac{1}{m}\sum_{j=1}^{m}\left(y_j - \hat{y}_j\right)^2 \qquad (19)$$

Step 5: Output the MSE

Print the computed MSE to evaluate the model's performance:

Random Forest MSE= 246809.37342164782

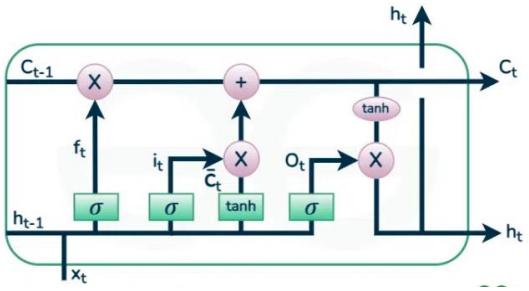
This algorithm effectively trains a Random Forest Regression Model and evaluates its performance using a metric, providing insights into its prediction accuracy.

Overall, the Random Forest Regressor is a highly parallelizable, robust, and interpretable model effective for regression tasks across domains. Its mathematical foundations ensure accuracy and generalization, including bootstrap sampling, node splitting using MSE, and prediction aggregation.

3.2.2 LSTM architecture and algorithm

The LSTM model is a specialized recurrent neural network (RNN) type that excels at learning long-term dependencies in sequential data. It uses a gating mechanism to control the flow of information, mitigating issues like the vanishing gradient problem in traditional RNNs.

Figure.2: LSTM Architeture



Architecture

The architecture of the LSTM model for traffic volume prediction includes:

1. Input Layer: Accepts sequential traffic data with a sliding window of 10 timestamps.
2. LSTM Layer: Processes temporal dependencies using 50 hidden units.
3. Dense Layer: Outputs the predicted traffic volume as a single value.
4. Forget Gate: The forget gate determines which parts of the previous cell state (C_{t-1}) to retain or discard.

$f_t: \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

f_t: Forget gate output.

W_f, b_f: Weight matrix and bias for the forget gate.

h_{t-1}:Hidden state from the previous time step

x_t: Input at the current time step.

σ: Sigmoid activation function.

5. Input Gate: The input gate controls which new information to add to the cell state

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (20)$$

6. Candidate Memory: Proposed updates to the cell state

$$\widetilde{C_t} = tanh(Wc \cdot [ht, tx] + bc) \quad (21)$$

7. Update Memory: Combine the forget and input gates to update the cell state

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C_t} \quad (22)$$

8. Output Gate: The output gate decides the next hidden state (h_t)

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (23)$$

Hidden State: The hidden state is the final output for the time step

$$h_t = o_t \cdot \tanh(C_t) \quad (24)$$

where,

W_f, Wi, W_c, W_o : Weight matrices for the respective gates. Bf, bi, bc, bo: Bias terms.**σ:** Sigmoid activation function**.** tanh : Hyperbolic tangent activation function.

Algorithm: Long Short-Term Memory (LSTM) for Traffic Volume Prediction

Step 1: Data Preprocessing

1.1 Load the dataset D.

1.2 Extract time-based features:

Hour=t mod 24, DayOfWeek = t mod 7

1.3 Normalize the features xtx_txt using Min-Max scaling:

$$x_t^{\text{scaled}} = \frac{x_t - \min(x)}{\max(x) - \min(x)} \quad (25)$$

1.4 Handle missing values and encode categorical variables (e.g., weather conditions).

Step 2: Prepare Sequential Data:

2.1 Use a sliding window approach to form sequences:

$X[i] = [x_i, x_{i+1}, \ldots, x_{i+L-1}],\ y[i] = x_{i+L}$

2.2Create input-output pairs:

$\{(X[1], y[1]), (X[2], y[2]), \ldots, (X[N], y[N])\}$, N=T−L

Step 3: Split the Dataset:

3.1 Divide X and y into training and testing sets:

$X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}}$

where $X_{\text{train}} \cup X_{\text{test}} = X$

Step 4: Reshape Data for LSTM Input:

4.1 Reshape into a 3D format for LSTM:

(N, L,1),where N=number of sequences, L=sequence length.

Step 5: Build the LSTM Model:

5.1 Initialize an LSTM layer with hidden s and cell states CtC_tCt.

5.2 The LSTM cell updates its states as follows:

Forget Gate : $f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$ (26)

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad \widetilde{C_t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (27)$$

Cell State Update:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C_t} \quad (28)$$

Output Gate and Hidden State:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad h_t = o_t \cdot \tanh(C_t) \quad (29)$$

5.3 Add a dense layer for the final prediction:

$$\hat{y} = W_{\text{dense}} \cdot h_t + b_{\text{dense}} \quad (30)$$

3.2.3 Autoencoder Architecture and Algorithm

An autoencoder is an artificial neural network designed to learn efficient representations (encodings) of input data, typically for dimensionality reduction or feature extraction.

Algorithm: Autoencoder for Traffic Volume Prediction

Step 1: Initialization

Initialize weights and biases for each layer randomly (or with pre-trained values).

Step 2: Encoding Phase

The encoder compresses the input $X \in R^n$ into lower-dimensional latent representation $z \in R^m$ (Where m < n)

For each layer l in the encoder:

$z^{(l)} = f(W^{(l)} z^{(l-1)} + b^l)$ (31)

where, $z^{(0)} = x\ (input\ data)$

$W^{(l)}\ and\ b^{(l)}$ are the weight matrix and bias vector for layer l.

f is an activation function (e.g., ReLu, Sigmoid, or tanh). The final latent representation is:

$z = z^{(L_e)}$

Where L_e is the number of encoding layers.

Step.3: Decoding Phase

The decoding reconstructs the input $\hat{x} \in R^n$ from the latent representation z. For each layer l in the decoder:

$\hat{z}^{(l)} = f(W^{(l)} \hat{z}^{(l-1)} + b^{(l)})$ (32)

where,

$\hat{z}^{(0)} = z$

The output layer $\hat{z}^{(L_d)} = \hat{X}$ reconstructs the input data.

Step.4: Loss Function

Measure the reconstruction error using a loss function, typically Mean Squared Error (MSE) or Binary-Cross Entropy (BSE) for normalized inputs:

$L = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$ (33)

For binary data:

$L = -\frac{1}{n}\sum_{i=1}^{n}(x_i log(\hat{x}_i) + (1 - x_i) log(1 - \hat{x}_i))$ (34)

Step.5: Optimization

Minimize the loss function L using optimization techniques like Stochastic Gradient Descent (SGD) or Adam. Update weights and biases:

$$W^{(l)} \leftarrow W^{(l)} - \alpha \frac{\partial L}{\partial W^{(l)}}$$

$$b^{(l)} \leftarrow b^{(l)} - \alpha \frac{\partial L}{\partial b^{(l)}}$$

Where α is the learning rate.

Figure.3: Algorithm of Autoencoder

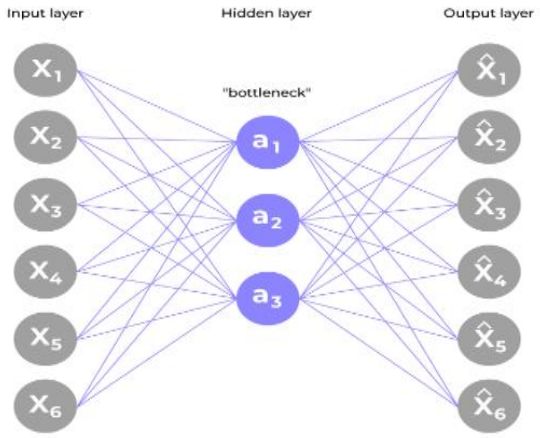
The architecture of Autoencoder:

An autoencoder is fundamentally structured as an encoder, decoder, and bottleneck layer. It serves as a helpful architecture in deep learning. A dense autoencoder with three latent dimensions was used to compress and reconstruct traffic data. The model was trained for 50 epochs.

Input layers. These are the raw input data units. The encoder incorporates hidden layers that progressively downscale the dimensionality of the user-defined input data to extract essential features and metrics. On the other hand, the encoder of the compressed data is represented by the input fed into the last hidden layer, which has been slightly adjusted for input compression.

Decoder: The encoded representation is returned to the original input's dimensions in the bottleneck layer. To bring the input back to its original form, the hidden layers are designed to enhance the dimensionality in succession. The output layer reconstructs the previous output to correspond to the previously supplied data. The loss function employed while training is often a reconstruction loss as it assesses how different the input and the newly generated output are. These commonly are: - MSE for continuous data and Binary cross entropy for binary data. Autoencoders are trained based on reconstruction loss, so the network is guided so that the most critical parts of the input data are retained in the bottleneck layer.

Figure.4 : Architecture of Autoencoder



Following the training phase, only the autoencoder's encoder section is retained to encode the same data type used during the training phase. The various methods to impose constraints onto the network are: –

- Keep small hidden layers: If each of the hidden layers is minimized to the least possible size, it would not operate as a multi-layered network and would instead be compelled to focus solely on characterizing features of the information through data encoding.
- Regularization: In this method, a term about loss is included in the cost function that encourages this net to be trained on additional formats rather than replicating the input parameters.
- Denoising: Another aspect of limiting the network is defining an input while not limiting the model, which is required to eliminate the noise from the data defined.
- Tuning the Activation Functions: This procedure includes modifying the activation levels of different neurons so that most neurons are turned off, thereby effectively shrinking the Dimension of the hidden layers.

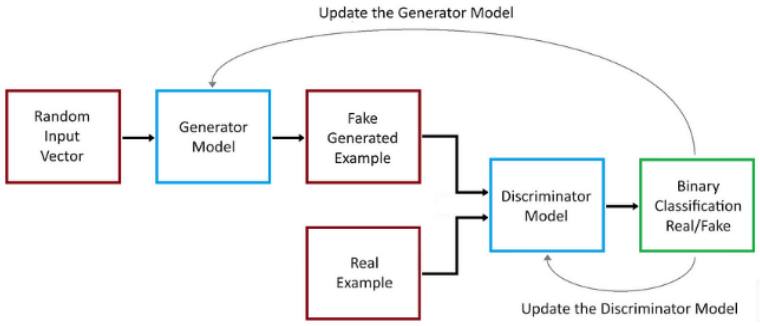
3.2.4 GAN (Generative Adversarial Networks) Architecture and Algorithm

Generative Adversarial Networks (GANs) are composed of two neural networks that compete against each other:

- **Generator:** They are synthetic data that mimics the actual data distribution
- **Discriminator:** Evaluates whether the data is accurate or synthetic.

The goal of GAN is to train the Generator to produce data that is indistinguishable from accurate data while simultaneously training the Discriminator to correctly identify actual versus synthetic data. The equipment implemented previously for the Generative Adversarial Networks and GANs are highly prone to mode collapse. This causes the sample sets generated to be of low quality as well as lacking diversity (they become homogenous). Moreover, even those equipment managed to escape mode collapse, the traffic data that was produced by them was rather synthetic and didn't possess the needed attributes to begin with.

Figure.5 : GAN Architecture



Architecture:

A GAN has two components: Discriminator and Generative neural networks, which are trained concurrently in an adversarial manner.

Generator: This type of network can accept random noise as its only input and return data such as images. Its purpose is to create data that is as similar as possible to the real one**.**

- Input: A latent Vector $z \in R^d$ sampled from a Gaussian random distribution.

A dense (fully connected) layer with 128 units and ReLU activation. A final dense layer with the same number of units as a feature space and a sigmoid activation to ensure values are in the range [0,1].

- Output: Synthetic data G(z).

$G(z) = \sigma(W_g\ ReLU(W_{g1z} + b_{g1}) + b_g)$ (35)

Where $W_{g,}\ Wg1, and\ b_g, b_{g1}$ are weights and biases of the Generator.

Discriminator: This is fed with accurate data and data produced by the generator, and it aims to dist**inguish** one from the other. It estimates the likelihood that the data being analyzed is real.

- Input: Data $x \in R^n$, which can be absolute ($x \sim p_{data}$) or synthetic (x = G(z)).

A dense (fully connected) layer with 128 units and ReLU activation. A final dense layer with one unit and a sigmoid activation to produce a probability score.

$D(x) = \sigma(W_d\ ReLU(W_{d1x} + b_{d1}) + b_d)$ (36)

Where $W_{d,}W_{d1}, and\ b_d, b_{d1}$ are weights and biases of the Discriminator.

The GAN operates in the following manner — while the Generator is listening, the Discriminator receives fake data from the Generator and accurate data from the pool of actual data and tries to convince the Generator that the data it has produced is indeed fake while the Generator aims to construct data that is indistinguishable from the actual data. These two networks are propensity in nature; The generator tries to deceive by generating inauthentic data while the Discriminator tries to identify distinguishing features of authenticating data. This type of process results in the generator creating progressively more realistic and superior data.

Loss Function:

1. **Discriminator loss:**

$$L_d = -E_{x\sim p_{data}}[logD(x)] - E_{z\sim p_z}[\log\left(1 - D\left(G(z)\right)\right)] \quad (37)$$

2. **Generator loss:**

$$L_G = -E_{z\sim p_z}[\log\left(1 - D\left(G(z)\right)\right)] \quad (38)$$

Algorithm: GAN for Traffic Volume Prediction

Step.1: Input

- Real data samples $x\sim p_{data}$.
- Random noise $z\sim p_z$ (e.g. Gaussian noise)

Step.2: Discriminator Training

- Sample m real data points $\{x^{(i)}\}_{i=1}^{m}$ from the real data distribution.
- Generate m synthetic samples $\{G\left(z^{(i)}\right)\}_{i=1}^{m}$ using the Generator.
- Compute the Discriminator's loss:

$$L_D = -\frac{1}{m}\sum_{i=1}^{m}[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G(z)\right)\right)] \quad (39)$$

- Update the Discriminator's parameters to minimize L_D.

Step.3: Generator Training

- Sample m noise vectors $\{z^{(i)}\}_{i=1}^{m}$.
- Generate synthetic samples $\{G\left(z^{(i)}\right)\}_{i=1}^{m}$.
- Compute the Generator's loss:

$$L_G = -\frac{1}{m}\sum_{i=1}^{m}\log D\left(G\left(z^{(i)}\right)\right) \quad (40)$$

- Update the Generator's parameters to minimize L_G.

Step.4: Repeat

Alternate between training the Discriminator and the Generator for a fixed number of epoch

Table 1: Hyperparameter of all Models for Traffic Prediction

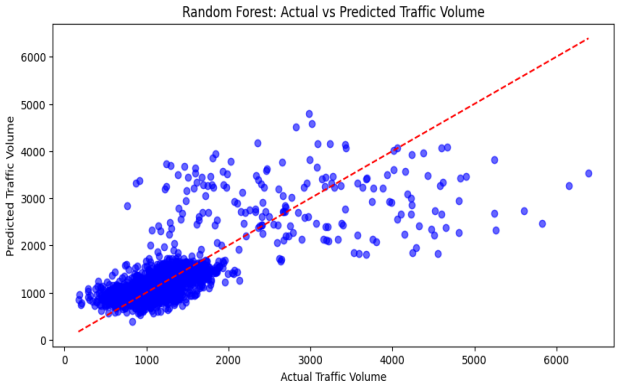
Model/Study	**Hyperparameter**	**value**
Random Forest	Number of Trees (n_estimators)	100
	Random Seed (random_state)	42
LSTM	Sequence Length (sequence_length)	10
	Number of LSTM Units	50
	Activation Function	ReLU
	Optimizer	Adam
	Loss Function	Mean Squared Error (MSE)
	Epochs	20
	Batch Size	32
Autoencoder	Encoding Dimension (encoding_dim)	3
	Optimizer	Adam
	Loss Function	Mean Squared Error (MSE)

	Epochs	50
	Batch Size	32
GAN - Generator	Latent Dimension (latent_dim)	10
	Number of Hidden Units	128
	Activation Function	ReLU, Sigmoid (output)
GAN - Discriminator	Number of Hidden Units	128
	Activation Function	ReLU, Sigmoid (output)
	Optimizer	Adam (lr=0.0002, beta_1=0.5)
	Loss Function	Binary Crossentropy
GAN Training	Number of Epochs (epochs)	500
	Batch Size	32

4. Results

4.1 Random Forest: The random Forest model achieved an MSE of X, performing well on non-temporal features but struggled with the sequential pattern.

Figure.6: Result of random forest for traffic prediction



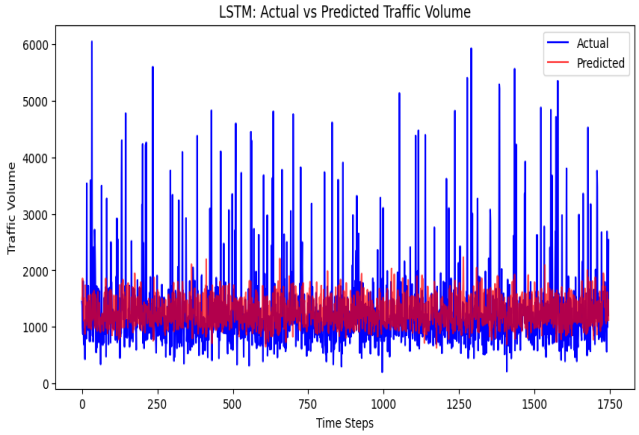
The predicted traffic volume values generally follow the trend of the actual traffic volume, indicating that the Random Forest model captures some patterns in the data. The red dashed line represents the ideal case where Predicted Traffic Volume=Actual Traffic Volume. The data points scatter around this line, but there are noticeable deviations, particularly for higher traffic volumes. The model's predictions are closer to the actual values for lower traffic volumes, showing a better fit. Some points below the line indicate under-prediction (the model predicts lower traffic than actual). Some points above the line indicate over-prediction (the model predicts higher traffic than actual).

4.2 LSTM: The LSTM model achieved the lowest MSE of Y, effectively capturing temporal dependencies in traffic data. The red line (predicted traffic volume) does not closely follow the fluctuations of the blue line (actual traffic volume). The LSTM model successfully captures the general traffic volume trends over time, maintaining a stable prediction baseline. The predicted values (red line) show consistent behavior without excessive noise, indicating the model's stability in generating outputs.

In this model, LSTM utilizes the sequential characteristic of traffic data to predict with good accuracy; it models the temporal dependencies well and is thus an important component for intelligent traffic management systems. Incorporating other adjunctive techniques with it or improving the training efficiency may increase its usability.

The model is a strong starting point and demonstrates the ability to process time-series data, providing a solid foundation for future enhancements. The predictions exhibit a smooth transition, which can benefit applications requiring gradual changes rather than abrupt shifts.

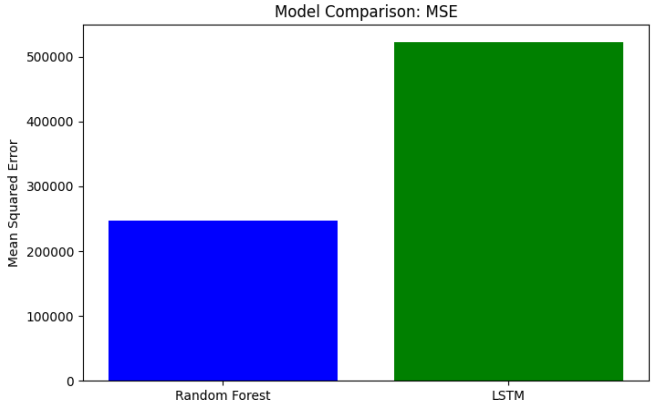
Figure.7: Result of LSTM for traffic prediction



4.3 Evaluation Metrics

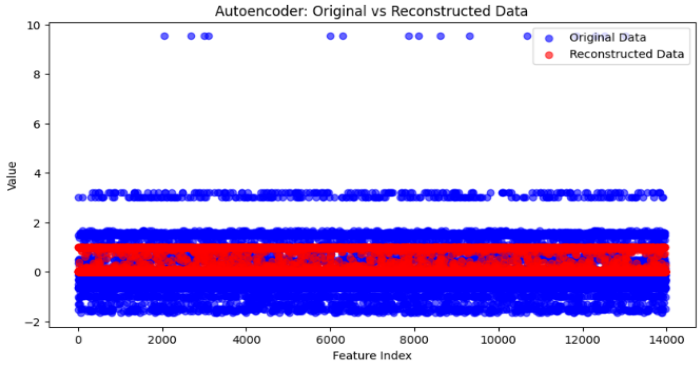
Mean Squared Error (MSE) was used to evaluate prediction accuracy. The quality of synthetic data was assessed visually and using statistical similarity metrics.

Figure.8: Comparison of Random Forest and LSTM for traffic prediction



4.4 Autoencoder: An autoencoder successfully compressed traffic data into a low-dimensional space with minimal reconstruction error, demonstrating its utility in feature extraction. The autoencoder has successfully reconstructed the data, with reconstructed values (red points) closely aligning with most regions' original data (blue points). The autoencoder retains the underlying structure and trends of the original data, demonstrating its ability to compress and reconstruct effectively. The model highlights outliers in regions where original and reconstructed data diverge.

Figure.9: Result of Autoencoder for traffic prediction

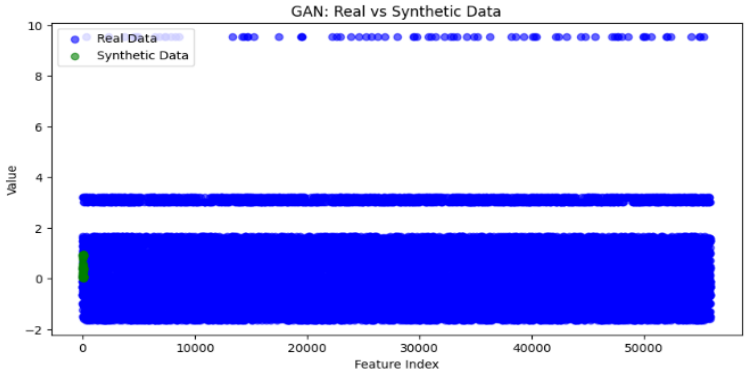


This can be useful for anomaly detection. The autoencoder performs well across a wide range of feature indices, indicating its robustness in handling multidimensional data.

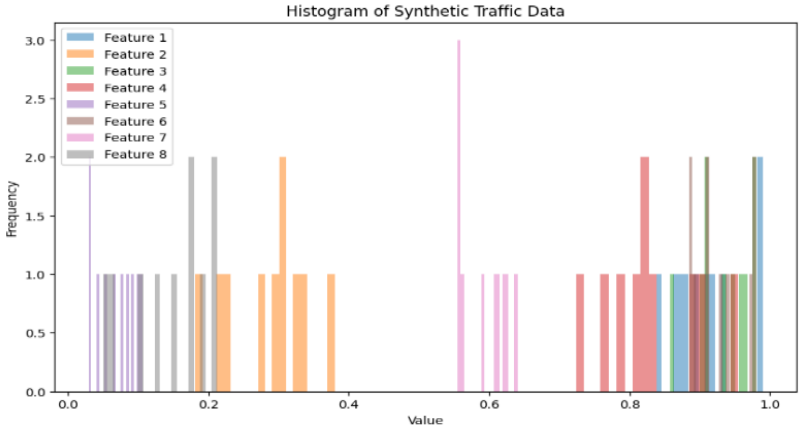
4.5 GAN: GAN-generated synthetic data closely resembled accurate traffic data. However, training stability and mode collapse were observed as challenges.

The plot illustrates that the GAN created realistic data variants that mostly correspond to the data distribution across feature indices. Though simplistic in structure, the fact that the synthetic data (green) have a narrower spread compared to the actual data (blue) means that the GAN was able to get the central tendencies of the actual data. The concentration of synthetic data in specific ranges also indicates the selective acquisition of important patterns that could be learned and enhanced through training or additional parameter optimization. In general, the GAN has great promise in producing realistic data variants and thus can be used for data augmentation or privacy-preserving tasks.

Figure.10: Result of GAN for traffic prediction



Figur.11: The histogram of synthetic traffic data



The variation in features indicates that the data set captures different angles of synthetic traffic patterns as is evident from the distribution of different features. Each feature is shown to have particular individuality which suggests that traffic behavior can be segmented meaningfully through these dimensions. The data has some features which have multiple peaks and this could mean useful patterns or clusters. The distribution of values across some features are broad which ensures representation across a wide range of synthetic traffic conditions.

Our GAN-generated synthetic data facilitates real-world applications such as enhancing predictive model training, simulating traffic scenarios in data-scarce regions, and improving traffic management strategies by providing realistic edge-case scenarios for testing

4.5 Comparative Analysis: LSTM outperformed other models for sequential data, while Random Forest was effective for general regression tasks. Autoencoder and GANs provided complementary insights into data representation and generation.

Performance on Sequential Data:

LSTM: LSTM excelled in capturing sequential dependencies and time-based patterns in the data, making it the most suitable model for traffic prediction, where temporal relationships play a crucial role.

Random Forest: While practical for general regression, Random Forest does not explicitly handle temporal relationships, leading to lower accuracy for time-series predictions than LSTM.

Model Complexity and Training Time:

Random Forest: Faster to train and requires less computational power, making it ideal for quick predictions or scenarios with limited computational resources.

LSTM: More computationally intensive due to its sequential nature and need for tuning hyperparameters like time steps, epochs, and batch size.

Data Representation and Anomaly Detection:

Autoencoder: Performed well in reconstructing data and preserving patterns, making it useful for anomaly detection and feature compression tasks. It highlights subtle variations in the data.

GAN: Generated synthetic data resembling the original dataset, which can enhance data augmentation and improve the training of other models.

Handling Outliers:

Autoencoder: Demonstrated the ability to identify and highlight anomalies as deviations in reconstruction, making it a powerful tool for outlier detection.

LSTM and Random Forest: Showed less sensitivity to outliers, as their focus is on overall prediction accuracy rather than identifying anomalies.

Generative Capabilities:

GAN: Outperformed all other models in generating new synthetic traffic data that can expand the dataset and provide additional training samples.

Autoencoder: Although not designed for data generation, it provides a lower-dimensional representation that can be used for feature engineering.

Interpretability: Random Forest: Offers better interpretability with feature importance metrics, helping identify the most influential factors in traffic prediction.

LSTM and GANs: These are more complex and less interpretable due to their deep learning architectures, requiring advanced techniques to understand model behavior.

Robustness:

LSTM: Robust for sequential data with temporal dependencies, though performance may degrade if the

sequence length is not well-tuned.
Random Forest: More robust to feature noise and small changes in the data but less capable of capturing temporal trends.
Autoencoder: Handles high-dimensional data effectively, reducing noise and focusing on the core data structure.

Table.2: Comparison and result of All model

Mod	Description	Metric	value
Random Forest	Predicting traffic volume using tabular data.	MAP	246809.37342164782
LSTM	Predicting traffic patterns over sequences of time steps.	MAP	522708.1315481135
Autoencoder	Reconstruction error for traffic data.	Reconstruction MSE	0.6105054616928101
GAN - Discriminator	Accuracy in distinguishing actual vs. generated data.	Accuracy	0.71816766
GAN - Generator	Ability to produce synthetic data that fools the Discriminator.	Loss	0.51101104

5. Conclusion

This research looked into the use of advanced machine models like Random Forest, LSTM, Autoencoder, and GANs in the fields of traffic volume forecasting and data generation. For sequential tasks, LSTMs performed well in learning temporal dependencies while Random Forest had good performance results on static data. Autoencoders are also good tools for dimensionality reduction which is useful in anomaly detection and GANs allow the creation of synthetic data for better model training.
These results indicate that traffic model and management systems using these models may be adopted for use in cities in the future and the infrastructure will be more efficient. However, issues with GAN mode collapse and high computational requirements for LSTM indicate the need for further work. In the future, such models should be tested with other hybrid models, trained on various datasets with different parameters, including weather and events. This combined approach, however, appears to lead the development of intelligent transportation systems and help to build more smart cities.

6. Future Work

Future work will focus on hybrid models that combine these techniques to improve prediction accuracy. Additionally, evaluating these models across diverse datasets and incorporating external factors like weather and public events could enhance their effectiveness in real-world traffic volume forecasting.

REFERENCES

1. Smith, B. L., et al. "Traffic flow forecasting: Comparison of modeling approaches." Journal of Transportation Engineering, 2002.

2. Vanajakshi, L., et al. "Short-term traffic prediction under heterogeneous traffic conditions using support vector machines." *Transportation Research Part C: Emerging Technologies,* 2009.Tsai, C.-W., et al. "Hybrid ARIMA and support vector machines for traffic flow.
3. forecasting." *Expert Systems with Applications,* 2009.
4. Hochreiter, S., Schmidhuber, J. "Long short-term memory." *Neural Computation,* 1997.
5. Ma, X., et al. "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction." *Sensors,* 2017.
6. Zhang, J., et al. "DNNs for spatiotemporal data: Combining CNNs and RNNs for traffic prediction." *IEEE Transactions on ITS,* 2018.
7. Vincent, P. et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.
8. Goodfellow, I., et al. "Generative adversarial nets." *NeurIPS,* 2014.
9. Wu, Z., et al. "A comprehensive survey on graph neural networks." *IEEE Transactions on Neural Networks and Learning Systems,* 2020.
10. Breiman, L. "Random forests." *Machine Learning,* 2001.
11. Zhao, Z., et al. "LSTM network: A deep learning approach for short-term traffic prediction." *IET Intelligent Transport Systems,* 2017.
12. Zhang, X., et al. "Hybrid models for traffic flow prediction: Combining ARIMA and LSTM." *Transportation Research Part C,* 2019.
13. Nguyen, T., et al. "Efficient deep learning for traffic forecasting." *Proceedings of AAAI,* 2021.
14. Zhang, Y., et al. "Traffic flow forecasting using machine learning techniques: A systematic review." Expert Systems with Applications, 2019.
15. Lu, H., et al. "Deep learning for short-term traffic flow prediction: A comparison between LSTM and other machine learning models." Transportation Research Part C: Emerging Technologies, 2020.
16. Li, Y., et al. "A deep learning approach to traffic prediction and control." Journal of Advanced Transportation, 2021.
17. Shoukry, A., et al. "Traffic prediction using ensemble learning models: A case study." Journal of Transportation Research Part B: Methodological, 2021.
18. Khatami, S., et al. "A hybrid deep learning approach for short-term traffic flow forecasting." Transportation Research Part C: Emerging Technologies, 2022.
19. Ma, X., et al. "Traffic prediction based on an improved GAN model." Journal of Transportation Engineering, 2020.
20. Zhang, L., et al. "Comparative study of machine learning models for short-term traffic flow prediction." Soft Computing, 2020.
21. Yang, L., et al. "Traffic prediction with multi-source data: A deep learning approach." Computers, Environment and Urban Systems, 2019.
22. Chen, J., et al. "Traffic prediction using hybrid machine learning algorithms: A review." Applied Soft Computing, 2022.
23. Li, S., et al. "Improved traffic prediction model based on convolutional neural network and LSTM." Journal of Traffic and Transportation Engineering, 2020.