Research Article

Aquila Optimizer-Based Hybrid Predictive Model for Traffic Congestion in an IoT-Enabled Smart City

Ayushi Chahal, Preeti Gulia, Nasib Singh Gill, and Nishat Sultana

1Department of Computer Science & Applications, Maharshi Dayanand University, Rohtak, Haryana, India
2Department of Computer Science & Engineering, Daffodil International University, Ashulia, Dhaka, Bangladesh

Correspondence should be addressed to Nishat Sultana; nishatsultana241@gmail.com

Received 22 November 2022; Revised 6 January 2024; Accepted 17 January 2024; Published 31 January 2024

Academic Editor: Alexander Hošovský

Effective traffic congestion prediction is need of the hour in a modern smart city to save time and improve the quality of life for citizens. In this study, AB_AO (ARIMA Bi-LSTM using Aquila optimizer), a hybrid predictive model, is proposed using the most effective time-series data prediction statistical model ARIMA (Autoregressive Integrated Moving Average) and sequential predictive Deep Learning (DL) technique LSTM (Long Short-Term Memory) which helps in traffic congestion prediction with a minimum error rate. Also, the Aquila optimizer (AO) is used to elevate the adequacy of the AB_AO model. Three road traffic datasets of different cities from the "CityPulse EU FP7 project" are used to implement the proposed hybrid model. In a time-series dataset, two components need to be handled with care, i.e., linear and nonlinear. In this study, the ARIMA model has been used to manage linear components and Bi-LSTM is used to handle nonlinear components of the time-series dataset. The Aquila Optimizer (AO) is used for hyperparametric tuning to enhance the performance of Bi-LSTM. Error measurement parameters like the Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE) are used to validate the results. A detailed mathematical and empirical analysis is given to justify the performance of the AB_AO model using an ablation study and comparative analysis. The AB_AO model acquires more stable and precise results with MSE as 18.78, MAE as 3.18, and MAPE as 0.21 than other models. It may further help to predict the vehicle count on the road, which may be of great help in reducing wastage of time in traffic congestion.

1. Introduction

Traffic congestion has become a major challenge to human society, which consumes a lot of time for a person. Road traffic cannot be negated if one wants to improve the quality of life in a smart city. Predictive analytics plays a very important role in every sector of a real-life scenario. With the advancement of technologies like IoT (Internet of Things), ML, and Big Data [1], prediction of any kind can be achieved. The traffic congestion problem can also be resolved if one can predict it before it happens.

It has been seen recently that lots of useful time gets wasted due to travelling. Traffic congestion with travelling time makes a deadly duo to spoil the useful time of a person. These days, time is the new currency for every economy to grow. Traffic congestion can cause economic losses. A survey reported that in Brazil, there is about 808 billion loss per year due to traffic congestion. The same is recorded as 1208 billion loss per year for the USA [2]. A scenario like this motivates us to make a genuine effort which can resolve or avoid the problem of traffic congestion. To achieve this, prediction or forecasting of traffic congestion is the only way out.

Deep learning models have immensely contributed to intelligent transportation systems. ARIMA and LSTM models are very popular statistical and DL models that are used for forecasting a time-series dataset [3]. Traffic flow prediction is performed for a particular time interval. If the prediction is performed for less than 15 min, then it is known as a short-term traffic flow prediction system; otherwise, it is known as a long-term prediction system. A hybrid stepwise modelling framework can predict dynamic states of traffic flow when there is congestion [4].
In this study, we have used the statistical model ARIMA and the DL model LSTM to detect traffic congestion. The proposed model AB_AO is implemented on three datasets from the “CityPulse EU FP7” project. It is an openly available dataset that contains traffic data of two smart cities for different time intervals. Hence, the dataset becomes suitable for traffic congestion prediction in a smart city. The main contribution of this research is given as follows:

(1) This study proposes an optimized hybrid predictive model for traffic congestion prediction in an IoT-enabled environment

(2) A hybrid predictive analytical model AB_AO such as ARIMA and Bi-LSTM is deployed using an optimization technique on a big time-series dataset named as “CityPulse” dataset

(3) The Aquila optimizer is exploited to tune hyperparameters of Bi-LSTM

The paper is organized as follows: The list of acronyms used throughout the manuscript is given in Table 1 for user understanding. The related literature review is given in Section 2. Section 3 explains ARIMA, LSTM, Bi-LSTM models and Aquila Optimizer in detail. Section 4 describes a stepwise approach and implementation of the proposed model used for prediction. Section 5 elaborates the dataset, the ablation study of proposed model, parameters utilized to measure performance of proposed model. This sections also explains the results obtained in the study and comparative analysis is also provided. Section 6 is a conclusive section that gives a brief summary of the whole manuscript and discusses the possible future work of the existing work.

2. Related Work

Traffic congestion or traffic flow prediction can be performed using statistical and machine learning models. Traffic congestion prediction is a complex process in which one must be specific about the characteristics on which they are working. It depends upon the dataset a researcher is using, to deduce the conclusion from its study. Datasets generally used for traffic congestion prediction have both spatial and temporal characteristics. For this study, the focus will be on the temporal properties of the dataset.

For time-series data analysis, it is very important to choose an effective model. Nowadays, the hybridization of models is a very common practice, but no one takes care of the various components of the time-series data. One needs to identify various components or features of the dataset. The identified component is then handled using a single model. After this, one can integrate differently chosen models by various means.

2.1. Statistical Models. Hamed et al. [5] used the ARIMA model for short-term traffic forecasting. The ARIMA model is one of the best statistical models that are used for time-series forecasting. There are many variants of the ARIMA model. Some of the most popular ARIMA models are ARIMA, SARMA, and SARIMAX. ARIMA aids in prediction of those datasets, which are nonseasonal in nature but have some patterns. When the dataset has certain seasonal characteristics, the SARMA model is applied. Datasets with both exogenous variables and seasonal characteristics are fitted using the SARMAX model.

Statistical models produce good results for traffic flow/congestion predictions if one is using a linear dataset. However, these models solely depend upon historical data and do not have space to store intermediate results for long-term prediction. The AB_AO model uses Bi-LSTM to handle nonlinearity as well as intermediate storage which enhances efficiency as compared to the simple statistical model.

2.2. Machine Learning and Deep Learning Models. Medina-Salgado et al. [6] gave a bibliographic review of different computational techniques used for traffic flow prediction. This review has focused on the intelligent transportation system in which the IoT and intelligent algorithms are incorporated into the transportation environment. This study helps in understanding different computational models that are used for intelligent transportation systems. Mehdi et al. [7] used a Convolutional
Neural Network (CNN) for traffic congestion prediction. First, labelling is performed using entropy on meta-parameters. In this study, the “CityPulse EU FP7 project” dataset is used. Multiple samples of the dataset are used for entropy-based labelling. After labelling, the regression problem is then converted to the classification problem. To classify the data, the CNN model is used. The CNN model uses the ReLU activation function, fully connected layer 1 with 384 parameters, fully connected layer 2 with 100 parameters, and an output layer with 1 parameter. Reshma Ramchandra and Rajabhushanam [8] elaborated on different ML algorithms used for traffic forecasting. Four kinds of ML models are used, namely, Random Forest (RF), Deep Autoencoder (DAN), LSTM, and Deep Belief Network (DBN). All these four models are used for traffic forecasting. A comparative analysis of these models is conducted using matrices like accuracy, precision, and recall. This study has shown that the LSTM model is the best one for traffic forecasting of other ML models. LSTM outperforms others with 95.2% accuracy. Kang et al. [9] explained a short-term traffic flow prediction model using the LSTM recurrent neural network. Both upstream and downstream datasets are detected by the detector which helps in the improvement of the spatial-temporal prediction. Although ML and DL models are effective prediction methods that can process complex traffic flow data, for time-series data, one needs to capture both linear and nonlinear characteristics to achieve accuracy. In the AB_AO model, Bi-LSTM is used to capture nonlinear features of the temporal data and reduces the error rate as data from both directions are used in this model.

2.3. Hybrid Models. Pan et al. [10] proposed a hybrid FD-Markov-LSTM model using the fundamental diagram Markov model and LSTM to predict traffic state, i.e., congested and uncongested. The proposed work is applied on the dataset from three case studies of Beijing and Los Angeles. A comparative analysis with other state-of-the-art techniques has also been provided by the researchers. This comparative analysis shows that the FD-Markov-LSTM model decreased MAE by 39%, RMSE by 35%, and MAPE by 7.4%. Miao et al. [11] proposed a GDENet (Graph Differential Equation Network) to predict traffic flow. This approach has worked effectively for spatiotemporal correlational datasets. In this model, authors have used the gate structure to train a model for temporal properties and GNN (Graph Neural Network) for spatial properties of the dataset. STFI (Spatio-Temporal Feature Integrator) is used to integrate spatial and temporal features of the dataset. PeMS03, PeMS04, PeMS07, and PeMS08 are the datasets used to test the proposed model using MAE, MAPE, and RMSE as evaluation matrices.

Liu et al. [12] proposed a grey convolutional neural network model to predict traffic flow. This study has analyzed the effect of accidents on traffic flow. A small sample size in which features like traffic flow and traffic speed fluctuation rate is used to identify the accident area. Both spatial and temporal components of the traffic flow dataset are used to predict the traffic flow. This grey CNN model is verified on real-time data in Hangzhou Viaduct in China. Grey CNN outperformed the other models like ANN, ARIMA, and GRU. Xuecai et al. [13] proposed a hybrid model for traffic flow prediction. The Autoregressive Fractionally Integrated Moving Average (ARFIMA) model is integrated with Nonlinear Auto-Regressive (NAR) for traffic flow prediction. MAR is considered a neural network model. ARIFMA is used for long-term memory storage and handling linear components. NAR is used to address short-term memory for a nonlinear residual component. The results of the model are validated on an openly available dataset named PeMS database which contains traffic flow data of California freeways. Yan et al. [14] proposed a hybridized model to forecast traffic flow for the short term only. This model used support vector regression to reduce the negative effect of outliers in the traffic flow dataset. The efficiency of the model depends on the environment (like weather conditions, accidents, or traffic jams) in which the traffic data are collected. For this study, different traffic flow indicators are used like speed, traffic flow, and occupancy. Nonlinearity of the dataset is optimized with the help of an adaptive fruit fly optimization algorithm.

Chen et al. [15] proposed a novel hybrid model that can be used for forecasting purposes, particularly in a time-series dataset. This study has used the ARIMA model to perform linear modelling and other artificial intelligence models like Support Vector Machine (SVM), and multilayer perceptron models are applied for nonlinear modelling in a time-series dataset. Both models are combined using suitable nonlinear parameters. This study showed that nonlinear forecast models can perform better than any existing model. Syam and Girsang [16] predicted the traffic flow of 4G data over the network using a hybrid technique. The ARIMA model of order (0, 0, 6) and the LSTM model (windows = 100, 2 layer, 100 neuron) are used to create a hybrid model. The RMSE value for this model is 387.69. They concluded that if the prediction is made within 50 hrs, then only its accuracy is high.

Chen et al. [17] proposed a new hybrid model in the field of traffic congestion prediction. A DL technique “ANN” is used in Ensemble Empirical Mode Decomposition (EEMD), to predict traffic flow for different intervals. ANN and EEMD techniques when clubbed together help traffic data users to collect more accurate data. This further helps in better decision-making. Su et al. [18] proposed a lightweight tensor flow prediction method to predict traffic flow. This technique overcomes the drawbacks of neural network models. Maximum neural network models are not able to give accurate results with limited data. This approach establishes a multidimensional relationship between traffic flow values and time intervals. Then, missing values are constructed. Finally, the tensor is used for the prediction of traffic flow.

Izhar et al. [19] used a hybrid model for label generation on the CityPulse EU FP7 project, in order to predict traffic congestion. Izhar et al. used two features named vehicle_count and average_speed and concatenated them to generate labels as congested or not congested. Before labelling the dataset, class balancing using random
undersampling is performed on it. To validate their results, posterior analysis is performed using the Cumulative Distribution Function (CDF). Khan et al. [20] predicted traffic flow and pollution using a hybrid model based on CNN and LSTM. To validate the study, the authors have used the CityPulse dataset. Authors have used CNN to handle spatial data and LSTM to handle temporal data. Root Mean Square Error (RMSE), accuracy, and time consumption are taken as matrices to compare models. RMSE came out to be 49, and accuracy was 92.3%.

Hajirahimi and Khashei [21] used ARIMA, Multilayer Perceptron (MLP), and SVM models to make a hybrid model for time-series forecasting. Five datasets are used for prediction, i.e., British pound/US dollar exchange rate, Wolf's Sunspot, Nikkei 225 stock price, the Colorado wind speed, and Canadian Lynx. Accuracy is considered as a measure to tell the best hybrid model among SVM-ARIMA, MLP-ARIMA, ARIMA-SVM, and ARIMA-MLP. Z. Hajirahimi and Khashei [22] concluded that if one uplifts the weighted limitations from the sequential model, then it can give better results than traditional model results. Research presented a novel weighted sequential hybrid model in which first, components of time-series data are modelled, and then, different weights are sequentially assigned to the components. Weights given to the components are decided using the least square method. ARIMA and ANN models are hybridized to analyze a time-series dataset. Hajirahimi and Khashei [23] presented a review of the hybridization of different models for time-series prediction. Hybrid models are categorized into four kinds, namely, preprocessing-based hybrid models, parameter optimization-based hybrid models, component combination-based hybrid models, and postprocessing-based hybrid models. The study concluded that around 250 research papers have been published for hybrid models in recent years for time-series prediction. This study gave a clear scenario about how and when to make a hybrid model. Li et al. [24] contributed to the same area by introducing a hybrid model, using Radial Basis Function Artificial Neural Network (RBF-ANN) and ARIMA for predicting traffic flow. The validation study is conducted on the real-time traffic data from Nansha District, Guangzhou, China. Wang et al. [25] presented a hybrid technique to capture complete linear and nonlinear time-series components. The ARIMA model and the SVM model are used to predict traffic congestion. This hybrid model when applied to the dataset in a piecewise manner does not come out to be the best model.

Abualigah et al. [26] presented a smart parking system for vehicles using IoT and cloud technology. The kernel-based least mean square algorithm is used to calculate the free spots for parking using the autoregression machine learning technique. To validate the results, this study has used a smart city dataset. Authors have used MSE and MAE matrices to measure the error rate. Qiao et al. [27] proposed a hybrid Traffic Flow Model into Deep Learning (TFMDL) that is used to predict traffic state. The proposed model has the properties of both model-driven and data-driven approaches. An open-access PeMS dataset is used for this study. MAE, RMSE and MAPE are the evaluation parameters considered which resulted in 1.20, 2.24 and 2.75%, respectively.

This review provides all the possibilities of the hybridization of statistical, machine learning, or deep learning models in different possible scenarios. Metaheuristic optimizers are found to be much more effective and enhance the results of deep learning and machine learning models. AB_AO hybridizes ARIMA and Bi-LSTM, and to enhance its efficiency, AO optimizes the hyperparameters of Bi-LSTM.

2.4. Models with Optimization Techniques. Liu et al. [28] created a model using a Chaotic Particle Swarm Optimization algorithm-Smooth Support Vector Machine (CPSO/SSVM) used for urban traffic flow prediction. Urban traffic flow data are highly nonlinear and multiple-dynamic. Performing short-term prediction is very difficult in this kind of dataset. Liu et al. used a support vector machine to smooth the traffic flow nonlinear function and then applied the swarm optimization technique to improve the model’s efficiency.

Liu et al. [29] proposed a hybrid metaheuristic approach using AO and African Vultures Optimization (AVO) to increase the efficiency of the IoT-fog-cloud system. The proposed algorithm AO_AVOA provides the best scheduling solution. Authors have used two datasets, i.e., High-Performance Computing Center North (HPC2N) and NASA Ames iPSC/860 to get comparative results with Firefly Algorithm (FA), Particle Swarm Optimization (PSO), and Harris Hawks Optimization (HHO). In terms of mark-span time, AO_AVOA increases efficiency by 8.36%, 2.61%, 104.38%, 17.56%, and 94.05% as compared to AO, AVOA, PSO, HHO, and FA algorithms, respectively.

Nematollahi et al. [30] proposed an Improved Multi-objective Aquila optimizer (IMAO) for task offloading in an IoT environment. Due to the heavy load on the IoT environment, some of the tasks are transferred to cloud/fog servers. Here, fog computing is used to handle task transfer between IoT sensors and the cloud. IMAO is used to reduce the response time over the network. The average response time and the failure rate are found to be less when compared to other optimizers like PSO, FA, and Ant Colony Optimization (ACO).

Ekinci et al. [31] created an effective vehicle cruise control system using the Aquila Optimizer (AO). Authors enhance the performance of AO by using a chaotic local search strategy. In this study, detailed literature for the implementation of the PID+ controller is given which helps to describe the working of the proposed algorithm CmOBL-AO on the vehicle cruise control system. Comparative results with other metaheuristic optimization methods like particle swarm, grey wolf, and salp swarm show that AO is the best optimizer for a system like vehicle cruise control system.

Alkhalaf et al. [32] used an Aquila Optimizer to diagnose cancer on medical imaging. Authors have proposed an explainable Artificial Intelligence-enabled Cancer Diagnosis (AAOXAI-CD) technique for automatic cancer detection using medical images. The Adaptive Aquila Optimizer
(AAO) performs hyperparametric tuning for the faster SqueezeNet model. AAO enhanced the performance using the ensemble technique with three deep learning classifiers, i.e., Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and Bidirectional Long Short-Term Memory (Bi-LSTM).

Qiao et al. [27] used an improved Aquila optimizer with a temporal convolution network and random forest to predict multistep runoff rainfall. Authors have used a threetep prediction algorithm, i.e., RF-IAO-TCN which is analyzed on the dataset of 5 hydrological stations from 2009 to 2014. Nash efficiency coefficient (NSE), RMSE, MAE, and correlation coefficient (R) are four parameters used to check the performance of RF-IAO-TCN proposed. The proposed hybrid model RF-IAO-TCN outperforms all other stand-alone models and achieves the highest MAE values and the largest R values. Jiang et al. [33] proposed an early warning system called VMD-MMODA-ELM (Variational Mode Decomposition-Modified MultiObjective Dragonfly optimization Algorithm-Extreme Learning Machine) for traffic congestion. To train the model, eight different statistical parameters are used to fit characteristics. The accuracy for this proposed system came to be 97%.

Wang et al. [34] addressed the problem of fault detection in wind turbine by proposing an Adaptive Chaotic Aquila Optimization-based Support Vector Machine (ACAOSVM). The Refined Time-Shifted Multiscale Fuzzy Entropy (RTSMFE) method is used to extract data from the turbine system. Dimensionality of extracted data is reduced by using a supervised isometric approach. After dimensionality reduction, data are given to the SVM model optimized using AAO for fault detection. Wang et al. [35] also proposed the Beetle Antennae Search-based Support Vector Machine (BAS-SVM) method for fault diagnosis in wind turbine. In this method, the Multiscale Permutation Entropy (MPE) method is used to extract data from the turbine system. This system provides accuracy of 100%.

Table 2 shows that different optimization techniques are also used by researchers to enhance the efficiency of the existing state-of-art techniques for prediction.

Most of the previous related works have proposed predictive models for traffic flow or traffic congestion. Some of these proposed works have used accuracy as a matrix to measure the proposed model’s prediction, and some of them have used error rates for the same. The predictive model is considered as best if it has higher accuracy or the lowest error rate. This literature survey helps to conclude that there is a possibility to improve the predictive models for the time-series dataset, if one uses past and future data points to train the model. With the help of proposed work of this research, both linear and nonlinear components can be handled efficiently and improve performance of the predictive model in terms of error rate. Section 4 explains the proposed model in detail.

3. Background

3.1. ARIMA Model. The ARIMA model is based on the idea that values of the past will be used to forecast the future in time-series data. The ARIMA model works on three principles: AutoRegression (AR), Integration (I), and Moving Average (MA) [36].

ARIMA uses “p” as a parameter for AutoRegression (AR). “p” represents the autoregressive order of the model. In the autoregression model, the ARIMA model uses past regressive values in the time-series equation to predict future. It uses the relationship between past and current values to predict future values. “p” helps to decide the number of lags to be used for prediction. In the autoregressive model, output (Yt) depends upon the number of lags taken:

\[ Y_t = a + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + \epsilon_t, \]

where Yt-1 is lag no. 1 of the time series, \( \beta_i \) is the coefficient of lag no. 1, 2, 3, ..., \( p \) which is chosen by the model itself, \( \epsilon_t \) is an error term, and \( a \) is an intercept of the series. It is also chosen by the model.

ARIMA uses “q” as a parameter for Moving Average (MA). “q” represents the moving average order of the model. “q” helps to decide the number of lagged forecast errors to be fed into the ARIMA model. If Yt is considered as an output of the moving average, then it is given as the following equation:

\[ Y_t = a + \epsilon_t + \varnothing_1 \epsilon_{t-1} + \varnothing_2 \epsilon_{t-2} + \ldots + \varnothing_q Y \epsilon_{t-q}, \]

where \( \epsilon_t \) and \( \epsilon_{t-1} \) represent the error terms that came along with the input from the autoregression model.

To implement the ARIMA model, time-series data need to be stationary because the ARIMA model can work well on inputs which are not correlated and are independent of each other. To make time-series data stationary, differencing is one of the best choices. ARIMA uses “d” as a parameter for Integration (I). “d” represents the difference required to make the time-series dataset stationary. This means differentiating time series with the current value of time-series for “d” number of times. After differencing and combining AR and MA models, the final output (Yt) for the ARIMA model can be given as

\[ Y_t = a + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + \epsilon_t + \varnothing_1 \epsilon_{t-1} + \varnothing_2 \epsilon_{t-2} + \ldots + \varnothing_q Y \epsilon_{t-q}. \]

3.2. LSTM Model. The long short-term memory is considered a variant of RNN (Recurrent Neural Network). RNN is a powerful DL model that is sequentially used for prediction. But RNN must deal with a major problem named “vanishing gradient problem.” RNN is not able to retain the information of more than three or four past observations, while LSTM being a modified version of LSTM can retain information from a significant number of past observations.

Figure 1 elaborates on the working of RNN, which consists of three layers, i.e., input layer, hidden layer, and output layer. In the given figure, \( X \{x_0, x_1, x_2, \ldots, x_T \} \) is a set of inputs given into the RNN model and \( h[h_0, h_1, h_2, \ldots, h_T] \) is the output provided by the output layer of RNN. Unit A represents the hidden layer. Input X is passed
to the hidden layer; hidden layers further generate the output. All the hidden layers have the same activation function, weights, and bias. So, to standardize the RNN model, only one hidden layer is looped as often as required.

LSTM is a short-term memory neural network that can retain past information for a longer period [37]. Like RNN, LSTM also has a looped unit. But a looped unit of LSTM is a complex one. The working of LSTM can be described in Figure 2. In this repeating loop, three layers of neural networks are present which interact with each other. The cell works as a memory for the LSTM, while these gates work as traditional neurons that are used in multilayer feed-forward neural networks.

The first one is to forget the gate layer. This layer helps LSTM to decide how much data it should forget and how much data they need to retain. Next is the input gate layer. This layer allows LSTM to determine what information is important enough to pass through and which information it should skip. Third is the output gate layer. This layer helps LSTM to decide which cell state should make it to output.

All the information that is fed into the memory cell is controlled by the input gate ($i_t$). Results generated by the memory cell are tuned by the output gate ($o_t$). The forget gate ($f_t$) uses the previous output of the memory cell and current input to the memory cell, and then, the activation function of the forget gate is applied to it. LSTM helps to develop the linear relationship between memory cells ($C_t$) and its previous value $C_{t-1}$ (Equation (6)). Also, LSTM uses nonlinear activation functions for input gates and output gates:

\[
\begin{align*}
    i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1}), \\
    f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1}), \\
    c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(h(W_{xc}x_t + W_{hc}h_{t-1})), \\
    o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1}), \\
    h_t &= o_t \odot \tanh(c_t),
\end{align*}
\]

where $t$ is the timestep, $x_t$ is the input, $i_t$ is the input gate at $t$ (Equation (4)), $f_t$ is the forget gate at $t$ (Equation (5)), $W$ is the weight matrix, $o_t$ is the output gate at $t$ (Equation (7)), and $h_t$ is the LSTM output (Equation (8)).
Many variants of LSTMs themselves can be used according to the dataset or problem statement given for the solution. Based on time-series forecasting, LSTM can have the following variants.

3.2.1. Univariate LSTM. These models are those models in which a single input feature participates in future prediction:

(1) Vanilla LSTM uses a single input and output layer for prediction

(2) Stacked LSTM uses multiple hidden layers that are stacked on each other

(3) Bidirectional LSTM is trained by using information combined from both forward and backward sequences

3.2.2. Multivariate LSTM. The LSTM model in which there are multiple observations for a single time step is known as multivariate LSTM. Multivariate LSTM can be of two types:

(1) Multiple input series

(2) Multiple parallel series

3.2.3. Multistep LSTM. When a dataset or a problem statement needs to predict multiple time steps for an interval of observation, then multivariate multistep LSTM is used. There are two types of multistep LSTM models:

(1) Vector output model

(2) Encoder-decoder model

3.2.4. Multivariate Multistep LSTM. These models are used for a very complex problem. When one has multiple inputs for observations and needs to predict multiple time steps for the future, then multivariate multistep LSTM is used. There are two types of multivariate multistep LSTM models:

(1) Multiple input multistep output.

(2) Multiple parallel input and multistep output

3.3. Bi-LSTM. Bi-LSTM stands for the Bidirectional Long Short-Term model. The primary drawback of LSTM is that it solely uses historical or prior data values to create predictions. Future observations are irrelevant when using LSTM to make decisions. Bi-LSTM uses both past and future observations to improve the anticipated results. Data in the Bi-LSTM model can go both forward and backwards. This facilitates use of both historical and future data values to train the prediction model and choose the best course of action [38]. The bidirectional LSTM architecture is shown in Figure 3.

Output of Bi-LSTM is represented as \( y(y_0, y_1, y_2, \ldots, y_t) \). Dataflow in the forward direction is represented as \( \overrightarrow{h} \{ h_0, h_1, h_2, \ldots, h_t \} \), and in the backward direction, it is represented as \( \overleftarrow{h} \{ h_0, h_1, h_2, \ldots, h_t \} \). Dataflow in the Bi-LSTM model is given in the following equations:

\[
\overrightarrow{h}_t = \sigma \left( W_\rightarrow x_t + W_\rightarrow \overrightarrow{h}_{t-1} + b_\rightarrow \right),
\]

\[
\overleftarrow{h}_t = \sigma \left( W_\leftarrow \overleftarrow{h}_{t-1} + W_\leftarrow h_t + b_\leftarrow \right),
\]

\[
y_t = W_y \overrightarrow{h} + W_y \overleftarrow{h} + b_y,
\]

where \( \sigma \) and \( b \) are the logistic sigmoid and bias function, respectively, and \( t \) is a time stamp.

3.4. Aquila Optimizer (AO). Aquila optimizer is a meta-heuristic, population-based optimization technique. It is a nature-inspired approach that is based on the hunting method of Aquila bird. Aquila opts for four kinds of hunting strategies to hunt its prey [39]:

(i) When Aquila flies at a high level to the ground, it uses a vertical stoop for hunting.

(ii) When Aquila flies at a low level to the ground, it uses a short glide attack with contour flight for hunting. It is the most used method of hunting for Aquila.

(iii) Aquila uses a low flight slow attack method to invade steadily on prey's neck. In this method, Aquila lands on ground and attacks the prey when it has zero escape response.

(iv) Aquila uses a walking and grabbing technique to get the prey by walking on land and pulling it. This kind of technique is used to attack young and big size prey.

Like every optimizer, AO also focuses on exploration and exploitation of search space to get optimized results. The flowchart of AO is given in Figure 4.

To avoid trapping at local optima, the complete AO algorithm is divided into four sections as that for attacking methods of Aquila. The steps used in the Aquila optimizer are given as follows:

Step 1: Initialize the population \( P \), using

\[
P = \begin{bmatrix} P_{1,1} & \cdots & P_{1,j} & \cdots & P_{1,D-1} & P_{1,D} \\ P_{2,1} & \cdots & P_{2,j} & \cdots & P_{2,D-1} & P_{2,D} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{N-1,1} & \cdots & P_{N-1,j} & \cdots & P_{N-1,D-1} & P_{N-1,D} \\ P_{N,1} & \cdots & P_{N,j} & \cdots & P_{N,D-1} & P_{N,D} \end{bmatrix}
\]

\( P \) represents the initial search space of candidate solutions out of which an optimal result is to be found. These candidate solutions are produced using random variable \( r \) between the upper bound (U) and the lower bound (L) of the problem:
where $P_i$ is a outcome of the $i^{th}$ solution, $r$ is a random variable, $L_j$ is the $j^{th}$ lower bound, $N$ is total population, and $D$ is the number of dimensions of the problem.

Step 2: Initialize basic parameters of AO, i.e., $\alpha$, $\delta$.

Step 3: Calculate fitness values of all the candidate solutions using the fitness function.

Step 4: Find out the best value among the candidate solutions which will be named as $P_{\text{best}}(t)$.

Step 5: Calculate and update the mean values of the current best solution $P_m(t)$ as given in the following equation:

$$P_m(t) = \frac{1}{N} \sum_{j=1}^{D} P_j(t), \forall j = 1, 2, \ldots, D.$$  \hspace{1cm} (14)

Step 6: Calculate and update $x$, $y$, $M_1$, $M_2$, and $\text{levy} (D_j)$. 

**Figure 3: Bi-LSTM model (unconventional).**

**Figure 4: Flowchart of AO.**
\(x\) and \(y\) are the dimensional points which help to create a spiral shape used for searching the optimal solutions and can be calculated as
\[
x = n' \sin(\theta),
\]
\[
y = n' \cos(\theta),
\]
where \(n' = n_1 + R \times D_1, \forall n_1 = 1 \text{ to } 20, R = 0.00565, D_1 = 1 \text{ to } D\).

\(M_1\) represents different motions of AO used to track the prey/target solution and can be calculated as given in (17). \(M_2\) represents the slop used by AO to find the prey/target solution from the first location (1) to the last location (\(t\)). It is a decreasing value from 2 to 0 and is given by (18). Levy (\(D_s\)) is a flight distribution function of AO in dimensional space \(D_1\) and is given by (19):
\[
M_1 = 2 \times r - 1,
\]
\[
M_2 = 2 \times \left(1 - \frac{t}{T}\right),
\]
\[
\text{Levy}(D_s) = s \times \frac{u \times \sigma}{|v|^T/2},
\]
where \(u\) and \(\sigma\) are random numbers between 0 and 1.

Step 7: Check the condition \(t \leq (2/3) \times T\).
(Where \(t\) and \(T\) represent the current iteration and maximum number of iterations, respectively)
if true, then there is a need to explore of search space (in Step 6)
else, there is a need to exploit the search space (Step 7)

Step 8: EXPLORATION of search space.
Exploration part is divided into two parts, i.e., expanded exploration \((P_1)\) and narrowed exploration \((P_2)\).
Check condition \((r \leq 0.5)\)
If true, then step 8(a)
Else, then step 8(b)

Step 8 (a): Expanded exploration \((P_1)\)
It is regarded as a vertical stoop for the hunting method of Aquila. In this method, the AO survey from high soars to calculate the search space where the target solution is expected to be present. Expanded exploration can be calculated as given by
\[
P_1(t + 1) = P_{\text{best}}(t) \times \left(1 - \frac{t}{T}\right) + (X_M(t) - X_{\text{best}}(t) \times r).
\]
Update \(P_{\text{best}}\) and \(P(t)\).

Step 8 (b): Narrowed exploration \((P_2)\)
It is regarded as a short glide attack with the contour flight hunting method of Aquila. In this method, AO narrowly explores the selected search space where the target solution is likely to be found. Narrowed exploration is given by
\[
P_2(t + 1) = P_{\text{best}}(t) \times \text{Levy}(D_s) + P_x(t) + (y - x) \times r,
\]
where, \(P_x(t)\) is a random solution at \(t\)th iteration.
Update \(P_{\text{best}}\) and \(P(t)\).

Step 9: EXPLOITATION of search space.
Exploration part is divided into two parts, i.e., expanded exploitation \((P_3)\) and narrowed exploitation \((P_4)\).
Check condition \((r \leq 0.5)\)
If true, then step 9 (a)
Else, then step 9 (b)

Step 9 (a): Expanded exploitation \((P_3)\)
It is regarded as a low flight and slow descend hunting method of Aquila. In this method, AO exploits the selected search space of the target solution to reduce the distance as minimum as possible and then attack that target solution. Expanded exploitation can be calculated as given by
\[
P_3(t + 1) = (P_{\text{best}}(t) - X_M(t)) \times \alpha - r + (U - L) \times r + L) \times \delta.
\]
Update \(P_{\text{best}}\) and \(P(t)\).

Step 9 (b): Narrowed exploitation \((P_4)\)
It is regarded as a walk and grab-hunting method of Aquila. In this method, AO takes the target solution from the last location. Narrowed exploitation is given by
\[
P_4(t + 1) = Q_f \times P_{\text{best}}(t) - (M_1 \times P(t) \times r - M_2 \times \text{Levy}(D_s) + r \times M_1),
\]
where, \(Q_f(t)\) is a quality function value at \(t\)th iteration.
Update \(P_{\text{best}}\) and \(P(t)\).

Step 10: Gives out the best target solution \((P_{\text{best}})\).

4. AB_AO and Implementation

In this paper, a hybrid model AB_AO is proposed using ARIMA and Bi-LSTM. The ARIMA model uses a linear component of time series for future prediction. But dataset does not only have a linear relationship among data items that are used for prediction. Here comes the part of LSTM. The residuals of the ARIMA model are then fed to the LSTM model. The LSTM model is very well capable of handling nonlinear components of the time series which can handle residuals of the ARIMA model and then further improve prediction results. For this model, the univariate Bi-LSTM model is used for prediction because it uses both previous and next observations to predict new observations. The working model is explained with the help of Figure 5. The proposed model is only implemented for a short term, and
The prediction time horizon is 10 mins. The proposed AB_AO model used for vehicle count prediction follows the following steps:

1. The dataset is preprocessed and analyzed for different seasonality and trends.
2. To handle the linear component of the time-series dataset, the ARIMA model is used and generates forecasting values.
3. Nonlinear components are generated as residuals of the ARIMA model.
4. These residual values are fed into the optimized Bi-LSTM model using Aquila optimizer. While applying AO, the objective function is set as
   \[
   \text{Obj} = \min(\text{Err}).
   \]
   Where Err refers to the error that occurs due to the predictive model and Obj is the objective function passed to the predictive model to minimize errors.
5. After applying optimized Bi-LSTM on the ARIMA residuals, forecasting values of Bi-LSTM are generated.
6. Forecasting values of the ARIMA model and optimized Bi-LSTM model are then combined to get better results.

5. Experiment and Results

This section gives a detail explanation on the dataset and the results of the proposed model. It also discusses the performance of the proposed model using the ablation study and comparative analysis with other models.

5.1. Dataset Description. This study has used an openly available big IoT dataset named "CityPulse" [40]. This is a semantically annotated dataset collected to analyze smart cities like Aarhus (Denmark) and Brasov (Romania). The CityPulse dataset contains different data related to the smart city like road traffic data, weather data, parking data, and pollution data. This dataset is used to evaluate smart-city frameworks. This dataset contains road traffic data collected through sensors at different locations, and these sensors interact with each other via IoT technology [41]. The road traffic dataset used for this study is collected over time in four different durations, of which following three datasets are “trafficData190501,” “trafficData190126,” and “trafficData158324.” Different datasets used in the study are as follows:

(i) Dataset 1: “trafficData190501” is a dataset of road traffic in city named Ega between street Grenåvej 440 and street Grenåvej 481. The distance between two points is around 838 meters.

(ii) Dataset 2: “trafficData190126” is a dataset of road traffic in city named Aarhus between street Spanien 63 and street Norreport 18. The distance between two points is around 1508 meters.

(iii) Dataset 3: “trafficData158324” is a dataset of road traffic in city named Hinnerup between street Arhusvej 72 and street Arhusvej 72. The distance between two points is around 1030 meters.

This analysis uses traffic data collected from February 2014 to June 2014. The given dataset consists of the following features explained in Table 3. Dataset visualization is also provided in Figure 6. Figures 6(a)–6(c) show seasonality, trend, and residuals of dataset 1, dataset 2, and dataset 3, respectively.

5.2. Ablation Study. The ablation study is performed on dataset 1 to explain the prediction performance of the AB_AO model. The ablation experiment is performed with the following models:

1. Standalone ARIMA model: it is an ARIMA model without any other model’s attachment
2. Standalone Bi-LSTM model: it is a Bi-LSTM model without any other model’s attachment
3. Hybrid ARIMA-Bi-LSTM without any optimizer

The experiment results are shown in Table 4. Experimental results have shown that the AB_AO model has outperformed all the comparative models. He and Bai [42] have proposed a neural network prediction model using the “CityPulse” dataset. The dataset contains data between two smart cities, i.e., Aarhus and Denmark. Different predictive methods like ANN, linear regression, and Kalman filter are applied to predict parking occupancy using traffic data between two smart cities. MAE is 3.866 for ANN and 5.03 for linear regression. The AB_AO model performs better than both ANN and linear regression.

Based on dataset 1, it is concluded that the ARIMA model could not handle nonlinear features of data, and to handle nonlinearity, the Bi-LSTM model is used which also could not perform much better than ARIMA. The value of MSE, MAE, and MAPE for the ARIMA model came out to be 23.86, 4.63, and 0.36, and for the Bi-LSTM model, it came...
<table>
<thead>
<tr>
<th>S. no.</th>
<th>Feature</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Status</td>
<td>This feature gives the status report of the IoT sensor device used</td>
</tr>
<tr>
<td>2</td>
<td>avgMeasuredTime</td>
<td>It gives average measured time</td>
</tr>
<tr>
<td>3</td>
<td>avgSpeed</td>
<td>It gives the average speed of a vehicle for the given time</td>
</tr>
<tr>
<td>4</td>
<td>medianMeasuredtime</td>
<td>It provides a median of the time for the sensor measuring speed and count</td>
</tr>
<tr>
<td>5</td>
<td>TIMESTAMP</td>
<td>This feature gives the exact timestamp for which the vehicle counts and the average speed of the vehicle</td>
</tr>
<tr>
<td>6</td>
<td>vehicleCount</td>
<td>It gives the number of vehicles present on the street for a particular time interval</td>
</tr>
<tr>
<td>7</td>
<td>_id</td>
<td>It gives the ID of a sensor device</td>
</tr>
<tr>
<td>8</td>
<td>Report_id</td>
<td>Report id gives the id for a particular area location for which all the above features are being observed</td>
</tr>
</tbody>
</table>

**Figure 6:** Continued.
out to be 25.62, 4.2, and 0.31, respectively. So, to cover both linear and nonlinear features of the dataset, ARIMA and Bi-LSTM are hybridized. As shown in Table 4, AB_AO, an optimized hybridization, has enhanced the efficiency significantly. The proposed AB_AO model lowers the error values, with 18.74, 3.18, and 0.21 values as the mean square error and mean absolute error, respectively.

5.3. Results and Discussion. In this section, experimental flow with detailed explanation of results is given. Graphical results are given for all the datasets used in this study; however, a detailed explanation of these results is provided for dataset 1 only. The first step is to split the dataset between training, validating, and testing datasets.

After analyzing the dataset based on their seasonality, trend, and residuals, an ML model called ARIMA is considered to forecast future values. To get the best values of $p$, $q$, and $d$ for the ARIMA model, the “auto_arima” package is used. “auto_arima” is a package that helps to calculate the best fit ARIMA model concerning the data frame provided. One can fix the maximum and minimum values for ($p$, $d$, $q$) within which the auto_arima package can search for the best-fitted model. It works on the minimum AIC (Akaike information criterion) factor. This means that, after analyzing different combinations of ($p$, $d$, $q$), it will select that the model which will have the lowest AIC value. For this study, the ARIMA model with order (2, 1, 2) is selected with an AIC value of 521.226. Figure 7 presents the detailed diagnosis of the ARIMA (2, 1, 2) model for dataset 1.

From Figure 7, it is concluded that standard residual errors are fluctuating around the mean of zero for dataset 1. A histogram is a density plot that depicts that normal distribution with mean is centered. In the normal Q-Q plot, most of the blue dots fall on the red line. Some of the points are found to be skewed. Next is a correlogram which is also known as the ACF plot. This plot shows that residual errors are not correlated, that is why other values of $X$ are tried to get better results.

Figure 8 shows a plot between forecasted values of the ARIMA model and actual values of dataset 1, dataset 2, and dataset 3. Figure 9 depicts the residual values of the ARIMA model which are forwarded to the LSTM model for dataset 1, dataset 2, and dataset 3. After scaling the residual values using the min-max scaler function, values are given to the Bi-LSTM model. To implement the Bi-LSTM model, different packages of “Keras” packages are used. “Keras” is a python-based ML platform developed on “TensorFlow.” Layers like dense, LSTM, Bi-LSTM, and dropout are imported from “Keras.” A stacked Bi-LSTM sequential model is developed using imported packages.

For hyperparametric tuning of the Bi-LSTM model, AO is used which helps to fit the model with 99.0001 fitness value. After applying AO, hyperparameters are set as given in Table 5.

The proposed AB_AO model has two layers of Bi-LSTM, a dropout layer, and two dense layers. The first layer of the Bi-LSTM model uses “sigmoid” as a nonlinear activation function and 256 inputs. The second layer of the Bi-LSTM

<table>
<thead>
<tr>
<th>S. no.</th>
<th>Models</th>
<th>MSE</th>
<th>MAE</th>
<th>MAPE</th>
<th>Computational time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ARIMA</td>
<td>23.86</td>
<td>4.63</td>
<td>0.36</td>
<td>0.971</td>
</tr>
<tr>
<td>2</td>
<td>Bi-LSTM</td>
<td>25.62</td>
<td>4.2</td>
<td>0.31</td>
<td>33.87</td>
</tr>
<tr>
<td>3</td>
<td>ARIMA + LSTM</td>
<td>20.89</td>
<td>3.98</td>
<td>0.39</td>
<td>34.785</td>
</tr>
<tr>
<td>4</td>
<td>Proposed AB_AO model</td>
<td>18.74</td>
<td>3.18</td>
<td>0.21</td>
<td>35.24154</td>
</tr>
</tbody>
</table>
Figure 7: Diagnosis of the ARIMA model for dataset 1.

Figure 8: Continued.
Figure 8: Forecasted v/s actual values of the ARIMA model for (a) dataset 1; (b) dataset 2; (c) dataset 3.

Figure 9: Continued.
model uses “tanh” as a nonlinear activation function and 64 inputs. The third layer is a dropout layer which drops down neurons by 0.2 factors. The fourth layer is the dense layer, which gives 32 neuron values as output. The fifth layer is also a dense layer of the Bi-LSTM model, which gives only one neuron output that tends to be the predicted value. Weights of the dense layers are optimized using GlorotNormal() function.

ADAMAX optimizer [39] with a learning rate of 0.01 is used in the compilation of the Bi-LSTM model to minimize losses for dataset 1. To check on losses, Bi-LSTM is using the mean squared error as a loss parameter. Bi-LSTM uses 100 epochs to fit the model.

Figure 10 presents the loss values of training and validation dataset 1, dataset 2, and dataset 3 while applying Bi-LSTM. Loss counts of both training data and validation data came out to be very less for different epochs. This shows that the hybrid algorithm gives good results.

Figure 11 plots the predicted and actual values of the proposed AB_AO model for dataset 1, dataset 2, and dataset 3.

5.4. Performance Analysis. For comparative analysis of different models used in this study, two parameters are used:

(i) Mean Squared Error (MSE): MSE is regarded as an ideal metric for assessing the model’s error rate. A model is considered better when its MSE value is lower. The mathematical expression for MSE is given as follows:

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

where $n$ denotes number of data items, $Y_i$ denotes actual values, and $\hat{Y}_i$ represents the predicted value.

(ii) Mean Absolute Error (MAE): MAE is a variation between the expected and actual values in a paired observation. These paired observations are the ones causing the same phenomena which leads to error. The mathematical expression for MAE is given as follows:

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Activation function</th>
<th>Optimizer</th>
<th>Weight initializer</th>
<th>Learning rate</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>Tanh</td>
<td>Adamax</td>
<td>GlorotNormal()</td>
<td>0.01</td>
<td>100</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>Hard_sigmoid</td>
<td>Adam</td>
<td>GlorotNormal()</td>
<td>0.01</td>
<td>50</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>Sigmoid</td>
<td>Adam</td>
<td>GlorotNormal()</td>
<td>0.02</td>
<td>40</td>
</tr>
</tbody>
</table>
Figure 10: Training v/s validation losses for (a) dataset 1; (b) dataset 2; (c) dataset 3.

Figure 11: Continued.
where $n$ denotes the number of data items, $\tilde{Y}_i$ denotes the predicted values, and $Y_i$ represents actual correct values.

(iii) Mean Absolute Percentage Error (MAPE): MAPE is used to find out the relative errors. It is used to calculate relative percentage errors of predicted values concerning the actual values. The mathematical expression for MAPE is given as follows:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|Y_i - \tilde{Y}_i|}{\max E_i |Y_i|}$$

where $E$ is a small positive number, $n$ is the number of samples, $Y_i$ denotes actual values, and $\tilde{Y}_i$ represents predicted values.

5.5. Comparative Analysis. Table 6 shows the results of the AB_AO model in three datasets used, i.e., dataset 1, dataset 2, and dataset 3.

Table 7 provides a comparative analysis with other hybrid models. The results of the proposed model are promising as compared to those of existing hybrid models. In the study [17], ensemble empirical mode decomposition and ANN models are used to create a hybrid model named EEMD + ANN for traffic prediction. After experimental results, it is found that EEMD + ANN gave MAE as 3.93 and MAPE as 6.72. The study [36] used the ARIMA model and ANN model for the stock prediction time-series dataset which gave MSE as 94.75 and MAE as 303,777.34.

Lui et al. [43] proposed a model named TSARGCN. This model slices the dataset to handle time series and adds a new residual network to maintain the integrity of information transfer. The dynamic time wrapping (DTW) algorithm is used to find traffic flow sequences. The performance of this model is calculated using MAE and RMSE which came out to be 19.24 and 27.04, respectively. Though the model performed better than Conv-LSTM, the proposed AB_AO model is much better as shown in Table 7. Prajam et al. [45] studied different machine learning algorithms for traffic forecasting. Data were collected from private ISP located at centers in 11 European cities for this study. ANN, LSTM, and ARIMA models are applied to the dataset to forecast future traffic. MAE and MSE came out to be $2.89 \times 10^9$ and $1.54 \times 10^{19}$ for the ARIMA model, $2.93 \times 10^9$ and $1.54 \times 10^{19}$ for the LSTM model, and $2.94 \times 10^9$ and $1.53 \times 10^{19}$ for ANN. The AB_AO model outperforms all these models.

Xue et al. [44] used a bidirectional LSTM deep learning model for traffic flow prediction. Data from Guangwu Toll Station, Zhengzhou city, China, are used for the study. This study concluded that RNN-based deep learning models are better than statistical models like ARIMA. MAE for ARIMA, Bi-LSTM, LSTM, and RNN came out to be 18.34, 7.345, 7.986, and 9.531, respectively. MAE for AB_AO model is 3.3 which is far better than any of the RNN models used in the study.

\begin{table}[h]
\begin{center}
\caption{AB_AO performance.}
\resizebox{\textwidth}{!}{
\begin{tabular}{|l|c|c|c|}
\hline
Performance matrices & MSE & MAE & MAPE \\
\hline
Dataset 1 & 18.74 & 3.18 & 0.21 \\
Dataset 2 & 3.83 & 1.79 & 0.87 \\
Dataset 3 & 14.14 & 3.53 & 0.83 \\
\hline
\end{tabular}
}
\end{center}
\end{table}

\begin{table}[h]
\begin{center}
\caption{Comparative analysis of models.}
\resizebox{\textwidth}{!}{
\begin{tabular}{|l|c|c|c|c|}
\hline
S. no. & Models & MSE & MAE & MAPE \\
\hline
1 & EEMD + ANN [17] & — & 4.90 & 6.72 \\
2 & Parallel hybrid model [36] & 94.75 & 303,777.34 & — \\
3 & ANN [42] & — & 3.866 & — \\
4 & Linear regression [42] & — & 5.03 & — \\
5 & TSARGCN [43] & — & 19.24 & — \\
6 & LSTM [44] & $2.93 \times 10^9$ & $1.54 \times 10^{19}$ & 3.15 \\
7 & RNN [45] & — & 9.531 & 7.56 \\
8 & Proposed AB_AO model & 18.74 & 3.18 & 0.21 \\
\hline
\end{tabular}
}
\end{center}
\end{table}
Lilhore et al. [46] gave a framework for an intelligent transport management system. The authors presented Adaptive Traffic-Management system (ATM) to monitor vehicles, infrastructure, and events in a smart city. An unsupervised machine learning algorithm named DBSCAN is used to detect accidental prone zones. ATM uses the vehicle image processing module to detect motorized vehicles. The MATLAB traffic simulator is used to simulate and detect the anomalies of accidents. While the proposed method is used to predict traffic congestion. The current study uses effective ML models like ARIMA and Bi-LSTM to handle time-series datasets with its linear and nonlinear feature to enhance the predictive results.

6. Conclusion

This study is focused on developing a new hybrid approach that can improve forecasting results. This study has used the smart city big IoT dataset to forecast the vehicle count. It has used the ARIMA model to handle linear components of time-series datasets. To handle nonlinear components of a time-series dataset, LSTM is used. This paper provided a detailed overview of the LSTM model and the kinds of LSTM models that help forecast time-series data. The proposed model AB_AO is used for the prediction of the vehicle count in three different time-series datasets. It first uses the ARIMA model to manage linear features of the time series data and forecast the results and then passes its residual values to the Bi-LSTM model. The Bi-LSTM model now uses nonlinear features of the time-series dataset. Hyperparametric tuning of the Bi-LSTM model is performed using AO. The AB_AO model further predicts future values after scaling the residuals. Predicted values of both ARIMA and optimized Bi-LSTM are then combined which provides better results than these models. During comparative analysis of all the models, the AB_AO model came out to be the best with an MSE as 18.78, MAE as 3.18, and MAPE as 0.21. The AB_AO model can be easily used to reduce the traffic on roads to avoid traffic congestion. This in turn can help in saving a lot of precious time for daily travelers.

The limitation of the AB_AO model is that hybridization of statistical and deep learning model affects the computation time and speed as compared to the state-of-art models. In future, detailed interpretation analysis using various recent approaches such as LIME and SHAP will also be used to justify the results obtained. The future scope of this model is that it can be used in other IoT domain areas like smart parking systems, crop prediction, weather prediction, and disease prediction. The proposed model will also be used at a large scale with the help of federated learning. Traffic data collected by sensors at different geographical locations can be handled and analyzed easily with the help of federated learning.

Data Availability

Data are available on request from the submitting author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References


[46] U. K. Lilhore, A. L. Imoize, C.-T. Li et al., “Design and implementation of an ML and IoT based adaptive traffic-
