FMDB *Transactions on Sustainable Intelligent Networks*

Optimizing Traffic Prediction in Continuous Data Stream in Vehicular Networks with Triplet Transfer Learning and Dholes Hunting-Based Optimization

S. Venkatasubramanian[1,*], A. Abirami[2], R. Kannan[3], S. Senthil[4]

[1,2,3,4]Department of Computer Science and Business Systems, Saranathan College of Engineering, Trichy, Tamil Nadu, India.
veeyes@saranathan.ac.in[1], abirami7123@saranathan.ac.in[2], kannan7137@saranathan.ac.in[3], senthil7111@saranathan.ac.in[4]

Abstract: Traffic Congestion Detection Services (TCDSs) in network environments that use continuous data streams receive much information to detect and update road segments at low speeds and high vehicle density. Traditional traffic monitoring is laborious and inefficient. The biggest concern with manual monitoring is traffic controller security. Thus, in VANET (Vehicular Ad hoc Network), predicting traffic and monitoring congestion are essential to reduce delays and accidents. Numerous categorisation and prediction algorithms provide clear vehicle forecasts and collision-free network pathways. Traditional approaches have struggled to forecast path pictures accurately. Lack of prediction precision and sluggish processing speed prevent acceptable travel route decisions. We create a triplet transfer learning network (TTLN) to detect traffic congestion using input data to address these challenges. VGGNet will analyse input data for deep and statistical properties. A pre-trained triplet model represents three dimensions as an upstream transfer learning architecture task. Rebuilding the pre-trained model with the triplet model, temporal model, and auxiliary layer as the downstream job is the final step. Weights are fine-tuned to indicate traffic congestion. The suggested model's classification accuracy is fine-tuned using Dholes Hunting-Based Optimisation (DHO). To evaluate our model against the most popular traffic prediction algorithms utilising categorisation criteria. The simulation results show that the suggested model exceeds the single method in time and prediction stability.

Keywords: Vehicular Adhoc Network; Traffic Congestion Detection Service; Triplet Transfer Learning Network; Dholes Hunting-Based Optimization; Artificial Intelligence (AI).

Received on: 02/04/2024, **Revised on:** 11/06/2024, **Accepted on:** 02/08/2024, **Published on:** 09/09/2024

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSIN

DOI: https://doi.org/10.69888/FTSIN.2024.000286

Cite as: S. Venkatasubramanian, A. Abirami, R. Kannan, and S. Senthil, "Optimizing Traffic Prediction in Continuous Data Stream in Vehicular Networks with Triplet Transfer Learning and Dholes Hunting-Based Optimization," *FMDB Transactions on Sustainable Intelligent Networks*., vol.1, no.3, pp. 165–177, 2024.

1. Introduction

In the age of computer science is artificial intelligence (AI). Artificial intelligence has extended since its inception 50 years ago, with notable advancements in data mining, computer vision, expert organizations, robotics, natural language processing, and machine learning [1]. Of all the AI subfields, machine learning has the most followers. Probabilistic models, deep learning, ANNs, and game theory are other branches of artificial intelligence [2]. Numerous industries contribute to the creation and implementation of these classes. Particularly in predicting when traffic will become heavy, it has recently emerged as the primary focus of transportation engineering studies. Economic growth and public health are two areas that feel the effects of

[*]Corresponding author.

traffic congestion both immediately and over time [3]. Multiple researchers say the opportunity cost and fuel consumption attributed to traffic congestion is Rs. 1 million per day [4]. Individuals are also affected by traffic congestion. Traffic contributes to increased pollution and global warming and causes significant time losses, particularly during peak hours, and psychological stress [5].

A country can't progress without efficient traffic flow, which is essential for economic growth and road users' comfort [6]. Authorities are increasingly focussing on traffic congestion monitoring due to advancements in the transportation sector that involve collecting traffic data [7]. With traffic congestion prediction, authorities can allocate resources in advance to ensure travellers have a smooth journey. Applying various AI approaches to acquired traffic data, a traffic congestion prediction challenge aims to estimate parameters linked to future traffic within the next fifteen minutes to a few hours [8]. When keeping an eye on and trying to forecast traffic jams, there are typically five metrics to look at: volume, density, occupancy, congestion index, and trip time. Several AI methods are utilized to assess the congestion parameters, each tailored to the specifics of the acquired data [9]. According to statistics, 3,500 car accidents occur daily on a global scale. Deaths caused by traffic accidents are highest in India, according to the United Nations. Tragically, the drivers were not adequately informed. Crossroads are hotspots for vehicle collisions. Overcrowding occurs on urban roadways when their capacity is exceeded. Problems with traffic in smart cities can lead to air pollution, fuel waste, infractions, noise, accidents, and lost time [10]. Emergency vehicles are delayed as traffic waits for other routes to clear. Congestion is exacerbated by fixed signal timings when there is a strong one-way traffic flow. With the rise of the smart town, intelligent transport has expanded.

Furthermore, smart villages manage traffic [11]. ITS was developed to alleviate traffic congestion and improve traffic management through wireless communication technology. With the increasing number of accidents, bottlenecks, and jams caused by vehicle traffic, ITSs are gaining popularity [12]. ITSs upgrade network management and safety by integrating transport networks, vehicles, and users with ICTs. Weather, traffic regulation, and efficient planning are all provided by ITSs in real time [13]. ITS's primary goals—reducing traffic congestion, preventing accidents, and making the most efficient use of infrastructure—can be realized with VANETs or Vehicular Ad Hoc Networks. The use of VANETs is on the rise in control frameworks and traffic management [14]. By facilitating communication between vehicle networks and smart city districts, VANETs aid in traffic reduction and reduce obstruction, accidents, malfeasance, and cost overhead [15]. Vehicle-to-earth Networks (VANETs) enable context-free communication between vehicles and displacement control structures in local and distant networks. VANETs have an emphasis on safety, amusement, and ecological preservation. Unlike traditional networks, VANETs do not rely on a central server or infrastructure to transmit data between vehicle nodes [16]. To enhance network efficiency and traffic monitoring, these networks optimize data communication protocols and the transmission efficiency of wireless communication. This study develops a deep-learning prototype for predicting traffic congestion based on feature extraction. After feature extraction with the VGGNet model, the TTLN model is constructed and fine-tuned using the DHO algorithm. to run our studies on a publically available dataset that measures several things. Here is the breakdown of the paper: In Section 2, the relevant literature is reviewed; in Section 3, the technique is proposed; in Section 4, the consequences are discussed; and in Section 5, the conclusion is drawn.

2. Related Works

By evaluating various models, Tsalikidis et al. [17] developed a multi-step forecasting approach to enhance traffic congestion prediction, especially in regions with little historical data. To evaluate and select a set of interpretable predictive algorithms to manage urban traffic flow complexity and spatiotemporal features. At each stage, the forecasting method chooses the best model to ensure the highest level of accuracy. Results show that Light Gradient Boosting Machines (LGBMs) and other variating Ensemble Tree-Based (ETB) regressors outperform conventional Deep Learning (DL) approaches in a 24-hour step prediction. More effective daily urban transport scheduling is possible thanks to our study, which also significantly contributes to short-term projections of traffic congestion.

To directly predict traffic congestion, Kumar et al. [18] suggest a multilayered deep neural network (MLDNN) besides a congestion index (CI) that is based on the traffic density component. To evaluate the suggested model, data was collected from a specific site in Delhi, where video cameras were utilized during peak hours of weekdays (Monday through Sunday). The data was organized in a matrix style at five-minute intervals. The input matrix was divided into several intervals to train, validate, and test the MLDNN and baseline models—including support vector regression, multilayer perceptron, gated recurrent unit, long short-term memory, and convolutional neural network (CNN). This study's results demonstrate the viability of using MLDNN and the proposed CI for heterogeneous traffic congestion prediction.

The congestion index is a novel input characteristic suggested by Fang et al. [19]. Through this method, the intricate relationships between traffic conditions and fluctuations in particulate matter (PM2.5) concentrations can be better understood. A multiscale input model will be applied, and an ablation index will be performed to evaluate the efficacy of this approach.

The approach shows a remarkable performance increase compared to conventional models; on average, it decreases all benchmark models' root mean square error (RMSE) by 6.07 percent, and the top-performing classical reduces it by 12.06 percent. The fact that the RMSE at peak hours is still less than 9.83 μg/m3 demonstrates that the technique successfully tackles pollution hotspots, even when high traffic emissions. This work offers fresh perspectives on enhancing the quality of urban environments and public health, and it is expected to stimulate additional research in this area.

Known as BiSPNet, it was created by Kumar and Kumar [20]. From BiSPNet, two heads emerge. The two-dimensional convolutional long and three-dimensional convolution (Conv3D) layers were applied to each data from the brain. The dense layer then combines the two outputs from the heads. Because of these layers, the prototypical can consider the features of numerous inputs. The ground truth data acquired by the video cameras placed on the roads in Delhi, India, verify the suggested model. Experimental results demonstrate that the prediction of the BiSPNet model outperforms that of numerous state-of-the-art benchmark models. For instance, when comparing the BiSPNet to the MDLNet, the former achieves a mean absolute error reduction of 53.23% and 40.95%, besides 0.38% in the prediction of congestion scores for the 5, 10, and 15-minute time frames.

Using coordinated vehicle putting data gathered from fixed detectors, Li et al. [21] propose a new model for estimating besides predicting the congestion. Using data filtered by the Savitzy-Golay, a shockwave-based approach is developed to simulate the time- and space-dependent propagation trajectory of congestion. Bayesian ridge regression is used to find the range of probabilities for the congestion's propagation path. Wi-Fi location data from a segment of the Beijing-Kunming Freeway in China was used to evaluate the suggested strategy. Field traffic conditions calculated from loop data were compared to our method's results. The results reveal that the projected congestion start time and the observed time on the designated road stretch varied by about 200 seconds. There is a discrepancy of about 100s between the anticipated and actual start times of congestion.

To better understand how traffic patterns change in lane-less situations, Kumar et al. [22] presented a model called Traffic State Anticipation Network (TSANet). This model can predict future traffic states by examining arrangements of current traffic graphs. Furthermore, EyeonTraffic (EoT), a massive lane-less traffic dataset comprising three hours of aerial footage shot at three busy intersections in India, is presented. The effectiveness of our projected TSANet in accurately predicting traffic conditions across various spatial locations within an intersection is demonstrated by experimental findings on the EoT dataset. Further, demonstrates that TSANet performs adequately when applied to new types of intersections, which increases its usefulness for traffic scenario analysis since it doesn't require explicit training.

3. Proposed Methodology

In this section, the traffic congestion forecast model is designed by developing an advanced deep learning network graphically shown in Figure 1, and each of its blocks is discussed here.

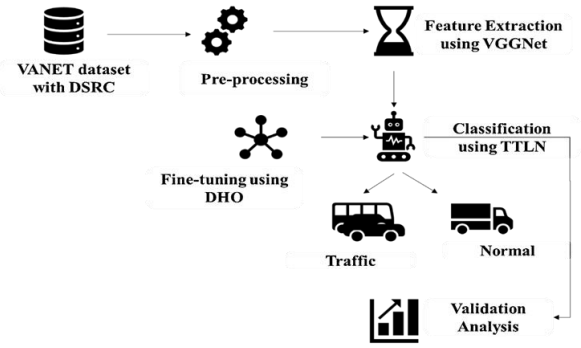

Figure 1: Workflow of the Research Work

3.1. Dataset Description

Researchers utilized a real-world VANET dataset that included DSRC-based communications in a realistic highway environment between automobiles and roadside equipment [23]. The trials occurred on I-75 between Exits 250 and 255 northwest of Atlanta, Georgia. Between hours 2 and 5, the chosen stretch of highway had five conventional lanes and one lane reserved for high-occupancy vehicles (HOVs). Most highways in American cities look like this [24]. The 822.11 ad hoc networks used GPS to get the data. Every two seconds, the GPS would report the vehicles' location, longitude, latitude, speed, and heading, among other features. The position data obtained by interpolation was accurate to about five to seven meters.

In addition, IPerf worked in tandem with GPS to read network parameters. The transmitter and receiver are positioned in adjacent lanes to test vehicle-to-vehicle communication in the following vehicles. On a bridge of varying heights, the RSU station was the receiver for the V2R communication measurements taken while vehicles were in motion [25]. The vehicle's sender remained in the rightmost lane and broadcasted the packets as it moved. At about 150 packets per second, the senders broadcast 1470 bytes in V2R communication. The following communication characteristics were recorded: "log time", "location information of both sender and receiver", "velocity", "packet sent/received," and "signal quality." All of these fields were linked, processed, and saved together. Network traffic happens when there is an increase in the number of packets sent and received through VANET communications, which happens when numerous vehicles are on the road. The VANET transmission data collected consisted of 39,998 records derived from the V2V and V2I datasets. Our issue, intelligent network traffic prediction as a classification challenge, is resolved. Regarding traffic prediction using VGGNe, all the useful aspects of the actual VANET dataset should be used. Predicting network traffic is the end goal, and packet reception is one network parameter to consider for this purpose. A binary class with values 0 (no traffic) and 1 (traffic) represents the target.

3.2. Data Pre-processing

Research has emphasized the advantages of feature selection and data preparation before feeding input data into ML models. Improving the model's performance is possible with high-quality data pre-processing, which includes tasks like filling in missing data, normalizing the data, and selecting the most relevant and crucial features [26]. To develop a high-performance predictive model, we initially focused on normalization and feature selection approaches in our model.

3.2.1. Normalization

The first step in this course was cleaning the raw dataset of any unnecessary, duplicate, or irrelevant data. After that, we do some basic pre-processing by normalizing our data using Standard Scaler normalization, which puts all our feature values in the [0,1] range. The equation is described as:

$$Z_{scaled} = \frac{(X-\mu)}{\sigma} \text{ (1)}$$

where X = input variable, μ = Mean and σ = Standard Deviation. Following normalization, choose the key elements to be covered in Section II. Ultimately, we divided our dataset into two parts: the training set and the testing set. X is an assumed variable where (i) is the number of selected features. Our goal variable, y, represents the traffic prediction, thus assigning it the labels "traffic" (1) and "no traffic" (0).

3.3. Feature extraction through VGG

This research employed a VGG model to determine how spatial features affect the prediction process via convolution. The model includes thirteen convolutional layers, three completely linked layers, and five pooling layers. The image matrix was scanned using a 3*3 Convolution Kernel to extract image features. The initial matrix data might be passed through a fixed-matrix convolution kernel to extract picture features. The convolutional techniques could reduce the image's dimensions while extracting its features from the original. To summarise the general law, the resulting matrix dimension equals the product of the image matrix dimension, the convolution kernel matrix dimension, and one. The convolution layer can only handle pooling sizes up to 2 × 2 and uses a continuous 3 × 3 convolution kernel. In a convolution, the kernel is directly proportional to the input size, a tiny subset of the output from the preceding layer. The convolutional layer was employed to examine the tiny portion of every above layer to obtain more abstract spatial information. Equation (2) defines the convolutional layer. This layer's activation function is the ReLU function, which helps to speed up training and efficiently prevents the disappearance of gradients. An equation defining the ReLU function is given by (3).

$$x_j^l = f\left(\sum_{i=M_j} x_j^{(l-1)}.k_ij^1 + b_j^1\right) \text{ (2)}$$

$$f(x) = max(0,x) \text{ (3)}$$

In Eqs. (2) and (3), l represents the l[th] network layer; M_j represents the receptive field output by the previous layer; b denotes the bias; k is the convolutional layer; and $f(\cdot)$ is the nonlinear activation function. To simplify the computational complexity of the neural network model, the pooling layers in the VGG model can lower the size of the input feature. The max-pooling layer is an algorithm that takes in characteristics and returns their maximum value.

$$x_j^l = f\left(\beta_j^l p\left(x_j^{l-1}\right) + b_j^l\right) \text{ (4)}$$

Where $p(\cdot)$ is the pooling function, and β is the weight. Before feature extraction using the VGG model, the spatial feature pictures were transformed into a 224 × 224 × 3 image format. At long last, the time-series high-dimensional features of the spatial images were successfully obtained. Principal component analysis was used to obtain time-series data with multi-dimensional spatial features. Because of this, we could make high-dimensional redundant features disappear as a noise source. Hydrometeorology data, pollutant data, and target pollutant error can be combined to build the input and output datasets of the hybrid deep learning model.

3.4. Classification using Triplet Network.

The triplet network model combines three distinct functional blocks to forecast when traffic will become heavy. A multi-head attention mechanism and a multi-feed-forward neural network (Multi-FFN) are integrated into the transformer block. The feature output of several convolution kernels from the depth-wise, spatial, and channel attention components are housed in the multiscale convolution module. Additionally, there is a normalization module, two convolutions, and a linked layer. Lastly, DenseNet is used with three dense blocks to lessen the likelihood of data loss in the first two blocks.

3.4.1. Transformer Block.

We rethought the transformer block's internal architecture, which includes a multi-head attention mechanism and a multi-FFN. A multi-head attention system can receive three possible sequences: query, key, and value, all data sequences. The length of the input query sequence matches the length of the output sequence of attention. How long is the query? L_{q}, and the length of the key besides value is L_{k}. There is at least one parallel cell structure that makes up multi-head attention. This kind of cellular arrangement is known as a head. This cellular structure is called one-head attention for the sake of convenience. The attention that spans more than one head is known as multi-head attention. Keep in mind that there are n heads in multi-head attention and that the ith head's weights are W^{Q}, respectively, W_i^Q, W_i^K, and W_i^V. then,

$$head_i = Attention(q.W_i^Q, k.W_i^K, v.W_i^V) \quad (5)$$

$$MultiHead(q,k,v) = Concat(head_1, head_2, \dots, head_n).W^O \quad (6)$$

All three of these heads receive matrices as inputs: q, k, and v. A new matrix is created by splicing the output matrices of dimensions. This matrix is then multiplied by the WO matrix to get the output. By splitting each attention action into its own "head," the multi-head attention mechanism can retrieve feature information from various dimensions. Q, K, and V are each transformed linearly by one of the three transformation tensors. Starting at the semantic level, each head begins to segment the output tensor to produce a set of Q, K, and V that may be used to calculate the mechanism. The multiple feed-forward neural network (MultiFFN) is constructed by fusing the three outputs produced by three separate feed-forward neural networks. Three blocks make up the structure: RBF, FC, and Conv. The multi-head attention mechanism is a part of this revamped transformer block, combining three feed-forward networks from a single feed-forward MLP network to extract multimodal fusion features. A total of P basis functions is chosen using the RBF method, with each basis function corresponding to a single training set of data. The following is the interpolation function that uses the radial basis function:

$$\phi(r) = exp\left(-\frac{r^2}{2\sigma^2}\right) \quad (7)$$

$$F(x) = \sum_{p=1}^{P} \omega_p \phi_p(\|X - X^P\|) \quad (8)$$

The input X is an m-dimensional vector, and the sample size is $P, P > m$. The radial basis function ϕ centres on the input data point X_p. The concealed layer's job is to transform the vector from m, the low-dimensional space, into P, the high-dimensional space. A low dimension can be linearly separable from a high dimension if indivisible. For the radial basis function, settle on the reflected sigmoidal functions ϕ. The Conv block includes three convolutional layers using a 3∗3 kernel. Data loss can be prevented by removing the down-sampling layer.

3.4.2. Multiscale Convolution Block.

The multiscale convolution block follows the transformer's internal structure. All channels are divided into many groups, and convolution is executed in each of these groups using group convolution. The convolution operation is executed from top to bottom, starting with dimension expanding (depth-wise convolution) and ending with dimension reducing (bottom to top) using the inverse bottleneck layer. This prevents the parameter amount from increasing and makes feature comparison easier after the 1 ∏ 1 convolution. The input is passed through three distinct depth-wise convolution kernel scales. The output of each layer is

used to extract the joint features of channel attention besides spatial attention. Then, two one-dimensional convolutions are used to generate the final multi-convolution fusion feature.

It should be mentioned that this module processes the input data to obtain three features for fusion using multiscale convolution kernels, specifically 7 ∏ 7, 5 ∏ 5, and 3 ∏ 3, in depth-wise convolution. To emphasize the landmark info and target site in the input data, serial channel attention is applied after getting the feature blocks. The network model's feature extraction capabilities can be enhanced by convolutional block attention (CBA) without substantially increasing computation or parameters. To create the final feature map, this module can sequentially generate information for an attention feature map in two dimensions: channel and space. It then uses adaptive feature correction to multiply this information with the original input feature map. There are two parts to it: channel attention and spatial attention. The former creates channel attention mapping by analyzing the connections between feature channels. Reducing the spatial feature mapping allows us to compute channel attention effectively. When aggregating geographical data, average pooling is the method of choice. Spatial attention maps are generated by utilizing the spatial interaction between features. Before calculating spatial attention, create effective feature descriptors by applying average and max pooling axes and concatenating them.

Channel Attention. We use average and max pooling techniques to reduce the input feature dimension to create two one-dimensional vectors. Global feedback is only provided when the retort is the greatest in the feature map during the calculation of gradient backpropagation, in contrast to global average pooling, which provides feedback for every pixel on the feature map. The features of the spatial dimension are aggregated using average pooling and max pooling to provide two spatial dimension descriptors: F_{max}^{c} and F_{avg}^{c}. Afterward, an MLP network generates the weight for every channel. The last step is to double the weight by the initial channel attention. Let me give you the formula:

$$M_c(F) = \begin{matrix} \sigma(MLP(AvgPool(F))) + (MLP(MaxPool(F))) \\ = \sigma\big(W_1(W_0\big(F_{avg}^{c}\big) + W_1\big(W_0(F_{max}^{c})\big)\big) \end{matrix} \quad (9)$$

Where F represents the input feature map, F_{avg}^{c} and F_{max}^{c} Are the features intended by global average pooling and global max pooling correspondingly, W_0 and W_1 represent two-layer parameters in features among W_0 and W_1 The multilayer perceptron model must be administered with ReLU as the activation function.

Spatial Attention. According to the author, apart from creating the attention model on the channel, the network has to know which sections of the feature map should respond more at the spatial level before applying mean and max operations on the channel dimension to the input levels. A two-channel feature map is created by stitching together two 2D features according to the channel dimension. A single convolution kernel is then convolved in a hidden layer. The input feature map and the output features should match up in terms of the spatial dimension. Max and average pooling operations are also utilized, which are carried out in the channel dimension. To train spatial attention, you must narrow the original feature's C-dimensional channel count down to one dimension. This equation can be expressed as:

$$M_s(F) = \begin{matrix} \sigma\big(f^{7\times7}([AvgPool(F); MaxPool(F)])\big) \\ = \sigma\big(f^{7\times7}\big[F_{avg}^{s}; F_{max}^{s}\big]\big) \end{matrix} \quad (10)$$

Where the feature map after max average pooling is defined as $F_{avg}^{s} \in R^{1*\mathrm{H}*\mathrm{W}}$ and $F_{max}^{s} \in R^{1*\mathrm{H}*\mathrm{W}}$ and σ represents function. The component displaying the convolution layer uses 7 x 7 convolution kernels. The following formula describes the relationship between channel attention and spatial attention:

$$\begin{cases} F' = M_c(F) \otimes F \\ F'' = M_s(F') \otimes F' \end{cases} \quad (11)$$

Where F is the feature map, $M_s(F')$ and $M_c(F)$ Focusing focus on space and channels ⊗ signifies the multiplication of elements one by one, while F′ and F″ stand for the output feature maps following channel attention besides spatial attention, correspondingly. You can install the convolutional block attention module wherever in the existing model because its input and output sizes are the same. After that, a GeLU activation function layer is sandwiched between two 1∗1 conventional convolution layers to keep the probability and input dependency intact and prevent the gradient from vanishing.

3.4.3. DenseNet.

DenseNet employs a more forceful dense connection technique and incorporates three dense blocks. Each layer will take in data from all the levels below it to supplement its input. Each dense block's feature map size is unified to simplify concatenation. As a network architecture, DenseNet makes use of dense block transitions. A multilayered module is called a dense block.

Every layer's feature map is of the same size. The layers are joined densely. To decrease the size of the feature map, the transition module pools data from two nearby dense blocks and joins them. Dense blocks and transitions (convolution with pooling) comprise the bulk of DenseNet's network architecture. Instead of referencing each layer individually, the feature transfer method concatenates all of the characteristics from each layer before it to the next.

3.4.4. Reconstructed Model.

Creating a temporal network, a two-layer fully linked layer, and a triplet network is the next step in the transfer learning model's downstream task. During pretraining, do not make any changes to the triplet network. Due to the sequential nature of the input data, first implant a temporal network that includes a bidirectional long short-term memory (BiLSTM) with attention to retraining and fine-tuning the initial network weights, then two fully linked layers. BiLSTM uses a temporal module and a two-layer internal extendable unit as the structure for the tune-tuning downstream duty. To compensate for the limitations of the triplet network, integrate its output features with those of the temporal network, maintain bidirectional information transmission between traffic sequence data frames, and enhance the matrix.

3.4.5. Fine-tuning using Dholes Hunting-Based Optimization (DHO)

In this work, the DHO is used to fine-tune the learning parameters of the proposed model that is discussed as follows: In contrast, to consider the location of prey to avoid being eaten as the optimal point of the set, where predators are difficult to reach. Prey are also sensitive when observing predators and always look for the best positions. As a result, the algorithm combines the optimal position of the hunter and the prey. We also need to strategize to ensure the members in the pack keep their distance for the best chasing, avoiding falling early into one local minimum location. Those main inspirations are described and simulated as mathematical equations in the below sections.

3.4.5.1. Initialization

A population of N dholes are placed on the ground, randomly initialized as follows:

$$x_{i,j} = x_{i,j}^{min} + rand_{i,j} * \left(x_{i,j}^{max} - x_{i,j}^{min}\right),\ i = 1, \dots, N; j = 1, \dots, D \quad (12)$$

where $x_{i,j}$ Is the vector of i − th dhole with D dimensions, $x_{i,j}^{min}$, and $x_{i,j}^{max}$ Denote the values for the j − th dimension of the i − th agent, and the rand is a uniform random value in the interval of $[0, 1]$. Let $f \in R^{\mathrm{D}}$ be the criterion function of m variables according to the problem space dimensions. The x_{best} is the optimal solution (agent) if function f (x_{best}) value if global minimum or maximum. In DHO, the position of prey is the optimal position the dhole tries to reach.

3.4.5.2. Clustering-Hunting Based

This subsection describes the first step in the main DHO algorithm. The algorithm is motivated by dholes' swarm hunting, ambushing tactics, and prey chasing. Moreover, they cannot all compete for each prey because of the large number of prey, so they will split into smaller groups to make hunting more efficient. A reasonable pack division, based on the K-means algorithm, should be separated into many groups so that the members of each group tend to be close together. At the same time, the distance between the groups is as far as possible. Using K-means as an essential step in helping dhole have good distance separation. In other words, the algorithm makes it a point to maintain its distribution diversity. When the group members are close to each other, the probability of them falling into the same valley is higher, so the gradient approximation can be applied to compute a local minimum. In addition, it is possible to replace the K-means algorithm by selecting neighbours with the K-nearest neighbour selection algorithm. In our study, the K-means algorithm instead of KNN ensures uniform clustering in the optimal domain. For the initial set with k random centroids $(c_1, c_2, \dots, c_k)$, each observation should be assigned to the cluster with the mean closest to it using the least-squares error (Eq. 13). For each cluster-allocated data, the update phase should recalculate the means centroids (Eq. 14).

$$C_i^{(t)} = \left\{x_p : \left\|x_p - c_i^{(t)}\right\|_2 \le \left\|x_p - c_j^{(t)}\right\|_2 \ \forall j, 1 \le j \le k\right\} \quad (13)$$

$$c_i^{(t)} = \frac{1}{\left|C_i^{(t)}\right|} \sum_{x_j \in C_i^{(t)}} x_j \quad (14)$$

3.4.5.3. Pack Hunting Strategy-Multi-Local Search Phase

This subsection describes the main algorithm based on swarm interaction, acceleration, and capture strategies. Dholes move around their territory in search of potential prey as a herd of elk, caribou, and wild pigs. When prey is spotted, the pack starts

approaching quietly from different directions as close as possible, trying to stay undetected. When they reach a good location, they run as fast as possible to attack by surprise. However, it will run in the opposite direction and try to find the locations where it feels safe that the dhole swarm can hardly find (local optimal point). Simultaneously, the pack members must surround the prey and keep a distance from other members. For mathematical modeling, suppose the position of the prey is x_{p}, and the members of the dhole swarm are $x_{\mathrm{d_i}}$, respectively ($2 \leq i \leq n_{\mathrm{members}}$ where n_{members} is the number of members in the dhole swarm). In the first step, the prey always has an advantage over the members of the dhole swarm. The complete process of pack hunting is presented, where ∇v is seen as the gradient vector and v_{support} helps guide the dholes to keep their distance from each other.

To use $f(x)$ has more than one local optimal point. However, for close sets of locations, we would expect them to lie in the same valley. Besides, this algorithm can still work well even if several points are not in the same valley or are in the case of an imperfect valley. However, one issue still exists: how to group dholes into the same packs so that the corresponding dholes belong to the same valley. Since points in one group are usually close to each other, it is expected that their location falls into a unique valley. In the chasing process, the dholes try to approach their prey while the prey runs away from all the dholes. The location of the dhole and prey are updated according to ∇v. However, the prey will be able to speed up and slow down depending on the terrain, so to use the kl_{target} coefficient to accelerate the convergence trend.

$$kl = \begin{cases} \frac{f(x_p^i) - f(x_p^{i-1})}{f(x_p^i) - \frac{\left(f(x_p^{i-2}) + f(x_p^{i-3})\right)}{2}}, & if\ 2 * f(x_p^i) \\ 1, & \neq \left(f(x_p^{i-2}) + f(x_p^{i-3})\right) otherwise \end{cases} \quad (15)$$

$$kl_{target} = \begin{cases} \frac{kl_{target}}{kl_{div}} & if\ kl \leq 1 \\ kl_{target} * kl_{mul}, & otherwise \end{cases} \quad (16)$$

$$kl_{mul} = kl_{mul} * r_1,\ if\ kl \geq 1 \quad (17)$$

$$kl_{div} = kl_{div} * r_2,\ otherwise \quad (18)$$

where $f(x_p^i)$ Is the fitness of dhole p in the i-th moves? kl_{div} Does the coefficient decrease the speed of divergence and kl_{mul} Does the coefficient increase the speed of divergence? kl_{div} and kl_{mul} Fluctuate from 1.0 to 2.0. n_{moves} is the number of moves in the pack-hunting strategy. It is different and thus different for the different optimization problems. After the packs catch their prey, they regroup into the super pack to share food and information about the local area.

In the hunting search phase, the algorithm adjusts the movement speed of dholes using the kl_{mul} and kl_{div} Parameters, which can increase or decrease the speed appropriately to help dholes achieve local optimization quickly. Simultaneously, a probability parameter is determined to avoid getting stuck in small local areas. Moreover, two acceleration parameters (r_1 and r_2) are used to vary kl_{mul} and kl_{div}, respectively. The acceleration parameters (r_1 and r_2) help to adjust the velocity of the dhole to change more rapidly in areas of high variability, allowing for faster search space exploration. However, the sensitivity of these two parameters is quite large when changing the domain space, and there is not a single fixed value. Therefore, evaluating many different benchmarks makes it very difficult to select the fixed value of these two parameters. In this study, these two parameters (r_1 and r_2) are selected to be small and fixed to avoid too much deviation. The value of n_{moves} has a significant influence on the running time of the procedure. However, n_{moves} does not necessarily need to be set to a very large value because the speed of the dhole can be increased exponentially by adjusting the numerical parameters r_1 and r_2. Use a fixed value for n moves in all benchmark sets during testing.

3.4.5.4. Re-forming packs-global search phrase

In forming packs and hunting, there can be many conflicts among the holes in the pack. Although they can stay close and communicate with each other, the food distribution issues make the local area's habitat less suitable than before. They may leave the pack and join another or re-form a new pack. In addition, dholes also face other dangers, such as being hunted by humans or other big predators like lions and leopards. Of course, if they cannot survive in the herd, they will have to find a way to reproduce (natural selection), adapt to the environment, or switch to another pack to live. In DHO, the weak dhole in the herd has to find a way to settle in another herd. If they find a new pack, they can survive and join it to hunt and share food. If they can't find other flocks, they may become lost, have to hunt alone, or be attacked by other predators (or possibly die due to food shortage). An unsuccessful re-forming pack process of dholes consists of two parts: the lost dholes have to go hunting alone, and the dholes starve or die by other predators, which shows the visualization of the pack reforming process.

3.4.5.5. Moving to Another Pack

If a dhole successfully joins the new herd, it must adapt to the new environment and learn how to hunt prey from the new leader and other holes in the new group. Therefore, our algorithm combines the features of three holes, including a current dhole, a leader dhole, and a random strong dhole, to form the integration process.

$$x_d^{new} = x_d^{old} + \frac{\delta\left(x_{d1}^{*} - x_d^{old}\right)}{2} \times rand_1 + \frac{(1-\delta)x_{d2}^{*}}{2} \times rand_2 \quad (19)$$

where x_d^{old} and x_d^{new} Are the old and new positions of the chosen dhole, δ is a user-defined parameter, x_{d1}^{*} Is the position of the leader dhole, and x_{d2}^{*} Is the position of the strong dhole selected randomly from one of the K best dholes currently ($x_{d1}^{*} \neq x_{d2}^{*}$). $rand_1$ and $rand_2$ are random vectors with the same dimension as x, values in (0, 1).

3.4.5.6. Straying from the pack

The dhole encounters many obstacles when finding a new herd from its current herd. For example, bad weather or being chased by other predators causes them to stray from the pack. In addition, dholes can also die from lack of food on the way, especially old dholes that do not have enough strength to move.

- For those dholes killed by the other predator or due to natural selection in searching for a new herd, a random dhole in the environment will reproduce and replace the dead one using Eq. 12.
- Meanwhile, dholes that survive being hunted by other predators or natural selection must possess unique characteristics and skills to adapt to new environments and the process of hunting alone. To simulate the foraging process of these holes, use the Levy-flight technique.

In the Levy-flight trajectory, the step length follows the heavy-tailed Levy distribution to model the foraging trajectory of dholes. Numerous researchers have demonstrated that many creatures, including birds, insects, and marine Levy-flight, can improve the efficiency and accuracy of natural adaptability. As a global searching operative, the Levy-flight trajectory searches for space using short-distance walking and long-distance jumping routes. Those two abilities help improve the population's diversity and local exploitation ability, especially with the approximate formula, which generates random statistics obeying distribution. In general, Levy's step size can be uttered as:

$$Levy(s) \sim |s|^{-1-\beta} \ \ with\ 0 < \beta \leq 2 \quad (20)$$

$$s = \frac{\mu}{|v|^{1/\beta}}, \mu \sim N\left(0, \sigma_\mu^2\right), v \sim N(0, \sigma_v^2) \quad (21)$$

$$\sigma_\mu = \left[\frac{\Gamma(1+\beta) \times \sin(\mu.\beta/2)}{\Gamma\left(\frac{(1+\beta)}{2}\right) \times \beta \times 2^{(\beta-1)/2}}\right]^{1/\beta}, \sigma_v = 1 \quad (22)$$

Where s is the step length of the Levy flight premeditated by Mantegna's procedure, μ, and v are chosen normal delivery, β in $[0, 2]$, and zero a function.

$$x_d^{new} = Levy(x_d^{old}) \quad (23)$$

where x_d^{old} and x_d^{new} The previous position and the current generated position of the current dhole are as follows:

3.4.5.7. Exploration and Exploitation Rate

In the DHO algorithm, four coefficients help the exploration and exploitation process and balance those two processes. α is calculated by Eq. 24 and is used to calculate δ (Eq. 25) and θ (Eq. 27). δ is a parameter mentioned in Eq. 25. Meanwhile, θ and γ are used to discount the convergence speed.

$$a = arctanh\left(-\left(\frac{g+1}{g_{max}} + 1\right)\right) \quad (24)$$

$$\delta = \frac{\exp(a)}{exp\left(arctanh\left(\frac{g_{max}-1}{t_{max}}\right)\right)} \quad (25)$$

$$\gamma = \left(1 - \frac{g+1}{g_{max}}\right) \times cos\left(\frac{\pi}{3} \times \frac{g+1}{g_{max}}\right) (26)$$

$$\theta = \frac{\left(1-\frac{g+1}{g_{max}}\right)}{2} \times exp\left(\frac{\alpha}{2}\right) (27)$$

Where g is the current generation (repetition/epoch), and $g_{\max}$ is the maximum number of generations. For termination with the maximum figure of function assessments, g and $g_{\max}$ are replaced by current evaluation count fees and maximum evaluation bound $fes_{\max}$.

3.4.5.8. Algorithm Complexity

With the assumption that to have a population with N dholes, D is the number of dimensions of the problem, the maximum sum of generations is $g_{\max}$, and the cost for the fitness function is C. The proposed algorithm uses n_{moves} moving steps to find the best local solution in the pack hunting strategy. The number of function evaluations in this step is $\frac{N*n_{move}}{G_{\text{size}}}$. In the re-forming pack phase, the DHO algorithm only uses common transforming operators. The number of evaluations in this step is $2N/G_{\text{size}}$. Consequently, the average complexity of the proposed algorithm is estimated as $O(g_{max} * N * n_{move} * C)$.

4. Results and Discussion

Specifically, the studies are carried out on a machine with 8 GB of internal memory and an $Intel\ Core\ i5-7200$ processor that can run at 2.7 GHz. The user interface (UI) and Jupyter Notebook (Python 3.7) are designed to perform the procedures on Windows 10, a 64-bit operating organization. Natural Setting.

4.1. Validation Analysis of projected model on diverse data ratio

Table 1 and Figure 2 provide the experimental analysis of the projected model on different training and testing data ratios.

Table 1: Experiment with different training and testing data

Train/Test Split	Accuracy	Precision	Recall	F1-Score
80/20	99.98	98.30	98.14	98
70/30	97.10	95.20	96.52	97
60/40	94.98	93.92	94.24	96.52

Results from experiments using dissimilar train/test splits show metrics of accuracy, precision, and recall besides F1-Score. For the 80/20 split, the model accomplished the uppermost performance with an accuracy of 99.98%, a precision of 98.30%, a recall of 98.14%, and an F1-Score of 98. When using a 70/30 split, the accuracy decreased to 97.10%, with a precision of 95.20% and recall of 96.52%, besides the F1-Score of 97. In the 60/40 split, the accuracy further decreased to 94.98%, with values of 93.92% and 94.24%, respectively, besides an F1-Score of 96.52. These results indicate that a larger training data proportion improves the model's presentation across all metrics.

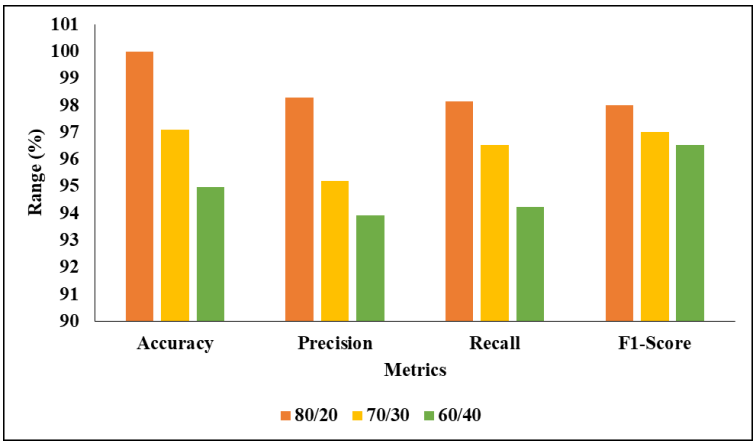

Figure 2: Visual Representation of the proposed model on various data ratios

Table 2 provides the timing analysis of the projected model with the usage of different training besides the testing ratio.

Table 2: Timing Analysis of the projected model

Training Time (s)	Train/Test Split	Prediction Time (s)
4.96	80/20	0.74
4.8	70/30	1.32
3.71	60/40	1.44

The table compares training and prediction times across different train/test splits. For the 80/20 time of 0.74 seconds, the higher training proportion led to faster predictions. With a 70/30 split, training time was slightly reduced to 4.8 seconds, but prediction time increased to 1.32 seconds. In the 60/40 split, training time was the shortest at 3.71 seconds, though prediction time rose to 1.44 seconds. These results show that a larger training set improves performance metrics and generally enhances prediction speed despite requiring slightly more training time.

4.2. Comparative Analysis

Table 3 provides the comparative investigation of the wished-for model with existing techniques in terms of different metrics. The existing models use different datasets, and therefore, the basic models are implemented using our datasets, and results are averaged for traffic congestion detection.

Table 3: Comparative Study of the Proposed Model

Models	Accuracy	Precision	Recall	F1-Score
LGBM [17]	0.8055	0.8006	0.8058	0.8000
MLDNN [18]	0.8541	0.8604	0.8544	0.8566
BiSPNet [20]	0.9041	0.9069	0.9036	0.9045
TSANet [22]	0.9071	0.9085	0.9060	0.9063
Proposed model	0.9416	0.9465	0.9402	0.9408

The LGBM model achieved an accuracy of 80.55%, with precision, recall, and F1-Score values around 80.06%, 80.58%, and 80.00%, respectively. The MLDNN model showed improved results with an accuracy of 85.41%, precision of 86.04%, recall of 85.44%, and F1-Score of 85.66%. BiSPNet further increased performance, achieving an accuracy of 90.41%, precision of 90.69%, recall of 90.36%, and F1-Score of 90.45%. TSANet showed slightly better results with an accuracy of 90.60%, besides the F1-Score of 90.63%. The projected model outperformed all previous models, achieving an accuracy of 94.16%, precision of 94.65%, recall of 94.02%, and an F1-Score of 94.08%, demonstrating its superior performance across all metrics (Figure 3).

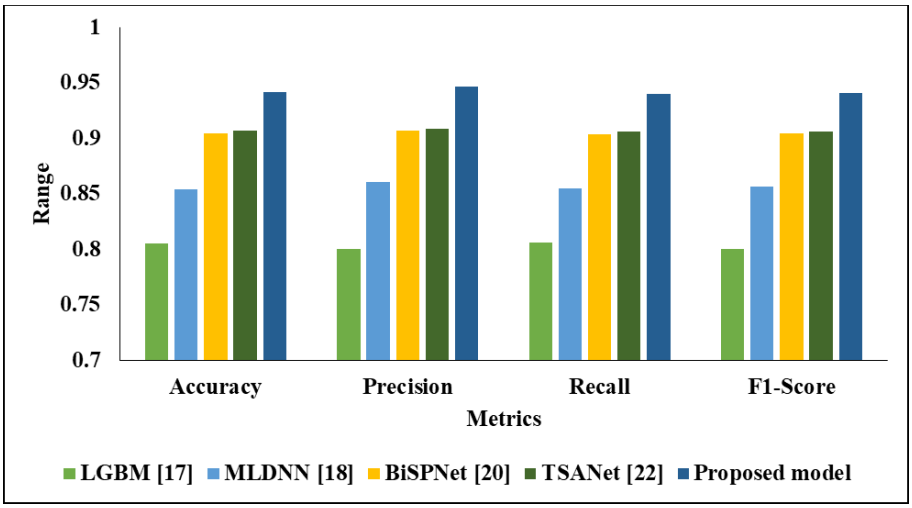

Figure 3: Visual Illustration of the wished-for model with existing methods

5. Conclusion

Our research presents a DL model for VANET traffic forecasting based on feature extraction. A reconstructed model achieves transfer learning for traffic prediction after features are retrieved using a VGGNet model. Following this, a pre-trained triplet hybrid model is created. The DHO algorithm enhances the classification accuracy by fine-tuning the suggested model. We researched and evaluated other popular ML models for traffic forecasting to develop our proposed model. Additionally, classification measures such as confusion matrix, score, and time were taken to assess the performance properly. Because DSRC admission knowledge is only suitable for short-range coverage, the offered model has this constraint. Furthermore, automobiles and roadside units communicate in a very simplistic manner. But in a real-world setting, we'll have multiple kinds of connectivity, such as V2X (vehicle-to-everything) and V2P (vehicle-to-cellular network). Consequently, the forms of communication should be expanded through various access technologies, such as 5G and LTE, which offer extensive coverage. By doing so, we can better understand how this prediction model handles traffic in real-world scenarios. Our long-term goal is to grow a reliable forecast model by integrating several technologies, such as vehicle-to-element (V2X) communications in VANETs and cellular networks for vehicular communication.

Acknowledgment: I am deeply grateful to my co-authors for their expertise and dedication, which greatly enriches this work. Special thanks to my friends for their unwavering support and encouragement throughout the research process.

Data Availability Statement: The data for this study can be made available upon request to the corresponding author.

Funding Statement: This manuscript and research paper were prepared without any financial support or funding.

Conflicts of Interest Statement: The authors have no conflicts of interest to declare. This work represents a new contribution by the authors, and all citations and references are appropriately included based on the information utilized.

Ethics and Consent Statement: This research adheres to ethical guidelines, obtaining informed consent from all participants. Confidentiality measures were implemented to safeguard participant privacy.

References

1. A. A. Shuvro, M. S. Khan, M. Rahman, F. Hussain, M. Moniruzzaman, and M. S. Hossen, "Transformer-based traffic flow forecasting in SDN-VANET," IEEE Access, vol. 11, no.5, pp. 41816–41826, 2023.
2. X. Liu, B. S. Amour, and A. Jaekel, "A reinforcement learning-based congestion control approach for V2V communication in VANET," *Applied Sciences*, vol. 13, no. 6, p. 3640, 2023.
3. K. Kaushik, V. V. Krishna A, and Abhilash, "A novel approach for real time traffic prediction using deep learning in VANET," in Proc. 14th Int. Conf. Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023.
4. M. Selvapriya, K. Manimekalai, R. Rajesh Kanna, P. Pratheep Kumar, and V. Rajkumar, "Risk identification and traffic prediction based on AI Technologies in VANET," Journal of Survey in Fisheries Sciences, vol. 10, no. 1S, pp. 4084–4089, 2023.
5. A. Chahal, P. Gulia, N. S. Gill, and I. Priyadarshini, "A hybrid univariate traffic congestion prediction model for IoT-enabled smart city," information, vol. 14, no. 5, p. 268, 2023.
6. S. R. Sahu and B. Tripathy, "A systematic VANET traffic congestion by eliminating recursion using intervention linear minimum spanning tree (ILMST) for traffic management system," in Springer Proceedings in Mathematics & Statistics, Cham, Springer Nature, Switzerland, pp. 175–186, 2024.
7. T. M. Aruna et al., "Geospatial data for peer-to-peer communication among autonomous vehicles using optimized machine learning algorithm," Sci. Rep., vol. 14, no. 1, p. 20245, 2024.
8. H. Ouhmidou, A. Nabou, A. Ikidid, W. Bouassaba, M. Ouzzif, and M. A. El Kiram, "Traffic control, congestion management and smart parking through VANET, ML, and IoT: A review," in 2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM), Istanbul, Turkey, 2023.
9. R. Pradeep, K. Revathi, A. K. Thomas, and R. Srikanth, "A Novel Optimized AI Based Model for Traffic Prediction in VANET," in 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA), IEEE, Coimbatore, India, pp. 1175–1180, 2023.
10. K. Suganyadevi, V. Swathi, T. Santhiya, S. S. Sankari, and V. S. Swathi, "Machine learning algorithm based VANET traffic management system," in 2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2023.

11. S. Tavasolian and M. Afzali, “Traffic Flow Prediction Based on VANET Data by Combining Artificial Neural Network and Genetic Algorithm,” Azerbaijan Journal of High Performance Computing, vol. 6, no. 1, pp. 91–112, 2023.
12. K. Aravinda, B. S. Kumar, B. P. Kavin, and A. Thirumalraj, “Traffic Sign Detection for Real-World Application Using Hybrid Deep Belief Network Classification,” in Advanced Geospatial Practices in Natural Environment Resource Management, IGI Global, Pennsylvania, United States of America, pp. 214–233. 2024.
13. H. Toulni, M. Miyara, Y. Filali, and S. C. K. Tékouabou, “Preventing urban traffic congestion using VANET technology in urban area,” in E3S Web of Conferences, vol. 418, EDP Sciences, 2023.
14. M. L. M. Peixoto et al., “FogJam: A fog service for detecting traffic congestion in a continuous data stream VANET,” Ad Hoc Netw., vol. 140, no. 1, p. 103046, 2023.
15. P. A. D. Amiri and S. Pierre, “An ensemble-based machine learning model for forecasting network traffic in VANET,” IEEE Access, vol. 11, no. 3, pp. 22855–22870, 2023.
16. S. Ghosh, I. Saha Misra, and T. Chakraborty, “Optimal RSU deployment using complex network analysis for traffic prediction in VANET,” Peer-to-Peer Networking and Applications, vol. 16, no. 4, pp. 1135–1154, 2023.
17. N. Tsalikidis et al., “Urban traffic congestion prediction: a multi-step approach utilizing sensor data and weather information,” Smart Cities, vol. 7, no. 1, pp. 233–253, 2024.
18. K. Kumar, M. Kumar, and P. Das, “Traffic congestion forecasting using multilayered deep neural network,” Transp. Lett., vol. 16, no. 6, pp. 516–526, 2024.
19. Y. Fang et al., “PM2.5 concentration prediction algorithm integrating traffic congestion index,” J. Environ. Sci. (China), 2024, Press.
20. M. Kumar and K. Kumar, “Bi-State Prediction Network Model for Mixed Traffic Congestion Prediction,” International Journal of Computational Intelligence and Applications, vol. 23, no. 3, p, 2450011, 2024.
21. Y. Li, J. Xu, Y. Li, Y. Xue, and Z. Yao, “Estimation and prediction of freeway traffic congestion propagation using tagged vehicle positioning data,” Transportmetrica B: transport dynamics, vol. 12, no. 1, pp. 1-21 2024.
22. K. N. Kumar, D. Roy, T. A. Suman, C. Vishnu, and C. K. Mohan, “TSANet: Forecasting traffic congestion patterns from aerial videos using graphs and transformers,” Pattern Recognit., vol. 155, no. 11, p. 110721, 2024.
23. R. M. Fujimoto, R. Guensler, M. P. Hunter, H. Wu, M. Palekar, J. Lee, and J. Ko. CRAWDAD Dataset Gatech/Vehicular, 2006. [Online]. Available: https://crawdad.org/gatech/vehicular/2006031, [Accessed by 03/05/2023].
24. H. Wu et al., “‘An empirical study of short range communications for vehicles,” in Proc. 2nd ACM Int. Workshop Veh. Ad Hoc Netw, Cologne, Germany, pp. 83–84, 2005.
25. A. Thirumalraj, V. Asha, and B. P. Kavin, “An improved Hunter-prey optimizer-based DenseNet model for classification of hyper-spectral images,” in AI and IoT-Based Technologies for Precision Medicine, IGI Global, Pennsylvania, United States of America, pp. 76–96, 2023.
26. B. Minh Nguyen et al., “Dholes hunting—A multi-local search algorithm using gradient approximation and its application for blockchain consensus problem,” IEEE Access, vol. 12, pp. no. 6, 93333–93349, 2024.