Traffic Jam Probability Estimation Based on Blockchain and Deep Neural Networks

Vikas Hassija, Vatsal Gupta, Sahil Garg and Vinay Chamola

Abstract—**The exponential surge in the number of vehicles on the road has aggravated the traffic congestion problem across the globe. Several attempts have been made over the years to predict the traffic scenario accurately and consequently avoiding further congestion. Crowdsourcing has come forward as one of the most adopted methods for predicting traffic intensity using live data. However, the privacy concerns and the lack of motivation for the live users to help in the traffic prediction process have rendered existing crowdsourcing models inefficient. Towards this end, we present an advanced blockchain-based secure crowdsourcing model. Not only does our model ensure privacy preservation of the users, but by incorporating a revenue model, it also provides them with an incentive to participate in the traffic prediction process willingly. For accurate and efficient traffic jam probability estimation, our work proposes a neural network-based smart contract to be deployed onto the blockchain network. The results reveal that the proposed model is highly efficient in terms of attaining high participation and consequently obtaining highly accurate predictions.**

Index Terms—**Deep Learning, Neural Networks, Long Short Term Memory, Traffic Jam, Ethereum, Blockchain.**

I. Introduction

In recent times, road traffic congestion has become an everyday affair, especially in metropolitan areas. Not only does it cause direct monetary losses in terms of fuel, and environmental losses in terms of increasing carbon footprint, but it also leads to various indirect losses [1]. Such losses include the loss of time, lack of productivity, rising stress levels among drivers, and unnecessary disputes among drivers arising out of frustration [2]. As per the Mobility Lab Report - 2019, each driver in New York City had to incur direct and indirect losses amounting to $2,982$ dollars on average [3]. Other than the lack of well-planned roads and reckless driving, accidents, demonstrations, marathons and other events are primarily responsible for traffic jams [4]. Although it may be beneficial to a certain extent to examine historical data for live traffic congestion prediction, it alone can not provide the best insights into the live traffic situation. There is a need to incorporate live data-based prediction models in traffic congestion prediction to yield accurate results.

Vikas Hassija and Vatsal Gupta are with the Department of Computer Science and IT, JIIT, Noida, India 201304 (e-mail: vikas.hassija@jiit.ac.in, vatsalgupt99@gmail.com).

Vinay Chamola is with the Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani Campus, India 333031 (e-mail: vinay.chamola@pilani.bits-pilani.ac.in).

S. Garg is with the Electrical Engineering Department, École de technologie supérieure, Université du Québec, Montréal, QC H3C 1K3, Canada. (email: sahil.garg@ieee.org)

Digital Object Identifier: XXXXXXXXXXXX

By employing the use of various tools, including incident detection, incident verification, incident response, and incident communication, intelligent transportation systems (ITS) play a significant role in mitigating traffic congestion [5]. In addition to ITS, various other efforts are being made in industry and academia to predict and mitigate traffic congestion problems effectively. In recent years, crowdsourcing has emerged as a prominent model for live traffic scenario estimation and real-time navigation [6]. Several crowdsourcing architectures have been proposed by various research and industry groups to solve urban traffic problems effectively [7].

The concept of crowdsourcing originated in the 18^{th} century when a substantial cash prize was offered to anyone who could succeed in developing a method for determining a vessel's length while at sea [8]. The core idea behind crowdsourcing is to use the talent, expertise and knowledge that is available to the public to solve specific problems. Wikipedia, launched in 2001 by Wales and Sanger, is an excellent example of an application using the concept of crowdsourcing. The collective knowledge of a large group of people can help in solving problems faster and more efficiently. Furthermore, most people directly deal with the problem under consideration and therefore know the exact pitfalls, corner-cases, and possible solutions. The use of crowdsourcing for traffic congestion prediction was first adopted by an Israeli start-up company named Waze in 2013 [9], which was later acquired by Google for 1.1 billion US dollars [10], [11].

Crowdsourcing, with all its benefits, faces two significant challenges - 1) the user privacy issues, and 2) the lack of motivation among users to participate. In existing crowdsourcing models, the participants have to share their private information such as their name, phone number and location information. Sensitive nature of such information makes it prone to malicious attacks and consequently compromises the privacy of the users. Crowdsourcing is one of the most significant components used in Google Maps for live traffic congestion prediction [12]. In such applications, the users are forced to share their location information in exchange for live traffic data. The location information shared by the users is then used to predict traffic conditions at different locations. Besides being able to look up routes and location, users have no other incentive to participate in such models.

Blockchain is a highly promising technology to resolve the issues limiting user participation in crowdsourcing projects as mentioned above. Blockchain is a highly secure, immutable, transparent, and privacy-preserving distributed ledger technology [16]–[20]. Initially, in 2009, the concept of blockchain was introduced for use as a digital currency system, but gradually,

TABLE I: Related works on traffic jam prediction and mitigation

References	Contributions	Strengths	Weaknesses
Bo Xie *et al.* [2]	Urban cell transmission model (UCTM) to evaluate traffic jam.	Simulation of urban traffic is done.	Only a mathematical model is presented.
Jiancheng Long [5]	Control strategies for incident based traffic jam in a two-way grid network.	Spatial topology of jam propagation is used.	Lack of prevention and prediction mechanisms.
Zuchao Wang *et al.* [13]	Visual traffic jam analysis based on trajectory data.	Better accuracy in traffic jam detection.	Only the vehicle speed is used to detect jam.
Myounggyu Won [14]	Vehicle to vehicle communication to mitigate phantom jam condition.	ML-based algorithms are used to detect jam.	The behavior of investors and developers is not considered.
Kazuhiro Saitou *et al.* [15]	Real-time traffic prediction for lagrangian traffic	20% improvement in the prediction results.	The live information from users is not used to improve the predictions.
Our proposed approach	Proposed a blockchain-based crowdsourcing platform to predict real-time traffic jam probability with a monetary incentive for the participants.	A distributed network that captures the interest of all the participating nodes.	Large number of frequent transactions cannot be supported (which is an inherent weakness of blockchain).

with the innovation of ethereum, blockchain technology has laid its roots in several domains [21]. The detailed system model involving blockchain network and crowdsourcing based traffic jam probability estimation is discussed in section III. The major contributions of the work are discussed as follows:

- A peer-to-peer, open network of vehicles is proposed where vehicles can securely share and request for the live traffic scenario at a particular location.
- An ethereum based smart contract is deployed for validating and storing the information shared by the users in the network.
- A neural network-based model is proposed to calculate the actual probability of traffic jam at a particular location based on live and historical data.
- An incentive model is proposed to increase the motivation of the users to participate in the crowdsourcing model for traffic jam probability estimation.
- Simulation results demonstrate that our proposed model achieves high participation from users, thereby giving better estimations.

II. Related Work

In recent times, research on blockchain-based crowdsourcing has become an emerging trend with the explosive growth of Blockchain 2.0 [22]–[24]. The authors of [25] have investigated the benefits that blockchain technology can bring to crowdsourcing systems by analyzing real-life crowdsourcing use cases. Ming Li *et al.* [26] have proposed a general-purpose decentralized crowdsourcing framework - CrowdBC, based on blockchain. Their model enables a crowd of workers to solve a requester's task without depending on any centralized party. Xiaolong Xu *et al.* [27] have also proposed a general-purpose blockchain-powered crowdsourcing model for privacy preservation in a mobile environment. Although numerous blockchain-based crowdsourcing architectures have surfaced in recent years, none of them has been implemented in the domain of traffic jam prediction and mitigation.

While blockchain-based crowdsourcing has not been adopted in the domain of traffic jam prevention and mitigation, several attempts employing different architectures have been made in the industry and academia to develop strategies facilitating smooth movement of traffic [4], [28]. Numerous researchers have proposed the use of modern technologies such as artificial intelligence, cloud computing and fog computing for traffic jam prevention. Bo Xie *et al.* [2] have proposed an Urban Cell Transmission Model (UCTM) to evaluate urban traffic jam condition. The model takes into account the density of each cell which consists of three different directions for inflow and outflow. Jiancheng Long *et al.* [5] have proposed control strategies for incident-based traffic jams in a two-way grid network. The control strategies proposed by the authors include diamond control, single-line control, area control, and multiline control. Zuchao Wang *et al.* [13] have proposed a scheme for visual traffic jam analysis based on trajectory data. The authors have proposed various strategies to derive accurate traffic jam information. The authors of [29] have formulated the traffic allocation problem for multipath routing as a lossy network flow optimization problem using the portfolio selection theory. The authors of [30] have proposed a jam absorption driving (JAD) strategy to guide the vehicles in a way that reduces traffic oscillations. Myounggyu Won *et al.* [14] have used the concept of the vehicle to vehicle communication to mitigate phantom jam conditions. A three-phase traffic theory and fuzzy interference system have been used by the authors to capture the traffic jam dynamics accurately.

Various researchers have employed the use of machine learning and game theory strategies for traffic jam detection and mitigation [31]. Kazuhiro Saitou *et al.* [15] have proposed a real-time traffic prediction and probing strategy for lagrangian traffic data. The authors of [32] propose an AI-based malicious network traffic detection in VANETs to tackle the malicious entities that deliberately want to cause traffic congestion. The authors use statistical network traffic analysis with data mining to detect malicious entities in vehicular ad-hoc networks. The authors of [33] use spatiotemporal correlation for discrimination and prediction of traffic congestion.

Although there have been various works in the direction of traffic jam detection and mitigation, most of them make use of prediction algorithms based on historical traffic data. The reasons and conditions leading to a traffic jam are dynamic and unpredictable. In such situations, the best way to know the live traffic situation at a particular location is to get the information directly from the people who are stuck at that particular location. This calls for a secure and efficient crowdsourcing model for traffic jam probability estimation. The proposed blockchain-based crowdsourcing model satisfies the required security constraints while ensuring a high level of participation due to an incentive-based approach [34], [35], [36]. A detailed system model explaining the steps involved

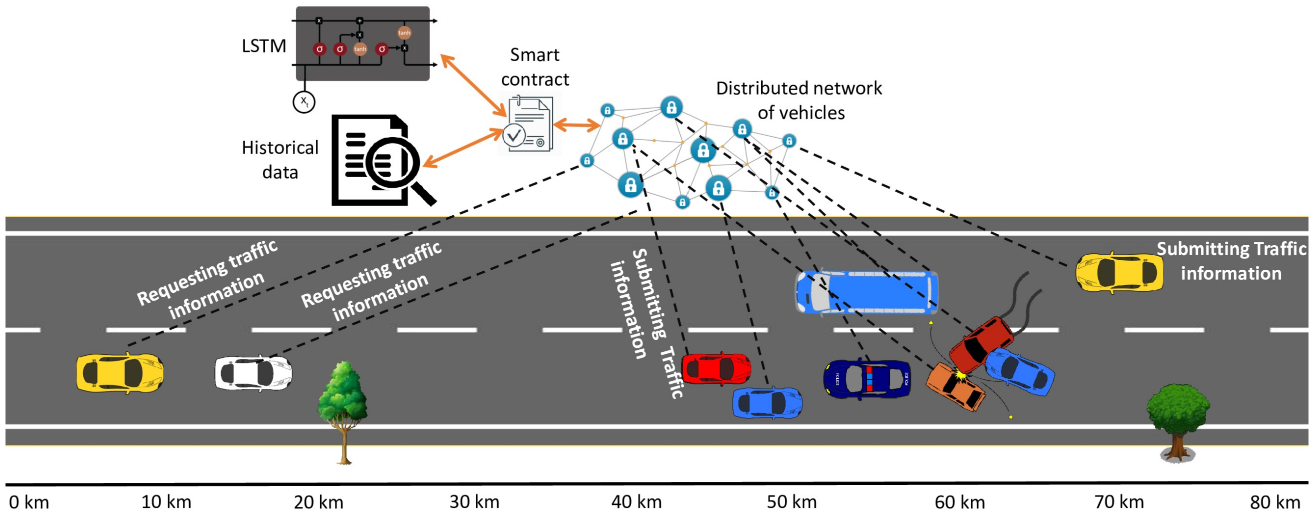

LSTM
Smart contract
Distributed network of vehicles
Historical data
Requesting traffic information
Requesting traffic information
Submitting Traffic information
Submitting Traffic information
0 km
10 km
20 km
30 km
40 km
50 km
60 km
70 km
80 km

Fig. 1: System model for proposed framework

in our crowdsourcing approach has been presented in the following section.

III. System Model

Fig. 1 illustrates the steps involved in the proposed model. The vehicles belong to the same blockchain network and are allowed to enter or leave the network at any given time. Owing to the use of resource-constrained mobile devices for running the ethereum wallet application and executing transactions, the generic proof-of-work consensus algorithm is not suitable for our model. Therefore, we deploy the proof-of-authority consensus algorithm in our model to validate the data as well as the transactions. A set of specialized nodes in the network with comparatively higher computation power act as authorities to validate the data. The steps involved in the execution of the framework are as follows:

1. An account is created as soon as a vehicle seeks to enter or participate in the network. Each account is associated with a unique set of account address, private key, and public key. The data shared by any user on the network is digitally signed using the allocated private key to prevent the issue of non-repudiation.
2. Whenever a user encounters an unusual situation on the road, the user may decide to share the incident on the network. The shared information will include some basic parameters such as the type of road, condition of the road, event or incident observed, or any other relevant detail.
3. The smart contract deployed on the network verifies all the events specified by the users and multiple entries for the same event from different users confirms the authenticity of the event. To ensure that the same users do not submit repeated information for monetary benefits, only the first user to share unique information is awarded some tokens. This motivates the users to share the information as accurately and as early as possible.
4. The verified information is passed through the Long Short Term Memory (LSTM) [37] neural network to predict the probability of traffic jam at a particular location at a particular time.
5. The historical data is also taken into account while calculating the probability of traffic jam. Historical data analysis is done in advance using a 3-layer feed-forward artificial neural network (ANN).
6. The probability estimates from both the models are combined to generate the final traffic jam probability estimate.
7. Any user in the network can request a traffic jam probability estimate of a particular location at any instance of time by making use of the earned tokens. Algo. 2 lists all the steps involved in final traffic jam probability estimation.
8. As a future extension of this work, we plan to introduce a feature that suggests alternative routes to the user based on the current probability estimate of a traffic jam at different locations.

The following section discusses the proposed network model and the mathematics behind the traffic jam probability calculation in detail.

IV. Proposed Network Model

Consider a set of vehicles $\mathcal{V} = \{V_1, V_2, V_3, \dots, V_i, \dots, V_n\}$ where n denotes the total number of vehicles in our blockchain network. Any vehicle, $V_i \in \mathcal{V}$, can either request for the traffic jam probability of a certain location at a specific time, or, offer its own location information in exchange for some tokens. In our work, two models, namely, Live Data-Based Probability Estimation (LDBPE) and Historical Data-Based Probability Estimation (HDBPE), have been used for traffic jam probability estimation, since both, live and historical data can provide useful insights into the traffic intensity at a particular location.

A. LDBPE using an LSTM Neural Network

Consider a day to be divided into 48 equal timeslots (30 minutes each). The set of vehicles offering their location information during a specific timeslot t, from a specific location l can be represented using:

$$\mathcal{V}_l^t = \{V_1, V_2, \dots, V_k, \dots, V_m\} \tag{1}$$

TABLE II: Values of ω, γ, and π

Type Of Road (ω)	Expressway
	National Highway
	State Highway
	Major District Road
	Other District Road
	Rural Road
Road condition (γ)	Smooth
	Moderate
	Poor
	Extremely Bad
Event on the Road (π)	Construction
	Accident
	Barricades
	Toll Booth
	Demonstrations or Protests
	Marathon
	Others

where $t \in [1, 48]$. Each vehicle, $V_k \in \mathcal{V}_l^t$, sends its location information vector, λ_k, which can be represented using:

$$\lambda_k = \{T_k, l_k, v_k, \alpha_k, \omega_k, \gamma_k, \pi_k\} \tag{2}$$

where $T_k \rightarrow$ Timestamp of the vehicle V_k,
$l_k \rightarrow$ Location of the vehicle V_k,
$v_k \rightarrow$ Velocity of the vehicle V_k,
$\alpha_k \rightarrow$ Anticipated Duration of Slowness on the road (in minutes) as given by the vehicle V_k,
$\omega_k \rightarrow$ Type of Road as given by the vehicle V_k,
$\gamma_k \rightarrow$ Condition of the Road as given by the vehicle V_k,
$\pi_k \rightarrow$ Special event on the road (if any) as given by the vehicle V_k.
All individual location information vectors, as sent by the vehicles $V_k \in \mathcal{V}_l^t$ must be aggregated to form a single live data vector for estimating the traffic jam probability at a particular location during a specific timeslot. The live data vector, Λ_l^t, can be represented as:

$$\Lambda_l^t = \{\mu_l^t, s_l^t, \Omega_t^l, \Gamma_l^t, \Pi_l^t\} \tag{3}$$

where

$$\mu_l^t = \frac{\sum_{k=1}^{m} v_k}{m} \tag{4}$$

represents the average velocity of vehicles at location l during timeslot t, and

$$s_l^t = \frac{\sum_{k=1}^{m} \alpha_k}{m} \tag{5}$$

represents the average anticipated duration of slowness at location l. The type of road, condition of the road, and any special event occurring on the road, as given by the maximum number of vehicles are represented by Ω_l^t, Γ_l^t and Π_l^t respectively. This vector is then fed into the LSTM neural network for estimating live traffic jam probability of that location at that particular timeslot.

B. HDBPE using a 3-Layer Feed Forward Neural Network

It is essential to incorporate historical data in the probability estimation of traffic jams since it can provide excellent insights into the traffic intensity trends. In our work, a simple three-layer feed-forward ANN has been used for HDBPE.

C. Training a Neural Network

For implementing any supervised machine learning or deep learning model, there is a need to train the model before deploying it to make predictions on new data. For LSTM neural network, all the examples in our training dataset include 5 input features (dependent variables): 1) the average velocity (μ_i), 2) the anticipated duration of slowness (s_i), 3) the type of road (Ω_i), 4) the condition of the road (Γ_i), 5) special event on the road if any (Π_i) and an output label (independent variable) y_i. In the case of feed-forward ANN, the input features include: 1) Timeslot of the day, τ, 2) day of the week, $\mathcal{D}_w$, 3) day of the month, $\mathcal{D}_m$, and 4) month of the year, $\mathcal{M}$, and an output label y_i. For both our models, the output label y_i can take only two values:

$$y_i = \begin{cases} 1 & \text{if traffic jam} \\ 0 & \text{if no traffic jam} \end{cases} \tag{6}$$

After pre-processing the data, the training dataset is split into mini-batches consisting of $\mathcal{S}$ examples. If $\mathcal{N}$ denotes the total number of examples in the training set, then:

$$\mathcal{Q} = \left\lceil \frac{\mathcal{N}}{\mathcal{S}} \right\rceil \tag{7}$$

denotes the total number of mini-batches formed. Let θ_t represent the parameter vector at batch-timestep t, i.e., the vector comprising of all the weight matrices and bias vectors associated with the neural network during the t^{th} mini-batch.

In our work, we have used the Adam optimization algorithm [38] over the traditional stochastic gradient descent method to update network weights iteratively (refer algo. 1). Adam is an extension of stochastic/mini-batch gradient descent and provides the benefits of both AdaGrad and RMSprop [39]. Adam algorithm requires us to maintain two additional vectors: first-moment vector, f_t, and second-moment vector, s_t, at each batch-timestep t. Initially, both the first and the second-moment vector are assigned null values:

$$f_0 = 0, s_0 = 0 \tag{8}$$

After random initialisation of all the weights and biases, our model makes predictions for each example of the first mini-batch. A cost function $J(\theta)$ is then used to calculate the error in the predictions made by our model. If $p(y_i)$ denotes the probability outcome of a traffic jam as given by our prediction model, then the log loss (also known as binary cross-entropy) cost function can be calculated as:

$$J(\theta) = -\frac{1}{\mathcal{S}} \sum_{i=1}^{\mathcal{S}} y_i * \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) * \log\left(1 - p\left(y_i\right)\right) \tag{9}$$

where,

$$0 \leq p(y_i) \leq 1 \tag{10}$$

Since a cost function should penalize wrong predictions, the function established in the equation (9) works very well for our model. For each example with $y_i = 1$, it adds the negative log probability of the occurrence of a traffic jam to the function, i.e. if our model outputs a low probability of a traffic jam occurring, then the negative log probability will be high which will increase the value given by cost function and vice versa. After our model is trained on the whole mini-batch, the gradient of error is calculated with respect to all the parameters in the parameter vector θ_t using the following equation:

$$g_t = \frac{\partial J(\theta)}{\partial \theta_t} = \frac{1}{\mathcal{S}} \sum_{i=1}^{\mathcal{S}} \frac{\partial J_i}{\partial \theta_t} \tag{11}$$

Algorithm 1 Training a Neural Network using Adam Optimizer

Input: Specified number of Epochs $\mathcal{E}$, Training Dataset with $\mathcal{N}$ training examples, Batch Size $\mathcal{S}$, Step Size κ, Exponential decay rates for moment estimates ρ_1 and ρ_2, Small constant ϱ for numerical stabilisation, Initial Parameter Vector θ_0

Output: Final Parameter Vector, θ_{final}

```
1:  Calculate the number of batches, $\mathcal{Q} = \frac{\mathcal{N}}{\mathcal{S}}$
2:  Initialise all the weight and bias matrices with random
    weights and biases respectively.
3:  Initialise $1^{st}$ and $2^{nd}$ moment vector; $f_0 = 0$, $s_0 = 0$
4:  Initialise batch-timestep $t = 0$
5:  for each epoch $\in (1, \mathcal{E})$ do
6:      for each mini-batch $\in (1, \mathcal{Q})$ do
          Forward Pass
7:          for each training example $i \in (1, \mathcal{S})$ do
8:              Use Procedure 1 (for LSTM neural
                network) or Procedure 2 (for feed-forward
                neural network) to make predictions for each
                example of the current mini-batch.
9:          end for
10:         Use equation (9) to compute the cost function (log
            loss) with the current values of parameters.
11:         Update batch-timestep, $t = t + 1$.
12:         Compute the gradient for all the weights and biases
            in the parameter vector $\theta_t$, using the formula
            established in equation (11).
13:         Update biased first and second moment estimate
            using equations (12) and (13) respectively.
14:         Compute bias-corrected first and second moment
            estimate using equations (15) and (16) respectively.
15:         Use equation (17) to update all the parameters in
            parameter vector $\theta_t$.
16:     end for
17: end for
```

The gradient, g_t, is then used to update the biased first and second moment estimates based on the following equations:

$$f_t = \rho_1 * f_{t-1} + (1 - \rho_1) * g_t \tag{12}$$

$$s_t = \rho_2 * s_{t-1} + (1 - \rho_2) * (g_t)^2 \tag{13}$$

where ρ_1 and ρ_2 denote the exponential decay rates for moment estimates.

$$0 \leq \rho_1, \rho_2 < 1 \tag{14}$$

Following this, bias-corrected first moment estimate $\hat{f}_t$ and second moment estimate $\hat{s}_t$ are calculated.

$$\hat{f}_t = \frac{f_t}{1 - (\rho_1)^t} \tag{15}$$

$$\hat{s}_t = \frac{s_t}{1 - (\rho_2)^t} \tag{16}$$

Finally, the weights and biases present in the parameter vector θ_t are updated using the following equation:

$$\theta_t = \theta_t - \kappa \frac{\hat{f}_t}{\sqrt{\hat{s}_t} + \varrho} \tag{17}$$

where κ is the step size and ϱ is a small constant (in the order of 10^{-8}) used for numerical stabilisation.

D. Working of the LSTM Neural Network to make predictions

In recent times, LSTM neural networks have emerged as one of the most suitable approaches for short term traffic forecast [40], [41]. To ensure reliable results, we have used a 3-layer stacked LSTM (refer fig. 2) for our problem. However, to prevent overfitting of the model on the training dataset, we add dropout regularisation between the LSTM layers [42]. Figure 2 shows the working and interaction of three LSTM cells across three LSTM layers.

Each input vector, $\mathcal{X}_i$, consisting of all the input features, passes through the LSTM cells present in all the LSTM layers before passing through the neurons in the final output layer. The gates responsible for the working of an LSTM cell are briefly explained below:

Forget Gate: The forget gate takes two input values - the previous hidden state, $\mathcal{H}_{i-1}$, and the current input vector, X_i. The input values are multiplied by the weight matrices, following which a bias present in the bias vector, $\mathcal{B}_f$, is added. The resultant value is passed through the sigmoid function, which outputs a value in the range of 0 and 1. The values closer to 0 are forgotten, while the values closer to 1 are kept.

$$f_i = \sigma(\mathcal{X}_i * \mathcal{U}_f^i + \mathcal{H}_{i-1} * \mathcal{W}_f^i + \mathcal{B}_f^i) \tag{18}$$

$$0 \leq f_i \leq 1 \tag{19}$$

Input Gate: The input gate is responsible for updating information in the cell state. The steps involved while updating the cell state are as follows:

$$\bar{c}_i = tanh(\mathcal{X}_i * \mathcal{U}_c^i + \mathcal{H}_{i-1} * \mathcal{W}_c^i + \mathcal{B}_c^i) \tag{20}$$

$$\mathcal{C}_i = f_i * \mathcal{C}_{i-1} + \mathcal{I}_i * \bar{c}_i \tag{21}$$

$$\mathcal{I}_i = \sigma(\mathcal{X}_i * \mathcal{U}_i^i + \mathcal{H}_{i-1} * \mathcal{W}_i^i + \mathcal{B}_i^i) \tag{22}$$

Output Gate: The output gate is responsible for determining the hidden state, $\mathcal{H}_i$ for the next timestep. The hidden state consists of all information about the previous inputs and is required to make predictions. Finding the hidden state for the next timestep is a two-step process:

$$\mathcal{O}_i = \sigma(\mathcal{X}_i * \mathcal{U}_o^i + \mathcal{H}_{i-1} * \mathcal{W}_o^i + \mathcal{B}_o^i) \tag{23}$$

$$\mathcal{H}_i = \mathcal{O}_i * tanh(\mathcal{C}_i) \tag{24}$$

A vector consisting of hidden states of all the LSTM cells in

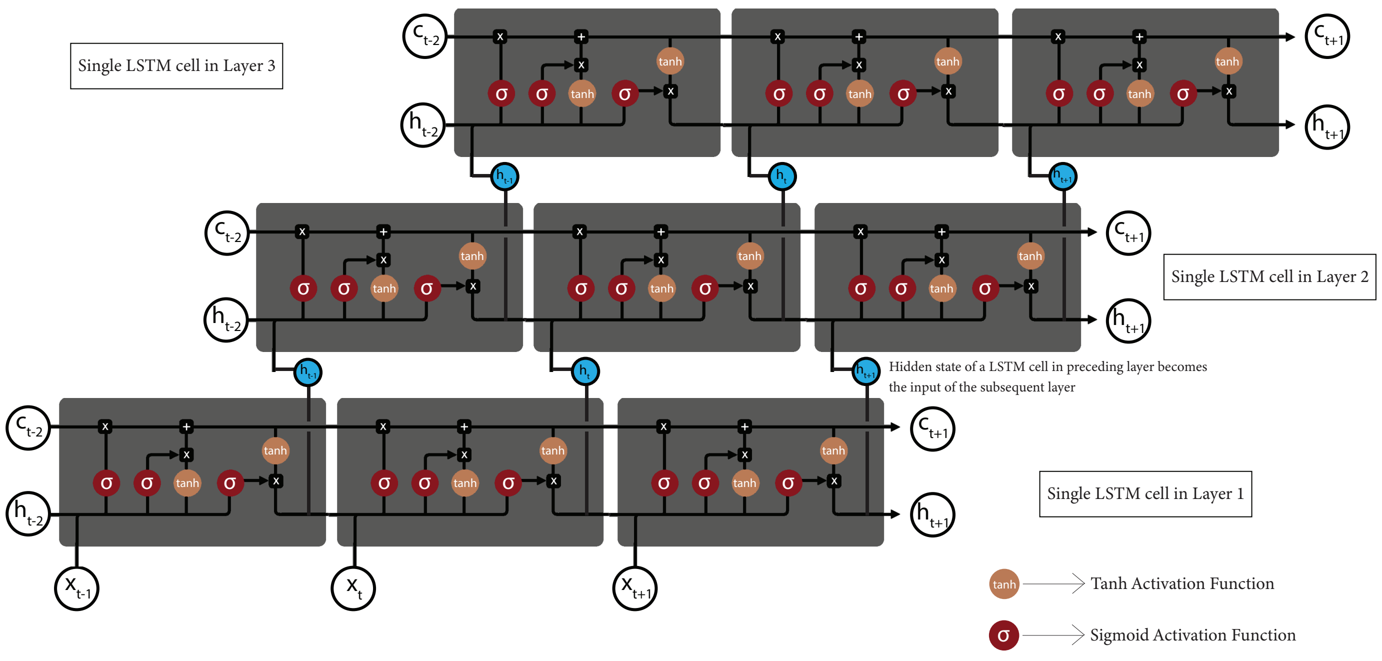

Single LSTM cell in Layer 3
Single LSTM cell in Layer 2
Single LSTM cell in Layer 1
Hidden state of a LSTM cell in preceding layer becomes the input of the subsequent layer
tanh → Tanh Activation Function
σ → Sigmoid Activation Function

Fig. 2: Stacked LSTM Neural Network

the first LSTM layer at timestep i acts as the input $X_i^{[2]}$ for LSTM cells in the next LSTM layer in the same timestep. If there are b number of LSTM cells in the first LSTM layer, then:

$$\mathcal{F}_i^{[1]} = (\mathcal{H}_i^1, \mathcal{H}_i^2, \dots, \mathcal{H}_i^b) \tag{25}$$

$$X_i^{[2]} = \mathcal{F}_i^{[1]} \tag{26}$$

Similarly, for the third LSTM layer,

$$X_i^{[3]} = \mathcal{F}_i^{[2]} \tag{27}$$

Finally, the hidden state vector of the final LSTM layer is fed into the output layer, which uses the sigmoid activation function to map the prediction as a probability between (0, 1).

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{28}$$

The final prediction made by our model can be mathematically modelled as:

$$p(y_i) = \sigma(\mathcal{F}_i^{[3]} * \mathcal{G}_i + \mathcal{E}_i) \tag{29}$$

where $\mathcal{G}_i$ and $\mathcal{E}_i$ represent the output layer weight matrix and bias vector at timestep i respectively.

E. Working of the feed-forward ANN

In a feed-forward ANN, output prediction of any example depends only on the current set of input features. Therefore, working of a cell in a feed-forward ANN is relatively simpler as compared to that of an LSTM cell. Suppose we have to make a prediction on an input vector, $\mathcal{X}_i$:

$$\mathcal{X}_i = \{\tau, \mathcal{D}_w, \mathcal{D}_m, \mathcal{M}\} \tag{30}$$

Initially, this input vector, $\mathcal{X}_i$, passes through the first hidden layer. In each cell of the first hidden layer, the input vector is multiplied by the weight matrices of the respective cells, after

Procedure 1 Working of the LSTM Neural Network to make predictions

Input: Previous Cell Memory $\mathcal{C}_{i-1}$, Previous Cell Output/ Hidden State $\mathcal{H}_{i-1}$, Current Input Vector $\mathcal{X}_i$, LSTM layer weight matrices for the current iteration $\mathcal{U}_f^i$, $\mathcal{W}_f^i$, $\mathcal{U}_c^i$, $\mathcal{W}_c^i$, $\mathcal{U}_p^i$, $\mathcal{W}_p^i$, $\mathcal{U}_o^i$, $\mathcal{W}_o^i$, LSTM layer bias vectors for the current iteration $\mathcal{B}_f^i$, $\mathcal{B}_c^i$, $\mathcal{B}_p^i$, $\mathcal{B}_o^i$, Output layer weight matrix $\mathcal{G}_i$ and bias vector $\mathcal{E}_i$ for the current iteration

Output: Probably outcome for the current example $p(y_i)$

```
for each LSTM layer k ∈ LSTM neural network do
    for each cell ∈ LSTM layer k do
        1. Forget Gate
        $f_i = \sigma(\mathcal{X}_i * \mathcal{U}_f^{i[k]} + \mathcal{H}_{i-1} * \mathcal{W}_f^{i[k]} + \mathcal{B}_f^{i[k]})$
        2. Input Gate
        $\mathcal{I}_i = \sigma(\mathcal{X}_i * \mathcal{U}_p^{i[k]} + \mathcal{H}_{i-1} * \mathcal{W}_p^{i[k]} + \mathcal{B}_p^{i[k]})$
        $\bar{c}_i = tanh(\mathcal{X}_i * \mathcal{U}_c^{i[k]} + \mathcal{H}_{i-1} * \mathcal{W}_c^{i[k]} + \mathcal{B}_c^{i[k]})$
        3. Cell State
        $\mathcal{C}_i = f_i * \mathcal{C}_{i-1} + \mathcal{I}_i * \bar{c}_i$
        4. Output Gate
        $\mathcal{O}_i = \sigma(\mathcal{X}_i * \mathcal{U}_o^{i[k]} + \mathcal{H}_{i-1} * \mathcal{W}_o^{i[k]} + \mathcal{B}_o^{i[k]})$
        $\mathcal{H}_i = \mathcal{O}_i * tanh(\mathcal{C}_i)$
    end for
    Create a vector of hidden states given by the cells of the current LSTM layer, $\mathcal{F}_i^{[k]} = (\mathcal{H}_i^1, \mathcal{H}_i^2, \dots, \mathcal{H}_i^j, \dots, \mathcal{H}_i^b)$ where $\mathcal{H}_i^j$ denotes the hidden state output of the $j^{th}$ cell in the $k^{th}$ LSTM layer.
    Initialise input vector, $\mathcal{X}_i$ for the next LSTM layer as the current hidden states vector, $\mathcal{F}_i^{[k]}$:
    $\mathcal{X}_i = \mathcal{F}_i^{[k]}$
end for
Let Final Layer Hidden State Vector, $\mathcal{F}_i = \mathcal{F}_i^{[k]}$
Final Ouput Layer
$p(y_i) = \sigma(\mathcal{F}_i * \mathcal{G}_i + \mathcal{E}_i)$
```

which the corresponding bias vectors are added.

$$\mathcal{Z}_i = \mathcal{X}_i * \mathcal{W}_i^{[k]} + \mathcal{B}_i^{[k]} \tag{31}$$

To limit the vanishing gradient problem, each hidden layer in our model uses a Rectified Linear Unit (ReLU) activation function, which can be mathematically modelled as:

$$R(x) = \max(0, x) \tag{32}$$

The final output, $\mathcal{A}_i$, from each cell is given as:

$$\mathcal{A}_i = \max(0, \mathcal{Z}_i) \tag{33}$$

A vector consisting of the outputs from the first hidden layer, $\mathcal{O}_i^{[1]}$ acts as the input vector for the second hidden layer. If there are m number of cells in the first hidden layer, then:

$$\mathcal{O}_i^{[1]} = (\mathcal{A}_i^1, \mathcal{A}_i^2, \ldots, \mathcal{A}_i^m) \tag{34}$$

$$\mathcal{X}_i^{[2]} = \mathcal{O}_i^{[1]} \tag{35}$$

Similarly, for the third hidden layer:

$$\mathcal{X}_i^{[3]} = \mathcal{O}_i^{[2]} \tag{36}$$

Procedure 2 Working of the Feed-Forward ANN to make predictions

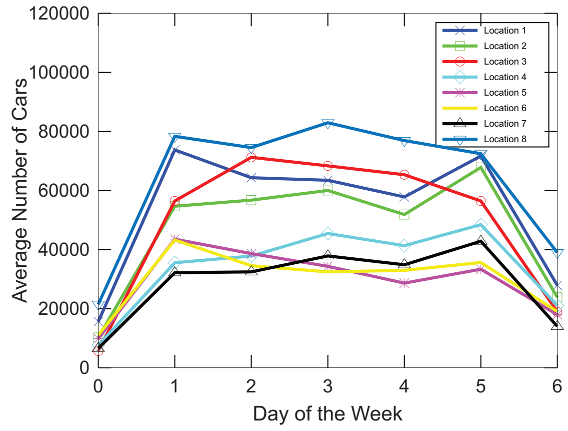
```
Input: Current Input Vector $\mathcal{X}_i$, Hidden layer weight matrices and bias vectors for the current iteration, Output layer weight matrix $\mathcal{G}_i$ and bias vector $\mathcal{E}_i$ for the current iteration
Output: Probably outcome for the current example $p(y_i)$
  for each hidden layer k $\in$ Feed-Forward ANN do
      for each cell $\in$ hidden layer k do
          $\mathcal{Z}_i = \mathcal{X}_i * \mathcal{W}_i^{[k]} + \mathcal{B}_i^{[k]}$
          $\mathcal{A}_i = \max(0, \mathcal{Z}_i)$
      end for
      Create a vector $\mathcal{O}_i$, consisting of outputs from the cells in the current hidden layer:
      $\mathcal{O}_i = (\mathcal{A}_i^1, \mathcal{A}_i^2, \ldots, \mathcal{A}_i^j, \ldots, \mathcal{A}_i^m)$ where $\mathcal{A}_i^j$ denotes the output from the $j^{th}$ cell in the current hidden layer.
      Initialise the input for the subsequent input layer as the output vector of the current hidden layer:
      $\mathcal{X}_i = \mathcal{O}_i$
  end for
  Final Output Layer
  $p(y_i) = \sigma(\mathcal{O}_i * \mathcal{G}_i + \mathcal{E}_i)$
```

Finally, the vector $\mathcal{O}_i^{[3]}$, consisting of final hidden layer cell outputs, passes through the output layer which uses a sigmoid activation function to output a probability. If $\mathcal{G}_i$ and $\mathcal{E}_i$ denote the weight and bias vector for the output layer respectively, then,

$$p(y_i) = \sigma(\mathcal{O}_i^{[3]} * \mathcal{G}_i + \mathcal{E}_i) \tag{37}$$

F. *Combining the two probabilities to give a final estimation*

Any vehicle at a given time would request for location information of a particular place. Using the data received from the vehicles in that same place during that time, we will use our LSTM network to output a traffic jam probability $\mathcal{P}_{LDBPE}$. Time of the day, the month of the year, day of the week, day of the month will be extracted from the current timestamp, which will act as the input for our feed-forward ANN. Our feed-forward ANN will make a prediction on this input $\mathcal{P}_{HDBPE}$. However, instead of sending two different probability outcomes to the requesting user, we will combine these two probabilities by performing conflation on the two probability outcomes to achieve a single output probability [43]. Conflation is an optimal method for consolidating data from independent sources that measure the same physical



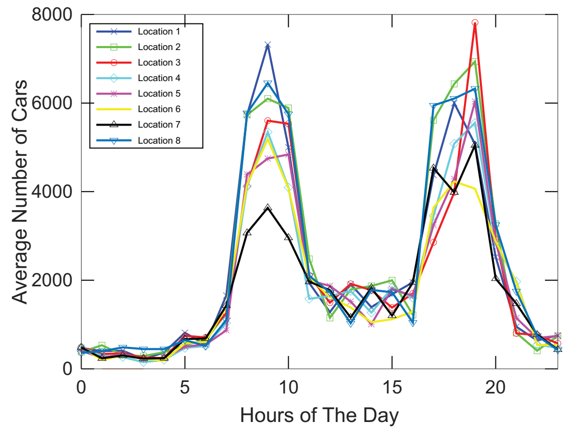


Fig. 3: Traffic Insights obtained from the Historical Data

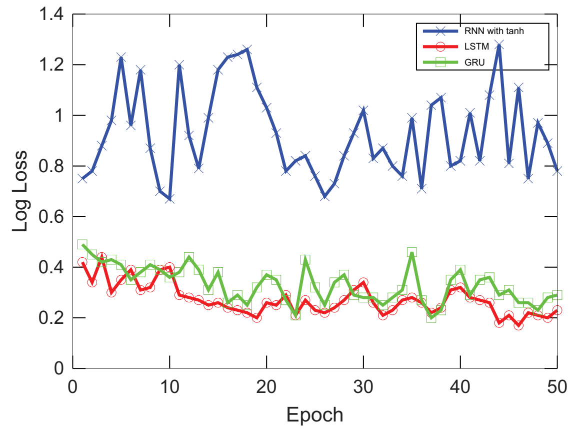


Fig. 4: Comparison of LSTM with other RNNs on our dataset

quantity in an unbiased manner. To calculate the conflated probability, the probabilities from different sources, in this case, $\mathcal{P}_{LDBPE}$ and $\mathcal{P}_{HDBPE}$, are multiplied and then renormalized [43].

$$\mathcal{P}_{final} = \frac{\mathcal{P}_{\mathcal{LDBPE}} * \mathcal{P}_{\mathcal{HDBPE}}}{\mathcal{P}_{\mathcal{LDBPE}} * \mathcal{P}_{\mathcal{HDBPE}} + (1 - \mathcal{P}_{\mathcal{LDBPE}}) * (1 - \mathcal{P}_{\mathcal{HDBPE}})} \tag{38}$$

Conflation has been chosen over other bayesian methods for consolidating probabilities since the conflation of probabilities a) preserves the proportionality of likelihoods, and b) minimizes the maximum loss of Shannon information.

Algorithm 2 Traffic Jam Probability Estimation based on Blockchain and Deep Neural Networks

Input: Request from a vehicle for traffic jam probability in location l at timeslot t.

Output: Final Probability Estimation of traffic jam at location l during timeslot t, p_l^t

```
1: Obtain location information from all the vehicles willing to send the information.
2: Assign few tokens to all the vehicles sending their location information vector $\lambda_k$.
3: Encapsulate the individual location information vectors into one live data vector $\Lambda_l^t$.
4: Feed the vector to our LSTM network for LDBPE, $\mathcal{P}_{LDBPE}$.
5: Feed the current timestamp into the feed-forward ANN for HDBPE, $\mathcal{P}_{HDBPE}$.
6: Use equation (38) to estimate the final traffic jam probability for the current timeslot at the location l, $p_l^t$.
7: Use our HDBPE model to estimate the probabilites of the next two timeslots, $p_l^{t+1}$ and $p_l^{t+2}$.
8: Provide the requesting vehicle with the current traffic jam probability estimate as well as the traffic jam probability estimate for the next two timeslots (using just HDPBE).
```

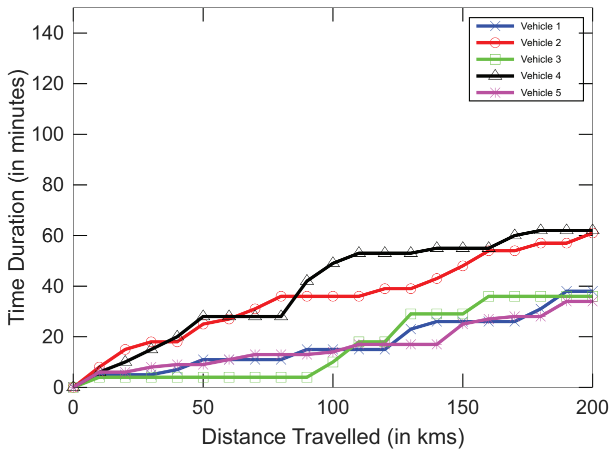
V. Numerical Analysis and Results

A. Simulation Settings

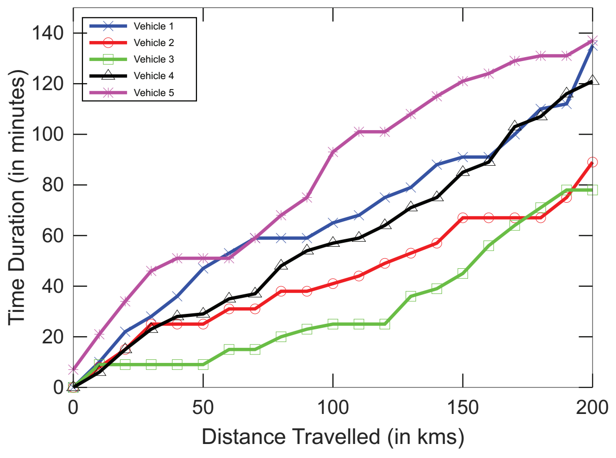
The LSTM has been implemented in python via the use of Keras framework on top of Google's TensorFlow, whereas the three-layer ANN has been implemented using the TensorFlow framework alone. To determine the best parameter values (batch size and number of epochs), we applied 10 fold cross-validation using the GridSearchCV class from the sklearn module in python. For our LSTM network, the optimal batch size and the number of epochs came out to be 28 and 47 respectively, whereas for our feed-forward ANN, batch size of 31 and 16 epochs provided highly accurate results with low loss (refer fig. 6). Furthermore, following the popular choice of step size, we have selected the step size as 0.001. For exponential decay rates, we have chosen the suggested default values of 0.9 and 0.999 for fast convergence.

B. Performance Evaluation

Some of the graphs obtained from the analysis of historical data are shown in fig. 3. These offer useful insights into the patterns of traffic volume at different days of the week and at different times of the day. It can be concluded from the graphs that weekdays usually have higher traffic than weekends, especially in the morning and evening hours.



(a) Participation in Existing Crowdsourcing Model



(b) Participation in Proposed Crowdsourcing Model

Fig. 5: Comparison of the Participation Level in Proposed Model and Existing Model

Fig. 4 compares the losses given by three different RNNs trained on our dataset. As it can be seen from the graph, LSTM and the Gated Recurrent Unit (GRU) outperform the vanilla RNN. Although both LSTM and GRU manage to achieve a low log loss, LSTM is able to do so consistently and is, therefore, the best option for our model.

Fig. 5 demonstrates that most vehicle owners used the existing architecture to look at traffic data only at the beginning or at the end of their journey, with only a few of them using the application for a long time. However, due to the added incentive, the same vehicle owners participated for an extended period of time in our model by sending location information regularly.

Fig. 6 evaluates the performance of different optimizers for the same feed-forward ANN. RMSProp achieves a 100% accuracy on the training dataset, but its low validation accuracy and high validation loss indicate that it overfits on the training dataset. While SGD does not achieve high accuracy on the

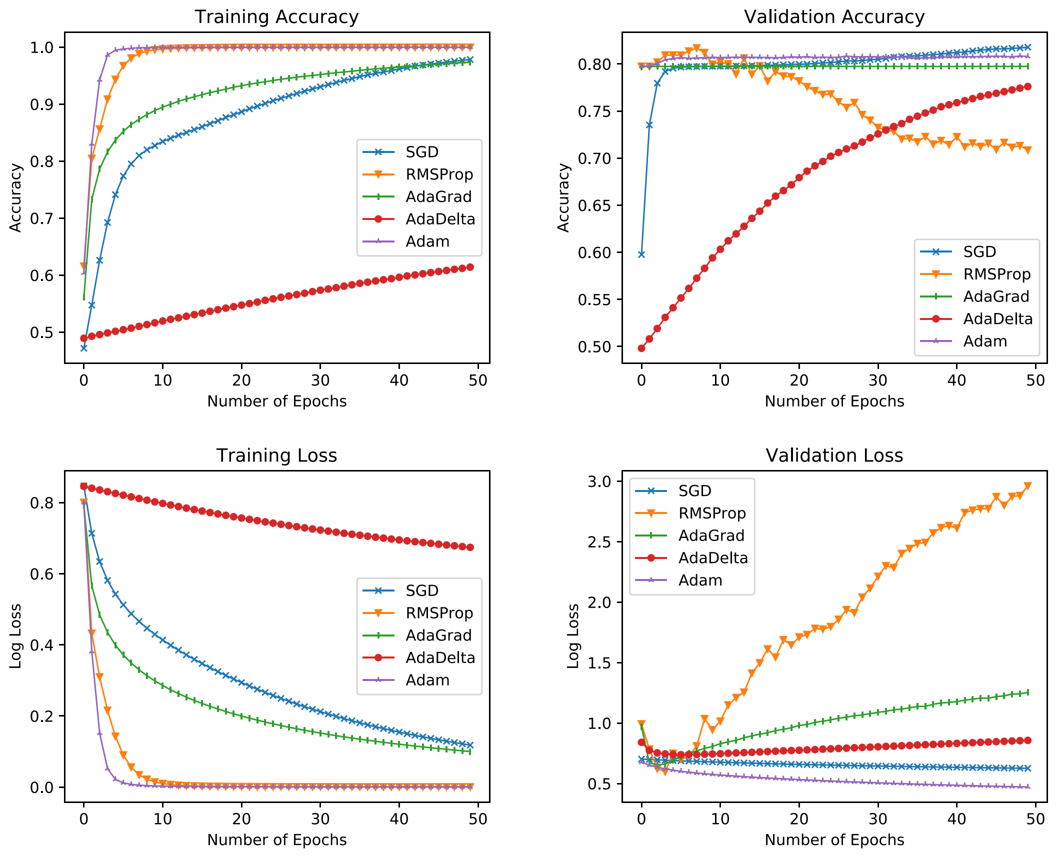


Fig. 6: Comparison of Adam with other Optimizers for our Feed Forward ANN

training dataset, its high validation accuracy and low validation loss indicate that it performs well on the validating dataset. However, among all the optimizers on our dataset, Adam is the clear winner as it achieves high validation accuracy and low validation loss while being able to converge faster.

VI. Conclusion

In light of the two fundamental limitations of existing crowdsourcing models, namely, privacy issues and lack of motivation, in this paper, we proposed a blockchain-based crowdsourcing-model for traffic jam probability estimation. As a part of our blockchain network, the user can earn tokens by sharing live traffic information with the network and can later use these tokens for obtaining traffic information from the network. We employed an LSTM neural network for live data based probability estimation of traffic jam while incorporating results from a feed-forward ANN trained on historical data. The results show that our model achieves a higher level of participation from the users due to the added incentive, which consequently resulted in highly accurate results.

References

[1] J. Ni, X. Lin, K. Zhang, and X. Shen, "Privacy-Preserving Real-Time Navigation System Using Vehicular Crowdsourcing," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Sep. 2016, pp. 1–5.

[2] B. Xie, Y. Chen, and M. Xu, "Evaluating urban traffic jam based on a urban cell transmission model (UCTM)," in *2012 12th International Conference on ITS Telecommunications*, Nov 2012, pp. 211–215.

[3] M. Lab, "U.S. is the world leader in traffic jams – USA Today," Nov 2019. [Online]. Available: https://mobilitylab.org/2018/02/06/u-s-is-the-world-leader-in-traffic-jams/

[4] T. James, "How to ...... cut traffic jams," *Engineering & Technology*, vol. 8, no. 1, pp. 44–47, February 2013.

[5] J. Long, Z. Gao, P. Orenstein, and H. Ren, "Control strategies for dispersing incident-based traffic jams in two-way grid networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 469–481, June 2012.

[6] X. Wan, H. Ghazzai, and Y. Massoud, "Mobile crowdsourcing for intelligent transportation systems: Real-time navigation in urban areas," *IEEE Access*, vol. 7, pp. 136 995–137 009, 2019.

[7] X. Zhang, Z. Yang, and Y. Liu, "Vehicle-based bi-objective crowdsourcing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3420–3428, Oct 2018.

[8] Thomas, "Crowdsourcing from its beginnings to the present," Sep 2018. [Online]. Available: https://www.clickworker.com/2018/04/04/evolution-of-crowdsourcing/

[9] R. Panko, "The popularity of Google maps: Trends in navigation apps in 2018," Mar 2018. [Online]. Available: https://www.themanifest.com/app-development/popularity-google-maps-trends-navigation-apps-2018

[10] R. Greenfield, "Why Waze is worth more than 1 billion," Oct 2013. [Online]. Available: https://www.theatlantic.com/technology/archive/2013/05/waze-google-1-billion/314914/

[11] A. DelColle, "Inside Waze's volunteer workforce," Nov 2017. [Online]. Available: https://www.popularmechanics.com/technology/a15624/waze-volunteer-work-force/

[12] A. Stefanidis, A. Crooks, and A. Croitoru, "How Google's geo-crowdsourcing is transforming the map," Jan 2015. [Online]. Available: https://www.citymetric.com/horizons/how-googles-geo-crowdsourcing-transforming-map-626

[13] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. Van De Wetering, "Visual traffic jam analysis based on trajectory data," *IEEE tran. on visualization and computer graphics*, vol. 19, no. 12, pp. 2159–2168, 2013.

[14] M. Won, T. Park, and S. H. Son, "Toward mitigating phantom jam using

vehicle-to-vehicle communication," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1313–1324, 2016.

[15] K.-C. Chu, R. Saigal, and K. Saitou, "Real-time traffic prediction and probing strategy for lagrangian traffic data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 497–506, 2018.

[16] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on IoT security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, July 2019.

[17] B. Zhou, A. Liu, V. Lau, J. Wen, S. Mumtaz, A. K. Bashir, and S. H. Ahmed, "Performance limits of visible light-based positioning for internet-of-vehicles: Time-domain localization cooperation gain," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[18] S. Garg, K. Kaur, S. Batra, G. Kaddoum, N. Kumar, and A. Boukerche, "A multi-stage anomaly detection scheme for augmenting the security in iot-enabled applications," *Future Generation Computer Systems*, vol. 104, pp. 105–118, 2020.

[19] V. Hassija, G. Bansal, V. Chamola, V. Saxena, and B. Sikdar, "BlockCom: A blockchain based commerce model for smart communities using auction mechanism," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2019, pp. 1–6.

[20] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924–935, Sep. 2019.

[21] Chen, Jianing and Wu, Jun and Liang, Haoran and Mumtaz, Shahid and Li, Jianhua and Konstantin, Kostromitin and Bashir, Ali Kashif and Nawaz, Raheel, "Collaborative trust blockchain based unbiased control transfer mechanism for industrial automation," *IEEE Transactions on Industry Applications*, 2019.

[22] V. Hassija, V. Chamola, G. Han, J. Rodrigues, and M. Guizani, "DAGIoV: A framework for vehicle to vehicle communication using directed acyclic graph and game theory," *IEEE Transactions on Vehicular Technology*, 2020.

[23] S. Garg, K. Kaur, S. Batra, G. S. Aujla, G. Morgan, N. Kumar, A. Y. Zomaya, and R. Ranjan, "En-ABC: An ensemble artificial bee colony based anomaly detection scheme for cloud environment," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 219–233, 2020.

[24] V. Hassija, V. Chamola, D. Nanda Gopala Krishna, and M. Guizani, "A distributed framework for energy trading between UAVs and charging stations for critical applications," *IEEE Transactions on Vehicular Technology*, 2020.

[25] D. G. Kogias, H. C. Leligou, M. Xevgenis, M. Polychronaki, E. Katsadouros, G. Loukas, R. Heartfield, and C. Z. Patrikakis, "Toward a blockchain-enabled crowdsourcing platform," *IT Professional*, vol. 21, no. 5, pp. 18–25, Sep. 2019.

[26] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J. Liu, Y. Xiang, and R. H. Deng, "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1251–1266, June 2019.

[27] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi, and W. Dou, "A blockchain-powered crowdsourcing method with privacy preservation in mobile environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1407–1419, Dec 2019.

[28] V. Hassija, V. Chamola, S. Garg, N. G. K. Dara, G. Kaddoum, and D. N. K. Jayakody, "A blockchain-based framework for lightweight data sharing and energy trading in V2G network," *IEEE Transactions on Vehicular Technology*, 2020.

[29] P. Tague, S. Nabar, J. A. Ritcey, and R. Poovendran, "Jamming-aware traffic allocation for multiple-path routing using portfolio selection," *IEEE/ACM Transactions On Networking*, vol. 19, pp. 184–194, 2010.

[30] Z. He, L. Zheng, L. Song, and N. Zhu, "A jam-absorption driving strategy for mitigating traffic oscillations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 802–813, 2016.

[31] L. Jia, Y. Xu, Y. Sun, S. Feng, L. Yu, and A. Anpalagan, "A game-theoretic learning approach for anti-jamming dynamic spectrum access in dense wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1646–1656, 2018.

[32] N. Lyamin, D. Kleyko, Q. Delooz, and A. Vinel, "AI-based malicious network traffic detection in VANETs," *IEEE Network*, vol. 32, no. 6, pp. 15–21, 2018.

[33] Z. Chen, Y. Jiang, D. Sun, and X. Liu, "Discrimination and prediction of traffic congestion states of urban road network based on spatio-temporal correlation," *IEEE Access*, 2019.

[34] V. Hassija, V. Saxena, and V. Chamola, "Scheduling drone charging for multi-drone network based on consensus time-stamp and game theory," *Computer Communications*, 2019.

[35] X. Lin, J. Wu, S. Mumtaz, S. Garg, J. Li, and M. Guizani, "Blockchain-based on-demand computing resource trading in IoV-assisted smart city," *IEEE Transactions on Emerging Topics in Computing*, 2020.

[36] V. Hassija, V. Saxena, and V. Chamola, "A mobile data offloading framework based on a combination of blockchain and virtual voting," *Software: Practice and Experience*, 2020.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[40] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[41] Yuan-yuan Chen, Y. Lv, Z. Li, and F. Wang, "Long short-term memory model for traffic congestion prediction with online open data," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2016, pp. 132–137.

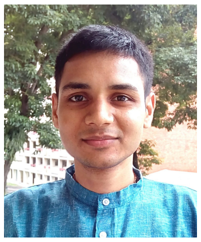
[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[43] T. Hill, "Conflations of probability distributions," *Transactions of the American Mathematical Society*, vol. 363, no. 6, pp. 3351–3372, 2011.

Vikas Hassija received the B.Tech. degree from Maharshi Dayanand University, Rohtak, India, in 2010, and the M.S. degree in telecommunications and software engineering from the Birla Institute of Technology and Science (BITS), Pilani, India, in 2014. He is currently pursuing the Ph.D. degree in IoT security and blockchain with the Jaypee Institute of Information and Technology (JIIT), Noida, where he is currently an Assistant Professor. His research interests include the IoT security, network security, blockchain, and distributed computing.

Vatsal Gupta is currently pursuing a B.Tech. degree from the Jaypee Institute of Information Technology (JIIT), Noida. He has completed a few projects in the field of blockchain applications, machine learning and data analytics. He is currently (the summer of 2020) pursuing his research internship at the Birla Institute of Technology and Science (BITS), Pilani under Dr. Vinay Chamola. His research interests include distributed ledger technology, machine learning and deep learning.

Sahil Garg received his Ph.D. degree from the Thapar Institute of Engineering and Technology, Patiala, India, in 2018. He is currently working as a post-doctoral research fellow at École de technologie supérieure, Université du Québec, Montréal, Canada. He has many research contributions in the area of machine learning, big data analytics, security & privacy, internet of things, and cloud computing. He has over 50 publications in high ranked Journals and Conferences, including 25+ IEEE Trans./Journal papers. He also received the IEEE ICC best paper award in 2018 at Kansas City, Missouri. He serves as the Managing Editor of Springer's Human-centric Computing and Information Sciences (HCIS) and Wiley's International Journal of Communication Systems (IJCS). He also serves as the Lead Guest Editor for special issue of IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Industrial Informatics, IEEE Internet of Things Journal, IEEE Network, and Future Generation Computer Systems (Elsevier). He has served as the workshop chair/publicity co-chair for several conferences including IEEE Infocom, ACM Mobicom, IEEE Globecom and IEEE ICC.



Vinay Chamola received the B.E. degree in electrical and electronics engineering and master's degree in communication engineering from the Birla Institute of Technology and Science, Pilani, India, in 2010 and 2013, respectively. He received his Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2016. In 2015, he was a Visiting Researcher with the Autonomous Networks Research Group (ANRG), University of Southern California, Los Angeles, CA, USA. He also worked as a post-doctoral research fellow at the National University of Singapore, Singapore where he worked in the area of Internet of Things. He is currently an Assistant Professor with the Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani Campus where he heads the Internet of Things Research Group / Lab. He has over 45 publications in high ranked SCI Journals including more than 25 *IEEE Transaction* and Journal articles. His works have been published in Journals like *IEEE Transactions on Communications*, *IEEE Transactions on Vehicular Technology*, *IEEE Journal on Selected Areas in Communications*, *IEEE Communications Magazine* etc. Furthermore, his works have been accepted and presented in reputed conferences like *IEEE INFOCOM, IEEE GLOBECOM, IEEE ICC, IEEE PerCom* to name a few. He has also served as a reviewer for several IEEE/Elsevier Journals. His research interests include IoT Security, Blockchain, 5G network management and addressing research issues in VANETs and UAV networks. He is an Associate Editor of the *IET Quantum Communications* journal and also a Guest Editor in the Computer Communication journal, Elsevier.