Adaptive Model Aggregation for Decentralized Federated Learning in Vehicular Networks

Mahtab Movahedian*, Mahdi Dolati†, and Majid Ghaderi*
*Department of Computer Science, University of Calgary, Calgary, Canada.
†School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.
Emails: {mahtab.movahedianatt, mghaderi}@ucalgary.ca, m.dolati@ipm.ir

***Abstract*—Decentralized federated learning (DFL) enables collaborative training of machine learning models without sharing sensitive data. As such, its application in vehicular networks has gained significant attention. At the core of DFL is the so-called model aggregation, in which each vehicle combines its locally trained model with those received from its neighboring vehicles to generate an updated model that, over time, converges to a global model shared by all vehicles. However, due to high mobility and wireless communication, vehicle-to-vehicle communication is lossy. As a result, a vehicle may receive its neighboring vehicle models only partially. A technical challenge in DFL is how to efficiently utilize such partial models to speed up model training without increasing the training overhead. In this paper, we present an Adaptive Model Aggregation (AMA) algorithm to address this challenge. Our algorithm runs asynchronously on each vehicle and uses a threshold to decide whether to use a partially received model for aggregation. We show that due to the mobility of vehicles, a static threshold is not sufficient and subsequently develop an algorithm based on the contextual multi-arm bandit theory to adaptively compute an optimal threshold for each vehicle based on the dynamics of the network. We evaluate the performance of AMA in realistic environments that include different mobility patterns. Our results show that AMA can decrease DFL aggregation overhead by 83% without reducing the training accuracy compared to non-adaptive aggregation.**

I. Introduction

Modern vehicles are equipped with sensors, cameras, and communication systems that generate a variety of data, including information about traffic conditions. As vehicles become more connected and autonomous, the opportunity arises to leverage this data for improved road safety and driving performance through communication and data sharing among vehicles. However, ensuring privacy and security is crucial when sharing data to protect the drivers and passengers [1].

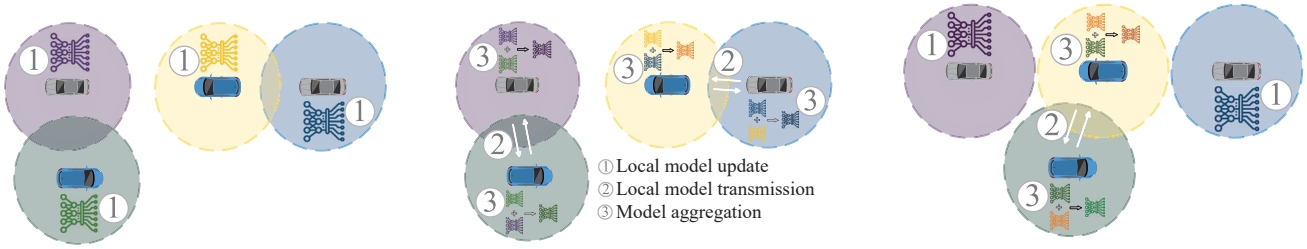
In this context, federated learning [2] is a promising solution for privacy-preserving machine learning (ML) in vehicular networks [3]. In traditional federated learning, a central entity, such as a parameter server, coordinates model training across distributed vehicles. Decentralized federated learning (DFL) [4] extends this approach by enabling vehicles to autonomously coordinate model training without a central entity. In DFL, each vehicle acts as a local learning agent, training its local model based on its own data and then periodically sharing model parameters, such as the weights and biases of a deep neural network, with other vehicles in the network. Upon receiving model parameters from others, a vehicle *aggregates* the parameters of those models with its own model parameters (*e.g.*, by taking their average [2]) to generate an improved model. Over time, each vehicle's local model converges to a global model shared by all vehicles without the need to share raw data among vehicles [5]. Fig. 1 illustrates the three stages of DFL in a vehicular network, namely 1) local model training, 2) model exchange, and 3) model aggregation.

However, deploying DFL in a vehicular network presents technical challenges. In a vehicular network, vehicles use wireless communication (*e.g.*, onboard WiFi) to exchange model parameters with each other. The wireless communication environment is prone to signal degradation due to propagation impairments and interference, resulting in transmission losses at the receivers. Vehicle mobility further exacerbates the *unstability* of the environment. Not only mobility increases transmission errors, but also results in frequent changes in the network topology and neighborhood structure. Consequently, maintaining stable connections for exchanging model parameters between vehicles is challenging in vehicular networks, especially when considering the short range of vehicle-to-vehicle communication and the brief duration of time that vehicles come into contact with each other. When exchanging model parameters over an unstable wireless link, some data packets carrying model parameters may be lost (due to transmission errors, for example). This results in vehicles receiving only partial model parameters from neighboring vehicles. Discarding these partial model parameters would significantly slow down model training [6], particularly in networks with frequent packet losses.

Thus, to ensure efficient model training, it is essential to consider the impact of mobility and wireless communications in vehicular networks. While there is extensive research on DFL in vehicular networks (*e.g.*, see the comprehensive survey [7] and references therein), most existing studies either do not take these aspects into account, as they focus solely on the decentralized learning process itself [8], [9], or they only address one aspect of the problem, such as wireless communication or mobility. For example, in [10], communication is limited to one-hop neighbors, while [11] assumes an ideal communication environment with no packet losses. Similarly, works like [6], [12], [13] assume a static network topology, which fails to capture the dynamic nature of vehicular networks caused by mobility.

In this paper, our objective is to design an aggregation algorithm for decentralized federated learning in vehicular

978-3-903176-59-1 ©2023 IFIP



(a) Vehicles train their models with local data. (b) Vehicles share their models with neighbors. (c) Vehicles aggregate their models with the received models.

Fig. 1: Decentralized federated learning in a vehicular network. The vehicles participate in the training process by exchanging their model parameters with neighboring vehicles. Each vehicle then updates its local model by aggregating the received models with its own.

networks that effectively handles packet losses due to unreliable wireless communication, while minimizing computational overhead. Specifically, we focus on the critical problem of how to utilize *partially received models* in the aggregation process to allow a recipient vehicle to accelerate its model training. As mentioned earlier, simply discarding partial models is not efficient, as it fails to take advantage of the information that is received correctly. Alternatively, blindly incorporating every partially received model during aggregation (*e.g.*, by aggregating only the correctly received parameters [6]) could also negatively impact the model training process. First, when the received partial model is very small, *i.e.*, only a small number of model parameters are received correctly, a bias towards the parameters of the aggregating vehicle may be introduced, leading to an imbalanced final trained model. Second, aggregating and retraining the local model for each received partial model not only results in significant computational overhead, but also creates a lot of traffic in the network to exchange updated model parameters after each aggregation, further exacerbating the packet loss problem.

Our key insight is that if only a small fraction of the parameters of a model are received correctly, it is likely that including the model would not significantly improve the training process. Conversely, if a large fraction of the parameters is received correctly, incorporating the model could potentially enhance model training and convergence. This leads to the question: *how to decide which models should be discarded and which models should be aggregated?* To address this question, we design an algorithm, called Adaptive Model Aggregation (AMA), that uses a threshold on the fraction of the model parameters that are received correctly to determine which partially received models should be included or excluded from the aggregation process. The AMA algorithm is designed to be general and can be combined with existing model aggregation methods, such as FedAvg [2], to improve model training performance in vehicular networks. We show that the optimal threshold value depends on communication reliability and vehicle mobility. To automate threshold selection for different network environments, we design an algorithm based on the contextual multi-arm bandit (CMAB) theory [14]. This algorithm continuously updates the local threshold on each vehicle by considering the communication reliability and mobility patterns of the vehicles in the network.

Our main contributions in this work can be summarized as follows:

- We present a decentralized federated learning approach tailored to vehicular networks, considering both wireless communication and vehicle mobility.
- To handle transmission losses and effectively use partially received models, we design AMA, a threshold-based algorithm to control which models are used in model aggregation, discarding those that would have little to no impact on model accuracy.
- We propose an algorithm to automatically compute the threshold used in AMA by considering dynamic link reliability and mobility patterns, which reduces the algorithm's computational complexity while ensuring high model accuracy.
- We evaluate AMA in different mobility and communication scenarios where vehicles may have different classes of data, demonstrating the effectiveness of our approach in such situations.

Paper Organization. This paper is organized as followed. Section II describes the related works. In Section III we present our aggregation algorithm. Section IV describes the contextual multi-armed bandit formulation and our adaptive threshold selection method. In Section V we provide performance evaluation and analysis. Section VI concludes the paper.

II. Related Works

In this section, we review the existing works that are most relevant to ours. A summary of the related works is presented in Table I.

Decentralized Federated Learning. Federated learning [2] utilizes the Federated Averaging (FedAvg) algorithm, enabling local computation of model updates by multiple devices. A central server aggregates these updates into a global model through weighted averaging. This iterative process continues until convergence is achieved. Numerous extensions of the basic technique are proposed in the literature to, for example, accommodate partial work across devices [15], reduce communication overhead with non-IID data sets [16] or a segmented gossip approach [8], adjust local updates based

TABLE I: Related work overview.

Works	DFL	Communication	Mobility	Description
[2]	✗	Ideal	✗	Aggregation by averaging
[15]	✗	Ideal	✗	Accommodates partial work across devices
[16]	✗	Ideal	✗	Reduces communication overhead
[17]	✗	Ideal	✗	Adjusting local updates based on local loss
[18]	✗	Ideal	✗	Minimize performance variance across devices
[19]	✓	Ideal	✗	Peer-to-peer network
[20]	✓	Ideal	✗	Model exchange and aggregation with one-hop neighbors
[8]	✓	Ideal	✗	Reducing communication overhead using a segmented gossip approach
[9]	✓	Ideal	✗	Introduces an n-out-of-n secret sharing schema
[21]	✓	Ideal	✗	Privacy-preserving protocol, model exchange with one-hop neighbors
[6]	✓	Unreliable	✗	Loss handling and aggregation optimization
[10]	Hybrid	Ideal	Gaussian	Clusterheads collect data and communicate with MEC servers
[13]	✓	Ideal	✗	Improvement in privacy preservation
[12]	✓	Ideal	✗	Improving model stability and handle imbalanced data distribution problems
[11]	✓	Ideal	RWP, CSE	Improving aggregation process
Our Work	✓	Unreliable	RWP, CSE	Dynamic threshold selection

on local loss [17], and minimize performance variance across devices [18]. Decentralized federated learning eliminates the need for a central server by offloading model aggregation to individual devices. This is achieved by forming a peer-to-peer network among the devices that participate in collaboratively training a model [19]. Similar to federated learning, numerous papers have studied extending the basic model aggregation algorithm, for example, a Bayesian-like approach [20], a random selection approach [9], and a hierarchical aggregation technique [21]. The most relevant work to ours is [6], which proposes to replace missing parameters in partially received models with the corresponding parameters from the local model on the recipient device so that every partial model can be used for aggregation. However, as discussed in Section I, blindly incorporating every partial model in the aggregation process leads to learning bias as well as increased computation and communication overhead.

Decentralized Federated Learning in Vehicular Networks. Decentralized federated learning in vehicular networks allows learning with limited neighbors despite communication disruptions. The work [10] proposes a cluster-based approach with a designated cluster head responsible for data aggregation. In [13], a Byzantine fault-tolerant algorithm is introduced for autonomous vehicles. FADNet [12] addresses model convergence and imbalanced data distribution in autonomous driving, but its evaluation is limited to an indoor environment using a mobile robot. Blockchain technology offers a secure and decentralized platform for facilitating the exchange of updates among vehicles, ensuring trust through its consensus mechanism [22]–[25]. However, these approaches still leverage the use of a global model. It is important to note that in the context of fully decentralized federated learning, the notion of a global model is not applicable. The authors in [11] study model aggregation and propose an algorithm that is better suited to non-IID data sets in vehicular networks. However, communication between vehicles is assumed to be ideal, with no loss occurring during transmission.

In summary, the existing works in this area either assume ideal communication between vehicles (*e.g.*, [10], [11]) or ignore neighborhood changes due to mobility (*e.g.*, [6]). In our work, we propose a model aggregation algorithm that jointly considers both of these aspects, which are inherent features of vehicular networks.

TABLE II: Table of Notations.

Notation	Description
N	Number of vehicles
$\mathcal{D}_i$	Local dataset for vehicle i
Θ_i^t	Model parameters of vehicle i
B	Local minibatch size
η	Learning rate
$\overline{\Theta}_i^t$	Model parameters of vehicle i after local update
$\gamma_{i,j}^t$	Model parameters of vehicle j received by neighbor i
$M_{i,j}^t$	Loss characterization matrix from vehicle i to vehicle j
$p_{i,j}^t$	Link reliability between vehicles i and j
$\boldsymbol{P}^t$	Link reliability vector
δ^t	Model aggregation threshold
$\hat{\gamma}_{i,j}^t$	Repaired model of vehicle j received by neighbor i
C_i	Set of neighbors for vehicle i
M	Number of arms
$H(\cdot)$	Accuracy given the input model parameters
R_i^t	Difference in vehicle i's model accuracy
$\Phi(\cdot)$	Oracle function
Q_m^t	Estimated value for arm a_m at time t
R_1^m, R_0^m	Number of 1 and 0 value rewards for arm a_m

III. Adaptive Decentralized Federated Learning in Vehicular Networks

In this section, we present our Adaptive Model Aggregation (AMA) algorithm. The notation used in the rest of the paper is summarized in Table II.

A. Vehicular Network Model

We consider a vehicular network composed of N vehicles equipped with communication and computing capabilities [26]. Each vehicle collects relevant data from its onboard systems and its environment. The collected data by the vehicles in the network is used to collaboratively train a machine learning model on each vehicle following the decentralized federated learning framework. Without loss of generality, we assume that vehicles use the Federated Averaging (FedAvg) algorithm [2] to aggregate their local model parameters with those received from other vehicles. Each vehicle manages its operations in a step-by-step fashion, where each step is indexed by $t \in \{1, 2, 3, \dots\}$. Let $\mathcal{D}_i^t$ denote the set of collected data by vehicle i in step t. We use Θ_i^t to represent the machine learning model (*i.e.*, parameters of the neural network) on vehicle i at the end of step t. Typically, Θ_i^t is represented as a matrix.

Vehicles use UDP broadcasts over the wireless vehicular network to communicate with each other. Using UDP alleviates the need for a connection setup phase (as required in TCP), resulting in faster and more efficient data transmission. Recall that due to mobility, vehicles may be within each

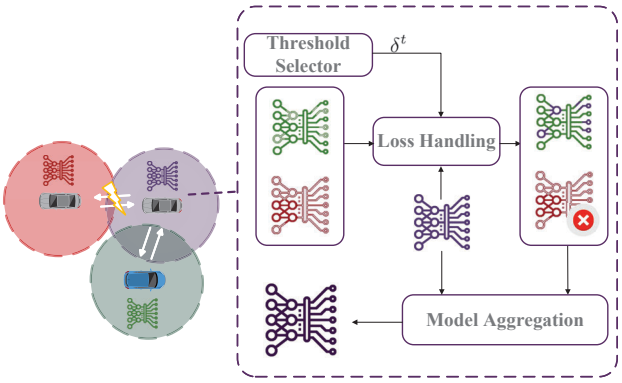


Fig. 2: High-level depiction of the Adaptive Model Aggregation (AMA) algorithm. AMA dynamically selects and aggregates model parameters based on δ_t, adapting to the communication environment and local model accuracy.

others transmission range for only a brief amount of time, making connection-oriented transports such as TCP inefficient. UDP efficiency, however, comes at the cost of loosing some packets during transmission. Each packet carries a portion of the model parameters in its payload. The receiving vehicle then reconstructs the model using the payloads of the received packets. Due to vehicle mobility and the unreliable nature of wireless communication, packets can be corrupted during transmission, resulting in the receiver obtaining only a subset of the model parameters from another vehicle.

B. Local Model Training

Training in DFL is performed locally on each vehicle, and the model parameters are updated using an optimization algorithm such as the Stochastic Gradient Descent (SGD) algorithm. The local training algorithm is presented in lines 1 to 4 of Algorithm 1. In the algorithm, Θ_i^{t-1} represents the model of vehicle i at step $(t-1)$. The vehicle begins step t by collecting relevant local data. Once the size of the collected data reaches a certain threshold or after a certain amount of time, the vehicle initiates the procedure to improve its current model. First, the vehicle splits the local data into batches of size B. It then uses these batches to train the local model using SGD, as follows:

$$g_i^t = \nabla F_i(\Theta_i^{t-1}, b), \tag{1}$$

$$\overline{\Theta}_i^t = \Theta_i^{t-1} - \eta \cdot g_i^t, \tag{2}$$

where, η is the learning rate, and g_i^t represents the average gradient on vehicle i's local data at the current model Θ_i^{t-1}. We used $\overline{\Theta}_i^t$ to denote the resulting model after training on the local data in step t. To control the complexity and overhead of the training phase, the vehicle only considers each batch once and completes the training after one pass over all the batches.

C. Adaptive Model Aggregation

After obtaining the improved local model by conducting the local training phase, the vehicle initiates a round of federated learning-based model improvement by broadcasting $\overline{\Theta}_i^t$. Upon receiving this new model, neighboring vehicles that possess a new model also broadcast their local models. Each vehicle i collects the models it receives from its neighbors and stores them in a set called Γ_i^t (lines 5 to 6).

Due to collisions and wireless transmission errors, some models received at vehicle i may miss a subset of their parameters. Let $\gamma_{i,j}^t$ denote the parameter matrix of vehicle j's model that is received at vehicle i in step t. We model the lossy reception of $\gamma_{i,j}^t$ as follows:

$$\gamma_{i,j}^t = M_{i,j}^t \odot \overline{\Theta}_j^t, \tag{3}$$

where, $\overline{\Theta}_j^t$ represents the local model of vehicle j after local training at time t, $M_{i,j}^t$ is a binary matrix indicating the presence or absence of the corresponding parameters with 0 and 1 respectively, and $\odot$ denotes element-wise matrix multiplication. Since all vehicles have the same type of model, each vehicle can easily construct the binary matrix $M_{i,j}^t$ by setting the elements to 1 or 0 based on whether the corresponding parameters are received or lost.

If the fraction of lost model parameters is high, using the remainder of the received parameters of the corresponding model can incur a computational overhead without a significant improvement to the local model. To eliminate such cases and lower the computation overhead, vehicle i uses an *aggregation threshold* on the size of the received parameters

Algorithm 1: AMA: Adaptive Model Aggregation.

Require: Current model Θ_i^{t-1}, Local data $\mathcal{D}_i^t$
Ensure : Update of Model Θ_i^t

1 $\mathcal{B} \leftarrow$ split($\mathcal{D}_i^t$)
2 **for** $b \in \mathcal{B}$ **do**
3 $\quad g_i^t \leftarrow \nabla F_i(\Theta_i^{t-1}, b)$
4 $\quad \overline{\Theta}_i^t \leftarrow \Theta_i^{t-1} - \eta \cdot g_i^t$
5 broadcast($\overline{\Theta}_i^t$)
6 $\Gamma_i^t \leftarrow$ receive()
7 $G \leftarrow |\Gamma_i^t|$
8 **for** $\gamma_{i,j}^t \in \Gamma_i^t$ **do**
9 $\quad p_{i,j}^t = \frac{|\gamma_{i,j}^t|}{|\overline{\Theta}_i^t|}$
10 $\boldsymbol{P}_i^t \leftarrow \left[p_{i,1}^t, \ldots, p_{i,G}^t\right]$
11 $\delta_i^t \leftarrow$ Adaptive-ATS($\boldsymbol{P}_i^t$) /* see Alg. 2 */
12 $C_i^t \leftarrow \{\}$
13 **for** $\gamma_{i,j}^t \in \Gamma_i^t$ **do**
14 $\quad$ **if** $p_{i,j}^t \geq \delta_i^t$ **then**
15 $\quad\quad M_{i,j}^t \leftarrow$ fill-matrix($\gamma_{j,i}^t$)
16 $\quad\quad \hat{\gamma}_{i,j}^t \leftarrow \gamma_{i,j}^t + (1 - M_{i,j}^t) \odot \overline{\Theta}_i^t$
17 $\quad\quad C_i^t$.append(j)
18 $\Theta_i^t \leftarrow \frac{1}{|C_i^t|+1}\left(\overline{\Theta}_i^t + \sum_{j \in C_i^t} \hat{\gamma}_{j,i}^t\right)$
19 $R_i^t \leftarrow H(\Theta_i^t) - H(\Theta_i^{t-1})$
20 Oracle-Train(δ_i^t, $\boldsymbol{P}_i^t$, R_i^t) /* see Alg. 3 */

to eliminate the received models that would not contribute significantly to the accuracy of its local model.

To compute the aggregation threshold, denoted by δ_i^t, the algorithm calls a subroutine called Adaptive-ATS in line 11. The threshold must be computed dynamically to take into account the environment's communication conditions (*i.e.*, how lossy the environment is). The detailed explanation of the algorithm for computing δ_i^t is provided in Section IV. To provide the information about the loss, vehicle i computes the ratio between the number of received parameters (denoted by $|\gamma_{i,j}^t|$) and the total number of parameters (denoted by $|\overline{\Theta}_j^t|$) for each neighbor j and stores it in the variable $p_{i,j}^t$, which we refer to as the *link reliability* parameter:

$$p_{i,j}^t = \frac{|\gamma_{i,j}^t|}{|\overline{\Theta}_i^t|}\,. \tag{4}$$

Then, the vehicle also creates a vector called the *reliability vector* by bundling variables $p_{i,j}^t$. The reliability vector of vehicle i in step t is represented by $\boldsymbol{P}_i^t$. The reliability vector is passed to the threshold computation algorithm, as indicated in line 11 where the model aggregation threshold δ_i^t is acquired.

After obtaining the aggregation threshold δ_i^t, the vehicle compares the value of $p_{i,j}^t$ and δ_i^t for all received models. If the fraction of present parameters is greater than or equal to the threshold, the vehicle repairs the model to be aggregated with the local model. The repair process fills the missing parameters with local parameters at the start of the current step. To this end, vehicle i constructs the matrix $M_{i,j}^t$ and applies the following formula:

$$\hat{\gamma}_{i,j}^t = \gamma_{i,j}^t + (1 - M_{i,j}^t) \odot \overline{\Theta}_i^t\,. \tag{5}$$

Here, $\hat{\gamma}_{i,j}^t$ represents the repaired model obtained from received model $\gamma_{i,j}^t$ of neighbor j. The vehicle stores the index of selected and repaired models in a set called C_i^t in line 17. As the result of algorithm's work in lines 12 to 17, the content of set C_i^t is equal to the following:

$$C_i^t = \{j | j \neq i, p_{i,j}^t \geq \delta_i^t\}\,. \tag{6}$$

Then, the vehicle aggregates the models that meet the requirement specified by the threshold with the local model to obtain an improved model, as follows:

$$\Theta_i^t = \frac{1}{|C_i^t| + 1}(\overline{\Theta}_i^t + \sum_{j \in C_i^t} \hat{\gamma}_{i,j}^t)\,. \tag{7}$$

Finally, the algorithm sends the selected aggregation threshold, the reliability vector, and the change in the model's accuracy after the aggregation to sub-routine Oracle-Train. This sub-routine, which is explained in Section IV, is responsible for improving the underlying parameters used to compute the threshold in Adaptive-ATS algorithm. Let $H(.)$ be the function that computes the accuracy of a given model. Vehicle i uses the following formula to compute the change in the model's accuracy in step t in line 19:

$$R_i^t = H(\Theta_i^t) - H(\Theta_i^{t-1})\,. \tag{8}$$

Then, the vehicle invokes Oracle-Train in line 20. Fig. 2 presents a high-level overview of the AMA algorithm.

IV. Aggregation Threshold Selection

In this section, we first formulate the aggregation threshold selection problem as a contextual multi-armed bandit (CMAB) problem. We then present the Adaptive Aggregation Threshold Selection (Adaptive-ATS) algorithm to solve the problem efficiently.

A. Problem Formulation

In CMAB, we encounter a sequential decision-making problem where, at each step (referred to as a decision epoch), we must select an arm from a given set of arms based on the available context information. Upon selecting an arm, a reward is obtained, and the objective is to find a strategy for arm selection that maximizes the cumulative reward over time [14]. To this end, we define the set of arms, specify the context, and design a suitable reward function to the aggregation threshold that maximizes model accuracy in each given context.

Arms. We discretize the interval $[0, 1]$ into M sub-intervals, each representing an arm. We denote the m-th sub-interval as a_m, where $m \in 1, \ldots, M$. Once vehicle i selects a_m in step t, the value of aggregation threshold δ_i^t is set to the midpoint of sub-interval a_m, as follows:

$$\delta_i^t = \frac{2m - 1}{2M}\,. \tag{9}$$

Reward. We use a binary reward function for arms. In each step t, vehicle i compares R_i^t (see (8)) with a parameter ρ_t that specifies the minimum acceptable accuracy increase. If $R_i^t \geq \rho_t$, a reward of one unit is assigned to the selected arm in step t. Otherwise, the reward is assumed to be zero.

Context. The value of each arm represents its impact on the accuracy of the local model after aggregation. As discussed before, the increase of accuracy in our system depends on the fraction of parameters that have been received successfully by the vehicle. Therefore, we consider the reliability vector $\boldsymbol{P}_i^t$ as the context. As a result, the value of each arm is assumed to be a function of the reliability vector $\boldsymbol{P}_i^t$.

B. Threshold Selection Algorithm

To select an aggregation threshold, each vehicle estimates the value of arms given the reliability vector $\boldsymbol{P}_i^t$ using historical rewards collected from interactions with the environment. The value of an arm represents the expected reward that can be obtained by playing it, and it is not known a priori. In this regard, we assume the existence of an oracle function $\Phi_m(\boldsymbol{P}_i^t)$ associated with each arm a_m. This oracle function takes the reliability vector as input and provides a relative value for arm a_m compared to other arms, ranging from zero to one. A common approach in CMAB is to learn the oracle $\Phi_m(\cdot)$ using a supervised learning algorithm. In our case, we utilize linear regression-based binary classifiers to implement the oracle functions and learn their parameters using data collected during interactions with the environment.

To facilitate learning from interactions with the environment and avoid getting stuck in local optima, vehicles utilize an adaptive ϵ-greedy algorithm [27], [28]. In this algorithm, a vehicle chooses to use oracles with a probability of $(1-\epsilon)$ and selects a random arm with a probability of ϵ. The value of ϵ is adjusted based on the maximum value among the arms. Specifically, if the value of the selected arm increases, ϵ is decreased, promoting exploitation. Conversely, if the value of the selected arm decreases, ϵ is increased, encouraging exploration to potentially find a better arm. The adaptive ϵ-greedy algorithm employed in our system is presented in Algorithm 2.

The algorithm begins by estimating the value of each arm in lines 1 to 5. In this process, Q_m^t represents the estimate for the reward obtained by using the aggregation threshold computed from arm a_m at step t, with the context $\boldsymbol{P}_i^t$. During the initial steps of the algorithm, the accuracy of the estimates returned by $\Phi(\cdot)$ may not be highly precise due to insufficient training data for the oracles. To handle this, we initially adopt a Bayesian multi-armed bandit policy for each threshold, utilizing a Beta prior [28] without incorporating the context covariates. Once a minimum number of observations from each reward value have been collected for a specific arm, we transition to a contextual bandit policy based on $\Phi(\cdot)$. Specifically, each vehicle utilizes the Beta prior to collect K ones and K zeros for each arm before utilizing its corresponding oracle function. Once K zeros and ones have been accumulated, the vehicle trains the parameters of the oracle and employs it to estimate the value of that arm. To employ the Beta prior, the vehicle keeps track of the current counts of ones and zeros for arm m using two counters, denoted as R_1^m and R_0^m, respectively. Consequently, the general expression for estimating the value of each arm is as follows:

$$Q_m^t = \begin{cases} \mathrm{Beta}(R_1^m + 1, R_0^m + 1), & \min\{R_1^m, R_0^m\} < K \\ \Phi_m(\boldsymbol{P}^t), & \text{otherwise} \end{cases} \tag{10}$$

where, $\mathrm{Beta}(.,.)$ denotes the beta distribution.

After obtaining an estimate for the value of arms, the algorithm applies the ϵ-greedy technique. In line 6, the algorithm generates a random number between zero and one. If this random number is greater than ϵ, the algorithm selects the arm with the highest estimated value in line 18. On the other hand, if the random number is smaller than ϵ, the algorithm selects a random arm in line 16. In this case, the algorithm also utilizes the estimated values to dynamically update the value of ϵ. To update ϵ, the algorithm maintains a global variable named $\widehat{R}$ that stores the maximum estimated value encountered so far. The algorithm compares the current maximum estimated value with $\widehat{R}$. If the current maximum is smaller, the value of ϵ is decreased using a decay factor d_1. However, if the current maximum is larger, the value of ϵ is reset to 0.5. If the two values are equal, ϵ remains unchanged.

It is important to note that the update of ϵ is not performed in every call of the algorithm. To increase the stability of the algorithm, the update happens every L_1 calls. The algorithm keeps track of the variable c_1, which counts how many times the exploration mode has been executed since the last change in the value of ϵ. In lines 8 to 15, the algorithm determines whether to change the value of ϵ once c_1 reaches the limit L_1. After setting the new value of ϵ, the variable c_1 is reset, and $\widehat{R}$ is updated to R.

Algorithm 2: Adaptive-ATS: Adaptive Aggregation Threshold Selection.

Global : $\widehat{R} = 0$, $c_1 = 0$, $d_1 = 0.8$, ϵ, R_1^m, R_0^m, Exploration Limit L_1
Require: Reliability Vector $\boldsymbol{P}^t$
Output : Aggregation Threshold

1 **for** $m \in \{1, \ldots, M\}$ **do**
2 **if** $R_1^m < K$ **or** $R_0^m < K$ **then**
3 $Q_m^t \leftarrow \mathrm{Beta}(R_1^m + 1, R_0^m + 1)$
4 **else**
5 $Q_m^t \leftarrow \Phi_m(\boldsymbol{P}^t)$
6 **if** uniform$(0,1) \leq \epsilon$ **then**
7 $c_1 \leftarrow c_1 + 1$
8 **if** $c_1 = L_1$ **then**
9 $R \leftarrow \max_{m \in \{1,\ldots,M\}} \{Q_m^t\}$
10 **if** $R - \widehat{R} > 0$ **then**
11 $\epsilon \leftarrow \epsilon \times d_1$ /* scale down ϵ */
12 **if** $R - \widehat{R} < 0$ **then**
13 $\epsilon \leftarrow 0.5$ /* reset ϵ */
14 $c_1 \leftarrow 0$
15 $\widehat{R} \leftarrow R$
16 $m \leftarrow$ random-choice$(\{1, \ldots, M\})$
17 **else**
18 $m \leftarrow \mathrm{argmax}_{m \in \{1,\ldots,M\}} \{Q_m^t\}$
19 **return** $\frac{2m-1}{2M}$

C. Training Reward Oracles

When a vehicle performs an aggregation based on a threshold, it sends the change in the model's accuracy before and after aggregation, along with the reliability vector and the selected threshold, to Oracle-Train, outlined in Algorithm 3. Oracle-Train adjusts the oracle responsible for selecting that threshold. In line 2, Oracle-Train determines the selected arm based on the value of the aggregation threshold. Then, it compares the increase in accuracy with an improvement threshold ρ_t. If the increase in accuracy is greater than ρ_t, it considers the choice of the arm that led to the threshold value as beneficial. Consequently, the oracle corresponding to arm a_m is trained to produce a higher value in the given context. To achieve this, the context $\boldsymbol{P}^t$ is labeled as 1 and the context-label pair is saved for arm a_m in $\mathcal{J}_m$. Additionally, the value of the counter R_1^m used in the Beta prior is incremented. Otherwise,

Algorithm 3: Oracle-Train: Oracle Model Training.

Global : $d_2 = 0.9$, R_1^m, R_0^m, L_2, $c_2 = 0$, ρ_t, $\mathcal{J}_m$
Require: Selected Threshold δ_i^t, Reliability Vector $\boldsymbol{P}^t$, Accuracy Change R_i^t

1 $c_2 \leftarrow c_2 + 1$
2 $m \leftarrow (2M \times \delta_i^t + 1)/2$
3 **if** $R_i^t \geq \rho_t$ **then**
4 $R_1^m \leftarrow R_1^m + 1$
5 $\mathcal{J}_m \leftarrow \mathcal{J}_m + \{(\boldsymbol{P}^t, 1)\}$
6 **else**
7 $R_0^m \leftarrow R_0^m + 1$
8 $\mathcal{J}_m \leftarrow \mathcal{J}_m + \{(\boldsymbol{P}^t, 0)\}$
9 **if** $c_2 = L_2$ **then**
10 **for** $m \in \{1, \ldots, M\}$ **do**
11 train($\Phi_m(.)$, $\mathcal{J}_m$)
12 $c_2 \leftarrow 0$
13 $\rho_t \leftarrow \rho_t \times d_2$

$\boldsymbol{P}^t$ is labeled as 0, which decreases the probability of selecting arm a_m when the context is similar to $\boldsymbol{P}^t$. Similarly, the value of the counter R_0^m is incremented (lines 3-8).

The oracles are not trained immediately after labeling $\boldsymbol{P}^t$. Instead, the context and label are saved for future training. The training of the oracles occurs after every L_2 steps. To keep track of the number of steps, the algorithm maintains a counter c_2. Once c_2 reaches the limit L_2, the oracles are trained using the accumulated context-label pairs (line 9). After the training is completed, the counter c_2 is reset to zero to start counting the steps for the next training cycle. As the model is expected to converge, the increase in accuracy is anticipated to decrease over time. Therefore, the value of the improvement threshold ρ_t is decreased in line 13 using a decay factor named d_2.

V. Evaluations

In this section, we evaluate AMA and Adaptive-ATS under different mobility scenarios and compare their performances with several existing baselines.

A. Methodology

Setup. The implementations are done in the Python programming language, and the machine learning models are built using the PyTorch library. The experiments are conducted on a desktop computer with a 3.80 GHz Intel i7 processor, 16 GB of memory, and an NVIDIA GeForce RTX 3080 Ti graphics card. Throughout the evaluations, we simulate a network consisting of 20 vehicles. The data in each vehicle is generated from a non-IID CIFAR-10 dataset [29], where the number of samples of each class differs across vehicles. There is no overlap in the data between vehicles, and the data distribution is randomly generated at the beginning of each run. The data is divided into 1000 shards, each consisting of 50 data samples, and each vehicle is assigned at least one shard.

TABLE III: Variants of the CSE Model.

Name	CSE 1	CSE 2	CSE 3	CSE 4	CSE 5
p	0.5	0.5	0.5	0.9	0.1
k	0.5	0.3	0.7	0.5	0.5

ML Model. The machine learning model used in each vehicle is a ResNet-20 model [30] trained on the CIFAR-10 dataset. The weight decay is set to 0.0001, and the batch size is 32. The initial learning rate is 0.1, and the momentum is set to 0.9. The aggregation threshold values are quantified from 0.0 to 0.9 with increments of 0.1. The initial value of ϵ is set to 0.5 and M is set to 10. A linear regression model serves as the oracle for each arm. K is set to 2 and L_1 and L_2 are set to 10 and 15, respectively. The initial value of ρ_t is set to approximately 4% and gradually decreases to 0.04% as the model converges to its final accuracy value.

Link Reliability. The link reliability between any two vehicles i and j is given by the following expression:

$$p_{i,j}^t = p_{j,i}^t = k^{(\frac{d_{i,j}^t}{r})^2},$$

where k is an exponential decay factor between 0 and 1, $d_{i,j}^t$ is the distance between vehicles at time step t, and r is the range of communication [6]. A higher value of k indicates that the wireless signal does not weaken quickly with distance, resulting in more reliable inter-vehicle communications.

Vehicle Mobility. We have implemented two mobility models for the vehicles in the network:

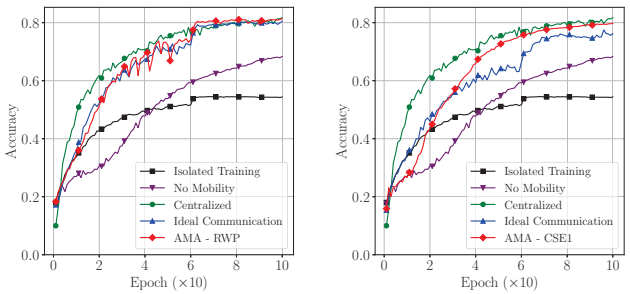
- Random Waypoint (RWP) [31]: Each vehicle moves towards a random destination in a 1500m by 1500m grid at a constant speed. After the training and communication phases, the vehicles' positions are updated to their positions 50 steps ahead. The travel speed is randomly chosen from the range of $[50, 150]$ m/epoch.
- Community Structured Environment (CSE) [32]: There are five communities of vehicles, with each community spanning a 500m by 500m square. Vehicles within a community move together. Each vehicle spends 3 to 6 learning steps within a community and has a probability $p \in [0.1, 0.9]$ of changing its community. We consider several variants of the Community Structured Environment (CSE), which are listed in Table III.

B. Model Training Performance

In this set of experiment, our objective is to access the training performance of AMA in terms of convergence speed.

Implemented Training Approaches. We compare the performance of AMA with four baseline approaches:

- Isolated Training: Each vehicle trains its model only on its local data without communicating with other vehicles, *i.e.*, no federated learning.
- No Mobility: Vehicles collaboratively train their models, but there is no mobility so that each vehicle always communicate with the same set of neighbors.



(a) Dynamic network (RWP Mobility). (b) Dynamic network (CSE Mobility).

Fig. 3: Average model accuracy with RWP and CSE mobility.

- IDEAL COMMUNICATION: All packet transmissions over links with $p_{i,j}^t \geq 0.7$ are error-free, *i.e.*, the models are received fully. Error-free models are always aggregated.
- CENTRALIZED: All vehicles send their local models to a central server for aggregation. Vehicles are stationary (mobility is not relevant in this model) and communication is ideal. This approach is expected to achieve the highest training accuracy.

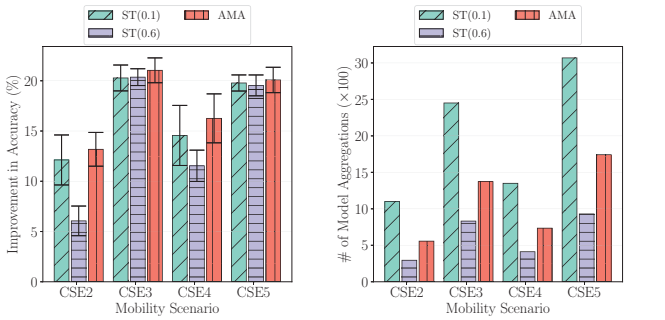
Results. Fig. 3 shows the accuracy trend per epoch for each approach under different mobility models. In ISOLATED TRAINING, where communication between vehicles is lacking, the accuracy remains consistently below 60%. However, in both static and dynamic networks where communication is reliable, the training process becomes efficient, leading to increased accuracy similar to conventional federated learning. Comparatively, convergence is achieved faster in AMA and IDEAL COMMUNICATION than in other baseline approaches except CENTRALIZED. Notably, AMA surpasses IDEAL COMMUNICATION by 1% with RWP mobility and 3% with CSE mobility, while achieving a performance close to that of CENTRALIZED, regardless of the mobility model.

C. Threshold Selection Benchmarks

In this set of experiments, our objective is to study the performance of Adaptive-ATS and compare it with other potential threshold selection algorithms. To have a fair comparison, we use the same AMA algorithm for federated learning, but replace the default Adaptive-ATS with other algorithms.

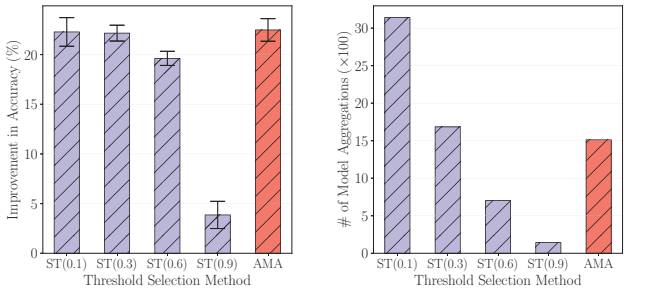
Implemented Selection Algorithms. We compare Adaptive-ATS with the following algorithms:

- STATIC: The aggregation threshold is set to a fix value. The static threshold algorithm is denoted by $ST(\alpha)$, where α is the fix threshold value.
- EPSILON-GREEDY: The aggregation threshold is set to a random value with probability ϵ; otherwise, the value with the highest average reward is chosen.
- ADAPTIVE-GREEDY: Similar to Epsilon-Greedy, however, the value of ϵ decreases over time [27].
- CONTEXTUAL-GREEDY: Same as Epsilon-greedy except that the expected reward for each arm is predicted using a linear regression model based on the context.



(a) Effect of threshold on accuracy. (b) Effect of threshold on number of model aggregations.

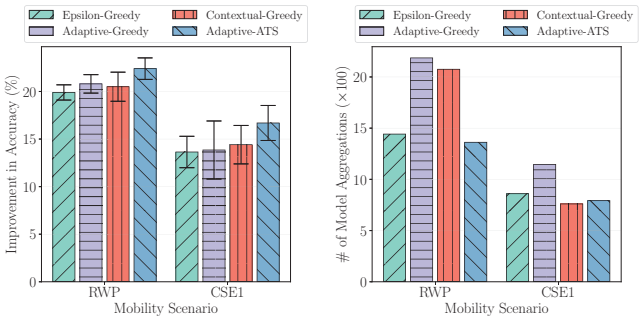
Fig. 4: Effect of the aggregation threshold on accuracy and model aggregations in different CSE mobility scenarios.



(a) Effect of threshold on accuracy. (b) Effect of the aggregation threshold on number of model aggregations.

Fig. 5: Effect of aggregation threshold on accuracy and model aggregations in RWP mobility.

Static Threshold Selection. We compared the final model accuracy and the number of aggregations in different CSE mobility scenarios for $ST(0.1)$, $ST(0.6)$, and AMA. Fig. 4(a) shows the improvement in accuracy achieved by each algorithm compared to the fixed baseline threshold 0 (*i.e.*, always aggregating all received models). As can be seen, the improvement is consistently positive, thus highlighting the importance of using a threshold in the system. The 0.6 threshold yields better results in more dense environments with higher link reliabilities, and the 0.1 threshold generates favorable results regardless of the environment. Notably, AMA achieves higher accuracy compared to the static threshold algorithms in all scenarios. While these results indicate that a 0.1 threshold is a good choice in the considered scenarios, as shown in Fig. 4(b), the 0.1 threshold requires an average of 83% more model aggregations than AMA. This makes AMA the best choice as it provides higher accuracy with reduced computational overhead compared to the 0.1 threshold. We have also compared the performance AMA under the RWP mobility model with static aggregation threshold values 0.1, 0.3, 0.6, and 0.9. The results are presented in Fig. 5. As can be seen from the figure, while lower threshold values, such as 0.1 and 0.3, produce higher improvement in training accuracy, AMA achieves similar accuracy with 51% and 10% fewer aggregations than $ST(0.1)$ and $ST(0.3)$, respectively.



(a) Effect of dynamic threshold selection method on accuracy.

(b) Effect of dynamic threshold selection method on number of model aggregations.

Fig. 6: Effect of dynamic threshold selection method on accuracy and model aggregations in different mobility scenarios.

Dynamic Threshold Selection. We compare the three greedy algorithms (that dynamically select the aggregation threshold) to Adaptive-ATS under both mobility models. The results, depicted in Fig. 6, demonstrate that Adaptive-ATS performs the best in terms of accuracy among all the algorithms while requiring fewer aggregations. This indicates that incorporating contextual information into the threshold selection algorithm can significantly enhance its performance, and the use of an adaptive epsilon value enables a balanced trade-off between exploration and exploitation.

VI. Conclusion

In this paper, we proposed AMA for federated learning in vehicular networks, taking into account the unreliability of wireless communications caused by vehicle mobility. Our approach utilizes a dynamic threshold to adapt the participation of vehicles in the federated learning process, and we evaluated it using two different mobility models. The results demonstrate that our method surpasses other approaches in terms of final accuracy and the number of aggregations required. For future work, we recommend exploring more advanced reinforcement learning methods that consider additional context information such as vehicle direction and speed. Additionally, investigating alternative machine learning algorithms to enhance the model's accuracy or training multiple models simultaneously, could be worthwhile avenues of research.

References

[1] A. Nanda, D. Puthal, J. J. Rodrigues, and S. A. Kozlov, "Internet of autonomous vehicles communications security: overview, issues, and directions," *IEEE Trans. Wireless Commun.*, vol. 26, no. 4, 2019.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, 2020.

[4] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *NeurIPS*, 2018.

[5] J. Posner, L. Tseng, M. Aloqaily, and Y. Jararweh, "Federated learning in vehicular networks: Opportunities and solutions," *IEEE Network*, vol. 35, no. 2, 2021.

[6] H. Ye, L. Liang, and G. Y. Li, "Decentralized federated learning with unreliable communications," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 3, 2022.

[7] Z. Du, C. Wu, T. Yoshinaga, K.-L. A. Yau *et al.*, "Federated learning for vehicular internet of things: Recent advances and open issues," *IEEE Open Journal of the Computer Society*, vol. 1, 2020.

[8] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *arXiv preprint arXiv:1908.07782*, 2019.

[9] T. Wink and Z. Nochta, "An approach for peer-to-peer federated learning," in *Proc. IEEE/IFIP DSN Workshops*, 2021.

[10] A. Taïk, Z. Mlika, and S. Cherkaoui, "Clustered vehicular federated learning: Process and optimization," *IEEE Trans. Intell. Transp. Syst.*, 2022.

[11] H. Ochiai, Y. Sun, Q. Jin, N. Wongwiwatchai *et al.*, "Wireless ad hoc federated learning: A fully distributed cooperative machine learning," *arXiv preprint arXiv:2205.11779*, 2022.

[12] A. Nguyen, T. Do, M. Tran, B. X. Nguyen *et al.*, "Deep federated learning for autonomous driving," in *IEEE IV*, 2022.

[13] J.-H. Chen, M.-R. Chen, G.-Q. Zeng, and J.-S. Weng, "BDFL: a byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, 2021.

[14] T. Lu, D. Pál, and M. Pál, "Contextual multi-armed bandits," in *AIStats*, 2010.

[15] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi *et al.*, "Federated optimization in heterogeneous networks," *Proc. MLSys*, vol. 2, 2020.

[16] E. Jeong, S. Oh, H. Kim, J. Park *et al.*, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[17] L. Huang, Y. Yin, Z. Fu, S. Zhang *et al.*, "Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data," *arXiv preprint arXiv:1811.12629*, 2018.

[18] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[19] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab *et al.*, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[20] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," *arXiv preprint arXiv:1901.11173*, 2019.

[21] Q. Chen, Z. Wang, Y. Zhou, J. Chen *et al.*, "CFL: Cluster federated learning in large-scale peer-to-peer networks," in *Proc. ISC*, 2022.

[22] I. Aliyu, M. C. Feliciano, S. Van Engelenburg, D. O. Kim *et al.*, "A blockchain-based federated forest for sdn-enabled in-vehicle network intrusion detection system," *IEEE Access*, vol. 9, 2021.

[23] H. Liu, S. Zhang, P. Zhang, X. Zhou *et al.*, "Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, 2021.

[24] S. Otoum, I. Al Ridhawi, and H. T. Mouftah, "Blockchain-supported federated learning for trustworthy vehicular networks," in *Proc. IEEE GLOBECOM*, 2020.

[25] S. R. Pokhrel and J. Choi, "A decentralized federated learning approach for connected autonomous vehicles," in *Proc. IEEE WCNCW Workshops*, 2020.

[26] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk *et al.*, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 4, 2011.

[27] A. dos Santos Mignon and R. L. d. A. da Rocha, "An adaptive implementation of ε-greedy in reinforcement learning," *Procedia Computer Science*, vol. 109, 2017.

[28] D. Cortes, "Adapting multi-armed bandits policies to contextual bandits scenarios," *arXiv preprint arXiv:1811.04383*, 2018.

[29] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016.

[31] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, "Stochastic properties of the random waypoint mobility model," *Wireless Networks*, vol. 10, no. 5, 2004.

[32] H. Ochiai and H. Esaki, "Mobility entropy and message routing in community-structured delay tolerant networks," in *Proc. AINTEC*, 2008.