V2V-Assisted Timely Hierarchical Federated Learning

Jintao Yan, *Student Member, IEEE,* Zhaojun Nan, *Graduate Student Member, IEEE* and Sheng Zhou, *Member, IEEE*

***Abstract*—To meet the timeliness and data privacy requirements of vehicular applications, federated learning (FL) has been applied in edge training for connected vehicles. However, FL in vehicular networks needs to meet stringent latency requirements due to the high mobility of vehicles. This paper proposes a vehicle-to-vehicle (V2V)-assisted hierarchical federated learning (VAHFL) architecture to reduce the communication latency for model uploading. We formulate the communication latency minimization problem as a sequential decision-making problem. This problem is too complex and we divide it into three sub-problems: V2V link scheduling, bandwidth allocation, and sequential node classification. The first two problems can be solved using optimization methods, while the third problem is still hard to solve due to unavailable noncausal information. We develop a graph neural network (GNN)-reinforcement learning (RL) based framework to solve the third problem in an online manner. Experimental results show that our proposed scheme can reduce the latency by at least 41.99% compared with the benchmarks. Additionally, the convergence speed and the testing accuracy of our proposed scheme exceed that of the benchmarks under both i.i.d. and non-i.i.d. settings.**

I. Introduction

The rapid developments of vehicular networks give rise to many vehicular applications, such as autonomous driving, cooperative perception, and route planning. These vehicular applications generate massive data and require *timely* training of machine learning (ML) models to adapt to changing road conditions [1]. In the traditional machine learning framework, data is uploaded to the central server for model training, which brings risks of privacy disclosure and large delays. In this context, federated learning (FL) is a promising framework to enable privacy conservation.

FL is a distributed ML framework where clients train models locally and send updated parameters to servers for aggregation. Despite growing interest in FL for vehicular networks, challenges arise from stringent latency requirements and fewer participating clients compared to traditional wireless networks. Thus, hierarchical federated learning (HFL) is considered, leveraging the large coverage of cloud servers and low latency of edge servers.

For implementing HFL in vehicular networks, the communication latency is a major bottleneck. Current methods to reduce the communication burden in HFL include gradient sparsification, resource allocation, and model quantization [2]-[4]. In [2], periodic averaging and gradient sparsification techniques are adopted to improve the communication efficiency of HFL. In [3], a joint resource allocation and edge association scheme is proposed in HFL to minimize both the aggregation latency and the energy consumption. In [4], model quantization is adopted to reduce the communication latency in HFL.

Meanwhile, some efforts consider device-to-device communications for FL. In [5], the concept of collaborative federated learning is introduced, where some devices upload their model parameters with the help of their neighboring devices. In [6], a topology-optimized federated edge learning scheme is presented to enhance communication efficiency. A semi-decentralized FL algorithm is designed in [7] that utilizes collaborative relaying to improve the convergence performance. However, the above works presume a static network topology during model uploading. The vehicular network topology is often dynamic, which necessitates making online decisions at each hop during the transmission. This brings challenges in designing the transmission scheme for model uploading.

This paper presents a vehicle-to-vehicle (V2V)-assisted HFL (VAHFL) architecture and an uploading scheme to minimize communication latency. The latency minimization problem is formulated as a sequential decision-making problem and decomposed into three sub-problems: V2V link scheduling, bandwidth allocation, and sequential node classification. While optimization methods can be used to solve the first two sub-problems, the node classification problem remains challenging due to unavailable noncausal information. To address this, a graph neural network (GNN) and reinforcement learning (RL)-based algorithm is proposed to make online decisions. Experiments on the Simulation of Urban MObility (SUMO) platform demonstrate that our approach significantly reduces latency and improves FL training convergence.

II. System Model

We consider an ML model trained in a hierarchical FL manner, where there is one cloud server, N roadside units (RSU) and M vehicles, as illustrated in Fig. 1. The set of RSUs and vehicles are defined as $\mathcal{U} = \{1, ..., N\}$ and $\mathcal{V} = \{1, ..., M\}$ respectively. The vehicles are partitioned into N clusters based on the coverage of RSUs. Due to mobility, the elements in each cluster may vary during the FL training process. The vehicles upload their parameters to the RSUs

Jintao Yan, Zhaojun Nan and Sheng Zhou are with the Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. (email: {yanjt22, nzj660624}@mails.tsinghua.edu.cn, sheng.zhou@tsinghua.edu.cn).

The work is sponsored in part by the Natural Science Foundation of China (No. 62341108, No. 62022049, No. 62111530197) and Hitachi Ltd.

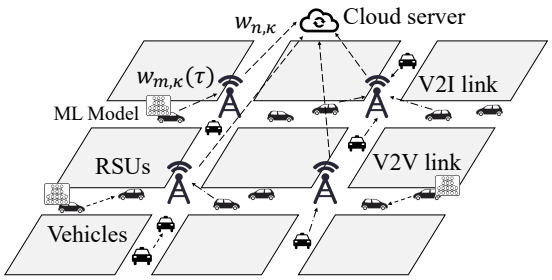

Fig. 1. The VAHFL architecture.

via orthogonal channels (V2I), and communicate with other vehicles via sidelinks (V2V) [8].

A. VAHFL Architecture

Each vehicle m holds a local dataset $\mathcal{D}_m$. There is an input feature $\mathbf{x}_i$ and a labeled output y_i in each sample $\mathbf{d}_i \in \mathcal{D}_m$ of the dataset. $f_m(\mathbf{w}, \mathbf{d}_i)$ denotes the loss function on a data sample $\mathbf{d}_i$, and $f_m(\mathbf{w}) = \frac{1}{|\mathcal{D}_m|}\sum_{\mathbf{d}_i \in \mathcal{D}_m} f_m(\mathbf{w}, \mathbf{d}_i)$ is the local loss function of vehicle m based on the model parameter $\mathbf{w}$. Considering all vehicles over the entire system, the global loss function of the HFL task is defined as $F(\mathbf{w}) = \frac{1}{D}\sum_{m=1}^{M} |\mathcal{D}_m| f_m(\mathbf{w})$, where $D = \sum_{m=1}^{M} |\mathcal{D}_m|$.

The goal of the HFL training is to minimize the global loss function by optimizing the global parameter $\mathbf{w}^*$, where $\mathbf{w}^* = \arg\min F(\mathbf{w})$. The HFL training includes the following four stages: local update, V2V-assisted uploading, edge aggregation and cloud aggregation.

1) Local Update: At the start of κ^{th} edge round, RSU n broadcasts their model parameters $\mathbf{w}_{\kappa-1}$ to the scheduled vehicles, denoted by $\mathcal{V}_{n,\kappa}$. Every $m \in \mathcal{V}_{n,\kappa}$ overwrites its local parameter as $\mathbf{w}_{m,\kappa}(0) = \mathbf{w}_{\kappa-1}$, and uses stochastic gradient descent (SGD) algorithm to update the model parameter:

$$\mathbf{w}_{m,\kappa}(\tau) = \mathbf{w}_{m,\kappa}(\tau-1) - \eta_\kappa \nabla f_m(\mathbf{w}_{m,\kappa}(\tau-1)), \quad (1)$$

where $\tau = 1, 2, ..., \tau_1$ denotes the updating steps, τ_1 is the number of repeated updating steps, and η_κ is the learning rate. The gradient ∇f_m is calculated based on a randomly sampled subset $\mathcal{D}_{m,b} \in \mathcal{D}_m$, where b denotes the batch size.

2) V2V-Assisted Uploading: After all scheduled vehicles complete the local updates, they upload their model parameters to the RSU n for edge aggregation. Unlike conventional uploading schemes, where parameters are directly uploaded to the RSUs, vehicles in our scheme upload their model parameters via either V2V or V2I communications. The aggregation process is divided into S steps, where S is an optimization variable, denoting the number of steps required to upload the parameters of all vehicles remaining in the cluster. In each step (denoted by the superscript s), the vehicles leaving the cluster send their parameters to the relay vehicles via V2V links, while the vehicles close to the RSUs directly send their parameters to the RSUs via V2I links.

3) Edge Aggregation: Upon receiving model parameters from $\mathcal{S}_{n,\kappa}$, the RSU n aggregates the model parameters and begin a new edge round: $\mathbf{w}_{n,\kappa} = \frac{\sum_{m\in\mathcal{S}_{n,\kappa}} |\mathcal{D}_m| \mathbf{w}_{m,\kappa}(\tau)}{\sum_{m\in\mathcal{S}_{n,\kappa}} |\mathcal{D}_m|}$.

4) Cloud Aggregation: After κ_1 rounds of edge aggregation, assuming that κ_1 is a given parameter, RSUs upload their aggregated parameters to the cloud server. The cloud server aggregates all received parameters to obtain a global model: $\mathbf{w}_\kappa = \frac{\sum_{n=1}^{N} |\mathcal{D}_n| \mathbf{w}_{n,\kappa}}{\sum_{n=1}^{N} |\mathcal{D}_n|}$, where $\mathcal{D}_n$ is the aggregated dataset under RSU n. We assume that the cloud aggregation latency is constant, and is not an optimization goal in this work. After cloud aggregation, a cloud round is completed and the model parameters are broadcast to RSUs to start a new cloud round.

B. Communication Latency

In this part, we model the communication latency for the V2V-assisted uploading process in the VAHFL framework. We consider the FL training under an arbitrary round κ, so we omit the subscript κ in the following without ambiguity.

The communication for model uploading includes V2V and V2I communication. Therefore, the vehicle set in s^{th} step $\mathcal{V}^{(s)}$ can be divided into three subsets, i.e., $\mathcal{V}^{(s)} = \mathcal{V}_1^{(s)} \cup \mathcal{V}_2^{(s)} \cup \mathcal{V}_3^{(s)}$, where $\mathcal{V}_1^{(s)}$ and $\mathcal{V}_2^{(s)}$ are defined as the sender and receiver sets for V2V communications, respectively, and $\mathcal{V}_3^{(s)}$ is defined as the sender set for V2I communications.

We consider Rayleigh fading channel as the channel model, and the channel gain between node i and node j is given by $h_{ij} = g_0(d_0/d_{ij})^\beta \xi_{ij}$, where g_0 is the pathloss constant, d_0 is the reference distance, d_{ij} is the distance between sender i and receiver j, β is the pathloss exponent, and $\xi_{ij} \sim \mathcal{CN}(0,1)$ is the small-scale fading coefficient.

For the V2V communications between the sender $i \in \mathcal{V}_1^{(s)}$ and the receiver $j \in \mathcal{V}_2^{(s)}$, we assume that there is no interference between V2V links, and the available bandwidth for transmission is B_{ij}. Therefore, the transmission rate (bit/s) is $r_{ij}^{(s)} = B_{ij}\log_2\left(1 + \frac{P_i |h_{ij}^{(s)}|^2}{B_{ij} N_0}\right)$, where P_i is the transmit power of vehicle i, and N_0 denotes the noise power density. The communication latency between i and j is $t_{ij}^{(s)} = \frac{Q_{\text{model}}}{r_{ij}^{(s)}}$, where Q_{model} denotes the size of the model parameter in bits. The V2V communication latency in s^{th} step is determined by the slowest transmission among all transmission pairs. To model this, we define $\boldsymbol{C}^{(s)} = (c_{ij}^{(s)})_{|\mathcal{V}_1^{(s)}| \times |\mathcal{V}_2^{(s)}|}$ as the transmission matrix, where $c_{ij}^{(s)} = 1$ if i sends to j, and $c_{ij}^{(s)} = 0$ otherwise. Therefore, the V2V communication latency in s^{th} step is given by $t_{\text{V2V}}^{(s)} = \max_{i\in\mathcal{V}_1, j\in\mathcal{V}_2} \left\{ t_{ij}^{(s)} c_{ij}^{(s)} \right\}$.

For V2I communication, vehicles upload their parameters to the RSU n via orthogonal frequency bands. We define $\boldsymbol{\delta}^{(s)} = [\delta_1^{(s)}, \delta_2^{(s)}, ..., \delta_{K^{(s)}}^{(s)}]$ as the bandwidth allocation ratio. The total V2I bandwidth B_n is allocated to $K^{(s)}$ vehicles by $\delta_k^{(s)} B_n$, where $k \in \mathcal{V}_3$ is the index of the vehicle, and $\sum_{k\in\mathcal{V}_3} \delta_k^{(s)} \leq 1$. Therefore, the achievable V2I transmission rate (bits/s) is $r_k^{(s)} = \delta_k^{(s)} B_n \log_2\left(1 + \frac{P_k |h_k^{(s)}|^2}{\delta_k^{(s)} B_n N_0}\right)$, and the communication latency of vehicle k is $t_k^{(s)} = \frac{Q_{\text{model}}}{r_k^{(s)}}$. The V2I communication in the s^{th} step is given by $t_{\text{V2I}}^{(s)} = \max_{k\in\mathcal{V}_3} \left\{ t_k^{(s)} \right\}$.

The total communication latency of the model uploading process is given by

$$T = \sum_{s=0}^{S-1} \max\left\{t_{\text{V2V}}^{(s)}, t_{\text{V2I}}^{(s)}\right\}. \tag{2}$$

III. Problem Formulation

Our objective is to minimize the communication latency for model uploading, given in (2). Here we describe the optimization variables.

In each step, first we need to decide the senders and the receivers, that is, to divide $\mathcal{V}^{(s)}$ into three subsets:$\mathcal{V}^{(s)} = \mathcal{V}_1^{(s)} \cup \mathcal{V}_2^{(s)} \cup \mathcal{V}_3^{(s)}$. To simplify the expression, we define $\mathcal{F}^{(s)}$ as a mapping from the vehicle set to three subsets. When three subsets are given, the V2V link scheduling strategy $\boldsymbol{C}^{(s)}$ and the V2I bandwidth allocation strategy $\boldsymbol{\delta}^{(s)}$ need to be determined for model uploading in this step. When all vehicles send their parameters to other nodes, this step ends and the next step begins. Vehicles that have sent its parameters to other nodes are removed from the cluster, and the vehicle set in the next step can be represented by

$$\mathcal{V}^{(s+1)} = \mathcal{V}_2^{(s)}, \tag{3}$$

The V2V-assisted uploading process ends when all vehicles scheduled either transmit their parameters to the RSU n, i.e.,

$$\mathcal{V}^{(S)} = \emptyset. \tag{4}$$

Now we formulate the communication latency minimization problem by optimizing the node classification strategy $\Pi \triangleq \{\mathcal{F}^{(0)}, \mathcal{F}^{(1)}, ..., \mathcal{F}^{(S-1)}\}$, the V2V link scheduling strategy $\mathbf{C} \triangleq \{\boldsymbol{C}^{(0)}, \boldsymbol{C}^{(1)}, ..., \boldsymbol{C}^{(S-1)}\}$ and the V2I bandwidth allocation strategy $\boldsymbol{\delta} \triangleq \{\boldsymbol{\delta}^{(0)}, \boldsymbol{\delta}^{(1)}, ..., \boldsymbol{\delta}^{(S-1)}\}$ in each step:

$$\mathbf{P0:} \min_{\Pi, \mathbf{C}, \boldsymbol{\delta}, S} \quad T \tag{5a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{V}_1} c_{ij}^{(s)} = 1, \forall j \in \mathcal{V}_2 \tag{5b}$$

$$\sum_{j \in \mathcal{V}_2} c_{ij}^{(s)} \leq 1, \forall i \in \mathcal{V}_1 \tag{5c}$$

$$\sum_{k \in \mathcal{V}_3} \delta_k^{(s)} \leq 1, \tag{5d}$$

$$c_{ij}^{(s)} \in \{0,1\}, \forall i \in \mathcal{V}_1, \forall j \in \mathcal{V}_2 \tag{5e}$$

$$0 \leq \delta_k^{(s)} \leq 1, \forall k \in \mathcal{V}_3 \tag{5f}$$

$$(3) \text{ and } (4). \tag{5g}$$

This is a *sequential decision-making problem* with both integer and non-integer optimization variables. We decompose this problem into three sub-problems in the following way.

IV. GNN-RL Based Framework

We decouple **P0** into the V2V link scheduling problem **P1**, the V2I bandwidth allocation problem **P2** and the sequential node classification problem **P3**. Given the node classification strategy, **P1** and **P2** can be solved and the optimal solution can be obtained. Based on the optimal solution of **P1** and **P2**, we approximately solve the node classification problem with a GNN-RL based approach.

A. V2V link scheduling

P0 is decoupled as follows. First, given the node classification strategy in s^{th} step, the V2V link scheduling problem in s^{th} step can be written as

$$\mathbf{P1:} \min_{\boldsymbol{C}^{(s)}} \max_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} \left\{t_{ij}^{(s)} c_{ij}^{(s)}\right\} \tag{6a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{V}_1} c_{ij}^{(s)} = 1, \forall j \in \mathcal{V}_2 \tag{6b}$$

$$\sum_{j \in \mathcal{V}_2} c_{ij}^{(s)} \leq 1, \forall i \in \mathcal{V}_1 \tag{6c}$$

$$c_{ij}^{(s)} \in \{0,1\}, \forall i \in \mathcal{V}_1, \forall j \in \mathcal{V}_2. \tag{6d}$$

P1 is a *bottleneck bipartite matching problem* and the optimal solution can be reached according to the algorithm in [9]. The optimal value is denoted by $t_{\text{V2V}}^{(s)*}$.

B. V2I bandwidth allocation

Likewise, given the node classification strategy in s^{th} step, the V2I bandwidth allocation problem in s^{th} step is written as

$$\mathbf{P2:} \min_{\boldsymbol{\delta}^{(s)}} \max_{k \in \mathcal{V}_3} \left\{ \frac{Q_{\text{model}}}{\delta_k^{(s)} B_n \log 2\left(1 + \frac{P_k |h_k^{(s)}|^2}{\delta_k^{(s)} B_n N_0}\right)} \right\} \tag{7a}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{V}_3} \delta_k^{(s)} \leq 1, \tag{7b}$$

$$0 \leq \delta_k^{(s)} \leq 1, \forall k \in \mathcal{V}_3. \tag{7c}$$

This problem can be solved using Algorithm 1 in [10], and the optimal value is denoted by $t_{\text{V2I}}^{(s)*}$.

C. GNN-RL based sequential node classification

Now, since the optimal solution for **P1** and **P2** can be achieved, the communication latency in the s^{th} step is determined when the three subsets are given. Therefore, the key problem is how to divide the vehicle set into three subsets in each step. This problem can be formulated as

$$\mathbf{P3:} \min_{\Pi, S} \sum_{s=0}^{S-1} \max\left\{t_{\text{V2V}}^{(s)*}, t_{\text{V2I}}^{(s)*}\right\} \tag{8a}$$

$$\text{s.t. } (3) \text{ and } (4). \tag{8b}$$

P3 is still difficult to solve. Although latency in the s^{th} step can be obtained by solving **P1** and **P2**, there is no closed-form expression for the latency. Also, traditional approaches to solving sequential decision-making problems such as dynamic programming require complete noncausal information on vehicle positions, which is impossible to acquire in advance. One approach to solving such problems is deep reinforcement learning [11], but typical neural network architecture such as fully connected multi-layer perceptions or convolutional neural networks is not suitable for the dynamic topology of vehicular networks and the varying number of vehicles. These years, GNN has been applied in wireless networks and shown its potential in optimization problems in networks. It can

outperform other neural network architectures in terms of convergence speed, scalability and generalization, especially for dynamic network topology [12]. We leverage the advantages of GNN and RL to propose GNN-RL based method to solve this problem.

The RL framework contains four elements: state, policy network, action and reward. The GNN policy network takes the state as the input and output action. Based on the current state and action, the reward and next state are obtained. These four elements in our framework are defined as follows:

State: The state in our model is defined as the vehicular network topology $\mathcal{G}^{(s)} = (\mathcal{V}^{(s)}, \mathcal{A}^{(s)}, \boldsymbol{N}^{(s)}, \boldsymbol{E}^{(s)})$, where $\mathcal{V}^{(s)}$ is the vehicle set, $\mathcal{A}^{(s)}$ is the adjacency matrix. $\boldsymbol{N}^{(s)} \in \mathbb{R}^{|\mathcal{V}^{(s)}|\times 2}$ is the node feature, and each row of $\boldsymbol{N}^{(s)}$ is denoted by $\boldsymbol{n}_i^{(s)} = \boldsymbol{N}_{(i,:)}^{(s)} = [d_i^{(s)}, v_i^{(s)}]$, where $d_i^{(s)}$ is the distance between vehicle i and the RSU, and v_i is the speed of vehicle i. $\boldsymbol{E}^{(s)} \in \mathbb{R}^{|\mathcal{V}^{(s)}|\times|\mathcal{V}^{(s)}|}$ is the edge feature, and each element of $\boldsymbol{E}^{(s)}$ is denoted by $d_{ij}^{(s)}$, representing the distance between vehicle i and vehicle j.

GNN Policy Network: The policy network is a message passing GNN. The input of the GNN is the node feature $\boldsymbol{N}^{(s)}$ and the edge feature $\boldsymbol{E}^{(s)}$. This GNN consists of an embedding layer, two graph convolutional layers, a fully connected layer, and a softmax layer. The graph convolution operation of i^{th} node in a^{th} layer is specified as

$$\boldsymbol{n}_i^{(a)} = \sum_{j\neq i} \sigma(\boldsymbol{W}_1^{(a-1)}\boldsymbol{n}_i^{(a-1)} + \boldsymbol{W}_2^{(a-1)} d_{ij} + \boldsymbol{W}_3^{(a-1)}\boldsymbol{n}_j^{(a-1)}) \tag{9}$$

where $\boldsymbol{W}_1^{(a-1)}, \boldsymbol{W}_2^{(a-1)}$ and $\boldsymbol{W}_3^{(a-1)}$ are learnable weights in a^{th} layer, and $\sigma(\cdot)$ denotes the sigmoid activation function. Here we omit the superscript s since we consider the operation in an arbitrary step. The output of the last graph convolutional layer are inputted into the fully connected layer and then the softmax layer to classify the node into three labels: V2V sender, V2V receiver and V2I sender.

Action: Taking the current environment state as the input, the policy network can output the action. The action is the node classification strategy, denoted by $\mathcal{F}^{(s)} = \text{GNN}\left(\mathcal{G}^{(s)}\right)$.

Reward Function: When the action is determined, **P1** and **P2** can be solved to obtain $t_{\text{V2V}}^{(s)*}$ and $t_{\text{V2I}}^{(s)*}$. The reward function in $s^{\text{th}}$is defined as the negative of the latency in this step, i.e., $R^{(s)} = -\max\{t_{\text{V2V}}^{(s)*}, t_{\text{V2I}}^{(s)*}\}$.

One episode finishes when $\mathcal{V}^{(S)} = \emptyset$, and the overall return $G^{(s)}$ for every step is calculated by $G^{(s)} = \sum_{o=s}^{S-1} \gamma^{o-s} R^{(s)}$, where γ is a given discounting factor. Then, the GNN model weights are updated according to $\boldsymbol{W} \leftarrow \boldsymbol{W} + \alpha G^{(s)} \nabla \ln \Pr(\mathcal{F}^{(s)}|\mathcal{G}^{(s)}, \boldsymbol{W})$, where $\boldsymbol{W}$ denotes all learnable weights in this GNN, and α is the learning rate. The training process is summarized in **Algorithm 1**.

The implementation of the GNN-RL model has two phases: offline training and online decision-making. In the offline training phase, the GNN-RL model is trained based on the collected massive vehicular network state information. In the online decision-making phase, the RSUs obtain the positions and speeds of vehicles through a global position system to get the node and edge features and make dynamic decisions based on the pre-trained model.

Algorithm 1 GNN-RL based framework

```
Initialize the weights of the GNN model W.
for episode in episodes do
  while V^(s) ≠ ∅ do
    Classify nodes according to F^(s) = GNN(G^(s))
    Calculate the reward according to (9), R^(s) = −t^(s)
    Store G^(s), F^(s) and R^(s) in memory
    Obtain the next state according to (5)
  end while
  for s = 0 to S − 1 do
    Calculate the return for this step according to G^(s) = Σ_{o=s}^{S−1} γ^{o−s} R^(s)
    Updates the weights of GNN according to W ← W + αG^(s) ∇ ln Pr(F^(s)|G^(s), W)
  end for
end for
```

V. Experiment

In this section, we evaluate our proposed scheme for an image classification task. A large traffic network is built based on SUMO [13], which is 1500 m × 1500 m in size with 4 crossroads. An RSU is placed at each crossroad. There are 200 vehicles in the traffic network, and each is connected to the nearest RSU. The vehicles move according to the Manhattan mobility model with an average speed of 30 km/h. The RSU schedules all vehicles within 250 meters of itself. The V2I bandwidth is $B_n = 30$ MHz, the V2V bandwidth is $B_{ij} = 10$ MHz. The pathloss exponent is $\beta = 3.86$. The transmit power of vehicles is set to be $P_m = 9$ dBm, and the power spectrum density of the noise is $N_0 = -110$ dBm/MHz.

The dataset we use for FL training is CIFAR-10 [14], which contains 50000 training images and 10000 testing images with ten labels. For the independent and identically distributed (*i.i.d.*) setting, the dataset is uniformly partitioned into 200 subsets, and each subset contains 10 labels. For the *non-i.i.d.* setting, the data samples are sorted according to their labels, and each device holds a disjoint subset of data with 2 labels.

Based on the dataset, we train a convolutional neural network (CNN). This CNN includes six convolutional layers. Three of them are followed by a ReLU layer and a max pooling layer, and the other three are followed by a ReLU layer and a normalization layer. The last convolutional layer is followed by a fully connected layer and a softmax output layer. The learning rate is set to 0.1, and the batch size is 32.

We compare the performance of our proposed VAHFL framework with benchmark schemes. For the first benchmark scheme, we consider that all the vehicles are directly connected to the RSU via orthogonal frequency division multiplexing (OFDM), and the RSU equally allocates bandwidth to each vehicle (denoted by EA). For the second benchmark scheme, the RSU allocates bandwidth to each vehicle according to the policy in [10] (denoted by SA). The total bandwidth

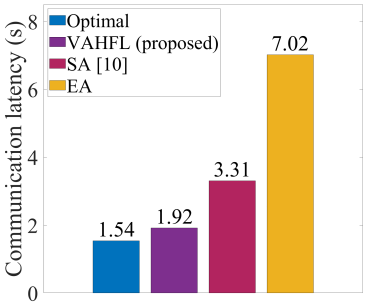

Fig. 2. The communication latency for model uploading.

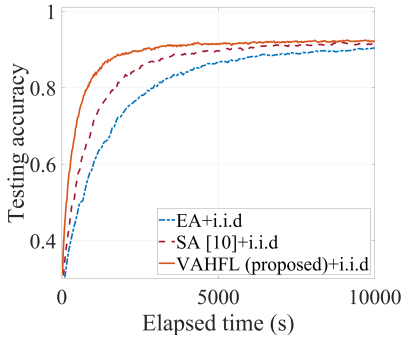

Fig. 3. The performance of the VAHFL compared with benchmarks under *i.i.d.* setting.

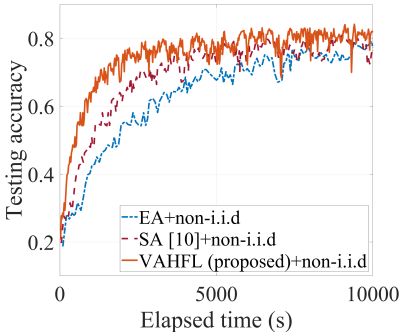

Fig. 4. The performance of the VAHFL compared with benchmarks under *non-i.i.d.* setting.

for both schemes is 40 MHz. Also, we consider the optimal scheme reached by exhaustive searching, where all non-causal information is known in advance.

Firstly, we compare our proposed scheme with the benchmark in terms of the average latency, as shown in Fig. 2. As expected, the optimal policy obtains the lowest latency for model uploading and gives a lower bound to the other policies. VAHFL achieves the lowest latency among all other schemes, which is 72.65% and 41.99% lower than EA and SA. This is because for VAHFL, both V2V and V2I communications are conducted so that each vehicle can occupy more bandwidth for transmission. Also, vehicles are more likely to communicate with other nodes (either vehicles or the RSUs) that are closer to them, so the communication distance is reduced.

We then evaluate our proposed scheme in terms of convergence performance. The testing accuracy of the proposed scheme compared with the benchmark is illustrated in Fig. 3 and Fig. 4. For the *i.i.d.* setting, both VAHFL and the benchmark achieve high testing accuracy, while the convergence speed of VAHFL is much faster than the benchmark since the aggregation latency is reduced. For the *non-i.i.d.* setting, the convergence speed and the highest testing accuracy of the VAHFL scheme exceed two benchmarks. After 1000 seconds of training, VAHFL achieves a testing accuracy of 65.62%, exceeding EA and SA over 20.75% and 12.98%. After 10000 seconds of training, the highest achievable accuracy is 86.24% for VAHFL, 80.13% for EA, and 81.98% for SA.

VI. Conclusion

This work has proposed a VAHFL architecture that leverages the V2V communications to reduce the communication latency for uploading model parameters. The problem was modeled as a sequential decision-making issue, which was then divided into three sub-problems. The first and second problems have been solved based on optimization methods, while the third problem is difficult to solve due to unavailable noncausal information on vehicle positions. A GNN-RL based framework has been proposed to solve it in an online manner. A simulation platform has been built based on SUMO, and our model is evaluated on an image classification task. Experimental results show that our proposed VAHFL scheme can reduce communication latency by at least 41.99% compared with benchmarks. Also, the convergence speed and the testing accuracy of our proposed framework exceed the benchmarks under both *i.i.d.* and *non-i.i.d.* settings.

References

[1] J. Yan, T. Chen, B. Xie, Y. Sun, S. Zhou, and Z. Niu, "Hierarchical federated learning: Architecture, challenges, and its implementation in vehicular networks," *ZTE Communications*, vol. 21, no. 1, pp. 38–45, 2023.

[2] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*. IEEE, 2020, pp. 8866–8870.

[3] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, 2020.

[4] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Hierarchical federated learning with quantization: Convergence analysis and system design," *IEEE Trans. Wireless Commun.*, vol. 22, no. 1, pp. 2–18, 2023.

[5] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 48–54, 2020.

[6] S. Huang, S. Wang, R. Wang, and K. Huang, "Joint topology and computation resource optimization for federated edge learning," in *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2021, pp. 1–6.

[7] M. Yemini, R. Saha, E. Ozfatura, D. Gündüz, and A. J. Goldsmith, "Semi-decentralized federated learning with collaborative relaying," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 1471–1476.

[8] Y. Sun, B. Xie, S. Zhou, and Z. Niu, "Meet: Mobility-enhanced edge intelligence for smart and green 6g networks," *IEEE Communications Magazine*, vol. 61, no. 1, pp. 64–70, 2023.

[9] A. P. Punnen and K. Nair, "Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem," *Discrete Applied Mathematics*, vol. 55, no. 1, pp. 91–93, 1994.

[10] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 453–467, 2020.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[12] Y. Shen, J. Zhang, S. Song, and K. B. Letaief, "Graph neural networks for wireless communications: From theory to practice," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2022.

[13] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic Traffic Simulation using SUMO," in *Proc. IEEE Int. Conf. Intell. Transport. Syst. (ITSC)*. IEEE, 2018.

[14] A. Krizhevsky, V. Nair, G. Hinton *et al.*, "*the cifar-10 dataset.*" 2014. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html