Mobility-Aware Semi-Synchronous Hierarchical Federated Learning for Internet of Vehicles

Zih-Heng Huang, Yan-Wei Chen, and Li-Hsing Yen
Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.

***Abstract*—Hierarchical Federated Learning (HFL) is well-suited to the Internet of Vehicles (IoV), as it mitigates communication bottlenecks and addresses the challenges of spatially non-independent and identically distributed (non-IID) datasets through edge-based client (vehicle) selection and model aggregation. However, traditional fully synchronous aggregation protocols struggle with vehicle heterogeneity, while asynchronous methods often suffer from instability during training and excessive communication costs. Furthermore, existing client selections typically preclude high-mobility vehicles, thereby underutilizing their potential to mitigate non-IID effects. To address these limitations, we propose an adaptive semi-synchronous HFL architecture. Unlike static approaches, our alternating reinforcement learning framework adaptively manages the degree of semi-synchrony at each edge server and jointly employs a client-selection mechanism that leverages heterogeneity context, such as mobility and computational capability, derived from vehicle microscopic features to ensure robust and efficient HFL training. Simulations with SUMO-controlled vehicle mobility confirm robust convergence and demonstrate higher accuracy and lower global training loss than counterparts.**

***Keywords*—Hierarchical Federated Learning, Semi-Synchronous Aggregation, Non-Independent and Identically Distributed, Client Selection, Reinforcement Learning, Internet of Vehicles, Vehicle Edge Computing**

I. Introduction

With the advancement of on-board computing capabilities, upcoming connected and autonomous vehicles (CAVs) can collect massive amounts of data for advanced driving systems [1] or for future intelligent transportation management. While Vehicular Edge Computing (VEC) infrastructures and wireless technologies facilitate connectivity between vehicles and roadside units (RSUs) [2], traditional centralized processing of collected raw data incurs bandwidth costs and privacy risks. Federated Learning (FL) mitigates these issues by allowing vehicles to upload only model updates rather than raw data. It enables distributed and collaborative training without privacy leakage and raw data transfer. However, given the large scale of the Internet of Vehicles (IoV), data possessed by speed-varying and spatially-distributed vehicles may exhibit significant non-independent and identically distributed (non-IID) characteristics induced by mobility and location [3]. A way to effectively manage this heterogeneity is to follow the client-edge-cloud Hierarchical Federated Learning (HFL) framework [4] to structure IoV into three tiers: global cloud server, RSU edge server, and vehicle client, where RSU edge servers serve as intermediate and regional aggregators. This hierarchical architecture alleviates communication bottlenecks [4, 5] and buffers the impact of non-IID data [6].

Despite these structural advantages, aggregation remains a challenge, especially between clients and edge servers. Existing studies primarily adopt either full-synchronous [7] or fully-asynchronous [8, 9] aggregation. Synchronous aggregations may suffer from the straggler effect due to computation-constrained clients [10, 11], while asynchronous approaches induce heavy communication loads and instability due to frequent model updates [12, 13]. Although some semi-synchronous studies exist, they either do not apply to HFL [14] or usually rely on static or simple metrics [15] to determine synchrony without capturing the dynamics of IoV.

In addition to the aggregation strategy, client selection also plays a critical role in FL for IoV. Early studies noted that high-mobility vehicles often generate poor-quality data, straggle, and drop out [16, 17], while heterogeneity across vehicles leads to varying computational capabilities and data quality [11, 18]. Prior studies tended to select clients based on presumed patterns, such as mobility, data quality, and resource availability. They also exhibited a strong bias against high-speed vehicles. However, recent works offer a paradigm shift. Chen et al. [19] argued that high mobility actually promotes rapid data and model fusion, particularly for non-IID distributions in large-scale scenarios. Additionally, Bian and Xu [20] exploited high-speed vehicles as relay nodes to improve connectivity. Recognizing that mobility impacts are highly scenario-dependent, an effective selection must move beyond traditional naive criteria and leverage high-mobility vehicles. Thus, the latest studies considered mobility as a critical factor to coordinate client selection or semi-synchronization strategies [5, 11, 14].

This paper adopts the HFL framework for IoV, in which the cloud and edge servers use synchronous and semi-synchronous model updates/aggregations, respectively. Edge servers at RSUs 1) manage the degree of semi-synchrony between the edge and clients by dynamically partitioning the cloud's global aggregation round into shorter aggregation slots for clients' model updates, and 2) select vehicle clients for each aggregation slot. These two tasks are tightly coupled: slot length constrains the feasible client set, while heterogeneous client characteristics call for flexible slot lengths. Nevertheless, prior work treated synchronization control and client selection separately. We develop an alternating reinforcement learning (RL) framework to jointly optimize both tasks. Though RL has been widely adopted for FL control, existing studies mainly focused on client selection [11, 21] or resource allocation [21] and overlooked dynamic semi-

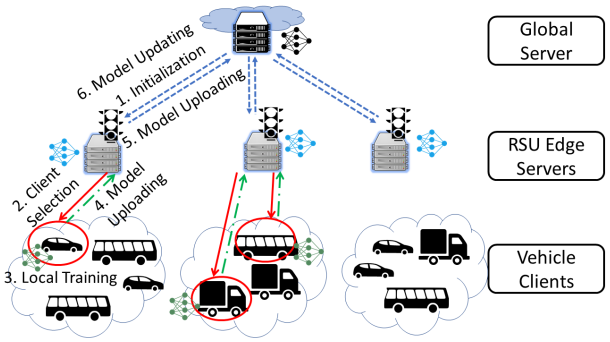


Figure 1: The Client-Edge-Cloud HFL Framework in IoV

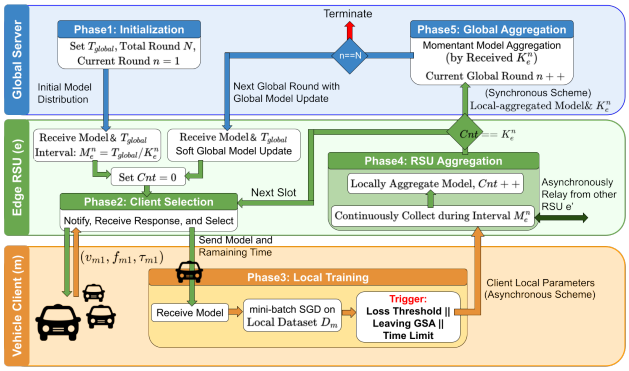


Figure 2: HFL System With Two-Level Model Updates/Aggregations

synchrony regulation. By jointly coordinating semi-synchrony control and characteristic-aware client selection, the proposed approach avoids suboptimal decoupled designs and adapts effectively to IoV dynamics. The main contributions are summarized as follows:

- HFL with Two-Level Model Aggregations: We separate the aggregation policy for edge servers from that for the cloud. While the cloud uses traditional synchronous model updates/aggregations to maintain model stability, the edge uses a semi-synchronous method to dynamically adjust the degree of synchronization in response to client heterogeneity.
- Alternating RL for Joint Optimization: We propose an alternating RL scheme for adaptive semi-synchrony control and client selection. The approach effectively manages task dependencies yet adaptively drives the system toward efficient, stable convergence.
- Realistic Spatiotemporal Simulation: Unlike existing works that rely on simplified or static mobility models, we integrate high-fidelity microscopic traffic simulation to validate the proposed framework under dynamic, non-IID conditions driven by continuous vehicular mobility.

The remainder of this paper is organized as follows: Sec. II details the system model and problem formulation. Sec. III presents the proposed alternating RL framework. Sec. IV provides the experimental results, and the last section concludes this study with future research discussion.

II. System Model

Fig. 1 depicts the proposed three-tier HFL system, referred to as the VEC architecture, which consists of vehicles, edge servers at RSUs, and a global server. Operating within a dynamic IoV environment, the architecture is tailored to manage high mobility and the pronounced spatially non-IID characteristics of vehicle-collected data. Fig. 2 delivers the detailed interactions between each tier.

A. Global Server, Edge Servers, and Vehicle Clients

The global server orchestrates the entire Global Server Area (GSA) and maintains a global model across all edge servers. The global server performs synchronous model update with edge servers. On the other hand, edge servers perform semi-synchronous FL with vehicles. Edge servers collect vehicle models over a short time interval and aggregate them at the end of the interval. A very short time interval makes edge servers collect and aggregate models from zero or one vehicle, rendering the scheme an asynchronous FL. Conversely, edge servers synchronously incorporate all vehicle models during a sufficiently long time interval (e.g., equal to the length of a global round). The time required to collect and aggregate vehicle models thus determines the degree of semi-synchrony. Specifically, each edge server e manages semi-synchrony in the n-th global round by dividing the global round into K_e^n local time slots and collecting/aggregating vehicle models in each time slot.

We assume that each vehicle m processes a static dataset D_m with batch size ξ_m proportional to its computing power f_m. Each D_m contains spatially-dependent non-IID data with mobility-induced motion blur [16]. Furthermore, each vehicle has the ability to estimate the remaining time to exit the GSA.

B. Synchronization Between Global and Edge Servers

Each round in the synchronous FL is of a fixed interval T_{global}. The global server transfers the global model to each edge server at the beginning of each round. At the end of each round n, the global server collects edge model $\omega_e^{n,*}$ from each edge server e and performs momentum aggregation [22] with coefficient η to get the global model for the next global round $n+1$:

$$\omega_g^{n+1} = \eta \cdot \sum_{e\in E} \frac{K_e^n}{\sum_{e'\in E} K_{e'}^n} \omega_e^{n,*} + (1-\eta)\cdot \omega_g^n,\ n = 0 \sim N-1, \tag{1}$$

where edge e's aggregation weight is proportional to K_e^n. The global server then transfers ω_g^{n+1} to each edge for the next round. Each edge e does not use ω_g^{n+1} to replace its edge model directly but employs a soft update with coefficient ψ to merge ω_g^{n+1} with its old model $\omega_e^{n,*}$ for the next round:

$$\omega_e^{n+1,0} = (1-\psi)\cdot \omega_e^{n,*} + \psi \cdot \omega_g^{n+1},\ n = 0 \sim N-1. \tag{2}$$

C. Local Training of Vehicle Clients

Each edge server updates its edge model by (2) at the beginning of each new round and uses it as the initial edge

model for the first time slot of this round. At the beginning of each time slot i, each edge e notifies clients within its coverage of the remaining time in this round and releases its edge model $\omega_e^{n,i}$ to a selected set of vehicles $\mathbf{x}_e^{n,i}$ (i.e., FL clients) based on their reported microscopic attributes. Upon receiving $\omega_e^{n,i}$ from e, each vehicle m initializes its vehicle model $\omega_m^{n,i,0}$ as $\omega_e^{n,i}$ and optimizes its vehicle model for classification with standard cross-entropy loss via mini-batch SGD with learning rate ζ. The training data ξ_m are sampled from D_m. The SGD update rule for the vehicle client is

$$\omega_m^{n,i,t+1} = \omega_m^{n,i,t} - \zeta \nabla \mathcal{L}_{loc}\big(\omega_m^{n,i,t}; \xi_m\big)\ ,\ t = 0, 1, 2, \cdots\ , \quad (3)$$

where $\omega_m^{n,i,t}$ are m's model parameters in slot i of round n after t-epoch training and $\mathcal{L}_{loc}$ is mini-batch cross-entropy loss.

Unlike traditional FL with fixed local epochs, our design requires vehicles to terminate tasks and update their models in response to system dynamics. Vehicle uploads are triggered upon leaving the GSA, upon reaching the global timeout, or upon training completion. A vehicle m terminates its local training normally if $\Delta\mathcal{L}_{loc} < 0.01\mathcal{L}_{loc}^{*}$ holds for 5 consecutive epochs, where $\Delta\mathcal{L}_{loc}$ denotes m's local training loss improvement and $\mathcal{L}_{loc}^{*}$ is m's lowest local training loss.

D. Semi-Synchronous Aggregation of Edge Models

Each edge collects vehicle models during each time slot and aggregates them at the end of the slot. Let $S_e^{n,i}$ be the set of vehicles that upload their models to edge e in slot i of round n. Let $\omega_m^{n',j,*}$ be the vehicle model uploaded by $m \in S_e^{n,i}$. Edge e will discard the received model (and remove m from $S_e^{n,i}$ accordingly) if it is obsolete, namely $n' < n$. For the vehicle model that was not inherited from e's edge model due to vehicle mobility, e relays it to the original edge that released it. On the contrary, the edge model with $n' = n$ and $j \le i$ is deemed a fresh update originating in the current or previous slot of the same round from e. Hence, edge e performs aggregation and assigns received $\omega_m^{n,j,*}$ an aggregation weight $\alpha_{e,m}^{n,i,j}$ that is proportional to $|D_m|$, i.e., $\alpha_{e,m}^{n,i,j} = |D_m| / \sum_{m' \in S_e^{n,i}} |D_{m'}|$ and uses a staleness factor $\beta_j^i = (1 + \delta(i-j))^{-1}$ to penalize a possible outdated late model update, where δ is a positive number. The edge model $\omega_e^{n,i+1}$ that will be distributed by e at the beginning of the subsequent slot $i+1$ in round n is given by:

$$\omega_e^{n,i+1} = \sum_{m \in S_e^{n,i}} \alpha_{e,m}^{n,i,j} \beta_j^i\, \omega_m^{n,j,*},\ i = 0 \sim K_e^n - 1, \quad (4)$$

where $\omega_m^{n,j,*}$ is uploaded model from vehicle m to edge server e in slot i of round n.

E. Problem Definition and Decision Tradeoff

We aim to minimize the global classification loss $\mathcal{L}(\boldsymbol{\omega}_g^N)$ after N rounds. $\mathcal{L}(\boldsymbol{\omega}_g^N)$ is evaluated on a centralized validation dataset at the global server to obtain a uniform, independent performance metric. The decision variables for each edge server e include the slot count K_e^n in each round n and the vehicle selection vector $\mathbf{x}_e^{n,i}$ in each slot i. The problem is defined as:

$$\begin{aligned} \min_{\{K_e^n,\, x_{e,m}^{n,i}\}} \quad & \mathcal{L}(\boldsymbol{\omega}_g^N) \\ \text{s.t.} \quad & K_e^n \in \{1, \ldots, 10\}, \quad x_{e,m}^{n,i} \in \{0,1\}, \quad (1),\ (4), \\ & \forall m \in \mathcal{V}_e^{n,i},\ e \in \mathcal{E},\ i = 0, \ldots K_e^n - 1,\ n = 0, \ldots N-1, \end{aligned} \quad (5)$$

where $\mathcal{V}_e^{n,i}$ is the set of vehicles within edge RSU e's coverage in slot i of global round n, and $\mathcal{E}$ is the set of edge RSUs.

To maintain computational tractability under a wide-ranging control of semi-synchrony, we discretize K_e^n into a finite set of candidate integers. Specifically, we set $K_e^n \in \{1, \ldots, 10\}$ in (5). This range provides a sufficient granularity to explore the trade-off between aggregation frequency and non-IID impacts without exhaustive computation. A large K_e^n value accelerates global convergence through frequent updates but increases communication load and compromises stability due to non-IID variance across shorter slots. Small K_e^n ensures robust, diverse aggregation but slows down the training. When it comes to vehicle client selection, selecting high-speed vehicles alleviates non-IID impact via data fusion but faces challenges of data blurring and limited connection. Conversely, low-speed vehicles provide stable, high-quality updates but intensify label locality, inducing global divergence. The two decision trade-offs above pose critical challenges for our research.

III. Proposed Approach

Unlike rule-based methods that fail at scale and in highly dynamic IoV, the joint optimization of the semi-synchrony degree and client selection is inherently a localized, sequential decision-making problem for edge servers in a non-stationary environment. RL emerges as the ideal role here, as it can leverage the environment's feedback to balance the long-term trade-off between aggregation frequency and non-IID impacts. We further decompose the edge-level problem into coupled decisions: 1) determining the degree of semi-synchrony via the number of waiting slots in each global round, and 2) selecting appropriate clients per waiting slot. To address the mutual dependency between these tasks, we propose a strategy in which two decisions are alternately optimized.

We use the Deep Q-Network (DQN) algorithm [23] for the two decomposed decisions. The Semi-Synchronization Agent (SSA) employs a two-hidden-layer MLP to determine the number of waiting slots in every round. The SSA's output actions are delivered to the Client Selection Agent (CSA), which uses a similar structure to select clients for each waiting slot. The CSA's outputs are then fed back to the SSA for the next training episode (refer to Algorithm 1). This alternation allows the system to dynamically adapt semi-synchrony and selections, significantly improving training efficiency.

A. Markov Decision Process (MDP) for SSA and CSA

$\mathcal{M}_z = \langle \mathcal{S}_z, \mathcal{A}_z, \mathcal{P}_z, R_z, \gamma_z \rangle$ denotes the MDP for task $z \in \{\text{SSA}, \text{CSA}\}$. We omit the transition $\mathcal{P}_z$ due to model-free.

- **State:** $\mathcal{S}_{\text{SSA}}$ incorporates macroscopic features of edge server and vehicles within coverage, including the global

round number, previous number of waiting slots, number of vehicles in its service coverage, and average vehicle speed, computing capability, and remaining time to exit the GSA; $\mathcal{S}_{\text{CSA}}$ contains per-vehicle microscopic attributes, namely velocity, computational capability, and remaining time of each covered vehicle, along with the number of waiting slots and the global round number.

- **Action:** $\mathcal{A}_{\text{SSA}} \in \{1..10\}$ is the discretized number of waiting slots in a global round; $\mathcal{A}_{\text{CSA}} \in \{0,1\}^*$ is a mask to select vehicles.
- **Reward:** r_z for agent $z \in \{\text{SSA}, \text{CSA}\}$ in round n is

$$r_z = \tanh(\frac{\Delta\mathcal{L}_z}{\lambda_z}) \cdot (0.5 + 0.5\frac{n}{N}), \tag{6}$$

where $\Delta\mathcal{L}_z$ is global loss drop scaled and λ_z = 3 for SSA and $\Delta\mathcal{L}_z$ is RSU relative loss drop and λ_z = 0.5 for CSA. This design is to maintain stability in the early stage while encouraging improvement in the later stage.

Table I summarizes the corresponding hyperparameters.

Algorithm 1 Alternating DQN Procedure

Input: E, global rounds N, $\gamma_{\text{SSA}}, \gamma_{\text{CSA}}, \alpha_{\text{SSA}}, \alpha_{\text{CSA}}, \tau$
Output: Trained model of SSA θ_{SSA} and CSA θ_{CSA}
Initialize: Global FL model ω_g^0, θ_{SSA}, $\bar{\theta}_{\text{SSA}} \leftarrow \theta_{\text{SSA}}$, θ_{CSA}, $\bar{\theta}_{\text{CSA}} \leftarrow \theta_{\text{CSA}}$, $\mathcal{D}_{\text{SSA}}, \mathcal{D}_{\text{CSA}}$, $\epsilon_{\text{SSA}}, \epsilon_{\text{CSA}}$

for episode $e = 1, 2, \ldots, E$ **do**
  Reset environment with initial observation s
  if e is odd **then** ▷ **Phase SSA**
    $\epsilon_{CSA} \leftarrow 0$ ▷ greedy CSA selects clients
    for global round $n = 1, 2, \ldots, N$ **do**
      Observe state S_{SSA}
      Sample slot-count $a_{\text{SSA}} \sim \pi(\epsilon_{\text{SSA}})$
      Execute a_{SSA} slots; assess global performance
      $r_{\text{SSA}} \leftarrow$ Eq. (6) with $z = \text{SSA}$
      Put $(S_{\text{SSA}}, a_{\text{SSA}}, r_{\text{SSA}}, S'_{\text{SSA}})$ in $\mathcal{D}_{\text{SSA}}$
      Update θ_{SSA} with minibatch $\sim \mathcal{D}_{\text{SSA}}$
        Compute TD-target; minimize TD-Error
        Soft update: $\bar{\theta}_{\text{SSA}} = \tau\theta_{\text{SSA}} + (1-\tau)\bar{\theta}_{\text{SSA}}$
  else ▷ **Phase CSA**
    $\epsilon_{\text{SSA}} \leftarrow 0$ ▷ greedy SSA chooses slot number K
    for global round $n = 1, 2, \ldots, N$ **do**
      for slot $i = 1, \ldots, K$ **do**
        Observe state S_{CSA}
        Get client selection $\mathbf{a}_{\text{CSA}} \sim \pi(\epsilon_{\text{CSA}})$
        Apply $\mathbf{a}_{\text{CSA}}$; evaluate edge aggregation
        $r_{\text{CSA}} \leftarrow$ Eq. (6) with $z = \text{CSA}$
        Put $(S_{\text{CSA}}, \mathbf{a}_{\text{CSA}}, r_{\text{CSA}}, S'_{\text{CSA}})$ in $\mathcal{D}_{\text{CSA}}$
        Update θ_{CSA} with minibatch $\sim \mathcal{D}_{\text{CSA}}$
          Compute TD-target; minimize TD-Error
          Soft update: $\bar{\theta}_{\text{CSA}} = \tau\theta_{\text{CSA}} + (1-\tau)\bar{\theta}_{\text{CSA}}$

IV. Performance Evaluation

We considered one global server and four edge servers on a 7×7 Manhattan grid as shown in Fig. 3. The GSA encompasses the entire area, while edge servers partition it equally into four square subareas under their respective coverage. We utilized SUMO [24] to control microscopic vehicular mobility in the GSA, where a vehicle entered the GSA from one of the four directions at a constant speed and departed through a different direction. We set D_m to fall into one of the following four distinct datasets depending on m's dictated entrance: $D_1 = \{y_0, y_1, y_2\}$, $D_2 = \{y_3, y_4\}$, $D_3 = \{y_5, y_6\}$, and $D_4 = \{y_7, y_8, y_9\}$. Although real vehicles generate sensing data, we utilize the standard CIFAR-10 dataset as a representative proxy for visual perception tasks (e.g., obstacle recognition) to evaluate the learning performance and non-IID mitigation of the proposed algorithm in IoV environment. Moreover, a vehicle m with speed v_m has $v_m\%$ of its dataset D_m blurred by a Gaussian blur [25] (e.g., a vehicle traveling at 20 m/s has 20% blurred data samples). Table II details simulation parameters.

Table I: Hyperparameters of DQNs for SSA and CSA

Parameter	**SSA**	**CSA**
Batch Size	32	8
Learning Rate α for Adam	1×10^{-3}	1×10^{-4}
Hidden Layer Sizes	(128, 128)	(64, 64)
Replay Buffer Size $\mathcal{D}$	10,000	
Discount Factor γ	0.99	
Target DQN Soft-Update τ	0.02	
Exploration ϵ (Start $\rightarrow$ End)	1.0 $\rightarrow$ 0.05 (Decay: 0.995)	
Total Episodes E	100	

We conducted experiments on an Ubuntu 24.04 system with an Intel Core i7-14700 CPU, NVIDIA GeForce RTX™ 4090, and 128 GB of memory. We created a separate process with GPU memory strictly capped at 1% for the vehicle to perform clients' training tasks. The temporal settings and grid size ensured sufficient local training iterations and high aggregation density within each global round.

The RL agent dynamically learns by interacting with the SUMO-simulated IoV environment. The interaction generates transitions during training, which are stored in a Replay Buffer

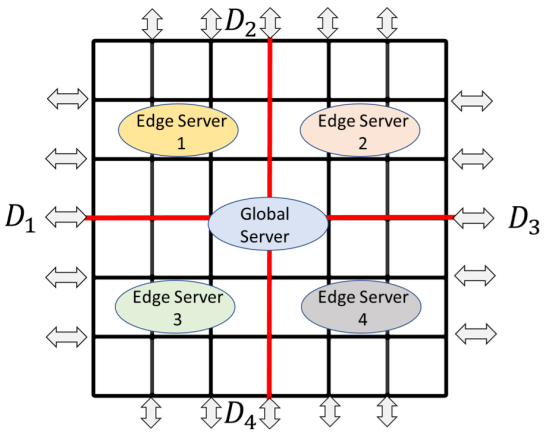


Figure 3: 7×7 GSA. Vehicular traffic continuously enters from 20 boundary gates across four cardinal directions, where each entry direction intrinsically dictates the specific non-IID dataset (D_1 to D_4) assigned to the vehicles.

Table II: Simulation Parameters

Parameter	Value	Parameter	Value
Grid Layout	7×7 (250m)	Global Rounds	15 (120 s each)
Roads	4-lane, 2-way	Vehicle Speed	5 – 20 m/s
Entrances/Exits	20/20 gates	Vehicle Gen. Rate	0 – 6 vehicle/s
Servers	1 Global, 4 Edge	Total Vehicles	≈ 850
Dataset	CIFAR-10	Model	ResNet18
δ (Staleness)	1	η (Momentum)	0.1
ψ (Edge Soft-Update)	0.7		

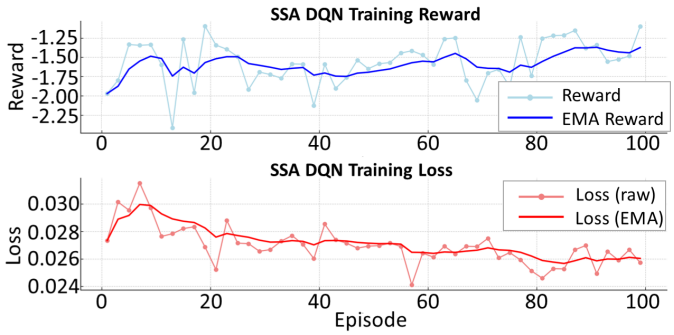


Figure 4: SSA DQN Training Log

Table III: Convergence analysis of the SSA DQN

Metric	Value	Interpretation
Reward slope (last 20%)	-0.0097	Near zero, plateau reached
Reward CV (last 20%)	0.115	Low variation, stable
Loss slope (last 20%)	6.2×10^{-5}	Near zero, stable loss
Loss CV (last 20%)	0.030	Very low variation

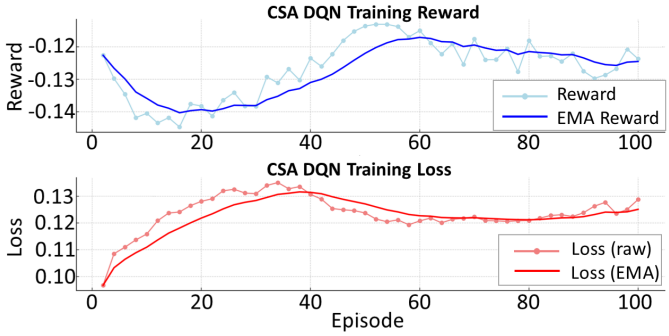


Figure 5: CSA DQN Training Log

for network updates. Fig. 4 illustrates the training log of the SSA DQN, and Table III provides the corresponding statistical results. We analyzed the convergence stability of the SSA DQN by statistical metrics over the final 20% of training episodes. Specifically, we examined the slope, derived from a linear regression fit of the reward and loss curves, and the coefficient of variation (CV), calculated as the ratio of the standard deviation to the mean. As presented in Table III, both the reward and loss curves demonstrate near-zero slopes, indicating stable learning. Furthermore, the low CV values reflect consistent performance with minimal fluctuation. These statistical results confirm that the SSA achieves robust convergence in the final stage of training.

Fig. 5 shows the training log of the CSA DQN, while Table IV lists the associated statistical results. Both the reward and loss slopes were nearly zero, while the CV values were extremely small, confirming that the RL training process was also highly stable. Moreover, compared with the SSA, the CSA exhibited even lower variability in both reward and loss.

To evaluate the effectiveness of the proposed framework, we compared it against two baseline strategies (Table V). The Random strategy randomly selects clients and sets the number of waiting slots. The Extreme strategy prioritizes the fastest vehicles and fixes the number of waiting slots to the maximum (i.e., $K = 10$).

We conducted experiments for these strategies across ten environments. Each of them had randomized vehicle trajectories and enter-exit points, along with fixed traffic distributions and vehicle speed. Each strategy was averaged across two execution runs to account for the stochastic nature of synchronization agent or client selection agent. Fig. 6 shows the average HFL performance of each strategy over ten environments. The proposed method achieved the best performance in both HFL accuracy and loss. This advantage stems from the agents' capacity to dynamically tailor the degree of semi-synchrony and client selection to the changing system state. The Random strategy ranked second, surpassing the Extreme strategy, owing to its inherent flexibility on well-behaved trajectories and the tolerance nature of HFL. Notably, the Extreme strategy yielded the poorest results, revealing that aggressive model updating combined with high-mobility clients can be counterproductive in HFL.

V. Conclusions

We have studied dynamic client selection and synchrony control in a semi-synchronous HFL framework for IoV with heterogeneous vehicles. Experimental results demonstrated convergence of the proposed framework, as well as higher accuracy and lower global training loss in HFL. Crucially, we refute the assumption that fast-moving vehicles are detrimental; instead, we account for them during selection to potentially promote data and model fusion. By dynamically adjusting the degree of semi-synchrony and client selection, the adaptive mechanism significantly outperforms rule-based baselines.

Future research will address key challenges in the HFL-IoV framework through three main directions. First, we aim to replace rule-based termination with experience-driven strategies that adaptively align with global model maturity. Second, we will rigorously analyze the impact of momentum and mobility on convergence using advanced metrics, such as volatility, curvature, flap rate, and monotonicity. Moreover, we also plan to explore other state-of-the-art strategies and separate optimization. Finally, we intend to elevate the architecture to a collaborative multi-agent framework, enabling inter-RSU coordination to optimize parameter filtering and global performance while accounting for realistic delay and drop conditions.

Acknowledgment

This work was supported by the National Science and Technology Council of Taiwan under grant numbers NSTC

Table IV: Convergence analysis of the CSA DQN agent

Metric	Value	Interpretation
Reward slope (last 20%)	-7.6×10^{-5}	Near zero, plateau reached
Reward CV (last 20%)	0.023	Very low variation, stable
Loss slope (last 20%)	3.0×10^{-4}	Near zero, stable loss
Loss CV (last 20%)	0.018	Very low variation

Table V: Decision Strategies

Strategy	Client Selection	Num. of waiting slots
Random	Randomly selected	Randomly set
Proposed	Selected by CSA DQN	Set by SSA DQN
Extreme	Fastest-vehicle first	Fixed to the maximum

114-2218-E-A49-017 and NSTC 114-2218-E-A49-018.

REFERENCES

[1] T.-K. Chi *et al.*, "An edge computing system with amd xilinx fpga ai customer platform for advanced driver assistance system," *Sensors*, vol. 24, no. 10, 2024.

[2] G. Yan *et al.*, "Edge intelligence for internet of vehicles: A survey," *IEEE Trans. Consum. Electron.*, vol. 70, no. 2, pp. 4858–4877, 2024.

[3] B. Li *et al.*, "Feel: Federated end-to-end learning with non-iid data for vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16 728–16 740, 2022.

[4] L. Liu *et al.*, "Client-edge-cloud hierarchical federated learning," in *IEEE ICC*, 2020, pp. 1–6.

[5] D. Wang *et al.*, "Mobility-aware reputation-based hierarchical federated learning for internet of vehicles," *IEEE Trans. Veh. Technol.*, pp. 1–12, 2025.

[6] N. Mhaisen *et al.*, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, p. 55–66, Jan. 2022.

[7] B. Xie *et al.*, "Mob-fl: Mobility-aware federated learning for intelligent connected vehicles," in *IEEE ICC*, 2023, pp. 3951–3957.

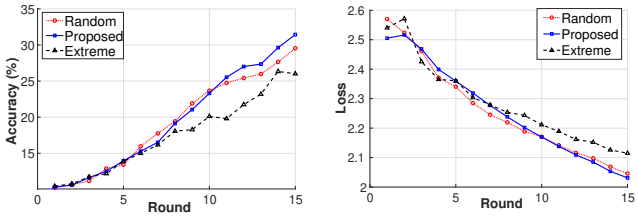
[8] S. Wang *et al.*, "Mobility-aware asynchronous federated learning for edge-assisted vehicular networks," in *IEEE ICC*, 2023, pp. 3621–3626.

[9] Z. Yang *et al.*, "Efficient asynchronous federated learning research in the internet of vehicles," *IEEE Internet Things J.*, vol. 10, no. 9, pp. 7737–7748, 2023.

[10] N. Lang, A. Cohen, and N. Shlezinger, "Stragglers-aware low-latency synchronous federated learning via layer-wise model updates," *IEEE Transactions on Commun.*, vol. 73, no. 5, pp. 3333–3346, 2025.

[11] Y. Li *et al.*, "Joint optimization for volatile federated learning in vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 74, no. 9, pp. 14 632–14 644, 2025.

[12] T.-D. Cao *et al.*, "Asyn2f: An asynchronous federated learning framework with bidirectional model aggregation," *IEEE Trans. Emerg. Top. Comput.*, vol. 13, no. 4, pp. 1618–1632, 2025.



(a) Average accuracy (b) Average loss

Figure 6: Accuracy and Loss of HFL over 10 environments w.r.t. strategy

[13] W. Gu and Y. Zhang, "Fedeem: a fairness-based asynchronous federated learning mechanism," *J. Cloud Comput.*, vol. 12, no. 1, p. 154, 2023.

[14] Z. Jin *et al.*, "Mobility-aware semi-asynchronous federated learning for vehicular networks," *IEEE Trans. Veh. Technol.*, pp. 1–12, 2025.

[15] F. Liang *et al.*, "Semi-synchronous federated learning protocol with dynamic aggregation in internet of vehicles," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4677–4691, 2022.

[16] H. Xiao *et al.*, "Vehicle selection and resource optimization for federated learning in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11 073–11 087, 2022.

[17] X. Chang *et al.*, "Mobility-aware vehicle selection strategy for federated learning in the internet of vehicles," in *Int'l Conf. on Commun., Comput., Cybersecur., and Inform. (CCCI)*, 2024, pp. 01–06.

[18] X. Li *et al.*, "Mitigating data imbalance in federated learning for the iov: A louvain-based participant selection strategy," *IEEE Trans. Veh. Technol.*, pp. 1–6, 2025.

[19] T. Chen *et al.*, "Mobility accelerates learning: Convergence analysis on hierarchical federated learning in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 74, no. 1, pp. 1657–1673, 2025.

[20] J. Bian and J. Xu, "Accelerating asynchronous federated learning convergence via opportunistic mobile relaying," *IEEE Trans. Veh. Technol.*, vol. 73, no. 7, pp. 10 668–10 680, 2024.

[21] F. Zheng *et al.*, "Fedaeb: Deep reinforcement learning based joint client selection and resource allocation strategy for heterogeneous federated learning," *IEEE Trans. Veh. Technol.*, vol. 73, no. 6, pp. 8835–8846, 2024.

[22] Z. Cheng *et al.*, "Momentum benefits non-iid federated learning simply and provably," in *The 12th Int'l Conf. on Learn. Represent. (ICLR)*, 2024.

[23] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[24] P. A. Lopez *et al.*, "Microscopic traffic simulation using sumo," in *The 21st IEEE Intell. Transp. Syst. Conf. (ITSC)*, 2018.

[25] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. Pearson Education, 2018.