Federated Training Generative Adversarial Networks for Heterogeneous Vehicle Scheduling in IoV

Lizhao Wu, Hui Lin, Xiaoding Wang

***Abstract*—In autonomous driving environments, Generative Adversarial Network (GAN) are often used to predict the future trajectories of objects in the scene, providing decision support for autonomous driving systems. However, integrating GAN models into the Internet of Vehicles (IoV) poses numerous challenges. Firstly, GAN models necessitate user data and extensive computing resources, whereas diverse Intelligent Connected Vehicle(ICV) possess limited bandwidth and computational capabilities, making it challenging to deploy models of the same scale as those in the cloud. Secondly, multi-faceted aspects, including energy consumption, computation, communication, and vehicle training scheduling, have yet to be thoroughly examined, particularly in the context of IoV's limited resources. To address above issues, we propose a novel federated learning framework, Heterogeneous-Vehicle-Scheduling-GAN (HVS-GAN), for training GANs in resource-constrained IoV environments. HVS-GAN balances GAN generation quality and training costs in IoV. It supports multiple ICVs training GAN models of different structures, breaking the strong assumption of uniform GAN model size constraints in previous works and enabling collaborative learning within IoV. Furthermore, to balance quality and training costs, we incorporate Deep Deterministic Policy Gradients learning to manage varying model size constraints, training delays, and training consumption across participating ICVs. Experimental results and analysis confirm the superiority of our proposed HVS-GAN solution, which achieves better outcomes in IoV scenarios with stringent model size constraints compared to state-of-the-art algorithms.**

***Index Terms*—Internet of Vehicles, Generative Adversarial Network, Federated Learning, Reinforce Learning, Deep Deterministic Policy Gradients.**

I. Introduction

Generative Adversarial Network (GAN) models have demonstrated immense potential in autonomous driving environments. They are not only widely used to predict the future trajectories of other vehicles, pedestrians, and other objects in the scene, providing precise and real-time decision support for autonomous driving systems and ensuring safe and efficient driving [1]; but also, GANs can generate highly personalized content based on users' driving habits, preferences, and even emotional states, such as customized navigation routes [2] and driving entertainment recommendations [3], significantly enhancing users' travel comfort and satisfaction.

However, applying GAN models to Internet of Vehicles(IoV) services poses several significant challenges. Firstly, GAN models heavily rely on user data and consume substantial computational resources, while many Intelligent Connected Vehicles (ICV), constrained by hardware and energy limitations [4], [5], often cannot provide sufficient computing power to support these demands. Furthermore, even in ICVS with adequate computational resources, effectively balancing computational and communication energy consumption to ensure system sustainability and efficiency remains an urgent issue.

To address the mismatch between GANs and IoV-supported services, recent research has delved into foundational strategies in this domain. Firstly, some studies have leveraged Federated Learning (FL) to facilitate GAN deployment in IoT scenarios. For instance, FedGAN [6] pioneered the extension of FL to GANs, setting distinct learning rates for different components of the GAN network and regularly communicating with the server. Another study, Cap-GAN [7], introduced a novel approach where discriminators were trained on User Equipment (UEs) and generators on the server, reducing overhead through this separated deployment.

However, the primary challenge in adopting FL systems stems from limited resources, as client devices participating in FL (e.g., smartphones, ICVs) exhibit significant variations in computing power, bandwidth, and other capabilities. Researchers have also endeavored to address FL under resource constraints. FedProx [8] allows each client to train in different number of iterations based on local resources, but it presupposes that all clients train the identical model. While this approach somewhat mitigates the heterogeneity issue, it does not fundamentally address the mismatch between model architectures and device computing capabilities. HeteroFL [9] introduced a novel approach that involves partitioning the global model horizontally, maintaining the entire depth of the Deep Neural Network (DNN) architecture on each client. This strategy adjusts the width partitioning ratio specifically to accommodate the varying capabilities of heterogeneous clients, thereby offering a tailored solution to the challenges posed by diverse client environments. Zhang [10], Zarandi and Tabassum [11], Chen [12], among others, focused more on energy consumption issues, formulating an optimization problem that considers computation delay, communication delay, bandwidth, and other factors. They also introduced Reinforcement Learning(RL) methods as schedulers to assist client selection in addressing these challenges.

Although current research has delved into GAN implementation methods, frameworks, and application cases in edge computing scenarios, and considered FL strategies in resource-constrained environments, these studies remain insufficient

Lizhao Wu, Hui Lin and Xiaoding Wang are with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, 350117, China, Email: melowlz@yeah.net,linhui@fjnu.edu.cn and wangdin1982@fjnu.edu.cn.

Corresponding authors: Hui Lin, Xiaoding Wang.

when confronted with the specific challenges of IoV. In the IoV environment, ICVs' computing power and communication bandwidth are strictly limited, necessitating careful selection of suitable ICVs to participate in GAN training to ensure both efficiency and high-quality GAN model performance under resource constraints.

Therefore, in this paper, we innovatively propose a framework named "Heterogeneous Vehicle Scheduling Generative Adversarial Network" (HVS-GAN) to tackle the challenges faced by GANs training in resource-constrained IoV environments, particularly within the context of FL deployment. We construct an optimization model that focuses not only on enhancing generation quality but also on optimizing training costs, while introducing stricter constraints in the form of differentiated model size limitations for each vehicle. To overcome this limitation, we integrate heterogeneous model technology, enabling HVS-GAN to effectively train standard-sized GAN models in the cloud environment, even under resource-constrained conditions. Furthermore, we employ the Deep Deterministic Policy Gradient (DDPG) algorithm to balance generation quality and training costs, determining participating ICVs for each aggregation round. This strategy significantly reduces training costs while ensuring unaffected GAN model generation quality, achieving an optimized balance between resource utilization and model performance. The main contributions of this paper are summarized as follows:

- We investigates the application of FL for GAN in the resource-constrained IoV scenario, where ICVs have limited computation and communication capabilities. Targeting this complex scenario, we innovatively formulate an optimization problem that encapsulates the trade-off between resource constraints and generation quality, aiming to achieve high-quality GAN model training under limited resource conditions.
- Taking into account the unique model size constraints faced by each vehicle, we innovatively propose the Heterogeneous-Vehicle-Scheduling-GAN (HVS-GAN) framework. This framework ingeniously incorporates heterogeneous model training methods, enabling different types of ICVs to run GAN models of varying sizes that match their own resources locally. Meanwhile, HVS-GAN ensures that these heterogeneous vehicle models can collaborate and contribute to the optimization of the global GAN model.
- In HVS-GAN, we tailored a DDPG agent specifically to learn about the resource conditions and generation quality of each ICVs, and to optimize the scheduling of vehicle participation in aggregation, thus addressing the optimization problem outlined above.
- Rigorous experiments have proven the superiority of our proposed HVS-GAN solution, which outperforms state-of-the-art algorithms in scenarios with stringent model size constraints, achieving better generated quality and lower communication cost.

The remainder of this article is organized as follows. The related work is summarized in Section II. Some preliminaries and the problem are described in Section III. Section V elaborates the proposed HVS-GAN framework to solve the problem. In Section VI, simulation results are presented and discussed. Finally, Section VII concludes this article.

II. Related Work

GAN: Goodfellow [13] et al. first introduce the concept of GAN, employing two neural networks in a competitive game-like framework. They taught these networks to learn the distribution of datasets, thereby enabling generative tasks. GANs have since garnered widespread adoption across diverse domains, including Anomaly Detection [14], the Vehicle Trajectory Prediction [15], the Healthcare System [16] and more, owing to their remarkable ability to generate data. Nevertheless, none of above algorithms have taken into account how to handle the heterogeneity of ICV in the IoV scenario.

FL-Based GAN: To address the above challenges, previous researchers began to explore how to deploy GANs in distributed scenarios. FL [17] was proposed for distributed general CNN model training without exchanging user data. With the development of FL, distributed GAN training algorithms leveraging this paradigm emerged. FedGAN [18] first extend FL to GAN, which employs two distinct learning rate for separate component of GAN network, and periodically communicate with the server. The parameters of GANs are notably larger than conventional models, typically encompassing both a generator and a discriminator, which means increase in energy consumption and computational demands during training process. Recently, CAP-GAN [7] proposed a method of separately training the discriminator on UEs and the generator on servers to achieve reduced overhead through separate deployment , while IFL-GAN [19] proposed maximum mean discrepancy to accelerate the convergence speed in heterogeneous data scenario. However, both of them still cannot support the heterogeneity of ICVs in the IoV network, particularly for ICVs with low computational capabilities, which are unable to deploy GAN models of normal size.

Heterogeneous Model Training FL: In IoV, different ICVs under the same network have different computing capabilities [20]. Therefore, it is difficult to ensure that all ICVs can run under the original model parameters when deploying FL in Iot. This problem is called Heterogeneous model training FL. Previous researchers have proposed a variety of works to solve this problem. FedProx [8] allows each client to train in different number of iterations based on local resources, but it presupposes that all clients train the identical model. While this approach somewhat mitigates the heterogeneity issue, it does not fundamentally address the mismatch between model architectures and device computing capabilities. HeteroFL [9] introduced a novel approach that involves partitioning the global model horizontally, maintaining the entire depth of the DNN architecture on each client. This strategy adjusts the width partitioning ratio specifically to accommodate the varying capabilities of heterogeneous clients, thereby offering a tailored solution to the challenges posed by diverse client environments. Recently, PerFedMask [21] proposed utilizing an optimized masking vector for sub-models in ICV segmentation. Moreover, the architectures of the above-mentioned

TABLE I
HOW THE WORK IN THE EXISTING LITERATURE HANDLES HETEROGENEOUS MODEL CLIENTS UNDER RESOURCE CONSTRAINTS ON TRAINING GAN

Design Choice	FL in IoV [10]–[12]	FL-Based GAN [7], [24]	Heterogeneous Model Training FL [9]	HVS-GAN
Model Partitions	Entire model on each Vehicle	Entire model on each client or set global generator on server	Partial parameters of model	**Partial parameters of discriminator and generator**
Client Scheduling	Scheduled according to resource	Random select	Random select	**Scheduled by DDPG according to resource**
Training GAN	No consideration	Consideration	No consideration	**Consideration**

algorithms are all targeted at CNN models, and have not been extended to the training of GANs.

IoV with FL: In IoV networks, where there are a large number of ICVs with a high number of parameters, selecting suitable ICVs with resources for aggregation becomes a challenging problem in FL. Zhang [10], Zarandi and Tabassum [11], Chen [12] et al. have studied the issue of energy consumption and formulated an optimization problem that takes into account factors such as computation delay, communication delay, bandwidth, etc. They also introduced RL methods as a scheduler to assist in client selection to solve this problem. Zhang et al. [22] focused on the context of the Industrial IoT and modeled the costs of training time and training quality. The total cost is obtained by combining these two costs, and the agent minimizes this cost by selecting nodes in each round. In [23], Bourbon and Nagellen issued a technical report on the integration of RL with FL. They designed and implemented a DDPG algorithm named SchedulerFL for training participants in FL settings. However, none of the aforementioned existing methods have considered the distributed training of GANs in IoV.

To highlight the distinctiveness of HVS-GAN, Table I succinctly compares key aspects like model partitioning, client scheduling, and GAN training with other prevalent methods.

III. PRELIMINARIES

A. FL with GAN

The GAN architecture comprises a discriminator $\mathcal{D}$ and a generator $\mathcal{G}$, engaging in a competitive game facilitated by neural networks. Their objectives are diametrically opposed: $\mathcal{G}$ strives to convert random noise z into data resembling the real distribution, while $\mathcal{D}$ endeavors to distinguish genuine from synthetic inputs. This adversarial interplay is governed by the objective function $V(\mathcal{G},\mathcal{D})$.

$$\begin{aligned}\min_{\mathcal{G}}\max_{\mathcal{D}} V(\mathcal{G},\mathcal{D}) = E_{x\in d_r}[log(\mathcal{D}_{\theta^{\mathcal{D}}}(x))] \\ +E_{x\in d_g}[log(1-\mathcal{D}_{\theta^{\mathcal{D}}}(G_{\theta^{\mathcal{G}}}(z)))]\end{aligned} \tag{1}$$

Here, z represents the noise input to the generator, while $\theta^{\mathcal{G}}$ and $\theta^{\mathcal{D}}$ denote the respective parameters of the generator and discriminator.

To facilitate the algorithm's applicability across diverse GAN models, we standardize the objective function in a universal format:

$$\min_{\mathcal{G}}\max_{\mathcal{D}} V(\mathcal{G},\mathcal{D}) = E_{x\in d_r}[\mathcal{L}(\mathcal{D}(x))] + E_{x\in d_g}[\mathcal{L}(1-\mathcal{D}(x))] \tag{2}$$

Here, x represents a sample drawn from a mixed dataset comprising both generated d_g and raw d_r distribution. In this context, raw distribution d_r represents the real data distribution that the GAN model needs to learn. On the other hand, generated distribution d_g is used to generate high-quality data from noisy data. The objective function employs a concave, increasing function $\mathcal{L}(\cdot)$, which can be tailored to yield various GAN training variants. For instance, adopting $\mathcal{L}(t) = ||\cdot||^2$. In our approach, we set $\mathcal{L}(t) = log(t)$ as depicted in equation (1), offering a unique perspective on GAN training.

FL encompasses a vast network of UEs $\mathcal{U}$, often numbering in the hundreds to thousands, where each UE trains a DNN model locally and contributes to a global update through a centralized server-orchestrated weight aggregation process. The FL try to find a global DNN model θ^* to optimize the objective function $F(\theta^*) = \Sigma_{u\in\mathcal{U}}\frac{n_u}{n}f_u(\theta)$, where $d_u = \frac{n_u}{n}$ denotes the UE's local sample size, and $f_u(\theta) = \frac{1}{n_u}\Sigma_{i\in d_u}\mathcal{L}_i(\theta)$ is the local objective function of the UE u. $\mathcal{L}_i(\theta)$ represents the training loss incurred by sample i and d_u signifies the local dataset maintained by UE u. With the training process about GAN in equation (1), The server aggregation phase can be concisely described as:

$$\begin{aligned}F(\theta^{\mathcal{G}}) = \Sigma_{u\in\mathcal{U}}\frac{n_u}{n}f_u(\theta^{\mathcal{G}}) \\ F(\theta^{\mathcal{D}}) = \Sigma_{u\in\mathcal{U}}\frac{n_u}{n}f_u(\theta^{\mathcal{D}})\end{aligned} \tag{3}$$

B. Reinforce Learning with Deep Deterministic Policy Gradient

RL is a goal-oriented learning method. In RL, an agent interacts with an environment. At each time-step, the agent observes its environment and selects an action s_t. Then the agent receives a reward r_{t+1} depending of its action on the environment. The environment changes over the time with state s_{t+1}. Generally speaking, the agent behaves like a decision maker which learns through a policy $\pi(s)$. Its goal is to optimize a long-term reward by interacting with the environment.

DDPG [23] is a general deep RL algorithm used to address problems with continuous action spaces [25], [26]. DDPG establishes a Q function as critic and a policy function as actor simultaneously. The action-value function $Q(s,a|\theta^Q)$, where θ^Q denotes for the critic network parameters, is learned by using Bellman equation[27]. Hence, the action-value function under any policy π can be defined as:

$$Q^{\pi}(s_t,a_t) = E[R(s_t,a_t)+\gamma Q^{\pi}(s_{t+1},\pi(s_{t+1}))] \tag{4}$$

And The agent's objective is to acquire the optimal policy $\pi^*(s)$:

$$\pi^*(s) = \max_a Q^*(s,a) \tag{5}$$

where $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ represents the optimal action-value function, the agent strives to attain.

IV. Problem Statement

We consider a system model of the FL consists of intelligent vehicles [28] which consists of one server and a set ICVs $\mathcal{U}$ in the road network, and each ICV have onboard sensors to collect data from surrounding environment.

Initially, each ICV develops its unique local models, denoted as $\mathcal{D}_u$ for discriminator and $\mathcal{G}_u$ for generator, tailored specifically to its individual local dataset d_u. The process of executing these computations locally entails both computational energy expenditure and latency, which can be mathematically formulated as:

$$E_u^{cmp} = \alpha_u C_u f_u^2 \tag{6}$$

$$T_u^{cmp} = \frac{C_u}{f_u} \tag{7}$$

Here, α_u is the computational efficiency factor, C_u quantifies the cumulative CPU cycles necessary for ICV u to process its local dataset, given by $C_u = \tau_u d_u$, where τ_u signifies the computational overhead for processing a single data; f_u represents the local processing capability of device u.

Since the local generator model $\mathcal{G}_u$ and discriminator model $\mathcal{D}_u$ training of ICV u completed, the updated local parameters $\theta_u^{\mathcal{G}}$ and $\theta_u^{\mathcal{D}}$ from ICVs participating in this round will be sent to the server. The latency and consumption incurred by the u in transmitting data to the server over a designated uplink resource block (RB) are specified as follows:

$$T_u^{up} = \frac{b(\theta_u^{\mathcal{G}}) + b(\theta_u^{\mathcal{D}})}{r_u} \tag{8}$$

$$E_u^{up} = \eta_u T_u^{up} \tag{9}$$

where $b(\theta_u^{\mathcal{G}})$ and $b(\theta_u^{\mathcal{D}})$ means the size of model parameters in bits, consisting of the model parameters of the discriminator and the generator, respectively. r_u denotes the transmission rate of device u, η_u denotes the transmission power of device u, β_u denotes peak transmission speed achievable from ICV to the serve, which is calculated by the following formula:

$$\beta_u = w_u log_2\left(1 + \frac{p_u g_u}{N_0 w_u}\right) \tag{10}$$

where w_u(in Hz) denotes the bandwidth of an allocated RB(in Hz) between the server and the u-th ICV; N_0 is the additive white Gaussian noise power at the server; g_u denotes the channel power gain from ICV u to the server, which is given by

$$g_u = L_u |h_u|^2 \tag{11}$$

where L_u signifies the resultant effect of the distance-based path loss combined with log-normal shadowing variations, h_u signifies the Rayleigh fading factor, adhering to the distribution specified by $\mathbb{CN}(0,1)$.

As the server aggregate the localized $\mathcal{G}_u$ and $\mathcal{D}_u$ models from ICVs, the processing latency and resource utilization at the server can be expressed as:

$$T_s^{cmp} = \frac{C_s}{f_s} \tag{12}$$

$$E_s^{cmp} = \alpha_s C_s f_s^2 \tag{13}$$

C_s signifies the aggregate CPU cycles needed by the server to process the associated data; f_s is the server processing capability, α_s is the effective capacitance coefficient of computation. After the server has aggregated the Generator $\mathcal{G}$ and global $\mathcal{D}$ based on updates from participated ICVs, the updated parameters of two models $\theta_u^{\mathcal{G}}$ and $\theta_u^{\mathcal{D}}$ are sent to all ICVs $u \in \mathcal{U}$ across the designated downlink connection communication resource blocks via Device-to-Device links. Hence, the latency and consumptionfor data transmission from the server to the ICV u can be formulated as:

$$T_u^{down} = \frac{b(\theta_u^{\mathcal{G}}) + b(\theta_u^{\mathcal{D}})}{r_u} \tag{14}$$

$$E_u^{down} = \eta_u T_u^{down} \tag{15}$$

The overall system delay per cycle comprises the ICV's model training latency, upload delay, download delay, and the server's model aggregation latency. As follows:

$$\hat{T} = \max_{u\in\mathcal{U}} T_u^{up} + T(\max\{\max_{u\in\mathcal{U}} T_u^{cmp}, T_s^{cmp}\} + \max_{u\in K} T_u^{down}) \tag{16}$$

where T is the total rounds of aggregation. Furthermore, the respective peak energy consumption is specified as:

$$E = \max_{u\in\mathcal{U}} E_u^{up} + T(\max_{u\in\mathcal{U}} E_u^{cmp} + E_s^{cmp} + \max_{u\in\mathcal{U}} E_u^{down}) \tag{17}$$

Therefore, the overall system cost is defined as the weighted combination of energy consumption and latency, expressed as follows:

$$Cost = \lambda E + (1-\lambda)\hat{T} \tag{18}$$

where $\lambda \in [0,1]$ represents the balancing factor between energy consumption and latency costs. When the entire system possesses limited power and places greater emphasis on energy consumption, the balancing factor can be assigned as $\lambda = 1$. Conversely, if the system is engaged in real-time or interactive tasks that prioritize processing latency, the balancing factor should be set to $\lambda = 0$.

In this paper, we consider a more realistic scenario of heterogeneity, where each ICV differs in terms of computational power, processing speed, and network communication bandwidth. Furthermore, given the large number of ICVs, the server needs to select appropriate ICVs for training the GAN. Our goal is to complete the training in the shortest possible time while ensuring the performance of the GAN under resources constraints. Therefore, the problem is to optimize:

$$\begin{aligned}
&\min_{\mathcal{G},A}\max_{\mathcal{D}}\left[\underbrace{\Sigma_{u\in\mathcal{U}}V_u(\mathcal{G},\mathcal{D})}_{Object1}+\underbrace{Cost}_{Object2}\right]\\
&s.t. \qquad \lambda\in[0,1],\\
&\qquad\qquad a_t^u\in[0,1]\\
&\qquad\qquad \Sigma_{u\in\mathcal{U}'}a_t^u\le N\\
&\qquad\qquad a_t^uT_u\le T_{max}\\
&\qquad\qquad \eta_{min}\le a_t^u\eta_u\le\eta_{max}\\
&\qquad\qquad f_{min}\le f_u\le f_{max}\\
&\qquad\qquad b(\theta_u^{\mathcal{G}})+b(\theta_u^{\mathcal{D}})\le b_{max}^u
\end{aligned} \tag{19}$$

where $A=[a_t^u]$ is a $T\times\mathcal{U}$ matrix for ICVs selection, with T the number of rounds and $\mathcal{U}$ the number of devices, $Cost$ represents the total utility of system.

$Object1$ describes the optimization goal of GAN performance, and $Object2$ describes the total utility cost of the scheduled ICVs. $\lambda\in[0,1]$ indicates the balancing factor of the system utility cost; a_t^u represents the limiting factor for the binary indicator of ICV selection decision; $\Sigma_{u\in\mathcal{U}'}a_t^u$ ensures that the number of ICVs selected does not exceed N in each round, where $\mathcal{U}'$ is the set of selected ICV each round; $a_t^uT_u\le T_{max}$ indicates that the processing latency for each ICV should not exceed a predefined maximum threshold; $\Sigma_{u\in\mathcal{U}'}w_u\le W_{max}$ represents the cumulative bandwidth utilization of all chosen ICVs remains within W_max; $\eta_{min}\le a_t^u\eta_u\le\eta_{max}$ imposes a limit on the transmission power level for each individual ICV; $f_{min}\le f_s\le f_{max}$ indicates that the allocated computational resources for local processing of each ICV must not surpass the respective available computational capacity; $b(\theta_u^{\mathcal{G}})+b(\theta_u^{\mathcal{D}})\le b_{max}^u$ indicates the model parameter size constraint for each ICV.

Directly solving problem (19) poses significant challenges. Firstly, aggregating GAN models with different parameter sizes trained on different ICVs into a global model is a difficult task. Although various methods have been proposed to aggregate heterogeneous models [8], [9], there is no works deployed specifically for GAN training. Furthermore, ICVs scheduling decisions also have a profound impact on the performance of the current model, which in turn influences subsequent resource allocation and client scheduling decisions, creating a complex interdependence that further complicates the problem.

V. Framework

To address the aforementioned issues, we propose the HVS-GAN framework, a method for federated training of GAN models in IoT scenarios with resource constraints. In the following chapters, We first introduce the main progress of HVS-GAN in chapter V-A. Then we introduce heterogeneous model aggregation techniques in chapter V-B , which enabling different devices to train GAN models of varying sizes and contribute to the global model on the server. Finally, we introduce ICVs selection based DDPG in chapter V-C, to tackle the problem of ICVs selection under resource constraints.

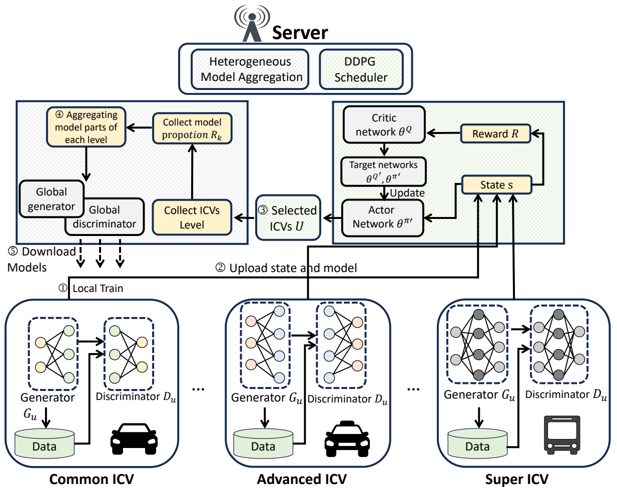


Fig. 1. HVS-GAN Overview. The workflow includes the following steps: ① ICVs perform local train on heterogeneous models; ② ICVs upload state and generator and discriminator model; ③ Server selects ICVs according to the state; ④ Server aggregates the models of different sizes; ⑤ ICVs download generator and discriminator;

A. HVS-GAN Framework

To resolving the $object1$ in problem (19), HVS-GAN allows heterogeneous ICVs to participate in training and contribute to the global model. As shown in Fig.1 and Algorithm 1, HVS-GAN follows the following steps:

Firstly, the server employs the DDPG module to make decisions and select several ICVs for training. The server get the selected ICVs a_t, send heterogeneous model to them. The selected ICVs receive heterogeneous models that are matched to their local devices for training. Then ICVs train the $\mathcal{G}$ and $\mathcal{D}$, the formulation as follows:

$$\mathcal{D}_k^{t+1}=\mathcal{D}_k^t+O(\nabla_{\mathcal{D}_k}\mathcal{L}_{\mathcal{D}}) \tag{20}$$

$$\mathcal{G}_k^{t+1}=\mathcal{G}_k^t+O(\nabla_{\mathcal{G}_k}\mathcal{L}_{\mathcal{G}}) \tag{21}$$

where $O(\cdot)$ is an optimization function. $\mathcal{L}_{\mathcal{D}}$ and $\mathcal{L}_{\mathcal{G}}$ is the loss function of discriminator $\mathcal{D}$ and generator $\mathcal{G}$.

Once the training is complete, the ICVs send model updates to the server. The server accepts updates from the ICVs and collects states S_t. Then,the server store experiences (S_t, S_{t+1}, a_t, r_t) to memory and update the actor network θ^Q and critic network θ^π. Next, the server performs heterogeneous model aggregation to get next round model $\mathcal{G}^{t+1}$ and $\mathcal{D}^{t+1}$ and uses the DDPG module to decide which ICVs participate as a_t in the next round of training. The server generates sub-models of generator $\theta_k^{\mathcal{G}}$ and $\theta_k^{\mathcal{D}}$ based on the limited resources of the selected ICVs and sends them to the ICVs. After completing these steps, the next round of training begins.

The framework primarily consists of heterogeneous model training and ICVs selection based DDPG . We will now talk about their details.

B. Heterogeneity Aggregation

Inspired by HeteroFL [9], we assign subsets of the global generator $\theta^{\mathcal{G}}$ and discriminator $\theta^{\mathcal{D}}$ to each ICV in HVS-GAN that satisfy their resource constraints, and aggregate their

Algorithm 1 HVS-GAN

```
Server:
Require: ICVs set $\mathcal{U}$, global GAN parameters $\theta^{\mathcal{D}}, \theta^{\mathcal{G}}$.
 1: for all communication round $t = 0, 1, ...T$ do
 2:     Select ICVs $\mathcal{U}'$ by Scheduler;
 3:     for all ICVs $u$ in $a_t$ do
 4:         Count model proportion $\mathcal{R}_u = b_u^{max}/2(b(\theta^{\mathcal{D}}) + b(\theta^{\mathcal{G}}))$;
 5:         Generate sub-models $(\theta_u^{\mathcal{D}}, \theta_u^{\mathcal{G}})$ based on $\mathcal{R}_u$;
 6:         Send $(\theta_u^{\mathcal{D}}, \theta_u^{\mathcal{G}})$ to selected ICVs;
 7:         Receive updated $(\theta_u^{\mathcal{D}}, \theta_u^{\mathcal{G}})$ from selected ICVs;
 8:     end for
 9:     for all layers $\mathcal{W}$ in $\theta^{\mathcal{D}}$ and $\theta^{\mathcal{G}}$ do
10:         Update each $\mathcal{W}$ according equation(29);
11:     end for
12:     Collect statement $s_t$;
13:     Update Scheduler acccording euqation(34) and equation(35);
14: end for
15: return model parameters $\theta^{\mathcal{D}*}, \theta^{\mathcal{G}*}$.
ICV:
Require: Sub-models $\theta_u^{\mathcal{D}}, \theta_u^{\mathcal{G}}$ from server.
 1: for all local round do
 2:     Update local discriminator $\mathcal{D}_u^{t+1} = \mathcal{D}_u^t + O(\nabla_{\mathcal{D}_u}\mathcal{L}_{\mathcal{D}})$;
 3:     Update local generator $\mathcal{G}_u^{t+1} = \mathcal{G}_u^t + O(\nabla_{\mathcal{G}_u}\mathcal{L}_{\mathcal{G}})$;
 4: end for
 5: return Updated sub-model parameters $(\theta_u^{\mathcal{D}}, \theta_u^{\mathcal{G}})$.
```

models at the server to solve problem(19). The details of this process are as follows:

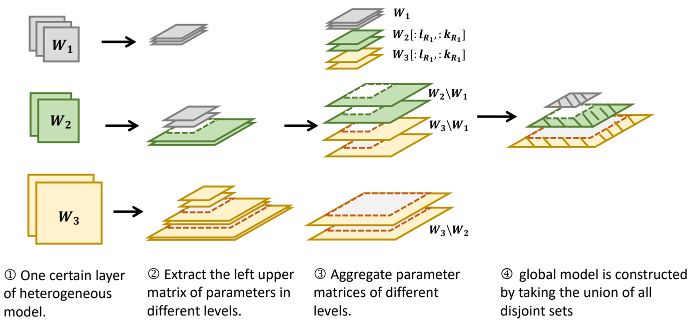
Once the server completes the aggregation of the global discriminator $\theta^{\mathcal{D}}$ and generator $\theta^{\mathcal{G}}$ for a certain round, the server utilizes a DDPG module to select the ICVs that will participate in the next round of training. For a specific ICV u, the server calculates its model proportion $\mathcal{R}_u$ based on its constraints, using a formula that is:

$$\mathcal{R}_u = \frac{b_u^{max}/2}{b(\theta^{\mathcal{D}}) + b(\theta^{\mathcal{G}})} \tag{22}$$

Here, $b(\theta^{\mathcal{D}})$ and $b(\theta^{\mathcal{G}})$ represent the model sizes of the global discriminator and generator in bits, respectively. b_u^{max} is the maximum model size that u can support in bits, and the division by two is due to the fact that the ICV needs to train both the generator and the discriminator models, and we assume that the sizes of these two models are similar. Additionally, we assume that each ICV will select a sub-model within its maximum tolerable model size b_u^{max}.

Taking the $\theta^{\mathcal{G}}$ as an example, let's assume that its certain hidden layer $\mathcal{W} \in R^{l_g \times k_g}$ within it has a dimension of l_g and k_g, where l_g and k_g represent the output and input channel size of this layer. By multiplying l_g and k_g with the model proportion $\mathcal{R}_u$, we can obtain the input channel l_u andd output channel k_u that are suitable for the ICV u.

$$\begin{aligned} l_u &= l_g \times \mathcal{R}_u \\ k_u &= k_g \times \mathcal{R}_u \end{aligned} \tag{23}$$



W_1
$W_2[:l_{R_1}, :k_{R_1}]$
$W_3[:l_{R_1}, :k_{R_1}]$
W_2
$W_2\backslash W_1$
$W_3\backslash W_1$
W_3
$W_3\backslash W_2$
① One certain layer of heterogeneous model.
② Extract the left upper matrix of parameters in different levels.
③ Aggregate parameter matrices of different levels.
④ global model is constructed by taking the union of all disjoint sets

Fig. 2. The process of Heterogeneity Aggregation, with $\mathcal{W}_1$(gray), $\mathcal{W}_2$(green), and $\mathcal{W}_3$(yellow) blocks representing a certain layer of models in different sizes.

After obtaining the ICV's input and output channel numbers, we can take the upper left submatrix of $\mathcal{W}$ with a size of $l_u \times k_u$ as the subset of this layer:

$$\mathcal{W}_u = \mathcal{W}[: l_u : k_u] \tag{24}$$

After performing such operations on each layer of the generator parameter $\theta^{\mathcal{G}}$ and discriminator parameter $\theta^{\mathcal{D}}$, the server can obtain a subset of the global generator $\theta_u^{\mathcal{G}}$ and a subset of the discriminator $\theta_u^{\mathcal{D}}$, then send these two models to the ICV for local training.

Once the training is completed by the ICV, the server collect all the updates for aggregation. First the server sorts the $\mathcal{R}_u$ of all ICVs, then get a list of model proportion $\{\mathcal{R}_u, ..., \mathcal{R}_{u'}\}$. ICVs with the same $\mathcal{R}_u$ value can be classified into one category, and we assume there are three different values $\{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3\}$. We assign a class level $\{c_1, c_2, c_3\}$ to each of these $\mathcal{R}$ values.

We exemplify the process using Fig.2. The equation (25) shows that the smallest level of model parameter(gray) c_1 is aggregated from all ICVs that contain it.

$$\mathcal{W}^{c_1} = \frac{1}{|\mathcal{U}'|}\Sigma_{u \in \mathcal{U}'} W_u^{c_1} \tag{25}$$

where $\mathcal{W}^{c_1}$ denotes the smallest subset of model parameters that corresponds to the level c_1.

Next, we discuss the aggregation of the next part, as shown in the equation(26) and the green blocks in figure. The set difference between part $\mathcal{W}^{c_2}$(green) and $\mathcal{W}^{c_1}$(gray) of model parameters is aggregated from ICVs having $\mathcal{W}^{c_2}$.

$$\mathcal{W}^{c_2} \backslash \mathcal{W}^{c_1} = \frac{1}{|\mathcal{U}'| - |\mathcal{U}'_{c_1}|}\Sigma_{u \in (\mathcal{U}' - \mathcal{U}'_{c_1})} W_u^{c_2} \backslash W_u^{c_1} \tag{26}$$

where, $\mathcal{W}^{c_2} \backslash \mathcal{W}^{c_1}$ denotes the set of elements including $\mathcal{W}^{c_2}$ but exclude $\mathcal{W}^{c_1}$; $\mathcal{U}' - \mathcal{U}'_{c_1}$ denotes among the selected ICVs but excluding the ICVs only with c_1 subset.

We aggregate the $\mathcal{W}^{c_3}$ in similar way as shown in yellow block and equation(27).

$$\mathcal{W}^{c_3} \backslash \mathcal{W}^{c_2} = \frac{1}{|\mathcal{U}'| - |\mathcal{U}'_{c_2}| - |\mathcal{U}'_{c_1}|}\Sigma_{u \in (\mathcal{U}' - \mathcal{U}'_{c_2} - \mathcal{U}'_{c_1})} W_u^{c_3} \backslash W_u^{c_2} \tag{27}$$

Therefore, We can generalize the formula, for c_n level, we can aggregate the corresponding parts in:

$$\mathcal{W}^{c_n}\backslash\mathcal{W}^{c_{n-1}}=\frac{\Sigma_{u\in(\mathcal{U}'-\mathcal{U}'_{c_{n-1}}-\cdots-\mathcal{U}'_{c_1})}W_u^{c_n}\backslash W_u^{c_{n-1}}}{|\mathcal{U}'|-|\mathcal{U}'_{c_{n-1}}|-\cdots-|\mathcal{U}'_{c_1}|} \tag{28}$$

Once all the sub-models of all levels are aggregated, the global model parameters $\mathcal{W}$ are constructed from the union of all disjoint sets of the partition.

$$\mathcal{W}=\mathcal{W}^{c_1}\cup\mathcal{W}^{c_2}\backslash\mathcal{W}^{c_1}\cup\cdots\cup\mathcal{W}^{c_n}\backslash\mathcal{W}^{c_{n-1}} \tag{29}$$

Descrption above applies to a specific layer within the global model and each layer of the model is updated and aggregated in the same way. In a practical framework, we perform the same operations for every layer of both the global discriminator $\theta^{\mathcal{D}}$ and the global generator $\theta^{\mathcal{G}}$.

C. DDPG-Based ICVs Selection

After resolving the training and deployment issues of GAN models under restrictions on model parameters, we aim to solve the $Object2$ of problem(19) through the DDPG module, with the reducing costs while maintaining the performance of GAN. The $Object2$ can be defined by a 3-tuple (S, a, r), where S, a, and r are the state, action, and reward respectively.

1) State S**:** During the $t-$th edge aggregation round, the system's state $s_t\in S$ is established as

$$s_t=\{T_{u,t-1}^{down}+T_{u,t}^{up}+T_{u,t}^{cmp},E_{u,t-1}^{down}+E_{u,t}^{up}+E_{u,t}^{cmp},b_{max}^{u},\epsilon_t^u\} \tag{30}$$

where $T_{u,t-1}^{down}$ and $E_{u,t-1}^{down}$represents the download delay and consumption of the ICV in the previous round; $T_{u,t}^{up}$ and $E_{u,t}^{up}$ represents the download delay and consumption of the ICV in this round; and $T_{u,t}^{cmp}$ and $E_{u,t}^{cmp}$ represents the delay and consumption of the ICV's computation in this round; and b_{max}^{u} represents the maximum model size in bits; ϵ_t^u indicates the lastest round ICV was selected.

2) Action A**:** We design the action in the t-th aggregation round to be $a_t\in A$. Here, a_t is the binary ICVs selection decision.

3) Reward r**:** Reward serves as an assessment of the performed action. For problem(19), a straightforward design way would define the instantaneous objective function in $Object1$ and $Object2$. However, the $\Sigma_{u\in\mathcal{U}}V_u(\mathcal{G},\mathcal{D})$ can't be observed. Therefore, we use the FID index (see chapter VI-A) as an observation indicator of $\Sigma_{u\in\mathcal{U}}V_u(\mathcal{G},\mathcal{D})$. Since $Object2$ can be observed directly, we calculate the reward value through the following formula:

$$r_t=\zeta_1[FID(t)-FID(t-1)]-\zeta_2 Cost(t) \tag{31}$$

where ζ_1 and ζ_2 is the hyperparameter to adjust the tradeoff between the generative quality and cost; $FID(t)-FID(t-1)$ describes the growth of GAN model performance. When the growth becomes smaller, it indicates that the training is approaching convergence. Due to the exceptionally significant difference between the FID and the COST in practical experiments,we employ two distinct parameters, ζ_1 and ζ_2, to control the reward value. **4) ICVs selection process** : We have introduce DDPG optimal goal in the previous chapter III-B, and in this chapter, we will introduce in detail how DDPG is applied to FedHVS. As we known, DDPG stands out among various DRL methodologies for problem (19) due to its proficiency in managing continuous state and action domains. Employing the actor-critic architecture, the DDPG model utilizes an actor network to model the policy π, and a critic network to estimate the action-value function $Q^{\pi}(s,a)$. A target network and an online network, both sharing the same architecture, constitute either an actor network or a critic network. This dual-subnetwork structure not only ensures learning stability but also effectively prevents overestimation, especially in large-scale problem scenarios.

Procedure 2 Scheduler

```
Select(·)
Require: State matrix $\hat{s_t}$;
1: Selection action $A_t=\pi(\hat{s_t}|\theta^{\pi})$;
2: return $A_t$.
Update(·)
Require: Updating times $T$.
1: for all $t=0,1,...,T$ do
2:     Sample $B=(s,a,r,s')$ from memory;
3:     Compute $y(r,s')\leftarrow r+\gamma Q'(s',\pi'(s'))$;
4:     Update Actor network:
       $\nabla_{\phi}\frac{1}{|B|}\Sigma_{(s,a,r,s')\in B}(Q(s,a)-y(r,s'))^2$;
5:     Update Critic network:
       $\nabla_{\theta}\frac{1}{|B|}\Sigma_{s\in B}Q(s,\pi(s))$;
6:     Update target network:
       $\phi'\leftarrow\rho\phi'+(1-\rho)\phi,\theta'\leftarrow\rho\theta'+(1-\rho)\theta$;
7: end for
8: return None.
```

Let $\pi(s_t|\theta^{\pi})$ and $Q(s_t,a_t|\theta^{Q})$ Denote the actor and critic online networks as such, respectively; θ^{π} and θ^{Q} are the parameters of these two model. Let $\pi'(s_t|\theta^{\pi'})$ and $Q'(s_t,a_t|\theta^{Q'})$ Designate the actor and critic target networks separately; $\theta^{\pi'}$ and $\theta^{Q'}$ are also the parameter of these two model. To optimize the critic network, the subsequent loss function is minimized:

$$L=\frac{1}{M'}\Sigma_t(Y_t-Q(s_t,a_t|\theta^{Q}))^2 \tag{32}$$

where $Y_t=r_t+\gamma Q(s_{t+1},\pi(s_{t+1}|\theta^{\pi'})|\theta^{Q'})$ and M' is the size of mini-batch. We optimize the actor network along the direction of $\nabla_{\theta^{\pi}}J(\theta^{\pi})\approx\frac{1}{M'}\Sigma_{a_t}Q(s_t,a_t|\theta^{Q})\nabla_{\theta^{\pi}}\pi(s_t|\theta^{\pi})$ in order to maximize the policy objective function outlined below:

$$J(\theta^{\pi})=E_{\theta^{\pi}}[Q(s_t,\pi(s_t|\theta^{\pi})|\theta^{Q})] \tag{33}$$

where $\nabla_{\theta^{\pi}}$ denotes the gradient with respect to θ^{π}. Subsequently, the actor and critic target networks undergo a soft update process as follows:

$$\theta^{\pi'}\leftarrow\phi\theta^{\pi}+(1-\phi)\theta^{\pi'} \tag{34}$$

$$\theta^{Q'}\leftarrow\phi\theta^{Q}+(1-\phi)\theta^{Q'} \tag{35}$$

where ϕ serves as a hyperparameter regulating the pace of learning.

The DDPG component integrated into HVS-GAN is outlined in the algorithm 2. During each iteration of edge aggregation, the agent gathers pertinent state information s_t from the surrounding environment. The ICVs contribute their localized observations alongside the trained models for central processing. The agent make decisions and select next round participated ICVs. The transmission latency and energy usage associated with state and action data are comparatively insignificant due to their significantly smaller size compared to model data. The actor online network outpus action a_t. Upon the actor online network outputting action a_t, we refine problem(19) to evaluate reward. At the end of the round t, the agent receives reward r_t and transitions to a new state s_{t+1}.

An experience replay buffer is incorporated into the agent to preserve the experience $\{s_t, a_t, r_t, s_{t+1}\}$ in each round. The learning procedure initiates once the experience buffer is sufficiently populated. Specifically, a mini-batch of M' experiences is randomly drawn from the replay buffer to facilitate the training of the DDPG network. Subsequently, the critic and actor online networks undergo updates by minimizing the loss function defined in equation(32) and maximizing the policy objective outlined in equation(33), respectively, followed by the update of their target networks according to equation(34) and equation(35). The model achieves convergence over multiple episodes by mitigating correlations among observations and exploring diverse environmental states.

VI. Experiments

To validate the effectiveness of the HVS-GAN framework, we conducted performance tests on the GAN model under the framework, as well as communication performance tests.

A. Experiment Configuration

GAN Model. Following the training configurations of the baseline models, we configured the GAN network with a generator comprising 5 layers of Deconvolutional layers and 5 layers of BatchNorm layers. The discriminator consists of 5 layers of Convolutional layers and 3 layers of BatchNorm layers.

DDPG Model. We defined two network structures in the DDPG algorithm: the actor network and the critic network. Both networks consist of three linear layers with 128 neurons each. The final layer of the actor network outputs a vector of length equal to the total number of ICVs, while the final layer of the critic network outputs a single value.

Experimental Environment. In the experiment, all devices and servers were simulated on a workstation. The workstation is equipped with a 2.4GHz Intel Core i9-12900 processor and a NVIDIA RTX A5000 Graphics card. Additionally, the workstation has 64 GB of memory.

Dataset. We trained the model on the CIFAR10 dataset. The CIFAR10 dataset comprises 60,000 32x32 color images distributed across 10 classes, with 6,000 images per class. It includes 50,000 training images and 10,000 test images. To simulate data heterogeneity in a real scenario, we applied the Dirichlet distribution [29] with $\alpha = 100$ and selected the top 7 categories with the least number of occurrences per device to simulates a scenario of label skewness.

Test Metrics. The Inception Score [30] (IS) and Frechet Inception Distance [31] (FID) are metrics used to evaluate the quality of images generated by generative models. IS is calculated based on the output of a single, pre-trained Inceptionv3 image classification model, while FID compares the distribution of generated images with that of a set of real images. We use both of them to evaluate the performance of global GAN model in all algorithm.

Simulation Settings. We defined three types of ICVs with varying levels: super, advanced, and common, each have different model size constraints and bandwidth constraints correspondingly. Super ICV has a model parameter limit of $27.28MB$ and a bandwidth limit of $9dBm/Hz$. Advance ICV has a model parameter limit of $19.10MB$ and a bandwidth limit of $7dBm/Hz$. Common ICV has a model parameter limit of $13.64MB$ and a bandwidth limit of $5dBm/Hz$. We assume that all ICVs are uniformly distributed within the coverage area of the server.

For other detailed parameters, each ICV is designed to process one sample data using 10^3 CPU cycles per bit, operating at a processing capability of $3GHz$. The additive white Gaussian noise power is set to $-174dBm/Hz$. The batch size for each ICV is 100, with a total of 100 ICVs involved in the system. The communication iteration is set to 20000, while each ICV performs a single local iteration $L = 1$. Tradeoff hyperparameters ζ_1 and ζ_2 are tuned to 0.65 and 0.02 respectively, and the cost balancing factor λ is set to 0.5. For the GAN training, a learning rate of 0.0002 is used, with the global GAN parameters occupying a total of $27.28MB$. For the DDPG, an experience replay buffer of size 4000 is employed, and separate learning rates of 0.0001 and 0.0002 are assigned for the Actor and Critic networks, respectively.

B. Experiment Results

1) Compared Experiments

Compared method. We choose three different baselines to compare the performance of HVS-GAN and all the compared methods adopt the heterogeneous model aggregation method of LG-FedAvg [32] for comparison:

- CAP-GAN [7]: CAP-GAN is the current SOTA method for FL of GANs, which focuses on enhancing its capabilities under the none independent and identically distributed(Non-IID) data. It proposes to divide the generator into shared layers and personalized layers.
- FL-GAN [24]: FL-GAN, a synthetic baseline of GAN algorithms rooted in FL, comprises a cloud server and a pool of devices, each equipped with a full model. Each ICV submits its locally trained model to the cloud server after completing 10 iterations.
- FeGAN [6]: FeGAN is a GAN based on FL that addresses the Non-IID problem by customizing the weights and lists of all participating ICVs in each round.

Experiment Scenario. We tested the algorithm in three different communication and device heterogeneity scenarios. We deliberately increased the proportion of common ICVs in more challenging scenarios, allowing us to simulate the

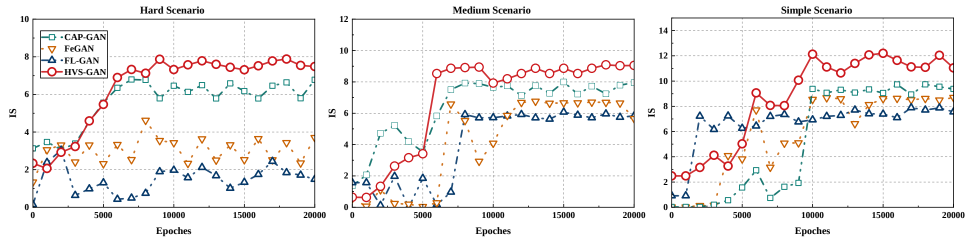


Fig. 3. The performance of four algorithms in terms of Inception Score metrics under different scenarios.

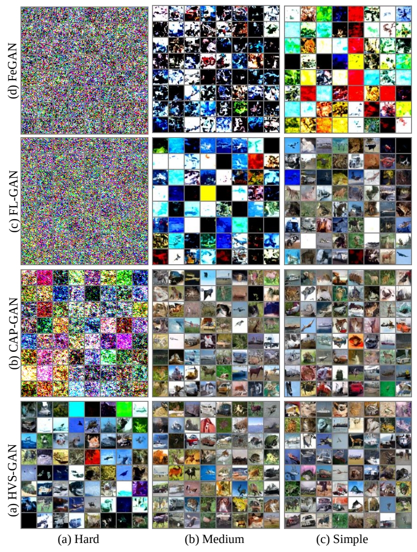


Fig. 4. The sample images are generated under three different scenario on CIFAR10.

impact of device heterogeneity and communication delays on the performance of different algorithms.

- Simple Scenario: we configure 80% of ICVs as super vehicles, 10% as advance vehicles, and 10% as common vehicles to simulate scenario of collaborative IoV for weak signal in tunnel.
- Medium Scenario: we configure 50% of ICVs as super vehicles, 30% as advance vehicles, and 20% as common vehicles to simulate scenario of delivery IoV networking.
- Hard Scenario: we configure 20% of ICVs as super vehicles, 50% as advance vehicles, and 30% as common vehicles to simulate scenario of public transport dispatching.

Results Analysis. The performance differences of FeGAN, FL-GAN and CAP-GAN are compared in Fig.4 and Fig.3. All results were used to generate images with the same Gaussian noise after 20,000 rounds of communication and calculation. To ensure fairness, we prioritized selecting ICVs with lower communication overheads among the three comparison methods, and with the same number of samples scheduled in each round as that of HVS-GAN.

Fig.4 show the image result of four algorithm. In the Simple scenario, due to its slow convergence speed, FeGAN still produces images with considerable noise, while the other three algorithms perform almost equally. When it comes to the Medium scenario, CAP-GAN and FedHVS exhibit high performance, while FL-GAN and FeGAN start to generate data with more noise. In the Hard scenario, CAP-GAN begins to produce more noise, and FL-GAN and FeGAN are unable to generate data with distinguishing features. However, FedHVS stands out in the Hard scenario, thanks to its scheduling method based on device communication and performance and its retention of model updates from heterogeneous ICVs. As a result, the data generated by HVS-GAN outperforms other methods in terms of FID metrics and IS. This is because in IoV scenarios, when only communication conditions are considered, ICVs with slower communication speeds participate less frequently, leading to inconsistent training paces among ICVs, which can cause deviations in their local discriminators and generators, potentially resulting in overfitting and making it difficult for the generator to learn the global and accurate distribution characteristics of the data.

On the other hand, HVS-GAN's superior performance is also attributed to heterogeneous model aggregation, which preserves the characteristics of different models, enabling the global model to learn the global data distribution. Although the other compared three methods also adopt the LG-FedAvg approach to handle heterogeneous characteristics, the results show that their generators in the Hard scenario have not yet developed the ability to generate images, which may require longer training time to converge. Although GAN is used in our method, this conclusion is mutually verified with HeteroFL [9].

2) *Selection Experiment*

No existing works has considered the resource constraints of ICVs scheduling with GAN training. Therefore, we maintain the same hyperparameters as the comparison experiment. We will observe the changes in the reward value in hard scenarios; the performance of DDPG by replacing the DDPG module in HVS-GAN with different baselines; the numbers of different ICVs and value of FID under different ζ_1. The baselines consist of replacing the DDPG module in HVS-GAN with the following modules:

1) Greedy Association(GA) [33]: ICVs are assigned to servers based on the highest available bandwidth.

2) Random Selection(RS) [17]: The server randomly selects a predetermined number of ICVs for each aggregation cycle, with the selection count being adjustable.

The left top one in Fig.5 plots the reward as the number of epochs increases, showcasing its convergence behavior. We observe that during the first 200 epochs, the reward remains

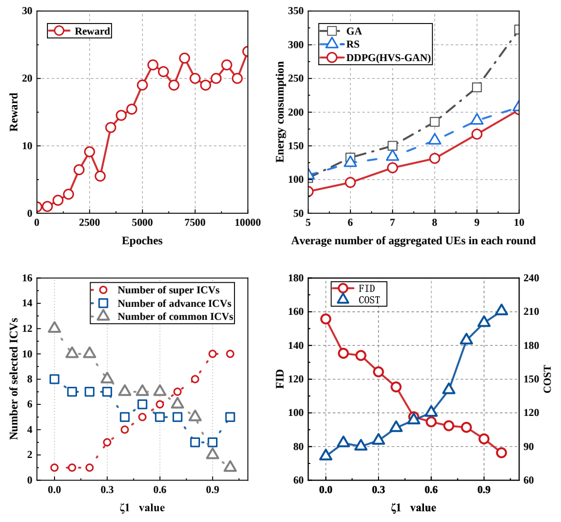


Fig. 5. Convergence of reward (left top), comparation with two baseline on cost (right top) ,ICV selection in different ζ_1 value(left bottom) and FID in different ζ_1 value(right bottom).

relatively low and stable. This is due to the fact that the DDPG parameters are initialized randomly and memory buffer is not filled. The DDPG converges to the maximum reward about 5000 epochs. In other words, it can schedule more ICVs to participate in the FL tasks through ICVs selection and association, enabling better GAN performance within a shorter cost consumption.

The right top one in Fig.5 depicts the relationship between the energy consumption and the average number of participating ICVs per round under different schemes. The results show that as long as the number of selected ICVs remains constant, the proposed DDPG algorithm consistently achieves the lowest cost in completing its tasks. Compared to the GA and RS, the improvement offered by the DDPG algorithm becomes more pronounced as the number of selected ICVs increases. For instance, when the number of selected ICVs is increased from five to ten, the extra cost required by the GA algorithm, compared to our approach, increases from 34.3% to 53%. When all ten ICVs participate in each aggregation round, the RS algorithm performs comparably to our proposed algorithm, yielding the same cost.

The bottom left in Fig.5 illustrates how different ζ_1 values affect the scheduling of DDPG. It can be seen that as ζ_1 increases, the reward tends to make DDPG pay more attention to the quality of GAN generation, so it will choose to schedule more super ICVs instead of common ICVs to participate in aggregation, while the number of advanced ICVs can maintain a relatively stable number. The bottom right also confirms this point. When the ζ_1 value increases, the FID value decreases significantly, indicating that the generation quality of GAN has been improved. However, due to the scheduling of more super ICVs for training, the cost also increases accordingly. This proves that our method can adjust the parameter ζ_1 to make DDPG adapt to the scheduling priorities of different scenarios.

VII. Conclusion

In this paper, we delve into the application of FL in GAN training within the context of Internet of Vehicles. Our proposed approach encapsulates an optimization model that not only strives to elevate the quality of generated content but also minimizes training costs. Subsequently, we introduce HVS-GAN, a tailored solution designed to refine the challenges encountered during the FL training process for GANs. Experimental findings conclusively demonstrate that HVS-GAN effectively sustains high generation quality while significantly reducing communication costs through its innovative DDPG selection mode.

References

[1] V. Mistry, B. Vaidya, and H. T. Mouftah, "Evaluation of lstm gan for trajectory prediction in connected and autonomous vehicles," in *2024 International Wireless Communications and Mobile Computing (IWCMC)*, 2024, pp. 226–231.

[2] Z. Wang, J. Zhan, C. Duan, X. Guan, P. Lu, and K. Yang, "A review of vehicle detection techniques for intelligent vehicles," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 3811–3831, 2023.

[3] R. Zhang, K. Xiong, H. Du, *et al.*, *Generative ai-enabled vehicular networks: Fundamentals, framework, and case study*, 2023. arXiv: 2304.11098 [cs.NI].

[4] H. Du, R. Zhang, D. Niyato, *et al.*, "Exploring collaborative distributed diffusion-based ai-generated content (aigc) in wireless networks," *IEEE Network*, vol. 38, no. 3, pp. 178–186, 2024.

[5] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5341–5351, 2021.

[6] R. Guerraoui, A. Guirguis, A.-M. Kermarrec, and E. L. Merrer, "Fegan: Scaling distributed gans," in *Proceedings of the 21st International Middleware Conference*, ser. Middleware '20, Delft, Netherlands: Association for Computing Machinery, 2020, 193–206, ISBN: 9781450381536.

[7] J. Zhang, L. Zhao, K. Yu, G. Min, A. Y. Al-Dubai, and A. Y. Zomaya, "A novel federated learning scheme for generative adversarial networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 3633–3649, 2024.

[8] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*, I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, Eds., mlsys.org, 2020.

[9] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," in *International Conference on Learning Representations*, 2021.

[10] Y. Zhang, J. Hu, G. Min, X. Chen, and N. Georgalas, "Joint charging scheduling and computation offloading in ev-assisted edge computing: A safe drl approach," *IEEE Transactions on Mobile Computing*, vol. 23, no. 9, pp. 8757–8772, 2024.

[11] S. Zarandi and H. Tabassum, "Federated double deep q-learning for joint delay and energy minimization in iot networks," in *IEEE International Conference on Communications Workshops, ICC Workshops 2021, Montreal, QC, Canada, June 14-23, 2021*, IEEE, 2021, pp. 1–6.

[12] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao, "Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things," *IEEE Trans. Ind. Informatics*, vol. 17, no. 7, pp. 4925–4934, 2021.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[14] C. Huang, J. Wen, Y. Xu, *et al.*, "Self-supervised attentive generative adversarial networks for video anomaly detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 11, pp. 9389–9403, 2023.

[15] L. Zhao, Y. Liu, A. Y. Al-Dubai, A. Y. Zomaya, G. Min, and A. Hawbani, "A novel generation-adversarial-network-based vehicle trajectory prediction method for intelligent vehicular networks," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 2066–2077, 2021.

[16] Z. Shen, F. Ding, A. Jolfaei, K. Yadav, S. Vashisht, and K. Yu, "Deformablegan: Generating medical images with improved integrity for healthcare cyber physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2584–2596, 2023.

[17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.

[18] M. Rasouli, T. Sun, and R. Rajagopal, *Fedgan: Federated generative adversarial networks for distributed data*, 2020. arXiv: 2006.07228 [cs.LG].

[19] W. Li, J. Chen, Z. Wang, Z. Shen, C. Ma, and X. Cui, "Ifl-gan: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 10 502–10 515, 2023.

[20] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2022.

[21] M. Setayesh, X. Li, and V. W. S. Wong, "Perfedmask: Personalized federated learning with optimized masking vectors," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[22] P. Zhang, C. Wang, C. Jiang, and Z. Han, "Deep reinforcement learning assisted federated learning algorithm for data management of iiot," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8475–8484, 2021.

[23] T. N. Benjamin BOURBON, "Federated reinforcement learning," Reporter about Reinforcement Learning in Federated Learning. 2023.

[24] Z. Ma, Y. Liu, Y. Miao, *et al.*, "Flgan: Gan-based unbiased federated learning under non-iid settings," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 4, pp. 1566–1581, 2024.

[25] J. Wang, J. Hu, J. Mills, G. Min, M. Xia, and N. Georgalas, "Federated ensemble model-based reinforcement learning in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 6, pp. 1848–1859, 2023.

[26] J. Wang, J. Hu, G. Min, W. Zhan, A. Y. Zomaya, and N. Georgalas, "Dependent task offloading for edge computing based on deep reinforcement learning," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2449–2461, 2022.

[27] R. Bellman, "Dynamic programming," *science*, vol. 153, no. 3731, pp. 34–37, 1966.

[28] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016. arXiv: 1610.05492.

[29] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[30] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.

[31] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[32] P. P. Liang, T. Liu, Z. Liu, R. Salakhutdinov, and L. Morency, "Think locally, act globally: Federated learning with local and global representations," *CoRR*, vol. abs/2001.01523, 2020. arXiv: 2001.01523.

[33] M. Mehta and C. Shao, "A greedy agglomerative framework for clustered federated learning," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 12, pp. 11 856–11 867, 2023.