UNIVERSITY OF HELSINKI

https://helda.helsinki.fi

# FedVisual : Heterogeneity-Aware Model Aggregation for Federated Learning in Visual-Based Vehicular Crowdsensing

**Zhang, Wenjun; Liu, Xiaoli; Zhang, Ruoyi; Zhu, Chao; Tarkoma, Sasu**

**2024-11-15**

Institute of Electrical and Electronics Engineers Inc.

http://hdl.handle.net/10138/592395

Zhang, W, Liu, X, Zhang, R, Zhu, C & Tarkoma, S 2024, 'FedVisual : Heterogeneity-Aware Model Aggregation for Federated Learning in Visual-Based Vehicular Crowdsensing', IEEE internet of things journal, vol. 11, no. 22, pp. 36191-36202. https://doi.org/10.1109/JIOT.2024.3456751

Downloaded from Helda, University of Helsinki institutional repository. https://helda.helsinki.fi

This is an electronic reprint of the original article.

This reprint may differ from the original in pagination and typographic detail.

Please cite the original version.

# FedVisual: Heterogeneity-Aware Model Aggregation for Federated Learning in Visual-Based Vehicular Crowdsensing

Wenjun Zhang, Xiaoli Liu, Ruoyi Zhang, Chao Zhu, Sasu Tarkoma

***Abstract*—With the advancement of assisted and autonomous driving technologies, vehicles are being outfitted with an ever-increasing number of sensors. Among these, visible light sensors, or dash-cameras, produce visual data rich in information. Analyzing this visual data through crowdsensing allows for low-cost and timely perception of urban road conditions, such as identifying dangerous driving behaviors and locating parking spaces. However, uploading such massive visual data to the cloud for centralized processing can lead to significant bandwidth challenges and also raise privacy concerns among vehicle owners. Federated learning (FL), in which vehicles serve as both data generators and computing nodes, presents a promising solution to address these challenges. Nevertheless, urban roads are complex and vehicles in different locations encounter completely different scenes, resulting in non-i.i.d. (non-independently and identically distributed) characteristics. Additionally, the diversity in dash-camera and onboard computation resources may lead to differences in the performance of locally trained models. Indiscriminate aggregating of local models from all vehicles can potentially degrade the global model's performance. To overcome these challenges, we introduce FedVisual, a model aggregation approach for FL in vehicular visual crowdsensing. FedVisual leverages deep Q-Network (DQN) to select appropriate local models, considering the heterogeneities in visual data contents and vehicles' specifications. By leveraging the historical training experience, an effective model selection strategy can be obtained without complex mathematical modeling. Through the extensive simulations of our self-collected driving videos, FedVisual reduces model aggregation latency by up to 3.8% while improving the model's performance by up to 3.2% compared to reference works.**

***Index Terms*—Autonomous Internet of Things (IoT) systems, deep Q-Network (DQN), Federated Learning (FL).**

## I. Introduction

With the widespread adoption of dash-cameras, visual-based vehicular crowdsourcing has become an emerging computing paradigm for driving environment sensing [1]. By collecting the onboard images/video from vehicles and processing them through machine learning methods, the measuring of phenomena of common interest, such as urban traffic monitoring [2], [3] and parking space detection [4], [5], can be significantly improved in terms of economic efficiency.

Traditionally, the raw data generated from the road is transmitted to the cloud server for model training. However, this approach raises pertinent concerns, particularly in the domains of data privacy and communication costs. A prospective solution is to train models directly on IoT vehicle devices via Federated Learning (FL), which allows data to be stored and models to be trained on devices without moving the raw data to the centralized cloud [6], thereby preventing privacy breaches. Additionally, it facilitates the coordination of learning across multiple nodes in a distributed environment, overcoming challenges related to large-scale data transmission and storage [7]. However, conventional FL typically focuses on training a global model without accounting for device heterogeneity. This heterogeneity, often referred to as unbalanced non-independently and identically distributed (non-i.i.d.) data, can lead to decreased accuracy [8]. Given the vastly different driving environments shown in Fig. 1a, such as urban streets, suburban roads, and highways, the content captured by dash cameras can vary significantly in terms of scene complexity and target types. This content variety implies that the data distribution generated from different vehicle cameras is likely non-i.i.d., which, in turn, has a significant impact on the model's aggregation accuracy. On the other hand, as shown in Fig. 1b and 1c, the hardware specifications of dash cameras (e.g., resolution) and computation resources equipped on vehicles (e.g., CPU cycles) vary, leading to variations in model quality inputs and local training round numbers. Indiscriminately aggregating these models from all vehicles would deteriorate the global model's performance and extend aggregation times.

In this paper, we introduce FedVisual, an innovative FL strategy that enhances global model performance through heterogeneity-aware vehicle selection and model aggregation

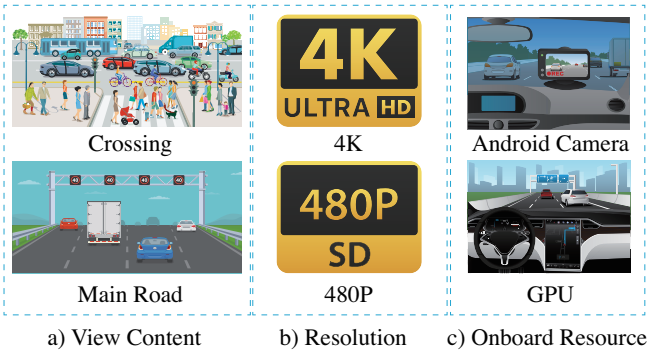



Fig. 1: The heterogeneity in the vehicular environment: a) view content, b) image resolution, and c) onboard computation resource.

W. Zhang, X. Liu and S. Tarkoma are with the Department of Computer Science, University of Helsinki, Finland.

C. Zhu* and R. Zhang are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. (Chao Zhu* is the corresponding author: chao@bit.edu.cn)

in visual-based vehicular crowdsensing scenarios. In FedVisual, normal vehicles train local models for specific common interest measurements (e.g., training YOLO models for parking space detection). Meanwhile, city buses are employed as vehicular fog nodes (VFNs) and act as edge computing nodes to aggregate local models uploaded from surrounding normal vehicles [1]. To select appropriate normal vehicles' local models for aggregation, we adopt an experience-based Deep Q-Network (DQN) algorithm [9]. However, the high dimensionality of the local model's parameter matrices poses a significant challenge to the state space in our DQN algorithm. To address this, we apply Principal Component Analysis (PCA) to reduce the dimensionality of the parameter space, simplifying the state representation. Through extensive experiments on our own collected real-world dataset, we find that FedVisual not only outperforms reference works with higher model accuracy but also achieves a faster aggregation rate.

The contributions of our work are threefold:

- We collect driving videos on three types of roads in Helsinki, including crossings, highways, and main roads. Based on the collected visual data, we analyze the heterogeneity in the real-world vehicular environment and its effect on the performance of FL.
- We propose FedVisual, a DQN-based model aggregation strategy tailored to select participating clients' local models in the FL, aiming to improve the global model's efficiency.
- Extensive experiments are conducted utilizing our collected driving images in Helsinki. The evaluation results demonstrate that FedVisual significantly outperforms the state-of-the-art methods.

The rest of the paper is organized as follows. The related work on vehicular crowdsensing and FL in vehicular networks is provided in Section II. We describe the system overview of our proposed FedVisual schema in Section III. In Section IV, we introduce FedVisual's framework and design principles of model aggregation. The analysis of the heterogeneity based on the real-world driving videos is presented in Section V and extensive empirical evaluations are conducted in Section VI. Finally, Section VII draws our conclusion and envision.

## II. Related Work

### A. Vehicular Crowdsensing

With the increasing use of dash cameras, many researchers have focused on analyzing sensor data collected from vehicles. These studies are designed to delve into the rich tapestry of information present within these visual feeds, covering a range of applications from object detection [3], [10], parking space identification [4], [5], [11], [12], vehicle tracking [13] to traffic monitoring [14]. Collectively, these studies significantly enhance the capabilities of comprehensive urban monitoring, offering nuanced insights into the dynamic urban milieu. Notably, Han et al. [10] have introduced a novel lidar and camera sensor fusion method dedicated to lane line classification. This method not only categorizes and extracts individual lane lines but is also capable of predicting the locations of obscured or missing lane markers. In the realm of parking space detection, Coric and Gruteser [11] have utilized crowdsensing data from vehicle parking sensors to differentiate between legal and illegal on-street parking spaces. Similarly, Grassi et al. [12] have harnessed video data from drivers' dash-mounted smartphones, facilitating real-time street analytics that is uploaded to the cloud as users navigate through urban spaces. Their system is bolstered by innovative lightweight algorithms that approximate the location of parked cars by merging data from various sources, including the vehicle's camera, GPS, and inertial sensors. This approach allows for effective tracking and quantification of parked vehicles within the camera's field of vision. Furthermore, Shi et al. in [4] have developed a crowdsensing parking system that offers drivers timely information about both current and future parking availability, leveraging the collective knowledge of the crowd. Zhu et al. in [5] have introduced a crowd-sensed parking system that not only accumulates but also disseminates details about available parking spots, capitalizing on the insights provided by crowd workers. In these works, however, all the raw data generated from the road must be transmitted centrally for model training without considering the sensitive private data and costly communication costs.

### B. FL in Vehicular Environment

Many works have already considered FL as a prominent role in vehicular networks. Zhu et al. [15] propose an FL-based travel mode (i.e., walking, biking, bus, driving, and train) identification system without accessing raw GPS data from users to protect users' privacy. Additionally, Zeng et al. [16] design an autonomous controller by training collaboratively among connected and autonomous vehicles (CAVs), considering the mobility of CAVs, the wireless fading channels, as well as the unbalanced and non-i.i.d. data across CAVs. However, these vehicular FL applications implicitly assume that it is possible for one single global model aggregated by weighted averaging to fit all decentralized nodes' data-generating distributions, rendering it poorly suited for heterogeneous user data in the vehicular domain.

Instead of training one single global model, Taïk et al. [17] propose an architecture where clusters of vehicles train models simultaneously, and only cluster heads are selected via vehicular-to-vehicular (V2V) communication to transmit their cluster aggregates to the MEC server, reducing communication costs, maintaining data integrity, and enhancing model performance. H. Yang et al. [18] propose a technique called Lead Federated Neuromorphic Learning (LFNL), a decentralized energy-efficient computing method based on spiking neural networks for efficient model training on resource-constrained edge devices. By utilizing various vehicular datasets for performance evaluation, LFNL can maintain high recognition accuracy while reducing data traffic and energy consumption, even with uneven data distribution. In addition, Gong et al. [19] apply a hierarchical cluster FL approach that characterizes the variation of model weights during training to find the proper time to quickly bi-partition the clusters of mobile devices in a hierarchical manner.

Despite these works applying FL to the vehicular network environment to a certain extent, protecting user privacy and

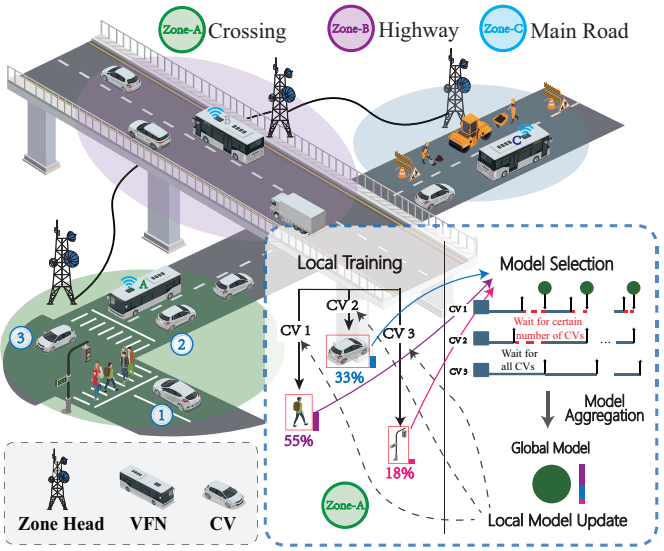



Fig. 2: Overview of FedVisual.

saving network bandwidth, they predominantly rely on simulations using public datasets such as MNIST and CIFAR-10, which only address manipulating data distribution heterogeneity. However, they have not considered the problem of data and system heterogeneity caused by the complexity of road conditions and the diversity of vehicles in the real world. In this article, we are the first to propose a heterogeneity-aware model aggregation scheme aimed at reducing heterogeneity's impact on the global FL model through differentiated local model selection from normal vehicles.

## III. System Overview

In this section, we define the related terms and present an overview of the workflow of FedVisual.

### A. Related Terms

In FedVisual, there are three main participants, namely zone heads, client vehicles (CVs) and vehicular fog nodes (VFNs).

**Service Zones and Zone Heads:** Due to the diversity of urban traffic, areas that are geographically distant from each other will exhibit different content characteristics. As illustrated in Fig. 2, vehicles traversing highways and crossings are exposed to entirely distinct views. Compared to using the same model across the entire city, we divide the city's vehicles into different clusters based on their geographical locations. To prevent the heterogeneity of training data from reducing model performance, each cluster independently maintains and updates its own global model without interacting with models from other clusters.

We assume that urban areas in modern cities are fully covered by cellular networks, making it convenient to group vehicles within the coverage area of a base station into a single cluster. In FedVisual, the city is divided into several service zones, and each service zone has a base station located at its center, which serves as the zone head. As illustrated in Fig. 2, three zones are presented, namely Zone A, Zone B, and Zone C. The zone head can detect vehicles entering and leaving its coverage area with the existing cellular registration mechanisms. Moreover, vehicles in a specific service zone regularly report their moving directions, locations, available capacities, and models to the corresponding zone head. By connecting base stations through underground fiber optics, these vehicles' data are visible to the control center of FedVisual.

**Client Vehicles and Local Models:** With the development of artificial intelligence, applications such as intelligent lane-changing assistance and cooperative driving are becoming increasingly widespread. These applications generate a series of tasks involving image processing, object detection, and target recognition. Currently, deep learning is commonly used to handle these tasks, often training on vehicles' own data. Therefore, ensuring model accuracy while protecting user privacy is a significant challenge. This has led to the demand for FL, where client vehicles do not transmit local data but instead send the trained local models to a server for aggregation.

In FedVisual, client vehicles (CVs) refer to normal vehicles performing tasks using deep learning, such as running YOLOv5n to identify vacant parking spots in the camera's view. Given the limited data sample size from a single vehicle, CVs enhance model accuracy through FL by transmitting their local models or only updates to a VFN for aggregation.

**Vehicular Fog Nodes and Global Models:** Although transmitting only models instead of raw data mitigates the consumption of transmission resources to some extent, the size of neural networks for visual processing tasks cannot be neglected. For instance, the size of the YOLOv5n model can reach hundreds of MBs. Frequent model aggregation may lead to significant bandwidth consumption.

Instead of assigning zone heads as aggregation nodes, we select certain vehicles equipped with capable computing and communication resources, such as buses, to serve as VFNs and designate them to aggregate local models from surrounding CVs while traveling. This approach offers two advantages: firstly, vehicles can transmit data via high-bandwidth one-hop connections, saving valuable cellular network data; secondly, the widespread distribution of these specific vehicles helps avoid network congestion that might result from centralized transmission. VFNs are responsible for maintaining and updating the global model and regularly communicating with the zone head to transmit the latest version of the global model.

### B. Workflow of FedVisual

Fig. 2 illustrates the FedVisual's workflow. The whole process consists of five steps.

1) *CVs detection and model distribution.* In the initial phase, VFN broadcasts signals to detect nearby CVs. Upon receiving acknowledgments, VFN requests the regional model from the zone head and distributes it to these CVs.
2) *Local initial training.* As illustrated in Fig. 2, taking Zone A as an example, CVs train locally upon receiving the regional model. After a certain period, the VFN will notify CVs to return the model weights updates $\Delta \boldsymbol{w}_k^0$, along with other information like the resolution $r_k$ and local training epochs $n_k$, collectively denoted as $\left\{\left(\Delta \boldsymbol{w}_k^0, r_k, n_k\right) \mid k \in [N]\right\}$.

TABLE I: Notations and Explanations

| Notation | Explanation |
|---|---|
| $k, D_k$ | CV, Local dataset |
| $\boldsymbol{w}$ | Model weights |
| $M$ | Bounding box number |
| $c, C$ | Class, Class set |
| $i, j$ | Image, Object in image |
| $y_{ic}$ | Binary indicator for if class $c$ is whether the correct classification for prediction $i$ |
| $\hat{y}_{ic}(\boldsymbol{w})$ | The predicted probability of class $c$ for prediction $i$ |
| $t$ | Training round |
| $\lambda$ | The weight of F1 |

3) *Models selection.* At the communication round t, where $t = 1, 2, \ldots, T$, upon receiving the CVs' information, the DQN agent computes $Q(s_t, a; \theta)$ for all CVs and selects K devices corresponding to the top-K values of $Q(s_t, a; \theta)$, $a = 1, \ldots, N$ for aggregation along with related information from previous communication round, simulating the model selection scenario in Fig. 2.
4) *Models aggregation.* The VFN aggregates the latest regional model $w_t$ based on the top-K devices selected by the agent applying the FedAvg. The corresponding copies of CVs' model weights stored on VFN are also updated.
5) *Local model update.* The devices participated download the latest regional model weights $w_t$ and perform specific epoch of SGD locally to obtain $\{\Delta \boldsymbol{w}_k^{t+1} \mid k \in [K]\}$ and upload them to the VFN.

Steps 3-5 will be repeated until the connection between the CVs and the VFN is disconnected. The VFN will send the latest model to the corresponding zone head. Note that VFNs may leave the current service zone and enter another one before model aggregation is completely processed. In this case, the VFNs transmit the generated global model to the new zone head, who then hands it over to the previous one.

## IV. Model Aggregation Framework

In this section, we introduce how model aggregation works in FL and present the design principles of the DQN-based model selection module. The notations are listed in Table I.

### A. Local Optimization

In our application scenario, we are moving beyond traditional single-label image classification task, e.g., MNIST and CIFAR-10. Instead, we aim to address multi-task object detection tasks that better reflect real-world scenarios. Consequently, we will utilize the YOLOv5n [20] model for local model training. The YOLOv5n loss function consists of three main components:

**The Complete Intersection over Union (CIoU) [21] loss**, incorporating model parameters, is critical for optimizing bounding box predictions in object detection models. The parameterized formula is given by:

$$\mathcal{L}_{\text{CIoU}}(\boldsymbol{w}) = 1 - \text{IoU}(\boldsymbol{w}) + \frac{\rho^2(\mathbf{b}(\boldsymbol{w}), \mathbf{b}^{gt})}{c^2(\boldsymbol{w})} + \alpha(\boldsymbol{w}) \cdot v(\boldsymbol{w}) \tag{1}$$

where $\text{IoU}(\boldsymbol{w})$ is the Intersection over Union calculated using the predicted bounding box $\mathbf{b}(\boldsymbol{w})$ and the ground truth $\mathbf{b}^{gt}$, $\rho(\mathbf{b}(\boldsymbol{w}), \mathbf{b}^{gt})$ represents the Euclidean distance between the centers of the $\mathbf{b}(\boldsymbol{w})$ and $\mathbf{b}^{gt}$, $c(\boldsymbol{w})$ is the diagonal length of the smallest enclosing box that encompasses both the $\mathbf{b}(\boldsymbol{w})$ and $\mathbf{b}^{gt}$, $v(\boldsymbol{w})$ measures the aspect ratio disparity, computed as $\frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w(\boldsymbol{w})}{h(\boldsymbol{w})}\right)^2$ where $w(\boldsymbol{w}), h(\boldsymbol{w})$ are the width and height of $\mathbf{b}(\boldsymbol{w})$, and $\alpha(\boldsymbol{w})$ is a scaling factor that adjusts the influence of the aspect ratio penalty, defined as $\frac{v(\boldsymbol{w})}{(1-\text{IoU}(\boldsymbol{w}))+v(\boldsymbol{w})}$, emphasizing aspect ratio consistency more as the overlap increases.

**The object confidence loss function**, incorporating the dependence on model parameters $\boldsymbol{w}$, is given by:

$$\mathcal{L}_{\text{obj}}(\boldsymbol{w}) = -\frac{1}{M}\sum_{i=1}^{M}[y_i \cdot \log(\sigma(\hat{y}_i(\boldsymbol{w}))) + (1 - y_i) \cdot \log(1 - \sigma(\hat{y}_i(\boldsymbol{w})))] \tag{2}$$

where $M$ is the number of bounding boxes, $y_i$ is the ground truth label indicating the presence with value 1 or absence with value 0 of an object in the $i$-th bounding box, $\hat{y}_i(\boldsymbol{w})$ represents the predicted probability that the $i$-th bounding box contains an object, and $\sigma$ denotes the sigmoid function, which maps the raw output of the neural network to a probability between 0 and 1.

**The classification loss function** for a model, reflecting the dependency on its parameters $\boldsymbol{w}$, is formulated as follows:

$$\mathcal{L}_{\text{cls}}(\boldsymbol{w}) = -\frac{1}{M}\sum_{i=1}^{M}\sum_{c=1}^{C}[y_{ic} \cdot \log(\sigma(\hat{y}_{ic}(\boldsymbol{w}))) + (1 - y_{ic}) \cdot \log(1 - \sigma(\hat{y}_{ic}(\boldsymbol{w})))] \tag{3}$$

where $C$ is the number of classes, $y_{ic}$ is a binary indicator (0 or 1) if class $c$ is the correct classification for prediction $i$, and $\hat{y}_{ic}(\boldsymbol{w})$ is the predicted probability of class $c$ for prediction $i$.

The total loss function combining these three components with respective weights is:

$$f_v(\boldsymbol{w}) = \lambda_{\text{box}}\mathcal{L}_{\text{CIoU}}(\boldsymbol{w}) + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}}(\boldsymbol{w}) + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}(\boldsymbol{w}) \tag{4}$$

We consider a typical FL ecosystem with one VFN and a set of CVs denoted by $\mathcal{V} = \{1, \ldots, N\}$. We introduce FL in the context of an object detection problem, defined over a compact feature space $\mathcal{X}$ and a label space $\mathcal{Y}$. The local dataset of a specific CV $k$ ($k \in \mathcal{V}$) is defined as $D_k = \{(x_i, Y_i)\}_{i=1}^{|D_k|}$, where $x_i$ is the $i$-th image, and $Y_i$ is a set of annotations for that image, with each annotation providing details about one detected object $Y_i = \{((b_{j1}, b_{j2}, b_{j3}, b_{j4}), c_j) \mid j \in J_i\}$, where $(b_{j1}, b_{j2}, b_{j3}, b_{j4})$ denotes the coordinates of the bounding box for the $j$-th object in image $i$, typically defined as the x and y coordinates of the top-left corner and the width and height of

the box. $c_j \in \{1, 2, \ldots, C\}$ represents the class label of the $j$-th object. $J_i$ is the index set of objects detected in image $i$.

In scenarios where each CV optimizes its own set of model parameters independently, the objective for each CV can be formulated as follows:

$$\underset{\boldsymbol{w}_k}{\text{minimize}}\ f_k(\boldsymbol{w}_k) \tag{5}$$

where $\boldsymbol{w}_k$ are the model parameters specific to CV $k$, and $f_k(\boldsymbol{w}_k)$ is the local loss function for CV $k$, tailored to its local dataset $D_k$.

To reduce communication overhead, each participating CV $k$ uploads the model weight difference $\Delta \boldsymbol{w}_k^t = \boldsymbol{w}_k^t - \boldsymbol{w}_k^{t-1}$. Upon receiving the updates from all participating CVs, the VFN performs FedAvg to update the global model $\boldsymbol{w}^{t+1}$ for the new communication round with the aggregated update $\Delta \boldsymbol{w}^t$ and previous global model $\boldsymbol{w}^t$:

$$\Delta \boldsymbol{w}^t = \sum_{k \in \mathcal{V}} \frac{|D_k| \Delta \boldsymbol{w}_k^t}{\sum_{k \in \mathcal{V}} |D_k|} \tag{6}$$

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \Delta \boldsymbol{w}^t \tag{7}$$

### *B. DQN for Model Selection*

*1) MDP Formulation:* Reinforcement Learning (RL) [22] is the process by which an agent learns to act within an environment in order to maximize cumulative rewards. Formally, RL is treated as a Markov Decision Process (MDP) [23], defined by a set of states $\mathcal{S}$, actions $\mathcal{A}$, and transition dynamics $p(s'|s, a)$ that describe the probability of transitioning to state $s'$ from state $s$ upon taking action $a \in \mathcal{A}$. The integration of deep learning into RL, leading to Deep Reinforcement Learning (DRL), has significantly improved decision-making capabilities from high-dimensional sensory inputs. DRL enhances RL by utilizing deep neural networks to approximate the policy $\pi$ or the value functions $V(s)$ or $Q(s, a)$, where the Q-function update rule in DQN is represented as:

$$Q(s, a; \theta) \leftarrow Q(s, a; \theta) + \alpha \left[ r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta') - Q(s, a; \theta) \right] \tag{8}$$

Here, $\theta$ represents the parameters of the Q-network. The DQN algorithm employs a neural network to approximate the Q-function $Q(s, a; \theta)$, with updates aimed at minimizing the loss between the predicted and the target Q-values:

$$\mathcal{L}(\theta) = \mathbb{E}\left[ \left( r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta') - Q(s, a; \theta) \right)^2 \right] \tag{9}$$

where the expression inside the expectation is the temporal difference error.

*2) Latency Model:* Each CV can transmit its model to the VFN through V2V communication. According to the Shannon theorem [24], the achievable data rate for CV $k$ to the VFN can be described as:

$$r_k = B \log_2 \left( 1 + \frac{P_k H_k}{N_0} \right) \tag{10}$$

where $B$ denotes the bandwidth of the V2V network communication, $P_k$ is the transmit power of CV $k$, $H_k$ is the channel gain from CV $k$ to the VFN, and $N_0$ is the power of background noise.

Furthermore, the Link Life Time (LLT) should be considered when downloading and uploading models. LLT defines link sustainability as the duration of time that two nodes remain connected. For the VFN and CV $k$, moving in the same or opposite directions, we denote the distance between the VFN and CV $k$ by $D_k$, and the velocities difference between them by $\Delta v_k$. Their LLT can be calculated as follows [25]:

$$LLT_k = \frac{-\Delta v_k \times D_k + |\Delta v_k| \times TR}{(\Delta v_k)^2} \tag{11}$$

where $TR$ denotes the transmission range.

Denote $T_k^{down}$ the downloading model time from VFN to CV $k$, $T_k^{train}$ the training time at CV $k$, and $T_k^{up}$ the uploading time from CV $k$ to the VFN. Then the transmitting time $T_k$ for CV $k$ can be described as:

$$T_k = T_k^{down} + T_k^{train} + T_k^{up} \tag{12}$$

To guarantee connectivity, the transmitting time $T_k$ must be less than or equal to $LLT_k$ as defined by equation (11).

We present the FL optimization problem formulation in this section. Let $a_{t,n} \in \{0, 1\}$ denote whether the CV $n$ is chosen to complete full local training during round $t$. The total processing latency for a training round $t$ can be expressed as:

$$H_t = \max_{1 \le n \le N} \left( T_n^{down} + T_n^{train} + T_n^{up} \right) a_{t,n} \tag{13}$$

Let $\text{F1}(T)$ denote the F1 score of the global model on a global test dataset after the last training round $T$. Our FL system optimization problem can be defined as:

$$\max_{A} \mathbb{E}\left[ \lambda \text{F1}(T) - (1 - \lambda) \sum_{t \in T} H_t \right] \tag{14}$$

where $\lambda \in [0, 1]$ is the weight representing the importance of F1. Our objective is to maximize the expectation of the weighted difference between the F1 score of the global model and the total processing latency.

*3) The Agent based on Deep Q-Network:* Suppose there is a FL job on $N$ available CVs. In each round, using a DQN, K CVs are selected to participate in the training.

**State**: Let the state of round $t$ be represented by a vector $s_t = \left( w_{t-1}^{(1)}, w_{t-1}^{(2)}, \ldots, w_{t-1}^{(N)}, T_{(1)}^{t-1}, T_{(2)}^{t-1}, \ldots, T_{(N)}^{t-1} \right)$, where $w_{t-1}^{(1)}, \ldots, w_{t-1}^{(N)}$ are the model weights and $T_{(1)}^{t-1}, \ldots, T_{(N)}^{t-1}$ are the corresponding transmission times of the $N$ CVs at round $t-1$. The agent collocates with the FL server and maintains a list of model weights $\{w_k | k \in [N]\}$.

The resulting state space can be huge, e.g., a CNN model can contain millions of weights. It is challenging to train a DQN with such a large state space. In practice, we propose to apply an effective and lightweight dimension reduction technique on the state space, i.e., on model weights, which will be described in detail in part IV-B4.

**Action**: At the beginning of each communication round $t$, the agent selects K CVs from the $N$ CVs. Note that the device number K is not always fixed for each communication round.

In particular, the agent is trained by selecting only one out of $N$ CVs to participate in FL per round based on DQN, while in testing and application, the agent will sample a batch of top-$K$ clients to participate in FL. Therefore, we define an action $a$ as the VFN receiving a weight update from a specific CV. Once the update from this CV is received, weight updates from subsequent CVs will no longer be incorporated into the global model update. The DQN agent learns an approximator of the optimal action-value function $Q^*(s_t, a)$, which estimates the action that maximizes the expected return starting from $s_t$.

**Reward**: To optimize the FL performance, the reward function should reflect the results in the test accuracy, processing latency, and communication cost after executing client selection decisions generated by the agents. The reward $r_t$ at training round $t$ is defined as:

$$r_t = \lambda \text{F1}(t) - (1 - \lambda) H_t \tag{15}$$

*4) Dimension Reduction:* In our study, we also address the challenge of training a DQN agent with a high-dimensional state space, which arises due to the use of deep neural networks like YOLOv5n that contain millions of weights. To effectively manage this complexity, our approach involves compressing the model weights using PCA.

Initially, we compute PCA vectors from local model weights $\{w_k^1 \mid k \in [N]\}$ obtained during the first training round at $t = 1$. These vectors are crucial as they capture the principal components that significantly represent the data variability with fewer dimensions. By doing this, we avoid the need to fit the PCA model in subsequent training rounds repeatedly.

For the following rounds $t = 2, 3, \ldots, T$, we utilize these predetermined PCA loading vectors to perform a linear transformation on the new model weights $\{w_k^t \mid k \in [N]\}$. This method allows us to consistently represent the state space with the first several principal components derived from the initial PCA fitting, thereby drastically reducing the computational overhead associated with recompressing the model weights.

*5) Training the Agent with Double DQN:* As shown in Algorithm 1, we propose using the Double Deep Q-Learning Network (DDQN) [26] as a method to effectively learn the optimal action-value function $Q^*(s_t, a)$. This approach builds on the foundations of Q-learning and its extensions to DQN, enhancing them to address specific challenges such as overestimation bias.

Initially, the Q-learning update rule is applied:

$$\theta_{t+1} = \theta_t + \alpha(Y_t^Q - Q(s_t, a_t; \theta_t)) \nabla_{\theta_t} Q(s_t, a_t; \theta_t) \tag{16}$$

where $\alpha$ is the learning rate, and $Y_t^Q$ is the target for the Q-learning update, defined as:

$$Y_t^Q \equiv R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t) \tag{17}$$

with $\gamma$ as the discount factor, $R_{t+1}$ as the immediate reward received after taking action $a_t$ in state $s_t$, and $s_{t+1}$ as the resulting state.

In the DQN framework, the action-value function is approximated using deep neural networks. The target Q-value ($Y_t^{\text{DQN}}$) is computed using a separate target network to provide more stable training outcomes:

$$Y_t^{\text{DQN}} \equiv R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t^-) \tag{18}$$

where $\theta_t^-$ are the parameters of the target network, periodically updated from the online network $\theta_t$.

DDQN refines the DQN by decoupling the action selection from the evaluation:

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t); \theta_t') \tag{19}$$

where $\theta_t'$ represents the parameters of a second network used to evaluate the greedy policy determined by the online network $\theta_t$. This enhancement reduces the overestimation often observed in DQN.

In the context of FL, the DDQN algorithm is used to train a DRL agent. The FL server initiates by performing random device selection to establish initial states. The DQN generates actions to select devices for the FL training rounds. After multiple rounds of training, having sampled numerous action-state pairs, the agent utilizes the learned policy to optimize:

$$\mathcal{L}(\theta_t) = (Y_t^{\text{DoubleQ}} - Q(s_t, a; \theta_t))^2 \tag{20}$$

This learning process ensures that the policy and network parameters are refined progressively for optimal decision-making in distributed environments.

**Algorithm 1** FedVisual Model Training

**Input:** observation state $s$, reward $r$
**Output:** action $a$

Set different regional models for different regions
Initialize FedVisual Model
**for** $each_train_round = 1, 2, 3 \ldots$ **do**
  VFN loads regional model based on corresponding region as global model
  Initialize all client models based on the global model
  **for** $each_communication_round = 1, 2, 3 \ldots$ **do**
    Place the global model for each client
    Obtain state $s$ using the arrival time and PCA-refined parameters from the previous communication round
    Agent selects client ID as action $a$ based on state $s$
    Based on the arrival time of selected client, client models that arrive not after selected client will participate in the aggregation process, while models that arrive later will be excluded from the aggregation
    The updates from participating clients are aggregated to obtain the global increment $\Delta \boldsymbol{w}$, then added to the previous global model to obtain a new one
    Test the score of the new global model based on VFN local dataset as a reward $r$ for FedVisual Model
    Record the quadruple $\{s, a, r, s'\}$, where $s'$ is assigned a value when obtaining the state $s$ of the next communication round
  **end for**
  Update regional model by new global model
  Update FedVisual Model by records of quadruple
**end for**




Fig. 3: Road trajectories and categories of driving videos.

### C. Computation Complexity Analysis

FedVisual's primary computational complexity arises from the PCA and DQN training. For PCA-based dimensionality reduction, the complexity of the operation is $\mathcal{O}(mN^2)$, where $m$ refers to the number of data points, and $N$ refers to the number of dimensions (or features) of each data point [27]. For DQN computation, since the dimensionality of the DQN inputs has been reduced, the DQN model used in FedVisual can be considered equivalent to a standard DQN. The computational complexity and resource requirements of a standard DQN can be analyzed from two aspects: the training phase and the decision output phase. In the training phase, the most computationally intensive part of DQN involves forward propagation, loss calculation, backpropagation, and parameter updates. The computation complexity of forward propagation and backpropagation is $\mathcal{O}(\sum_i^L n_i n_{i-1})$, where $L$ is the network layers and $n_i$ is the number of neurons in layer $i$. The complexity of loss calculation is $\mathcal{O}(n)$, where $n$ is the number of output neurons. The complexity of the parameter is $O(|\theta|)$, where $\theta$ is the number of model parameters. The training phase primarily involves forward propagation to compute Q-values for each state, the complexity remains $\mathcal{O}(\sum_i^L n_i n_{i-1})$.

## V. Heterogeneity in Vehicular Environment

In this section, we thoroughly discuss the collection and processing of real data we collected, analyzing the heterogeneity in the vehicular environment. Additionally, we investigate the potential impacts of this heterogeneity on model performance and evaluate its broader implications for FL frameworks.

### A. Data Collection and Preprocessing

In real-world vehicular environments, the disparity of visual data captured by dash cameras is natural and continuously changes in a spatio-temporal manner with the movement of vehicles. Therefore, compared to merely simulating data disparity on discrete public datasets, we prefer to collect real-world driving videos by ourselves to capture the natural occurrence of heterogeneity in visual data content. Based on this principle, we have collected driving videos along two trajectories in Helsinki, as shown in Fig.3. The first trajectory extends from Soittoniekanaukio to Brokadi Ravintola, while the second spans from Malminkartano Railway Station to Väinö Auerin Katu. These paths are meticulously planned to encompass a variety of urban and suburban environments to capture a wide array of traffic scenarios.

We divide these trajectories into road segments and categorize those segments into three categories: crossings, main roads, and highways. Meanwhile, we dissect collected videos into images by frame and classify them into these three categories. To improve the precision of our dataset labels, we first utilize the large-scale model, SAM [28], for initial pre-annotation. This is subsequently complemented by thorough manual verification and adjustments. To further reduce inaccuracies stemming from manual processes, we implement a strategy where several annotators independently assess and label identical images. Their individual labelings are then meticulously compared and merged to ensure consistency. Additionally, we conduct regular review sessions throughout the annotation cycle to identify and correct any errors, thereby harmonizing the labeling standards across the dataset. Note that we have processed the collected dashcam images for privacy, such as blurring license plate information, and uploaded them to GitHub.[1].

### B. Heterogeneity in Visual Data Content

We select six typical segments for each category (e.g., C1, C2, C3, C4, C5, C6 for crossings) and calculate the similarity among all 18 segments in terms of the density of pedestrians, vehicles, and traffic signs. Specifically, we begin by aggregating the counts of pedestrians, vehicles, and traffic signs across all images captured from each road segment—covering crossings, highways, and main roads. These aggregated counts are then normalized to establish a relative scale of similarity across different segments. As illustrated in Fig. 4, segments with counts closely are assigned a higher similarity score, indicated by a warmer (redder) color, whereas segments with larger deviations are marked with a lower similarity score, represented by a cooler (bluer) color.

Through the comparison of Fig. 4a, Fig. 4b, and Fig. 4c, we observe that the degree of diversity in video content varies for different recognition targets. For example, in Fig. 4a, we find that the number of pedestrians in videos collected from crossings and highways is quite similar. This is because pedestrian traffic is sparse on highways, and although crossings may have many pedestrians, the number that can be recognized is reduced due to vehicle obstructions. However, as shown in Fig. 4b, the counts of vehicles across different segments show a distinct pattern. We discover that the number of vehicles in videos collected from highways and main roads is more similar to each other than those collected from crossings. This is due to the broader field of view on highways and main roads, which allows more vehicles to be seen compared to crossings. As shown in Fig. 4c, the distribution of traffic signs is similar to that of vehicles in Fig. 4b, except the diversity in the number of traffic signs across segments is even greater

[1]https://github.com/xzqdll/FedVisual

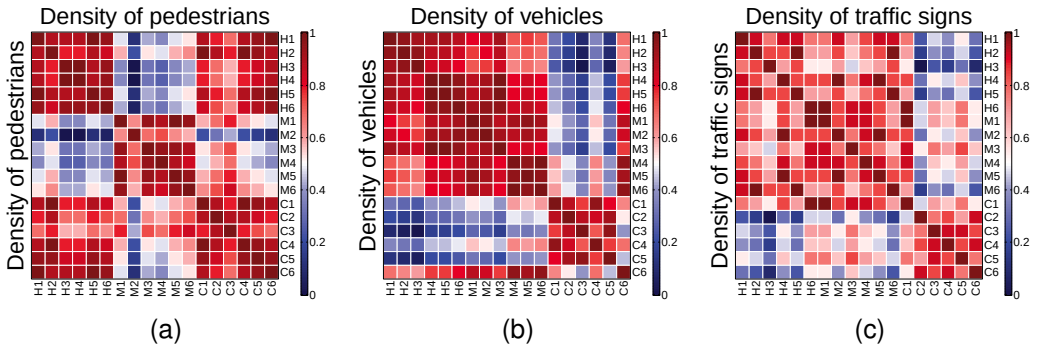



Fig. 4: The similarities in terms of the density of a) pedestrians, b) vehicles, and c) traffic signs.

(i.e., the colors are more chaotic). This is because the positions of traffic signs are fixed and are unevenly distributed along the trajectories, leading to a more varied distribution of traffic sign counts between segments compared to the distribution of vehicle counts. Given these observations, we can conclude that the road environment has a significant impact on the heterogeneity of video content for different recognition targets.

### C. Heterogeneity's Effect on Model Performance

*1) Visual Content Heterogeneity Effect:* As mentioned before, to assess the effect of video content heterogeneity on model performance, we dissect collected videos into images by frame and classify them as crossing, highway, and main road. After preprocessing the dataset, we deploy pre-trained YOLOv5n to train the dataset for recognizing the contents of the images, including car, bus, traffic light, traffic sign, person, scooter, bicycle, and bus station [29]. To evaluate the model's performance, we measure the model's precision and recall:

- Precision is the proportion of true positive predictions in all positive predictions made by the model. It is defined by the formula: $Precision = \frac{TP}{TP+FP}$, where $TP$ is the number of correctly predicted positive instances, and $FP$ is the number of instances wrongly predicted as positive when they are actually negative. A higher precision indicates that the model is more accurate when it predicts positive classes.
- Recall is the proportion of true positive predictions out of all actual positive instances. It is defined by the formula: $Recall = \frac{TP}{TP+FN}$, where $FN$ is the number of instances wrongly predicted as negative when they are actually positive. A higher recall value indicates that the model is better at covering the actual positive instances.

Fig. 5a illustrates the model precision under various scenarios, where *Mixed* refers to combining the image sets belonging to crossing, highway, and main road into one collection. We can observe that YOLOv5n exhibits higher precision on main roads and crossings than on highways. This is because, compared to other road conditions, highways have the least number of relevant elements, such as pedestrians and vehicles, leading to a higher proportion of false positives (such as trees or forests). Notably, the model's precision is the lowest in the mixed scenarios. This could be due to the fact that mixed

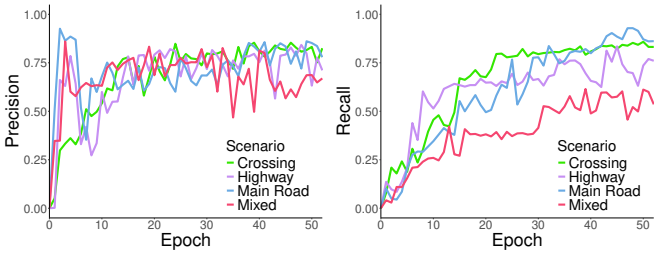



Fig. 5: YOLOv5n performance measured by a) precision and b) recall under different scenarios.

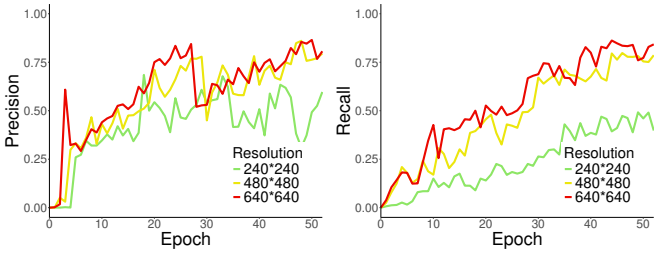
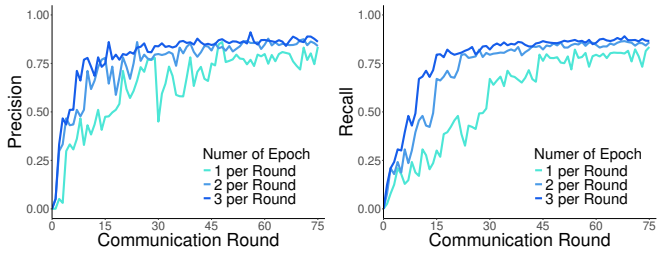



Fig. 6: YOLOv5n performance measured by a) precision and b) recall under different resolutions.

scenarios contain the most diverse content, highlighting their non-i.i.d. characteristics more prominently.

Fig. 5b illustrates the model recall under various scenarios. We can observe that the model's recall follows a similar trend to its precision. However, the variation in recall across different road conditions is more pronounced. For instance, after the model has converged, the recall rate in highway scenarios decreases by 18.39% compared to crossings, while in mixed scenarios, it decreases by 26.44% compared to crossings. This may be due to the increased difficulty of identifying true positives from more diverse video content.

*2) Dash-Camera Hardware Specification Effect:* In the vehicular environment, different vehicles may be equipped with dash cameras of varying parameters, affecting the resolution, frame rate, and field of view (FoV) of the collected videos. To meet the requirements of the network architecture, inputs of different resolutions are uniformly compressed to a




(a) Precision (b) Recall

Fig. 7: YOLOv5n performance under different number of epoch in each round: a) precision; b) recall.

specific resolution (e.g., $416 \times 416$) in YOLOv5n. However, images with varying resolutions may suffer differing degrees of information loss during the compression process, leading to variations in the performance of models trained on these images.

To explore this variation, we compress all images in the crossing collection to three different resolutions: $240 \times 240$, $480 \times 480$, and $640 \times 640$ to simulate the varying resolutions from different cameras. Then, they would be uniformly compressed to a specific resolution in YOLOv5n. As demonstrated in Fig. 6a, models trained on higher resolution images exhibit superior precision compared to those trained on lower resolutions. This improvement is attributed to the higher information entropy contained in the original high-resolution images. After being scaled down, these images still provide the model with richer context and feature information, enhancing object detection precision.

Fig. 6b shows the models' recall performance across different resolutions. We observe the trend in recall rates is similar to that of precision: the higher the resolution, the higher the recall rate. This is because high-resolution images contain more detailed information and a higher pixel density before compression. Therefore, even after being scaled down, these images continue to provide richer feature information. This aspect is particularly crucial for detecting small objects, which are more likely to be preserved during the compression process, making it easier to identify true positives from more diverse video content.

*3) Onboard Computation Resource Effect:* Besides the differences in dash cameras, the computation resources in vehicles also vary. Some cars boast significant floating-point operation capabilities, such as Teslas equipped with AMD Ryzen CPUs and dedicated GPUs, while some BMW or Mercedes are fitted with Qualcomm chips similar in computing power to smartphones. Some vehicles are simply equipped with smart dash cameras, offering limited graphics processing capabilities.

In FedVisual, we assume that vehicles periodically transmit their local models to a central node. Due to disparities in computation resources, the number of epochs completed by local models on different vehicles may vary. Within the same time limit, vehicles with superior floating-point computational power can complete more training epochs and vice versa. To explore the influence of computational resources on the performance of the global model in FL, We evenly split the crossing image collection into two parts, presuming each part was gathered by a different vehicle, and routinely merge the local models trained from these two vehicles at every communication round. Within one communication round, we assume that the local models undergo training for 1, 2, and 3 epochs, respectively. Fig. 7a and Fig. 7b display the precision and recall rates of the aggregated global model.

From the two figures, we observe that both the precision and recall rates of the converged global model improve as the number of epochs trained on the local models increases. Particularly in terms of recall rate, local models that underwent training for 3 epochs converged after approximately 15 communication rounds, while those trained for 2 and 1 epochs converged after 20 and 45 rounds, respectively. Therefore, within a vehicular network environment, assuming the vehicles observe similar content, aggregating local models from vehicles equipped with superior hardware configurations yields better outcomes than those from vehicles with inferior configurations.

Given these observations, we can infer that indiscriminate aggregation of local models from various vehicles, neglecting the influences of diverse video content, dash-camera specifications, and onboard computation resources, would lead to the performance degradation of the global model within an FL framework.

## VI. Implementation and Evaluation

In this section, we present the performance of FedVisual in loss convergence, investigate how the weights affect the F1 and latency for CVs selection and model aggregation, and compare FedVisual against the two baselines across various experiment settings.

*1) Loss Convergence:* The DRL agent is trained on three distinct datasets: crossings, highways, and main roads, utilizing one VFN and ten CVs for each dataset. Our experiments randomly distribute a single dataset among ten clients to simulate an FL environment. The DDQN model employed in the DRL agent consists of two four-layer MLP networks, with each consisting of three hidden layers and an output layer. Both the primary and the target networks adhere to this architecture, but they are updated at different intervals to enhance the stability of the learning process.

Fig. 8 illustrates the loss convergence across the three road categories when implementing our proposed FedVisual DQN approach. The loss curves for the three road segments (crossings, highways, and main roads) exhibit quite similar convergence patterns, with each rapidly stabilizing to a consistent value within 15 communication rounds. This rapid convergence demonstrates that our method enables rapid learning with less client communication overhead. Such efficiency in reducing communication exchanges is crucial for effectively addressing real-world challenges and can lead to significant improvements in deployment scenarios where communication costs are a concern.

*2) Weight effects of F1 and latency:* In DRL, the agent aims to maximize the cumulative reward it receives. By

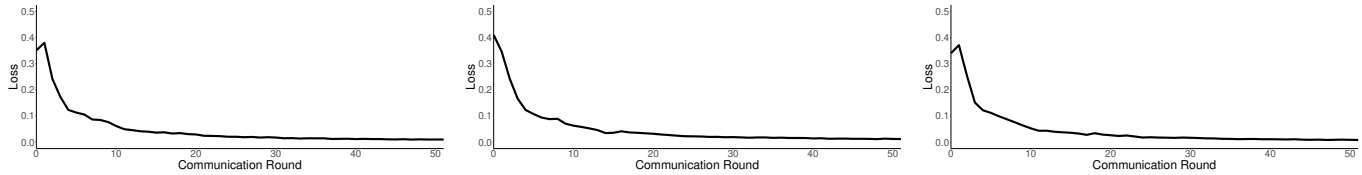



Fig. 8: The loss of proposed FedVisual DQN training in different communication rounds under different road categories: a) crossings, b) highways, and c) main roads.

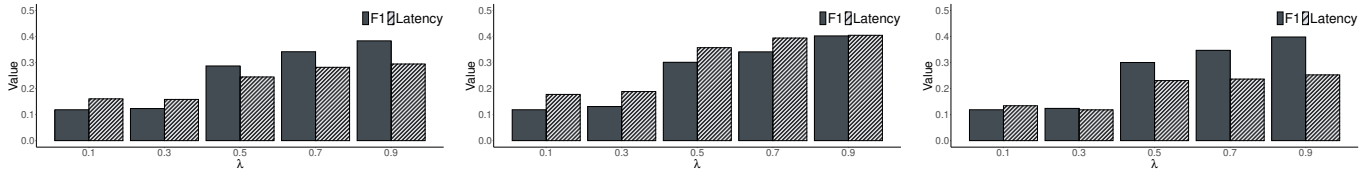



Fig. 9: Latency and F1 metrics of DQN-based strategies with different $\lambda$ under different scenarios: a) crossings, b) highways, and c) main roads.

adjusting the weights for the F1 score and latency in the reward function, the agent is guided to which aspects are more important. If the weight for the F1 score $\lambda$ is increased, the agent will tend to take actions that improve the F1 score, even it might lead to increased latency. Conversely, if the latency penalty is increased, the agent will prioritize strategies that reduce the overall latency, potentially at the expense of decreasing the F1 score. Such analysis can help decision-makers select the optimal weight configuration to achieve the desired balance between model performance and latency in specific applications.

We have assigned different weight coefficients to the F1 score and latency metrics to determine the most rational weight distribution for maximizing the reward return. From the charts shown in Fig. 9, we can observe that as the weight assigned to the F1 score $\lambda$ gradually increases (a corresponding decrease in the weight for latency), the value of the F1 score progressively rises, as does the latency value. However, we also observed that once the weight assigned to latency exceeds 0.5, the increase in latency values becomes marginal or even stabilizes without further changes. This is a surprising and positive finding. In practical applications, if we prioritize the F1 score, we can afford to increase $\lambda$ without worrying excessively about allocating too high a weight to latency. Conversely, when the importance of latency is a consideration, it is prudent to keep $\lambda$ very low to ensure more effective decision-making.

*3) Baselines comparisons:* We assigned equal weights to the F1 score and latency to demonstrate how FedVisual compares with two common baselines: random $K$-client selection [6] and all-in selection [30]. Fig. 10 displays the reward convergence graphs of FedVisual and the two baselines in three different types of road segments with their respective data. It is evident that in the FL model, engaging all CVs in the model integration process every round does not yield better results compared to selectively involving only a subset of CVs. Meanwhile, involving all clients in communication unnecessarily increases the communication burden. This observation suggests that optimizing the number of participating clients per round could reduce latency without compromising the model's performance. It is evident that, in the segments of highways and main roads, FedVisual, achieves higher rewards compared to the approaches that integrate all clients and randomly select $K$ clients for model training and integration. For crossings, however, the situation is reversed. FedVisual does not outperform the other two approaches. Crossings are inherently more complex than main roads and highways, resulting in higher data complexity. This increased complexity may limit the advantages of model aggregation, making it harder to achieve significant improvements over the other approaches.

*4) Communication resource comparison:* We have configured three reward function settings in our models based on the value of $\lambda$. Specifically, when $\lambda$ is set to 0.5, allocating equal weight to F1 and latency, the model is labeled as FedVisual-B; when $\lambda$ is increased to 0.7, prioritizing the F1 score, the model is referred to as FedVisual-F; and when $\lambda$ is reduced to 0.3, emphasizing the importance of latency, we label the model as FedVisual-L. We also include two baselines, the traditional random client selection methods and selecting all clients, to benchmark the performance of our proposed approach. As shown in Fig. 11, all those three FedVisual strategies and two baselines are compared under three communication resource scenarios characterized by different communication rounds: 25, 50, and 75 rounds.

We observe that the F1 scores for all five strategies tend to increase as the number of communication rounds increases for each road category, which aligns perfectly with the practical realities of FL. In FL, clients continuously refine their models and improve accuracy through ongoing communications.

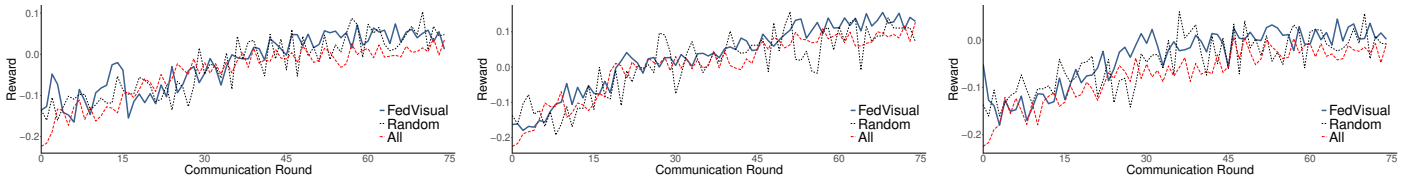



Fig. 10: Reward comparison of FedVisual with two baselines with $\lambda = 0.5$ under different scenarios: a) crossings, b) highways, and c) main roads.

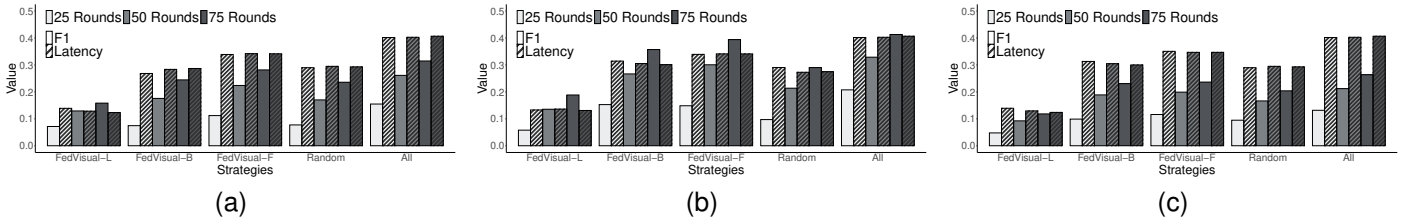



Fig. 11: Latency and F1 metrics of different strategies with different communication rounds under different scenarios: a) crossings, b) highways, and c) main roads.

However, the rate of increasing of F1 score decreases when communication rounds increases. This indicates the need for strategically planning the number of communications to optimize resource allocation across various aspects.

Regarding the latency metric, an interesting observation is that the latency does not significantly vary within each strategy under different communication rounds, which reflects real-world scenarios. However, comparing latency across different strategies reveals substantial differences. For each road category, the strategy of integrating all client's models' updates for model aggregation (marked with strategy "All" in Fig. 11) increases latency, whereas strategies reducing the number of clients participating in each communication round decrease latency. Take the crossings scenario for illustration, compared to the Random strategy, FedVisual-B reduces aggregation latency by 8%, 3.8%, 2%, and improves the F1 score by -3.5%, 3.2%, 3.7% under 25, 50, 75 communication rounds, respectively. These results indicate that our proposed FedVisual-(B, L, F), which balances the importance of F1 and latency, offers more strategic options for real-world applications.

Finally, we find that for relatively straightforward conditions such as highways, the overall F1 scores tend to be higher compared to the more complex conditions of crossings and main roads. This suggests that different communication frequencies should be chosen to optimize communication resources based on the specific road conditions being addressed.

## VII. Conclusion

In this research, we propose FedVisual, a practical model aggregation approach for FL in vehicular visual crowdsensing addressing practical multi-object detection tasks using real-world visual data. FedVisual employs DQN as an experiential learning method to facilitate optimal selection of local models. This approach effectively accommodates the heterogeneity inherent in visual data content and the diversity of vehicle specifications. By employing PCA, the computational overhead is significantly reduced, thereby enhancing the system's overall efficiency. Our experimental results confirm the aggregation efficiency and model performance of the FedVisual within the FL paradigm. The results also validate that FedVisual sustains robust performance across different communication rounds and settings, indicating its practical viability and scalability. Furthermore, analyzing the reward function's strategic allocation of weights to F1 scores and latency provides decision-makers with actionable insights to fine-tune the balance between accuracy and latency. In particular, exploring various strategies highlighted our approach's advantages and limitations under different traffic conditions, underscoring the adaptability of FedVisual and offering decision-makers crucial insights into optimizing communication resources.

Note that, FedVisual currently focuses solely on how the heterogeneity of vehicle visual data affects the efficiency of federated learning aggregation. However, the external environment, such as night-time driving conditions and varying weather conditions, also significantly impacts the visual data in terms of video quality and content richness. Exploring external influences and optimizing model aggregation accordingly is one of our future directions of work.

## References

[1] C. Zhu, G. Pastor, Y. Xiao, and A. Ylajaaski, "Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 58–63, 2018.

[2] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi, "Mobeyes: smart mobs for urban monitoring with a vehicular sensor network," *IEEE Wireless Communications*, vol. 13, no. 5, pp. 52–57, 2006.

[3] C. Zhu, Y.-H. Chiang, Y. Xiao, and Y. Ji, "Flexsensing: A qoi and latency-aware task allocation scheme for vehicle-based visual crowdsourcing via deep q-network," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7625–7637, 2021.

[4] F. Shi, D. Wu, D. I. Arkhipov, Q. Liu, A. C. Regan, and J. A. McCann, "Parkcrowd: Reliable crowdsensing for aggregation and dissemination of parking space information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 11, pp. 4032–4044, 2019.
[5] C. Zhu, A. Mehrabi, Y. Xiao, and Y. Wen, "Crowdparking: Crowdsourcing based parking navigation in autonomous driving era," in *2019 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, 2019, pp. 1401–1405.
[6] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016.
[7] 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.
[8] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.
[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning." [Online]. Available: http://arxiv.org/abs/1312.5602
[10] Y. Han, B. Wang, T. Guan, D. Tian, G. Yang, W. Wei, H. Tang, and J. H. Chuah, "Research on road environmental sense method of intelligent vehicle based on tracking check," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1261–1275, 2023.
[11] V. Coric and M. Gruteser, "Crowdsensing maps of on-street parking spaces," in *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, 2013, pp. 115–122.
[12] G. Grassi, K. Jamieson, P. Bahl, and G. Pau, "Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, ser. SEC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3132211.3134452
[13] H. Chen, B. Guo, Z. Yu, and Q. Han, "Crowdtracking: Real-time vehicle tracking through mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7570–7583, 2019.
[14] X. Wan, H. Ghazzai, and Y. Massoud, "Mobile crowdsourcing for intelligent transportation systems: Real-time navigation in urban areas," *IEEE Access*, vol. 7, pp. 136 995–137 009, 2019.
[15] Y. Zhu, S. Zhang, Y. Liu, D. Niyato, and J. J. Yu, "Robust federated learning approach for travel mode identification from non-iid gps trajectories," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 2020, pp. 585–592.
[16] T. Zeng, O. Semiari, M. Chen, W. Saad, and M. Bennis, "Federated learning on the road autonomous controller design for connected and autonomous vehicles," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 10 407–10 423, 2022.
[17] A. Taïk, Z. Mlika, and S. Cherkaoui, "Clustered vehicular federated learning: Process and optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25 371–25 383, 2022.
[18] H. Yang, K.-Y. Lam, L. Xiao, Z. Xiong, H. Hu, D. Niyato, and H. Vincent Poor, "Lead federated neuromorphic learning for wireless edge artificial intelligence," *Nature communications*, vol. 13, no. 1, p. 4269, 2022.
[19] B. Gong, T. Xing, Z. Liu, W. Xi, and X. Chen, "Towards hierarchical clustered federated learning with model stability on mobile devices," *IEEE Transactions on Mobile Computing*, 2023.
[20] G. Jocher, A. Chaurasia, J. Borovec, NanoCode012, A. Stoken, L. Pauli, Marc, T. Changyu, Laughing, Hogan, D. Almeida, AlexWang1900, P. Rai, and P. Skalski, "Yolov5," May 2020. [Online]. Available: https://github.com/ultralytics/yolov5
[21] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 12 993–13 000.
[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[23] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
[24] J. Chen, H. Wu, P. Yang, F. Lyu, and X. Shen, "Cooperative edge caching with location-based and popular contents for vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 291–10 305, 2020.
[25] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.
[26] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.
[27] C. He, J. Li, W. Liu, J. Peng, and Z. J. Wang, "A low-complexity quantum principal component analysis algorithm," *IEEE transactions on quantum engineering*, vol. 3, pp. 1–13, 2022.
[28] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023. [Online]. Available: https://arxiv.org/abs/2304.02643
[29] G. Jocher, "Yolov5 by ultralytics," 2020. [Online]. Available: https://github.com/ultralytics/yolov5
[30] C. Wu, F. Wu, L. Lyu, Y. Huang, and X. Xie, "Communication-efficient federated learning via knowledge distillation," *Nature communications*, vol. 13, no. 1, p. 2032, 2022.

## BIOGRAPHIES



**Wenjun Zhang** is currently pursuing a Ph.D. degree at the Department of Computer Science, University of Helsinki, Finland. Her research interests focus on Distributed Machine Learning, Artificial Intelligence, and Information Privacy. She is also active in Mobile and Ubiquitous Computing and Environmental Sensing and Monitoring.



**Xiaoli Liu** received the Ph.D. degree in mathematics and statistics from University of Helsinki in 2017. She is Research Coordinator at the Department of Computer Science, University of Helsinki, Finland. Her research interests include data analysis, distributed learning and inference, Internet of Things, and augmented reality.



**Ruoyi Zhang** received the B.S. degree in digital media technology from Hubei Normal University, Huangshi, China in 2016 and the M.S. degree in cyberspace security with the Beijing Institute of Technology, Beijing, China in 2024. He is currently pursuing the Ph.D. degree in computer technology with the Beijing Institute of Technology, Beijing, China. His research interests include machine learning, computer network and smart education.



**Chao Zhu** received the Ph.D. degree from the Department of Computer Science, Aalto University, Espoo, Finland, in 2021. He is currently an Assistant Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China.



**Sasu Tarkoma** (Senior Member, IEEE) is a Professor of Computer Science with University of Helsinki. He is a visiting professor with the 6G Flagship at the University of Oulu. He completed his Ph.D. in Computer Science at the University of Helsinki in 2006. He has authored four textbooks and has published over 500 scientific articles. He holds ten granted U.S. patents. His research interests include Internet technology, distributed systems, 6G, and mobile and ubiquitous computing.