

Asynchronous Federated and Reinforcement Learning for Mobility-Aware Edge Caching in IoVs

Kai Jiang, Yue Cao, *Senior Member, IEEE*, Yujie Song, Huan Zhou, *Member, IEEE*, Shaohua Wan, *Senior Member, IEEE*, and Xu Zhang *Member, IEEE*

Abstract—Edge caching is a promising technology to reduce backhaul strain and content access delay in Internet-of-Vehicles (IoVs). It pre-caches frequently-used contents close to vehicles through intermediate roadside units. Previous edge caching works often assume that content popularity is known in advance or obeys simplified models. However, such assumptions are unrealistic, as content popularity varies with uncertain spatial-temporal traffic demands in IoVs. Federated learning (FL) enables vehicles to predict popular content with distributed training. It preserves the training data remain local, thereby addressing privacy concerns and communication resource shortages. This paper investigates a mobility-aware edge caching strategy by exploiting asynchronous FL and Deep Reinforcement Learning (DRL). We first implement a novel asynchronous FL framework for local updates and global aggregation of Stacked AutoEncoder (SAE) models. Then, utilizing the latent features extracted by the trained SAE model, we adopt a hybrid filtering model for predicting and recommending popular content. Furthermore, we explore intelligent caching decisions after content prediction. Based on the formulated Markov Decision Process (MDP) problem, we propose a DRL-based solution, and adopt neural network-based parameter approximations for the curse of dimensionality in RL. Extensive simulations are conducted based on real-world data trajectory. Especially, our proposed method outperforms FedAvg, LRU, and NoDRL, and the edge hit rate is improved by roughly 6%, 21%, and 15%, respectively, when the cache capacity reaches 350 MB.

Index Terms—Edge caching; federated learning; content prediction; stacked autoencoder; deep reinforcement learning.

I. INTRODUCTION

With the popularity of vehicles on the road and the advancements in V2X technologies, the transportation system

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2022YFE0139300, the Ministry of Education- Industry University Research Innovation Program under Grant No. 2021LDA07005, the National Natural Science Foundation of China under Grant No. 62172255 and 62101437, the Wuhan AI Innovation Program under Grant No. 2022010702040056, and the Fundamental Research Funds for the Central Universities under Grant No. 2042022rc0020. (*Corresponding author: Yue Cao.*)

K. Jiang, Y. Cao, and Y. Song are with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430000, China. (e-mail: kai.jiang, yue.cao, Y. Song@whu.edu.cn).

H. Zhou is with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China. (e-mail: zhouhuan117@gmail.com).

S. Wan is with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China. (e-mail: shaohua.wan@uestc.edu.cn).

X. Zhang is with the Department of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China (e-mail: zhangxu@xaut.edu.cn).

Copyright (c) 20xx IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

has evolved into a data-driven, intelligent era [1]. Internet-of-Vehicles (IoVs) enable vehicles with diverse vehicular services, from multimedia to safety-related applications [2]. Technically, these applications require access to extensive Internet contents, relying on ultra-reliable and low-latency communications for content delivery. The significant surge in delay, unreliability, and redundant traffic have rendered cloud-based processing architectures infeasible due to the limited backhaul capacity and long transmission distance [3]–[5].

Fortunately, large-scale analysis has revealed that different contents often necessitate different priorities, which has prompted the emergence of edge caching in IoVs [6]. Expressly, edge caching is a promising technology to reduce backhaul strain and content access delay. It proactively caches frequently used contents in proximity to vehicles by sinking cloud functions to intermediate Roadside Units (RSUs). Consequently, vehicles can directly retrieve content from caching-enabled RSUs, alleviating redundant transmissions from remote cloud servers [7]. Meanwhile, integrating edge caching in IoVs brings additional technical benefits, such as scalability, privacy protection, and context awareness, derived from its distributed architectures and small-scale nature.

Generally, the finite caching capacity of RSUs makes the system performance heavily rely on well-designed caching strategies. The key to strategy design lies in knowing content popularity, which reflects content access preferences in the content library. Previous works [8], [9] often assume that content popularity is known in advance or obeys simplified models, like *Zipf* distribution and its variants. Nevertheless, such assumptions are unrealistic as content popularity exhibits non-stationary with uncertain spatial-temporal traffic demands in IoVs [10]. Therefore, to facilitate edge cache utilization, predicting content popularity is urgently required.

Recent strides in data analysis and computing power offer opportunities to improve predictions from a data-driven perspective. Based on this, Machine Learning (ML) has exhibited great potential in extracting dynamic features of content popularity by training Vehicular User (VU) data [11]. However, despite the continuous progress achieved in ML, adopting ML for content prediction in IoVs still faces two issues. First, the regional content popularity and validity are constantly changing. As vehicles connected to an RSU move and traverse the cell coverage rapidly, cached content can easily be outdated. An effective prediction method should be context- and mobility-aware to improve cache performance [12]. Second, most ML methods operate on the centralized framework, where vehicles must transmit VU data to an RSU for training.

It is inconsistent with the growing privacy concerns and limited communication resources in data transmission. The VU data usually involves sensitive information, and VUs are reluctant to share their data with others directly [13].

To address these issues, Federated Learning (FL) has attracted widespread concern as a privacy-preserving solution for distributed learning. Coordinated by RSUs, FL enables a shared global model update by aggregating local model parameters from vehicles. Each vehicle conducts training on its local VU data and uploads only model parameters to the RSU in proximity, instead of raw data [14]. Traditional FL operates aggregation synchronously, as the RSU must wait for all local updates from vehicles before aggregation. However, due to vehicle heterogeneity and communication uncertainties, some lagging vehicles may experience delays in local training, becoming "stragglers" [15]. Meanwhile, vehicle mobility will hinder model aggregation, as vehicles might leave the cell coverage of RSU before uploading their local updates. Such unavailability of vehicles inevitably disrupts the timely contribution of certain vehicles to the global model, leading to resource-wasting and diminished global model accuracy [16]. As a remedy, asynchronous aggregation enables the RSU to update its global model immediately for each arrived local update, without waiting for other vehicles to upload. Exploring asynchronous FL becomes imperative to enhance prediction adaptability, mitigate the impact of "stragglers", and ensure up-to-date global models in our work.

In addition, while prediction results offer valuable insights for caching decisions, the diversity of content often results in predicted popular content far surpassing the finite cache capacity of an RSU. To maximize the benefits of content prediction, content replacement mechanisms are indispensable for facilitating edge caching utilization [17]. Therefore, the RSU must determine where to cache the predicted popular contents and whether to replace already cached contents. Yet, intricate content requests and vehicle mobility pose new challenges for caching decisions, effectively addressed by Deep Reinforcement Learning (DRL). With solid cognitive ability, DRL handles dynamic control problems with high-dimensional and time-varying features. It is especially suited for sequential decision-making with long-term objectives under uncertainty [18], [19]. Thus, we can execute caching decision-making in IoVs by exploiting DRL.

Based on the above considerations, this paper investigates the mobility-aware edge caching by asynchronous FL and DRL. The main contributions are summarized as follows:

- 1) We conduct popular content prediction by Stacked AutoEncoder (SAE) models among vehicles and the RSU. The local updates and global aggregation of SAE models follow an asynchronous FL framework. Unlike traditional synchronous FL awaiting all local updates, this framework employs asynchronous aggregation, where the central RSU updates its global model immediately for each arrived local update. Meanwhile, we cope with the incurred stragglers in training and improve model convergence by several interventions.
- 2) Utilizing the latent features extracted by the SAE model, we adopt a hybrid filtering model for predicting popular

content. Based on the content ratings and personal information, this hybrid filtering model combines content-based collaborative filtering and demographic information during runtime. It determines the distance between two contents or two VUs through similarity metrics.

- 3) Furthermore, to maximize the benefits of content prediction, we formulate the caching process in an RSU as a Markov Decision Process (MDP) problem, intending to minimize the long-term content delivery delay in the system. Our proposed solution is based on DRL with adaptive and foresighted considerations, adopting neural network-based parameter approximations to circumvent the curse of dimensionality in RL.

The remainder of the paper is organized as follows. Section II outlines the related work. Section III provides the system model and definition. Section IV proposes a popular content prediction method under the FL framework. Section V explores caching decisions based on DRL to maximize the benefits of content prediction. Moreover, simulation results are elaborated in Section VI. Conclusion is summarized in Section VII.

II. RELATED WORK

A. Distributed Optimization

With the popularity of vehicles on roads and their enhanced computing capabilities, training models directly on these distributed vehicles has drawn increasing attention. Conventional multi-task models [20], however, prove to be unsuitable for edge vehicle training due to their inherent assumption that all vehicles partake in each training round. This assumption becomes unrealistic as vehicles frequently go offline during training for network unreliability or other factors. In light of this, distributed optimization is a critical paradigm to address the complexities of large-scale model training across vehicular networks. It strives to collectively optimize a global objective function while each vehicle performs local computations based on its own data. These local computations are then aggregated to yield a global solution.

Nevertheless, in recent years, various distributed optimization methods [17], [21]–[23] have encountered slow convergence, inefficiency, and straggler issues when deployed to real-world scenarios that involve large-scale and highly heterogeneous data with rich features. For instance, Zhou *et al.* [17] investigated cache replacement and multipoint collaboration by distributed multi-agent RL. Notably, however, this method exhibits exponential space complexity with respect to the number of agents. The computation of the Nash equilibrium significantly dominates the training time. Although they introduced parameter approximation to alleviate the convergence complexity, the convergence pattern remains nonlinear with the number of agents. Jin *et al.* [22] devised a distributed framework with alternating local and central learning by a soft confidence-weighted classifier. However, this asynchronous method assumes locally distributed data following a normal distribution, which imposes limitations on non-convex neural network objectives. Meanwhile, it lacks theoretical convergence assurances and necessitates transmitting a fraction of

local data from each device to the centralized server. Henna *et al.* [23] proposed a collaborative distributed ML method with leveraging edge topological dependencies. While this method ensures high accuracy compared to heuristics, complexity persists in the coordination among edge devices. The wireless bandwidth constraints and faculties exacerbate the coordination and become a bottleneck in the optimization process.

B. FL for Popular Content Prediction

FL, an extension of distributed optimization, offers a promising solution to tackle device diversity in ad-hoc networks. Diverging from traditional distributed optimization, FL places a pronounced emphasis on privacy preservation, facilitating training on sensitive data. Meanwhile, FL is tailored to handle challenges like non-uniform data distribution, communication bottlenecks, and network unreliability.

Recently, FL has acquired noteworthy application significance in edge caching, with several promising endeavours focused on popular content prediction [14], [16], [24]–[27]. Specifically, Yu *et al.* [16] introduced a mobility-aware proactive edge caching strategy using FL, where a context-aware adversarial AE is employed for content popularity prediction. This study significantly improves cache efficiency and eliminates the necessity for centralized data processing. However, coordinating vehicle learning through synchronous aggregation proves unsuitable for highly dynamic vehicular edge networks. Xie *et al.* [24] introduced an asynchronous aggregation for federated optimization involving weighted averaging of the global model. However, they did not consider the practical conditions where edge devices must deal with continuous streaming data. Furthermore, Wang *et al.* [26] focused on employing FL in edge caching to improve data security with consider the potential private data leakage in the trained model. They incorporated gradient clipping and model parameter limitation in model training, and proposed a privacy-preserving method for content popularity prediction by FL and Wasserstein generative adversarial network. Jiang *et al.* [27] proposed an FL-based model integration method that incorporates user context information for efficient user clustering. They employed the distributed approximate Newton algorithm with stochastic variance reduced gradient to learn the global popularity prediction model based on local models. However, this method does not consider user mobility and potential straggler issues during FL training.

C. DRL for Edge Caching Decision-Making

With the revival of artificial intelligence, DRL has been widely used in caching decision-making [6], [9], [18], [28]–[31]. These methods excel in sequential decision-making under uncertainty by acquiring critical attributes, including user historical request information and content popularity.

Specifically, Jiang *et al.* [6] conducted a comprehensive review of DRL-driven edge caching, discussing its implementation and outlook in depth. Zhou *et al.* [9] proposed a novel incentive-driven and DRL-based content caching strategy, intending to reduce the content delivery cost of the

TABLE I
NOTATIONS AND SYMBOLS

Notation	Explanation
\mathcal{N}	The set of all RSUs
\mathcal{F}	The index set of available contents
G_i	The limited cache capacity of RSU i
s_f	The size of each content f
$T_{u,i}$	The transmission delay between vehicle u and RSU i
$T_{i,j}, T_{i,N+1}$	The transmission delay between RSU-RSU and cloud server-RSU
$r_{u,i}$	The wireless downlink data rate between vehicle u and RSU i
$\mathcal{T}_{f,i,j}$	The total delay through different links
$a_{f,i,j}^t$	The caching decision of any RSU i for the requested content f
ξ	The Poisson distribution parameter
V_u	The velocity of vehicle u
μ, σ	The mean and standard deviation of vehicle velocities
P_u^t	The position of vehicle u
F_d	The diameter of cell coverage

content service provider. Considering vehicle mobility and stringent constraints, Tan and Hu [18] embarked on a bold step in designing, analyzing, and optimizing the cooperative coded caching placement at the vehicle and RSU levels in vehicular networks. They configured edge caching problems with a DRL-based multi-timescale framework, and developed mobility-aware reward estimation to relieve spatial complexity. Similarly, Qiao *et al.* [28] harnessed a comparable framework to optimize the content placement and delivery in the vehicular edge networks, with the aid of flexible trilateral cooperations among a macro-cell station, RSUs, and vehicles. Moreover, in [30], Chen *et al.* considered the content dependency, and addressed the collaborative service caching problem in a digital twin-empowered edge architecture by asynchronous advantage actor-critic. In [31], Kirilin *et al.* exploited cache admission and presented a DRL-driven method to determine the admission of requested objects into the CDN cache. In contrast to prior methods that rely on a limited set of criteria, this method weights an extensive array of features, including object size, recency, and frequency of access.

The aforementioned works demonstrate the efficacy of DRL for edge caching decision-making. Nevertheless, their reliance on centralized ways may not always be practical in real-world scenarios due to distributed network topology and privacy concerns.

D. Our Motivation

Inspired by existing studies, it's evident that fewer works have considered both vehicle mobility and privacy concerns when designing adaptive and foresighted caching decision-making. This motivates us to explore a mobility-aware edge caching strategy by exploiting asynchronous FL and DRL. We implement a novel asynchronous FL framework, and

conduct popular content prediction by SAE models among vehicles and the RSU. Compared to existing synchronous FL works, our work employs asynchronous aggregation while coping with straggler issues through several interventions. In addition, to maximize the benefits of content prediction and overall edge cache utilization, our work proposes a DRL-based solution based on the formulated MDP problem. It adopts neural network-based parameter approximations to circumvent the curse of dimensionality in RL. Finally, to verify the effectiveness of the proposed method, extensive simulations are conducted based on real-world trajectory.

III. SYSTEM MODEL

This section presents the edge caching architecture, including network infrastructure, content delivery, and vehicle mobility models. The main notations are listed in **Table I**.

A. Network Infrastructure

We consider an IoV architecture in the highway scenario comprising a cloud server and several small cells, each of which is equipped with a caching-enabled RSU, as illustrated in **Fig. 1**. All these RSUs in a cluster form are deployed equidistantly along the one-way street, and the central RSU is responsible for subsequent FL training. The set of all RSUs is denoted by $\mathcal{N} = \{1, 2, \dots, N\}$. Besides, let $\mathcal{F} = \{1, 2, \dots, F\}$ represent the total available contents, and s_f is the size of each content. The cloud server (denoted by $N + 1$) has abundant cache capacity to cache all contents. Each RSU $i (i \in \mathcal{N})$ is limited to a cache capacity of G_i , which indicates the maximum number of caching contents. Vehicles travel in the single direction along the road and frequently request content via V2X. These vehicles are assumed to be located in at least one cell coverage and will be served by the associated RSU therein. Particularly, each vehicle possesses its unique local dataset, which includes access requests and rating information for contents, as well as context information of VUs (age, gender, *etc.*). Meanwhile, we assume the system operates in multiple time slots (*e.g.*, batching or round) $t \in \{1, \dots, \Gamma\}$. Each time slot allows a vehicle to request a content, and the caching decision of each RSU is updated periodically. Notably, vehicle number in a cell coverage changes in each slot due to the mobility.

Each RSU caches a small portion of contents to eliminate the redundant traffic, ensuring that content requests can be accommodated nearby. Particularly, RSUs can determine where content requests are answered and which local caches should be replaced. Once a vehicle sends an access request within the cell coverage, the local RSU checks its cache space to determine whether the desired content is already cached. If not, the local RSU can retrieve it from adjacent RSUs or directly download it from cloud servers, and deliver it to the vehicle later. Additionally, all RSUs can replace their obsolete content with the popular one in each time slot. This involves a sequential decision-making problem, including efficient decision optimization under constraints.

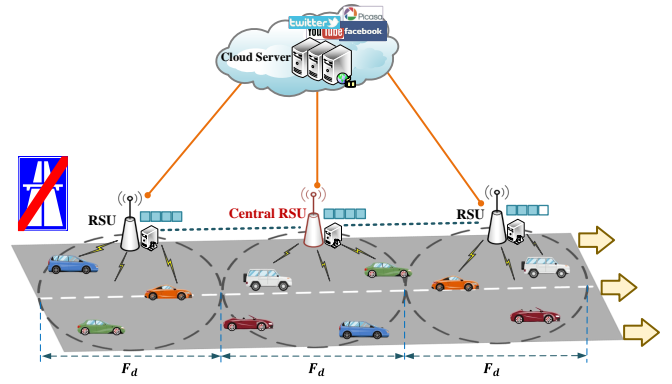


Fig. 1. Edge Caching-empowered IoV Architecture.

B. Content Delivery Model

Transmission delay of content f from any requested vehicle u to its connected RSU i is calculated as $T_{u,i} = s_f/r_{u,i}$, where $r_{u,i}$ represents the wireless downlink data rate between RSU i and vehicle u . Here, multiple channels are considered within each RSU, all with the same bandwidth allocation [17]. Considering the large-scale fading, $r_{u,i}$ is derived as follows:

$$r_{u,i} = B_{u,i} \log_2 \left(1 + \frac{P_i g_{u,i}}{\sigma^2 + \sum_{v \in \mathcal{U} \setminus \{u\}} P_i g_{v,i}} \right), \quad (1)$$

where $B_{u,i}$ is the channel bandwidth, P_i is the transmission power of RSU i , σ^2 and $g_{u,i}$ are the noise power and the channel gain, respectively. Notably, we neglect the transmission delay of sending request identifiers as it involves tiny data size and occurs over high link rates in most instances. Furthermore, as cloud server-RSU and RSU-RSU links are wired optical cables, the corresponding transmission delay $T_{i,N+1}$ and $T_{i,j}$ are set to constant values.

Here, we introduce binary variable $a_{f,i,j}^t \in \{0, 1\}$, $j \in \mathcal{N} \cup \{N + 1\}$ to denote the decision of any RSU i for the requested content f in time slot t . Upon a content request, the content delivery delay is analyzed as follows:

- $a_{f,i,i}^t = 1$ reveals that content f is cached in the RSU i at slot t and can be delivered to the vehicle directly. In this case, the total delay $\mathcal{T}_{f,i,i}$ is just the transmission delay $T_{u,i}$ from the RSU to the vehicle;
- $a_{f,i,j}^t = 1$ ($j \in \mathcal{N} \setminus \{i\}$) reveals that content f is not cached in the RSU i , but at least one of RSUs in the cluster has cached this content, thus the local RSU can inquire and obtain it from the neighbouring RSU j . Then, the total delay $\mathcal{T}_{f,i,j}$ consists of the transmission delay $T_{i,j}$ from the neighbouring RSU j to the local RSU i and the transmission delay $T_{u,i}$;
- $a_{f,i,N+1}^t = 1$ reveals that there is no expected content in all RSUs, and the RSU i has to forward the request identifier to the cloud server for processing at slot t . In this way, the total delay $\mathcal{T}_{f,i,N+1}$ consists of the transmission delay $T_{i,N+1}$ from cloud server to RSU i and the transmission delay $T_{u,i}$.

C. Vehicle Mobility Model

Vehicle mobility is an intrinsic feature in IoVs-related research. For each time slot t , we assume that the number of arrived vehicles in a cell coverage follows the Poisson distribution with parameter ξ . Here, parameter ξ is the average arrival frequency, which reflects the mobility intensity. Notably, due to the tiny slot size, we do not consider switching among cell coverages, and vehicle characteristics (velocity, position) remain unchanged throughout a time slot. This ensures a consistent number of arrived vehicles for each cell coverage in a time slot. Particularly, in the free-flow traffic state, vehicle velocities are assumed to follow independent and identically distributed (i.i.d.) distributions, as drivers can regulate them autonomously. Therefore, we hypothesize that the velocity of vehicle u is derived from a Gaussian distribution, with the probability density function as:

$$f(V_u) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(V_u-\mu)^2}{2\sigma^2}} \quad (2)$$

where V_u is the vehicle velocity; μ and σ is the mean and standard deviation of vehicle velocities, respectively.

Without loss of generality, we assume that vehicle velocities in urban districts are restricted within the upper and lower bounds (*i.e.*, $V_{\min} < V_u < V_{\max}$ for vehicle u). To avoid negative or near-zero velocities, we shall truncate the above Gaussian distribution as:

$$\hat{f}(V_u) = \begin{cases} \frac{f(V_u)}{\int_{V_{\min}}^{V_{\max}} f(s)ds} = \frac{2f(V_u)}{\text{erf}\left(\frac{V_{\max}-\mu}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{V_{\min}-\mu}{\sigma\sqrt{2}}\right)}, & V_{\min} < V_u < V_{\max} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $\text{erf}(x) = \left(\frac{2}{\sqrt{2\pi}}\right) \int_0^x e^{-t^2} dt$ is the Gaussian error function under the mean μ . Compared to the traditional Gaussian distribution or fixed velocity setting, the truncated version is more feasible and aligns better with practical traffic conditions.

Furthermore, we assume that vehicles maintain continuous connections with the RSU within its cell coverage. The connection time can be rather longer if vehicles travel at lower velocities. Here, let P_u^t represent the position of vehicle u in time slot t (*i.e.*, the traversed distance within the coverage of its connected RSU). Hence, for each vehicle u , the connection time in the cell coverage of the current RSU is expressed as:

$$T_{u,conn}^t = (F_d - P_u^t) / V_u^t \quad (4)$$

where F_d is the diameter of cell coverage.

IV. MODEL TRAINING AND CONTENT PREDICTION

This section proposes a popular content prediction method under the FL framework. We employ asynchronous FL for local updates and global aggregation of SAE models. The trained SAE model extracts latent features from VUs and contents. These features are used to construct a hybrid filtering model for recommending popular content.

A. Mobility-Aware Asynchronous FL based Model Training

FL enables collaborative training of SAE models among vehicles and the RSU, and preserves vehicle data remain local.

The schematic overview of model training is shown in **Fig. 2(a)**. The central RSU distributes the global model (*i.e.*, SAE model) to selected vehicles at the beginning of each round. Each vehicle initializes its local model with the received global model and trains it iteratively by local data. Following the local training, vehicles upload their local updates to the central RSU, which then performs asynchronous aggregation to update the global model. The above procedure is iteratively executed in each round until the global model converges or reaches a predefined accuracy threshold. Notably, we cope with the incurred stragglers in FL and improve model convergence by several interventions. Along with considering vehicle mobility, we incorporate regularized local loss functions and dynamic learning rates during local training. Additionally, we balance the previous and the current local gradients using a decay coefficient, and implement representation learning to mitigate the performance impact of asynchronous optimization.

1) *Vehicle Selection and Model Download*: The high-mobility may impose vehicles traversing the cell coverage rapidly, with insufficient time to complete the FL training. Thus, the central RSU should select eligible vehicles for FL training at the start of each round. The selection process considers favorable channel conditions, sufficient local training data, and the remaining connection time within the cell coverage. Thereinto, connection time determines whether a vehicle can participate in FL training and transmit its local update to the central RSU. As displayed in Eq. (4), the connection time of a vehicle largely depends on its position and velocity. Let T_{tra} and T_{inf} be the average training time and inference time in each slot, respectively. For each vehicle u , only if its connection time within the cell coverage exceeds the sum of these two means (*i.e.*, $T_{u,conn} > T_{tra} + T_{inf}$), will it have the opportunity to participate in FL training in the current round. Hereafter, the RSU distributes the global model to selected vehicles, enabling them to improve it through local training.

2) *Training on Vehicles*: For round t , a portion of vehicles is selected to download the global model parameter ω^t from the central RSU. Then, vehicles perform gradient optimizations and refresh their local model parallel based on the local data for multiple iterations. Let \mathcal{I}^t denote the set of selected vehicles, and the local data captured on each vehicle u is \mathcal{H}_u^t , with $h_u^t = |\mathcal{H}_u^t|$ being the number of data samples. The empirical local loss function of vehicle u is defined as:

$$\ell_u(\omega_u^t) = \frac{1}{h_u^t} \sum_{x \in \mathcal{H}_u^t} (x - \hat{x})^2 = \frac{1}{h_u^t} \sum_{x \in \mathcal{H}_u^t} f(\omega_u^t; x) \quad (5)$$

where $f(\omega_u^t; x)$ is the loss function for data point x under local model ω_u^t , and \hat{x} is the input reconstructed by SAE.

Notably, a regularization operation is implemented to ensure algorithm convergence during the local training. The regularized local loss function is defined as:

$$s_i(\omega_u^t) = \ell_u(\omega_u^t) + \frac{\lambda}{2} \|\omega_u^t - \omega^t\|^2 \quad (6)$$

where the second term is the regularization term to avoid the overfitting, with λ is the regularization parameter.

Generally, lagging vehicles (*i.e.*, stragglers) are inevitable in FL training due to statistical heterogeneity and commu-

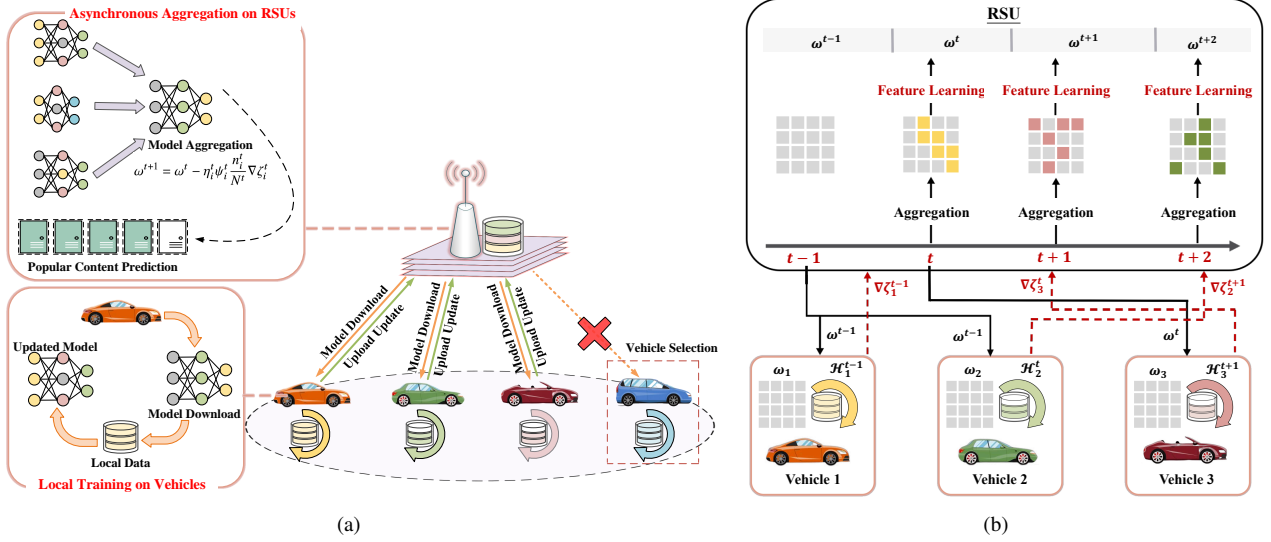


Fig. 2. (a) Mobility-Aware Asynchronous FL based Model Training Process. (b) Asynchronous Aggregation with the Representation Learning.

nication uncertainty. These stragglers may lag in uploading their updates as they have more data or poor communication conditions in the previous round. We define the gradient of the regularized local loss function as "local gradient", and distinguish the local gradient of stragglers in the previous round as "delayed local gradient". Empirical studies have revealed that stragglers negatively affect the accuracy and convergence of the global model. This is attributed to the parameter differences used in calculating the current local gradient compared to the delayed local gradient. To this end, balancing the previous and current local gradients during the aggregation in the current round is essential. That is,

$$\nabla \zeta_u^t = \nabla s(\omega_u^t) + \delta \nabla s_u^{(pre)}. \quad (7)$$

where $\nabla \zeta_u^t$ is the aggregated local gradient of vehicle u , δ is the decay coefficient, $\nabla s(\omega_u^t)$ and $\nabla s_u^{(pre)}$ are local gradient and delayed local gradient, respectively. Notably, $\nabla s_u^{(pre)} \rightarrow 0$ for uploading updates duly in the previous round.

Furthermore, to cope with the stragglers in asynchronous optimization, we incorporate a dynamic local learning rate into the local update. The rationale is that the corresponding learning step size for stragglers should be increased. With η_u^t being the time-related dynamic local learning rate of vehicle u in round t , the closed-form iterative solution for the local model update is defined by:

$$\omega_u^{t+1} = \omega_u^t - \eta_u^t \nabla \zeta_u^t \quad (8)$$

where multiplier $\eta_u^t = \eta_d \max\{1, \log(\bar{d}_u^t)\}$, η_d is an initial learning rate set for all vehicles, and $\bar{d}_u^t = \frac{1}{t} \sum_{\tau=1}^t d_u^\tau$ is the average time cost of the past t rounds. Following that, the actual learning step size is adjusted based on previous delays. By assigning larger step sizes to stragglers, this adaptive scaling effectively mitigates the adverse impact of stragglers on model convergence.

The local iteration process persists until the maximum number of epochs in FL training. After that, vehicle u uploads its local update (i.e., $\eta_u^t \nabla \zeta_u^t$) to the RSU. The global loss

function is formally defined as:

$$\mathcal{L}(\omega^t) = \sum_{u \in \mathcal{I}^t} \frac{n_u^t}{N^t} \ell_u(\omega_u^t) \quad (9)$$

where $N^t = \sum_{u \in \mathcal{I}^t} |\mathcal{H}_u^t|$ is the number of whole data among selected vehicles, $\frac{n_u^t}{N^t}$ is the proportion of local data to all data. Thus, vehicles with more data contribute more to the aggregated parameters, thereby preventing model contamination by vehicles with less data. Our goal of FL is to obtain a global model ω^* with:

$$\omega^* = \arg \min_{\omega^t} \mathcal{L}(\omega^t) \quad (10)$$

3) *Asynchronous Aggregation on the Central RSU*: The global model in central RSU gets updated by aggregating the local update from vehicles in each round. Traditional FL operates synchronously, which leads to resource-wasting as the RSU must wait for all local updates before aggregation. In contrast, we perform asynchronous FL where the central RSU updates its global model immediately for each arrived local update, without waiting for other vehicles to upload. Specifically, for the local update from vehicle u , the global model will be updated by:

$$\begin{aligned} \omega^{t+1} &= \omega^t - \frac{n_u^t}{N^t} (\omega_u^t - \omega_u^{t+1}) \\ &= \omega^t - \frac{n_u^t}{N^t} (\omega_u^t - (\omega_u^t - \eta_u^t \nabla \zeta_u^t)) \\ &= \omega^t - \eta_u^t \frac{n_u^t}{N^t} \nabla \zeta_u^t \end{aligned} \quad (11)$$

Meanwhile, inspired by [16], we recognize the significance of incorporating different weights during model aggregation, with considering vehicle mobility. Consequently, to enhance the global model accuracy and reduce the content transmission delay, the aggregation weight ψ is introduced with the intuition that vehicles with longer available training time should contribute more and be endowed with larger weights. Based on the wireless downlink rate and remaining

traversed distance in the cell coverage, the normalized aggregate weight of vehicle u can be derived as:

$$\psi_u^t = \alpha \frac{(F_d - P_u^t)}{F_d} + (1 - \alpha) \frac{r_{u,i}^t}{\max_{u \in \mathcal{I}^t} (r_{u,i}^t)} \quad (12)$$

where α is the tradeoff factor between wireless downlink rate and remaining traversed distance. As a result, the aggregated global model in Eq. (11) can be rewritten by incorporating the aggregate weight with it:

$$\omega^{t+1} = \omega^t - \eta_u^t \psi_u^t \frac{n_u^t}{N^t} \nabla \zeta_u^t \quad (13)$$

In addition, to mitigate the performance impact of asynchronous optimization, we implement representation learning on the central RSU. This enables us to capture embedding representations for most features, which are learned simultaneously with other parameters in the SAE model. The central RSU executes aggregation immediately for each arrived local update, and performs representation learning on aggregated parameters to derive a cross-vehicle feature representation. The illustration of asynchronous aggregation with representation learning is displayed in **Fig. 2(b)**. Specifically, our approach draws inspiration from attention mechanisms for feature identification and representation. Meanwhile, we incorporate weight normalization to reduce computation costs. We use feature extraction on the first layer after the input, and denote the parameters of this layer as $\omega_{(1)}^{t+1}$. For each element $\omega_{(1)}^{t+1}[a, b]$ in column $\omega_{(1)}^{t+1}[b]$ of $\omega_{(1)}^{t+1}$, the updated $\omega_{(1)}^{t+1}$ is obtained by:

$$\kappa_{(1)}^{t+1}[a, b] \leftarrow \frac{\exp\left(\left|\omega_{(1)}^{t+1}[a, b]\right|\right)}{\sum_b \exp\left(\left|\omega_{(1)}^{t+1}[a, b]\right|\right)} \quad (14)$$

$$\omega_{(1)}^{t+1}[a, b] \leftarrow \kappa_{(1)}^{t+1}[a, b] \cdot \omega_{(1)}^{t+1}[a, b] \quad (15)$$

Up to now, the asynchronous FL training during a round is completed. More details of the asynchronous FL are summarized in **Algo. 1**.

B. Convergence Analysis

Considering vehicle selection and asynchronous aggregation, we conduct a convergence analysis of the proposed asynchronous FL. Some definitions and assumptions are first given below:

Definition 1: (*L-smoothness*) The local loss function ℓ exhibits Lipschitz-smoothness with $L > 0$ if for $\forall x_1, x_2$,

$$\ell(x_1) - \ell(x_2) \leq \langle \nabla \ell(x_2), x_1 - x_2 \rangle + \frac{L}{2} \|x_1 - x_2\|^2. \quad (16)$$

Definition 2: (*μ -strongly convexity*) The local loss function ℓ exhibits μ -strongly convexity with $\mu > 0$ if for $\forall x_1, x_2$,

$$\ell(x_1) - \ell(x_2) \geq \langle \nabla \ell(x_2), x_1 - x_2 \rangle + \frac{\mu}{2} \|x_1 - x_2\|^2. \quad (17)$$

Definition 3: (*Bounded gradient dissimilarity*)¹ The aggregated local gradient $\nabla \zeta_u$ is Q -locally dissimilar at ω with

¹To quantify the dissimilarity between vehicles, Definition 3 is defined based on locally non-independent and identically distributed data.

Algorithm 1: Mobility-Aware Popular Content Prediction based on Asynchronous FL

1 Central RSU Execution:

Input: tradeoff factor α , the position and velocity of vehicles, wireless downlink rate for vehicles.

2 **Initialize** $\omega \leftarrow \omega^0$

3 **for** each round t **do**

4 \mathcal{I}^t : the set of selected vehicles

5 **for** each vehicle u **in parallel do**

6 $T_{u,conn}^t = (F_d - P_u^t) / v_u^t$

7 **if** $T_{u,conn}^t > T_{tra} + T_{inf}$ **then**

8 Add vehicle u to \mathcal{I}^t

9 **for** each selected vehicle $u \in \mathcal{I}^t$ **in parallel do**

10 Download the current global model ω^t

11 $\omega_u^{t+1} \leftarrow$ **Vehicle Update** (ω^t, u)

12 Update the global model immediately for each arrived local model ω_u^{t+1}

13 Compute aggregation weight ψ_u^t based on Eq. (12)

14 Update the global model with feature learning based on Eq. (13) - (15)

15 **Return** ω^{t+1}

16 Vehicle Execution:

Input: global model ω^t , regularization parameter λ , decay coefficient δ , $\nabla \zeta_u^t \leftarrow 0$, learning rate η_d

17 Compute the multiplier $\eta_u^t = \eta_d \max\{1, \log(d_u^t)\}$

18 **Vehicle Update** (ω, u):

19 **for** each local epoch **do**

20 Compute the local loss function based on Eq. (5)

21 Implement regularization based on Eq. (6)

22 Compute $\nabla \zeta_u^t$ based on Eq. (7)

23 Update local model ω_u^t based on Eq. (8)

24 **Terminate** until the maximum number of epochs

25 **Return** ω_u^{t+1}

26 Upload ω_u^{t+1} to the RSU for asynchronous aggregation

$Q > 0$ if

$$\mathbb{E} \|\nabla \zeta_k(\omega)\|^2 \leq \|\nabla \mathcal{L}(\omega)\|^2 Q^2. \quad (18)$$

Assumption 1: The expected squared norm of the stochastic gradient is uniformly bounded. This means that,

1.1 The global loss function $\mathcal{L}(\omega)$ has a lower bound, i.e., $\forall \mathcal{L}(\omega^*) = \mathcal{L}_{\min} > -\infty$,

1.2 There exists $\epsilon > 0$ such that $\mathbb{E} \|\nabla \zeta_k(\omega)\| \leq \|\nabla \mathcal{L}(\omega)\|$, and $\nabla \mathcal{L}(\omega)^\top \mathbb{E} \|\nabla \zeta_k(\omega)\| \geq \epsilon \|\nabla \mathcal{L}(\omega)\|^2$ hold for $\forall \omega$. Notably, $\nabla \zeta_k(\omega)$ is an unbiased estimator of $\nabla \mathcal{L}(\omega)$ when $\epsilon = 1$.

Next, we establish the convergence analysis through lemmas and theorems under **Assumption 1**.

Lemma 1: If the global loss function $\mathcal{L}(\omega)$ exhibits μ -strongly convexity, then with Assumption 1, we can obtain:

$$2\mu (\mathcal{L}(\omega^t) - \mathcal{L}(\omega^*)) \leq \|\nabla \mathcal{L}(\omega^t)\|^2. \quad (19)$$

Theorem 1: Assuming the global loss function $\mathcal{L}(\omega)$ demonstrates Lipschitz-smoothness and μ -strong convexity, and the aggregated local gradient $\nabla \zeta_u$ is Q -locally dissim-

ilar, with the satisfaction of Assumption 1. Then, let $\eta_u^t \geq \psi_u^t \frac{2\epsilon n_u^t}{LQ^2 N^t} \geq \eta_d$, the proposed asynchronous FL will converge to the global optimum ω^* after T^{max} update rounds. That is,

$$\mathbb{E} \left[\mathcal{L} \left(\omega^{T^{max}} \right) - \mathcal{L} \left(\omega^* \right) \right] \leq (1 - 2\mu\eta_d\kappa')^{T^{max}} \left[\mathcal{L} \left(\omega^0 \right) - \mathcal{L} \left(\omega^* \right) \right], \quad (20)$$

where $\kappa' = \epsilon - \frac{\eta_u^t LQ^2}{2}$.

Theorem 2: Assuming the global loss function $\mathcal{L}(\omega)$ demonstrates Lipschitz-smoothness and non-convexity, and the aggregated local gradient $\nabla\zeta_u$ is Q -locally dissimilar, with the satisfaction of Assumption 1. Then, if $\eta_u^t \leq \frac{2\epsilon-1}{LQ^2}$ holds for each round t , Eq.(21) is obtain after T^{max} update rounds:

$$\sum_{t=0}^{T^{max}-1} \frac{\eta_u^t}{2} \mathbb{E} \left[\left\| \nabla \mathcal{L} \left(\omega^0 \right) \right\|^2 \right] \leq \mathcal{L} \left(\omega^0 \right) - \mathcal{L} \left(\omega^* \right). \quad (21)$$

Overall, **Theorem 1** ensures convergence, specifically for the case of a convex global loss function, and provides an error bound for the general model aggregation form. **Theorem 2** indicates that the model convergence rate of the proposed asynchronous FL is governed by balancing the bounded dissimilarity value Q and learning rate η_u^t . The theoretical proofs for Lemma 1 and Theorem 1-2 are available in ref. [15], [24], [32]. Interested readers may refer to these sources for further details.

C. Hybrid Filter based Popular Content Prediction

Considering the sparseness of content requests, we adopt a hybrid filtering model based on SAE to predict the popular content. The trained SAE model is adopted to extract the latent feature representation of contents and VUs. Respectively, based on the extracted feature, the potential associations between the content pair and VU pair (*i.e.*, content similarity and VU similarity) can be measured by cosine distance. Through VU similarity, a VU group is formed by randomly selecting an active VU with its k neighbouring VUs. Based on the content similarity of their historical requests, a content recommendation list is generated for RSU to facilitate popular content prediction. The hybrid filtering model combines content-based collaborative filtering and demographic information throughout this process. It relies on similarity metrics to determine the distance between two contents or two VUs based on their contents ratings and personal information. **Fig. 3** illustrates the content prediction process for a vehicle within the cell coverage, consisting of the following procedures:

1) *Data Preprocessing*: Each vehicle generates a rating matrix \mathbb{F} from its local data based on the historical requests of VUs. The rows and columns of this matrix are VUs' ID and their ratings for different contents, respectively. Similarly, a VU information matrix is generated based on the personal information of VUs, where the rows and columns are user IDs and their corresponding personal information, respectively.

2) *Exploring Latent Representations*: Vehicles put the rating matrix as the input for the SAE to explore hidden features among VUs and contents. Along with the user information matrix, these features are used to compute similarity matrices for VUs and contents. Here, we adopt cosine

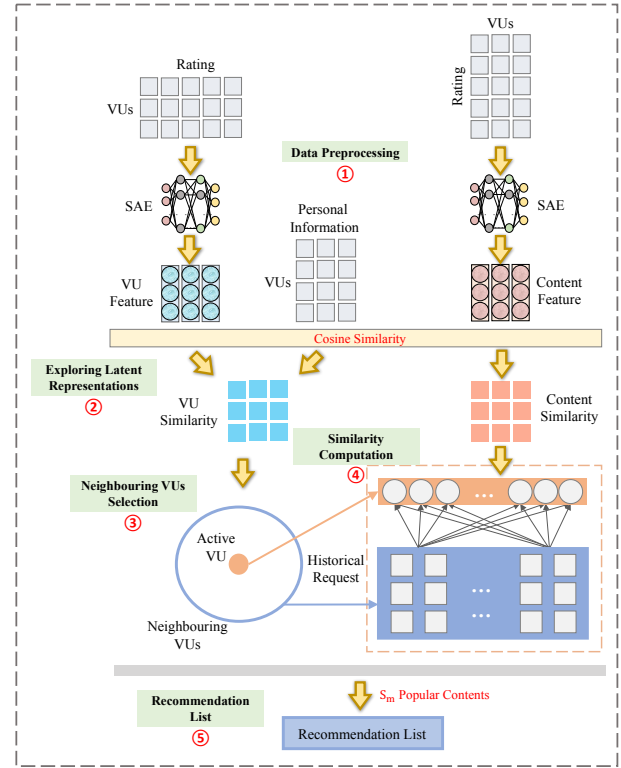


Fig. 3. Popular Content Prediction Process for a Vehicle.

similarity as the metric for similarity due to its effectiveness in sparse matrices. Cosine similarity distinguishes directional differences while being insensitive to absolute value changes, making it well-suited for feature differences measure. Indeed, VU similarity indicates content preference overlap among VUs, while content similarity represents the level of user interest in different contents.

3) *Neighbouring VUs Selection*: For simplicity, we assume that any current VU is active in this work. Based on the VU similarity matrix, we select the k nearest VUs as the neighbouring VUs of this active VU. Utilizing multiple neighbouring VUs helps balance training errors, thereby improving prediction accuracy. It is worth noting that the growth of k is not unbounded. Further increasing k would be unnecessary once the model converges with sufficient VUs. Next, based on the historical requests, a rating matrix \mathbb{K} is constructed, where the rows and columns represent the IDs of these neighbouring VUs and their ratings for different contents, respectively.

4) *Similarity Computation*: We extract a vector from the rating matrix \mathbb{F} to construct the rating matrix of the active VU. The content similarity matrix allows us to obtain the average similarity between each element in the rating matrix of the active VU and neighbouring VUs.

5) *Recommendation List*: Each vehicle generates a list of the first S_m highest similarity contents. After local training, the list is transmitted to the RSU as a recommendation list. The central RSU aggregates all recommendation lists from vehicles and ranks contents based on their occurrence frequency. The first S_m contents with the highest occurrence frequency are set as predicted popular contents (denoted by set \mathbb{S}).

V. DRL FOR EDGE CACHING DECISION

While the prediction results provide valuable insights for caching decisions, the diversity of content often results in predicted popular content far surpassing the finite cache capacity of an RSU. To efficiently manage the cache space of an RSU, content replacement mechanisms are indispensable in edge caching. Meanwhile, overall cache utilization and effectiveness are prone to underutilization without edge cooperation. Therefore, this section explores intelligent and adaptive caching decisions based on DRL, so as to maximize the benefits of content prediction in IoVs. Notably, to save computing resources, the prediction results generated by the central RSU are directly shared with neighbouring RSUs. From a macroscopic perspective, this makes sense as neighbouring RSUs are in proximity to the central RSU along the same path, and VU preferences are not expected to vary dramatically during a short period.

A. Markov Decision Process

The MDP model is commonly employed to depict almost all sequential decision-making problems. We model each RSU as an agent for the straightforward definition of state transition. The caching process in an RSU is formulated as an MDP problem.

State Space: The state is the immediate environment, including relevant information necessary for decision-making. Under the agent setting, the state is represented as $\mathbf{z}_i^t = \{\mathbf{a}_{\mathcal{F},i,i}^t, \mathbf{q}_{i,\mathcal{F}}^t\}$. The former set $\mathbf{a}_{\mathcal{F},i,i}^t$ is the current caching state respecting to the contents in RSU i . The latter set $\mathbf{q}_{i,\mathcal{F}}^t$ is content requests from vehicles. Specifically, $q_{i,f}^t = 1$ means whether at least one vehicle in RSU i requests for content f .

Action Space: After receiving a state in each slot, the agent is required to make a caching decision. This process begins by sorting the caching contents in RSU i in descending order of popularity, relying on the prediction results provided by the central RSU. With the current state \mathbf{z}_i^t , the action \mathbf{d}_i^t involves two parts, $\mathbf{a}_{\mathcal{F},i,j}^t$ and \mathbf{c}_i^t , where matrix $\mathbf{a}_{\mathcal{F},i,j}^t$ encodes the decision of RSU i for the requested contents, and decision variable \mathbf{c}_i^t indicates the current content replacement control in RSU i . Specifically, if $c_i^t = 1$, the RSU will randomly select $\frac{1}{2}G_i$ contents from those in prediction results \mathbb{S}_t but not currently cached at the edge. These selected contents will replace the least popular contents in its local cache.

Reward Function: The reward value serves as a metric for evaluating the action quality. Our objective is to minimize content delivery delay, in contrast to the goal of maximizing cumulative discounted reward for agents. To align the reward function with our optimization objective, we define the reward function as $r(\mathbf{z}_i^t, \mathbf{d}_i^t) = e^{-\sum_f^F \sum_i^N \sum_j^{N+1} q_{i,f}^t a_{f,i,j}^t \mathcal{T}_{f,i,j}^t}$, where a negative exponential function is adopted to transform the objective legitimately. Notably, punitive negatives will be incorporated into the reward for violating constraint conditions.

B. Value Function Approximation

In general, MDP problems can be tackled through linear or dynamic programmings. However, these methods may not

always be applicable in dynamic edge caching settings due to their quasi-static and myopic models, which yields unsatisfactory performance [33].

Therefore, we perform the decision via a function approximation structure, estimating the expected future value of executing an action. This approximate structure specifically emphasizes the agent. Each agent confronts the control and sequential decision-making under uncertainty. In time slot t , the agent observes the current environment state as z_t , chooses action d^t from the allowable action space based on its policy π , where π is a mapping from the current state to the corresponding action. Subsequently, the agent transitions to a new state z_{t+1} using the engineered transition function $p(z_{t+1} | z_t, d_t)$. Finally, the agent receives an immediate reward $r_t = r(z_t, d_t)$ and anticipates the reward as it progresses.

Considering the long-term influences, the recursive state value function $V^\pi(z_t)$ is defined under the policy $\pi(z_t)$. Specifically, it maps state z_t to the expectation of cumulative discounted reward value $U_t = \sum_{t=0}^{\Gamma} \gamma^t r_t$, where discount factor $\gamma \in (0, 1)$ indicates the importance of the predicted future rewards. By applying the Bellman Equation, the state value function is converted to the temporal difference form:

$$V^\pi(z_t) = \mathbb{E}_\pi \left[\left(r(z_t, d_t) + \sum_{t=1}^{\Gamma} \gamma^t r_t \right) \mid z = z_{t+1} \right] \\ = \left[r(z_t, d_t) + \gamma \sum_{z_{t+1}} p(z_{t+1} | z_t, d_t) V^{\pi^*}(z_{t+1}) \mid z = z_t \right], \quad (22)$$

where \mathbb{E} represents the expectation.

In conjunction with the process above, the agent aims to formulate an optimal policy $\pi_t^* \rightarrow d_t^*$ that maximizes the expectation of the cumulative discounted reward value. Subsequently, the optimization problem transforms into determining an optimal value function $V^{\pi^*}(z_t)$ under constraints, as represented by:

$$V^{\pi^*}(z_t) = \max_{\pi \rightarrow d_t} \left[r(z_t, d_t) + \gamma \sum_{z_{t+1}} p(z_{t+1} | z_t, d_t) V^{\pi^*}(z_{t+1}) \right] \\ s. t. \quad C1 : a_{f,i,j}^t \in \{0, 1\}, \quad \forall f, \forall i, \forall j \\ C2 : a_{f,i,j}^t \leq a_{f,j,j}^t, \quad \forall f, \forall i, \forall j \\ C3 : \sum_{j=1}^{N+1} a_{f,i,j}^t = 1, \quad \forall f, \forall i \\ C4 : a_{f,N+1,N+1}^t = 1, \quad \forall f \\ C5 : \sum_{f=1}^F a_{f,i,i}^t \leq G_i, \quad \forall i. \quad (23)$$

The sequential decision problem is resolved into a series of shorter, tractable time steps through this recursive value function, where actions must be determined at each step. The above constraints are summarized as follows: C1 ensures the integer nature of binary decisions; C2 and C3 are constraints on conflicting decisions, *i.e.*, each content request can only be served by an RSU with related content or the cloud server ultimately; C4 guarantees that cloud server has cached all

contents, and C5 is the cache usage constraint of each RSU.

Explicitly, the control policy π^* that adheres to Eq. (23) is assured to become the optimal policy. The optimal action for state z_t can be easily determined by $d_t^* = \operatorname{argmax}_{d_t} V^\pi(z_t, d_t)$.

C. Bellman Update Process

Next, to optimize the strategy and connect the current state with possible actions, we conduct Bellman updates around the action-value function² $Q(z_t, d_t)$. This function enables us to estimate the optimal state value function $V^{\pi^*}(z_t)$ under the current state, and the relationship between them is $V^{\pi^*}(z_t) = \max_{d_t} Q^\pi(z_t, d_t)$. Therefore, the expected cumulative reward after taking an action d_t is obtained as:

$$\begin{aligned} Q^\pi(z_t, d_t) &= r(z_t, d_t) + \gamma \sum_{z_{t+1}} p(z_{t+1}|z_t, d_t) V^{\pi^*}(z_{t+1}) \\ &= r(z_t, d_t) + \gamma \sum_{z_{t+1}} p(z_{t+1}|z_t, d_t) \max_{d_{t+1}} Q^\pi(z_{t+1}, d_{t+1}). \end{aligned} \quad (24)$$

Hereafter, the agent iteratively approximates the action-value function under the current state and executes an action d_t with the highest Q -value. The core process is value iteration, and the iterative formula for Q -value can be obtained as follows:

$$Q(z_t, d_t)^{new} = Q(z_t, d_t) + \beta \left[r(z_t, d_t) + \gamma \max_{d_{t+1}} Q(z_{t+1}, d_{t+1}) - Q(z_t, d_t) \right], \quad (25)$$

where $\beta \in (0, 1)$ is the learning rate parameter. The Q -value can definitely converge to the optimal value $Q^{\pi^*}(z_t, d_t)$ when an appropriate β is designed. Each agent regards other agents as part of the environment, and engages in repeated interaction to optimize the edge caching configuration gradually.

D. DRL based Edge Caching Decision-Making

Generally, RL can optimize the Bellman update process through its self-learning ability. However, RL methods are essentially tabular-based. The vastly diverse, dynamic, distributed edge caching, and cooperative network topology render the performance of most RL works unsatisfactory in IoVs. The fundamental problem is that RL often faces the curse of dimensionality, attributed to the vast action-state space in dynamic scenarios. Besides, recording Q -values in a large table may lead to lengthy searches and memory issues. Thus, DRL is employed for problems in RL trial-and-error interaction.

During the training process, neural networks anticipate input training samples to be independently distributed for both exploitation and exploration. However, the high correlation among continuous states can lead to non-uniform overestimation in action-value function estimations through on-policy updates. This introduces instability and adverse effects on the actual value. As a remedy, we implement off-policy

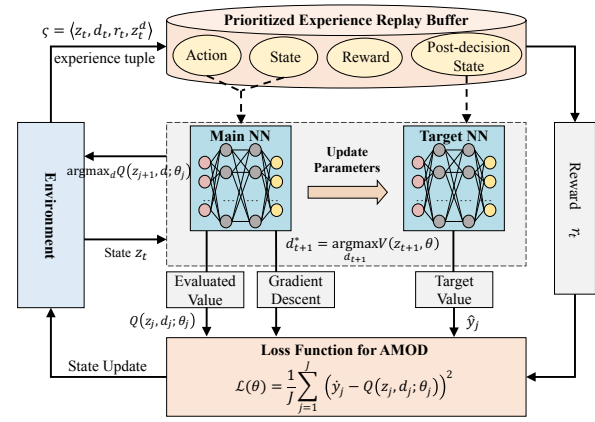


Fig. 4. The Update Process of Action-Value Function.

updates in practice. The basic principle in our proposed DRL method is to separate the policy of behavior and target by different state-action functions. By adopting neural network-based parameter approximations, the agent expects to make an optimal action by solving the sequential optimization in Eq. (25). It estimates the anticipated value of decisions with spatiotemporal dependencies, and approximates the optimal value $Q^*(z_t, d_t)$ with reconstructed Q -function $Q(z_t, d_t; \theta)$. Hereafter, the agent actuates the saved Q -value towards the target value through parameter updates.

Overall, the value update process in our proposed method largely aligns with the Double Deep- Q -Network. Instead of directly conducting bootstrapping, we introduce a target network mechanism to mitigate redundancy. Specifically, two neural networks with identical structures but distinct parameters are maintained. The target neural network focuses on capturing the temporal difference within the one-step return value, while the main neural network assesses the current Q -value. To maintain learning stability, the weight parameters θ of the target neural network undergo periodic updates (spaced a few training steps) using the corresponding θ parameters from the main neural network. The update rule is defined as $\hat{\theta} = \zeta\theta + (1 - \zeta)\hat{\theta}$ with $\zeta \ll 1$. It is worth noting that the one-step return method (i.e., TD(0)) in the target neural network relies solely on the immediate reward value. Then, the target network mechanism is used to generate the one-step target Q -value \hat{y}_t as:

$$\begin{aligned} \hat{y}_t &= r_t + \gamma \max_d Q(z_{t+1}, d; \hat{\theta}_t) \\ &= r_t + \gamma Q \left[z_{t+1}, \operatorname{argmax}_d Q(z_{t+1}, d; \theta_t), \hat{\theta}_t \right]. \end{aligned} \quad (26)$$

Moreover, for ensuring the stability of this neural network-based nonlinear approximation, a prioritized experience replay buffer \mathcal{M} is employed to reuse previous experiences, thereby breaking the correlation in data training. During training sample collection, the agent stores its experience tuple $\varsigma = \langle z_t, d_t, r_t, z_{t+1} \rangle$ into the experience replay buffer, which includes its current state and available action set. The collected experience samples are utilized to train the neural network parameters for value approximation. This entails extracting a minibatch of previous experience samples from the replay

²No distinction is made between "action-value function" and " Q -function".

Algorithm 2: DRL for Caching Decision-Making

Input: agent set \mathcal{N} , discount factor γ , learning rate β , replay buffer \mathcal{M} , minibatch size J , exploration factor ϵ

Output: neural network parameters θ

```

1 Each RSU  $i \in \mathcal{N}$  do
2 Initialize experience replay buffer, main neural
  network with random weight  $\theta$ , target neural network
  with  $\hat{\theta} = \theta$ .
3 for each episode do
4   Set  $t=0$ , obtain the initial state  $z_0$  by randomly
    caching
5   for each slots of episode do
6     Derive an action  $d_t$  based on the  $\epsilon$ -greedy
      strategy in the current state  $z_t$ 
7     Execute action  $d_t$ , observe reward  $r_t$  and the
      next state  $z_{t+1}$ 
8     Store observed experience tuple
       $\varsigma = \langle z_t, d_t, r_t, z_{t+1} \rangle$  into the replay buffer
9     if  $t \% \text{updateFrequency} == 0$  then
10      Randomly extract a minibatch of  $J$ 
        experience samples from the replay buffer
11      for each experience sample  $\{\varsigma_j\}_{j=1}^J$  do
12        Update parameter  $\theta$  by minimizing  $\mathcal{L}(\theta)$  as
        Eq. (27)
13      Update parameter  $\hat{\theta}$  periodically by  $\theta$ 
14      Let  $t \leftarrow t + 1$ 

```

buffer at each iteration, enabling the agent to reinforce its knowledge through replaying.

Since $Q(z_t, d_t; \theta_t) = \hat{y}_t$ is possible with a smaller error, we attempt to optimize parameters for the Q -function approximation. The loss function is interpreted as the Euclidean distance between target value and estimated value. For a minibatch of experience sample $\{\varsigma_j\}_{j=1}^J$, it is converted as:

$$\mathcal{L}(\theta) = \frac{1}{J} \sum_{j=1}^J \left(r_j + \gamma Q(z_{j+1}, \text{argmax}_d Q(z_{j+1}, d; \theta_j), \hat{\theta}_j) - Q(z_j, d_j; \theta_j) \right)^2, \quad (27)$$

where each experience samples $\varsigma_j = \langle z_j, d_j, r_j, z_{j+1} \rangle$ is used to update the parameter θ toward the target value by minimizing loss function $\mathcal{L}(\theta)$, and a gradient guiding updates of θ can be calculated by $\frac{\partial \mathcal{L}(\theta)}{\partial \theta}$.

Importantly, the optimal action is typically constrained within the search space, which depends on the quantity and quality of the training data. To introduce randomness in the action selection, we implement and adapt ϵ -greedy strategy to balance exploitation and exploration, with ϵ as a decreasing parameter. In particular, the agent selects action that maximizes the Q -value with a probability of $1 - \epsilon$ (exploitation), and other action with a small probability of ϵ (exploration).

Figure 4 shows the Q -value update process, and additional details are provided in **Algorithm 2**. We estimate the action-value function for each time slot using parameter approximation. The neural network updates parameters based on the constructed loss function, leading to optimal decisions. This value function approximation is iteratively solved until reaching a finite time horizon within an episode, continuing the process to the next slot. The value iteration concludes when it reaches the specified termination episode, resulting in the stabilization of the desired action-value function.

VI. PERFORMANCE EVALUATION

We consider a one-way highway scenario, with all experiments conducted in Python 3.8 and tested on a processor (AMD® Threadripper™ PRO5995WX@2.70GHz). The system comprises a cloud server and 5 RSUs, where the cell coverage is 500 m. Unless otherwise stated, the vehicle density is 15 vehicles/km, and the initial cache capability of each RSU is 100 contents. **Table II** provides details of other parameters.

The MovieLens 1M dataset [35] is utilized to emulate the content request of VUs. The MovieLens 1M dataset comprises 1,000,209 ratings from 6040 anonymized users for 3952 movies. The ratings range from 0 to 5. Each dataset entry includes indices for both user and movie, alongside the timestamp of the rating. Furthermore, the dataset encompasses pertinent user demographic information (gender, age, occupation, *etc.*). Particularly, we assume that a rating for a movie equates to a content request for that particular content. This conjecture aligns with the rationality that users generally rate movies they have already seen.

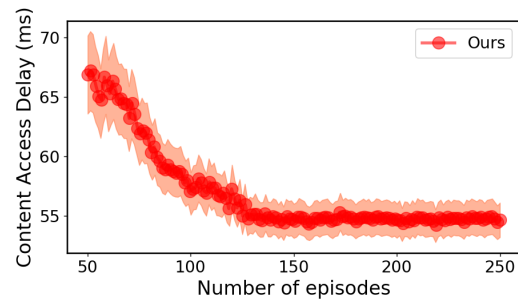


Fig. 5. Content access delay versus episodes.

For performance comparison, the following baselines are introduced:

- 1) **Oracle [36]:** Oracle is an omniscient algorithm that the knowledge of future demands is known in advance. It yields a performance upper bound for other baselines.
- 2) **Least Recently Used (LRU):** LRU replaces the cached content with the longest un-accessed time firstly in RSU.
- 3) **NoDRL:** NoDRL only predicts popular content but also doesn't use DRL for caching decisions. RSUs cache contents randomly from the predicted popular ones.
- 4) **Federated Averaging (FedAvg) [37]:** FedAvg is an synchronous FL method, where the RSU uses a weighted average to update the global model.

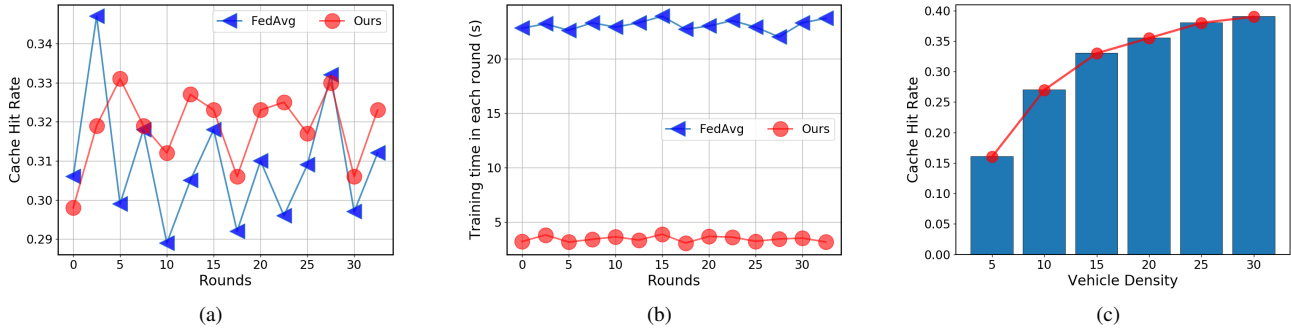


Fig. 6. (a) Content access delay versus episodes. (b) Cache hit rate versus the round in FL. (c) The training time for each round in FL. (d) Cache hit rate versus the vehicle density.

TABLE II
PARAMETERS SETUP

Parameter	Value	Parameter	Value
System Parameters			
V_{\max}, V_{\min}	60km/h, 100km/h	s_f	0.5MB
σ^2	-95dBm	B_i	2MHz
$T_{i,N+1}$	80ms	$T_{i,j}$	15ms
P_i	38dBm		
FL Parameters			
η_d	0.001	T_{tra}	3s
δ	0.001	T_{inf}	0.5s
α	0.4	k	8
λ	0.005		
DRL Parameters			
M	500	J	32
β	3×10^{-4}	ϵ	0.05
γ	0.98		

Quantitative metrics: 1) Cache hit rate: the ratio of requests served by an RSU to the total number of requests. 2) Content access delay: the average delay to access a content.

Fig. 5 presents the convergence curve, indicating a gradual decrease in content access delay, stabilising after about 150 episodes. This shows the learning process of RSU, which can achieve the optimal caching strategy after 150 episodes.

Fig. 6(a) shows the cache hit rate comparison between our proposed method and FedAVG versus rounds. Our proposed method maintains a relatively stable cache hit rate within a small fluctuation range (29.7% to 33.2%) during the 30 rounds. FedAVG exhibits up to 6% fluctuation in cache hit rate over the same period. This indicates that our proposed method slightly outperforms FedAVG regarding model prediction accuracy. This improvement can be attributed not only to the accuracy improvement through the mobility-aware asynchronous aggregation in our method, but also to a series of measures we adopted to cope with the learning instability caused by stragglers. Similarly, Fig. 6(b) compares the training time for each round between our proposed method and FedAVG. It is evident that our proposed method significantly reduces the training time due to asynchronous aggregation. The training time for FedAVG ranges from 22.1 to 23.9 seconds, as it must wait for all local updates from vehicles before aggregation. Besides, Fig. 6(c) illustrates that vehicle density positively impacts the cache hit rate. This is reasonable because as vehicle density increases, the amount of training data and uploaded local updates in the cell coverage also increase,

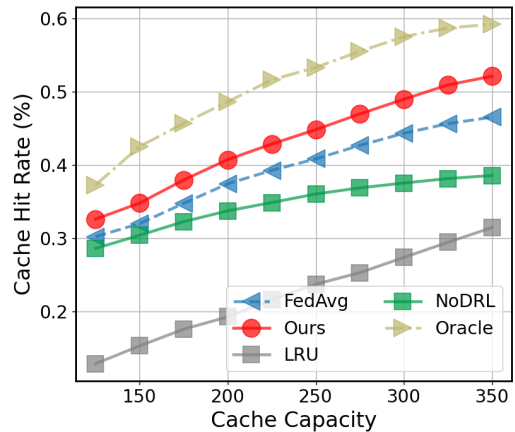


Fig. 7. Cache hit rate versus the cache capacity of RSUs.

further enhancing the model prediction accuracy.

Next, we study the effect of varying cache capacities on different methods, as shown in Fig. 7 and 8. Increasing RSU capacity resulted in higher cache hit rates and lower access delay across all methods. As expected, Oracle achieves the best performance, but its practicality is limited. Among other methods, our proposed method consistently outperforms them in terms of quantitative metrics. For example, when the cache capacity reaches 350 MB, the edge hit rate is improved by roughly 6%, 21%, and 15% compared to FedAvg, LRU, and NoDRL, respectively. At this moment, content replacement processes may infrequently occur in the RSUs.

VII. CONCLUSION

This paper investigated a mobility-aware edge caching strategy by exploiting asynchronous FL and DRL. Firstly, popular content prediction was conducted using the SAE model. We employed an asynchronous FL framework for local updates and global aggregation of SAE models. The trained SAE model extracts latent features from VUs and contents. These features construct a hybrid filtering model for popular content recommendation. Secondly, to maximize the benefits of content prediction, we explored intelligent caching decisions considering content delivery and cache replacement. Based on the formulated MDP problem, we proposed a

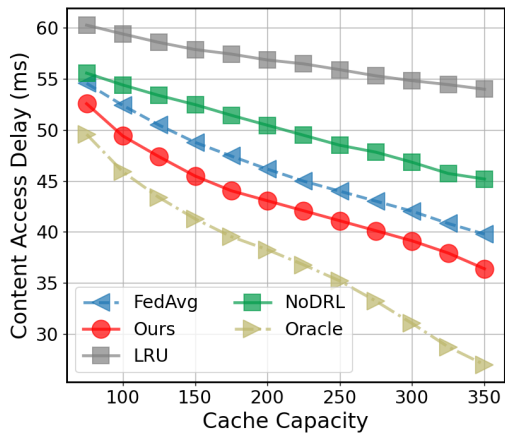


Fig. 8. Content access delay versus the cache capacity of RSUs.

DRL-based edge caching decision-making and adopted neural network-based parameter approximations. Extensive simulations were conducted based on real-world data trajectory.

REFERENCES

[1] P. Arthurs, et al., "A Taxonomy and Survey of Edge Cloud Computing for Intelligent Transportation Systems and Connected Vehicles," *IEEE Trans. Intell. Trans. Syst.*, vol. 23, no. 7, pp. 6206–6221, 2022.

[2] Y. Song, et al., "MP-VRCC: A Multi-Dimension and Priority-Based Vehicle-Road Collaborative Routing Protocol," *IEEE Trans. Veh. Technol.*, vol. 72, no. 8, pp. 10797–10812, 2023.

[3] J. Zhang, and K. B. Letaief, "Mobile Edge Intelligence and Computing for the Internet of Vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, 2020.

[4] P. Wang, et al., "Marginal Value-Based Edge Resource Pricing and Allocation for Deadline-Sensitive Tasks," *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1–10, 2023.

[5] L. Su, and V. K. N. Lau, "Data and Channel-Adaptive Sensor Scheduling for Federated Edge Learning via Over-the-Air Gradient Aggregation," *IEEE Internet of Things J.*, vol. 9, no. 3, pp. 1640–1654, 2022.

[6] K. Jiang, et al., "Intelligence-empowered mobile edge computing: Framework, issues, implementation, and outlook," *IEEE Network*, vol. 35, no. 5, pp. 74–82, 2021.

[7] J. Zhou, et al., "Data Caching Optimization With Fairness in Mobile Edge Computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 1750–1762, 2023.

[8] J. Zhao, et al., "Edge Caching and Computation Management for Real-Time Internet of Vehicles: An Online and Distributed Approach," *IEEE Trans. Intell. Trans. Syst.*, vol. 16, no. 10, pp. 2183–2197, 2021.

[9] H. Zhou, et al., "Incentive-driven Deep Reinforcement Learning for Content Caching and D2D Offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445–2460, 2021.

[10] C. Wang, et al., "Popularity Incentive Caching for Vehicular Named Data Networking," *IEEE Trans. Intell. Trans. Syst.*, vol. 23, no. 4, pp. 3640–3653, 2022.

[11] B. Jedari, et al., "Video Caching, Analytics, and Delivery at the Wireless Edge: A Survey and Future Directions," *IEEE Commun. Surveys & Tutorials*, vol. 23, no. 1, pp. 431–471, 2021.

[12] B. Feng, et al., "Proactive Content Caching Scheme in Urban Vehicular Networks," *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 4165–4180, 2023.

[13] D. Wang, et al., "Secure Long-Range Autonomous Valet Parking: A Reservation Scheme With Three-Factor Authentication and Key Agreement," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3832–3847, 2023.

[14] D. Qiao, et al., "Adaptive Federated Deep Reinforcement Learning for Proactive Content Caching in Edge Computing," *IEEE Trans. Paralle. Distr. Syst.*, vol. 33, no. 12, pp. 4767–4782, 2022.

[15] Y. Chen, et al., "Asynchronous online federated learning for edge devices with non-IID data," in *Proc. IEEE Int. Conf. Big Data*, pp. 15–24, 2020.

[16] Z. Yu, et al., "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Trans. Intell. Trans. Syst.*, vol. 22, no. 8, pp. 5341–5351, 2021.

[17] H. Zhou, et al., "Distributed Multi-Agent Reinforcement Learning for Cooperative Edge Caching in Internet-of-Vehicles," *IEEE Trans. Wire. Commun.*, early access, 2023, doi: 10.1109/TWC.2023.3272348.

[18] L. T. Tan, and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, 2018.

[19] H. Zhang, et al., "Power Control Based on Deep Reinforcement Learning for Spectrum Sharing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 4209–4219, 2020.

[20] J. Shu, et al., "Clustered Federated Multitask Learning on Non-IID Data With Enhanced Privacy," *IEEE Internet of Things J.*, vol. 10, no. 4, pp. 3453–3467, 2023.

[21] V. Smith, et al., "Cocoa: A general framework for communication-efficient distributed optimization," *Journal of Machine Learning Research*, vol. 18, no. 230, pp. 1–49, 2018.

[22] X. Jin, et al., "Collaborating between local and global learning for distributed online multiple tasks," in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, pp. 113–122, 2015.

[23] S. Henna, and A. Davy, "Distributed and Collaborative High-Speed Inference Deep Learning for Mobile Edge with Topological Dependencies," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 821–834, 2022.

[24] C. Xie, et al., "Asynchronous Federated Optimization," in *Proc. NeurIPS Workshop on Optimization for Machine Learning (OPT)*, pp. 1–11, 2020.

[25] L. Zhao, et al., "Towards cooperative caching for vehicular networks with multi-level federated reinforcement learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1–6, 2021.

[26] K. Wang, N. Deng, and X. Li, "An Efficient Content Popularity Prediction of Privacy Preserving Based on Federated Learning and Wasserstein GAN," *IEEE Internet of Things J.*, vol. 10, no. 5, pp. 3786–3798, 2023.

[27] Y. Jiang, et al., "Federated Learning-Based Content Popularity Prediction in Fog Radio Access Networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 3836–3849, 2022.

[28] G. Qiao, et al., "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet of Things J.*, vol. 7, no. 1, pp. 247–257, 2020.

[29] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep Reinforcement Learning-Based Edge Caching in Wireless Networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 48–61, 2020.

[30] L. Chen, et al., "A3C-Based and Dependency-Aware Computation Offloading and Service Caching in Digital Twin Edge Networks," *IEEE Access*, vol. 11, no. 10, pp. 57564–57573, 2023.

[31] V. Kirilin, et al., "RLcache: Learning-based cache admission for content delivery," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2372–2385, 2020.

[32] Q. Wu, et al., "Mobility-Aware Cooperative Caching in Vehicular Edge Computing Based on Asynchronous Federated and Deep Reinforcement Learning," *IEEE J. Sel. Top. Signal Process.*, vol. 17, no. 1, pp. 66–81, 2023.

[33] H. Zhou, et al., "Accelerating Deep Learning Inference via Model Parallelism and Partial Computation Offloading," *IEEE Trans. Paralle. Distr. Syst.*, vol. 34, no. 2, pp. 475–488, 2023.

[34] K. Jiang, et al., "Adaptive Dynamic Programming for Multi-Driver Order Dispatching at Large-Scale," *IEEE Trans. Cogn. Commun. Netw.*, early access, 2023, doi: 10.1109/TCCN.2023.3327578.

[35] F. M. Harper, and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2016.

[36] X. Wang, et al., "Federated Deep Reinforcement Learning for Internet of Things With Decentralized Cooperative Edge Caching," *IEEE Internet of Things J.*, vol. 7, no. 10, pp. 9441–9455, 2020.

[37] J. Konecny, et al., "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv:1610.02527. [Online]*, 2016. Available: <https://arxiv.org/abs/1610.02527>.



Kai Jiang (Student Member, IEEE) is currently pursuing the Ph. D. degree in cyberspace security at Wuhan University, China. His research interests include Deep Reinforcement Learning, Intelligent Transportation, Internet of Vehicles, and Mobile Edge Computing. He was a recipient of the Best Paper Award from the IEEE International Conference on Smart City and Informatization in 2022.



Shaohua Wan (Senior Member, IEEE) received Ph.D. degree from the School of Computer, Wuhan University in 2010. He is currently a full professor with Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China. From 2016 to 2017, he was a visiting professor at the Department of Electrical and Computer Engineering, Technical University of Munich, Germany. His main research interests include deep learning for Internet of Things. He is an author of over 150 peer-reviewed research papers and books, including over 50 IEEE/ACM Transactions papers such as TMC, TII, TITS, TOIT, TNSE, TMM, TGCN, TCSS, TOMM, TETCI, TAES, TALLIP, etc., and many top conference papers in the fields of edge intelligence.



Yue Cao (Senior Member, IEEE) received the Ph.D. degree from the Institute for Communication Systems (ICS) formerly known as Centre for Communication Systems Research, University of Surrey, Guildford, U.K., in 2013. Further to his PhD study, he had conducted a Research Fellow with the University of Surrey, and academic faculty with Northumbria University, U.K., Lancaster University, U.K., and Beihang University, Beijing, China. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, Wuhan,

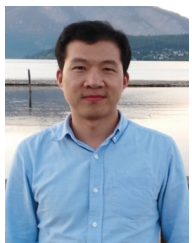
China. His research interests include intelligent transport systems, including E-Mobility, V2X, and edge computing.



Yujie Song is currently pursuing the Ph. D. degree in cyberspace security at Wuhan University, China. His research interests include Internet of Vehicles, Networking Transmission, Space-ground Integrated Information Network, and Trust Management.



Xu Zhang (Member, IEEE) received the B.Eng. degree in telecommunication engineering from Xidian University, Xi'an, China, in 2007, the M.Sc. degree from the University of Science and Technology, Beijing, China, in 2010, and the Ph.D. degree from the University of Surrey, Surrey, U.K., in 2015. She is currently a Lecturer with the Department of Computer Science, Xi'an University of Technology, Xi'an. Her research interests include networking and network management, including aspects such as content delivery networks, information-centric networking, and E-mobility.



Huan Zhou (Member, IEEE) received his Ph. D. degree from the Department of Control Science and Engineering at Zhejiang University. He was a visiting scholar at the Temple University from Nov. 2012 to May, 2013, and a CSC supported postdoc fellow at the University of British Columbia from Nov. 2016 to Nov. 2017. Currently, he is a Professor at the College of Computer, Northwestern Polytechnical University, Xi'an, China. He was a Lead Guest Editor of Pervasive and Mobile Computing, and TPC Chair of EAI BDTA 2020, Local Arrangement Chair

of I-SPAN 2018, and TPC member of IEEE MASS, ICCCN, Globecom, ICC, WCSP, etc. He has published more than 70 research papers in some international journals and conferences, including IEEE JSAC, TPDS, TMC, TWC and so on. His research interests include Mobile Social Networks, VANETs, Opportunistic Mobile Networks, and mobile data offloading. He receives the Best Paper Award of I-SPAN 2014 and I-SPAN 2018.