Mobility-Aware Proactive Edge Caching for Connected Vehicles using Federated Learning

Zhengxin Yu, Jia Hu, Geyong Min, Zhiwei Zhao, Wang Miao, M. Shamim Hossain

***Abstract*—Content Caching at the edge of vehicular networks has been considered as a promising technology to satisfy the increasing demands of computation-intensive and latency-sensitive vehicular applications for intelligent transportation. The existing content caching schemes, when used in vehicular networks, face two distinct challenges: 1) Vehicles connected to an edge server keep moving, making the content popularity varying and hard to predict. 2) Cached content is easily out-of-date since each connected vehicle stays in the area of an edge server for a short duration. To address these challenges, we propose a Mobility-aware Proactive edge Caching scheme based on Federated learning (MPCF). This new scheme enables multiple vehicles to collaboratively learn a global model for predicting content popularity with the private training data distributed on local vehicles. MPCF also employs a Context-aware Adversarial AutoEncoder to predict the highly dynamic content popularity. Besides, MPCF integrates a mobility-aware cache replacement policy, which allows the network edges to add/evict contents in response to the mobility patterns and preferences of vehicles. MPCF can greatly improve cache performance, effectively protect users' privacy and significantly reduce communication costs. Experimental results demonstrate that MPCF outperforms other baseline caching schemes in terms of the cache hit ratio in vehicular edge networks.**

***Index Terms*—Content Caching, Edge Computing, Federated Learning, Deep Learning, Vehicular Networks**

I. Introduction

With the advancement in wireless communications and Internet-of-Things (IoT), self-driving has been considered as a key enabling technology in Intelligent Transportation Systems (ITS) to decrease traffic congestion, improve traffic efficiency and enhance road safety [1]. Self-driving vehicles enable a wide range of applications, from infotainment applications to safety-related applications [2]. These applications may require large computation, communication and storage resources, and have strict performance requirements on network bandwidth and response time. Thus, supporting these applications imposes high pressure on the resource-constrained Vehicular Networks (VNs). Vehicular Edge Computing (VEC) is recognised as a promising paradigm to satisfy the increasing demands by integrating edge computing into VNs [2]. VEC allows data to be processed and stored at edge nodes, such as Roadside Units (RSUs) and Base Stations (BSs).

Caching content at edge nodes enables vehicles to fetch their requested contents within one transmission hop [3]. It is capable of reducing service latency and alleviating backhaul network burden. Due to the limited storage at edge nodes, the caching schemes need to identify and cache the popular contents that are interesting to most vehicular users. Caching schemes can be classified into two categories: reactive caching and proactive caching. Reactive caching utilises the observed users' request pattern to choose contents to be cached [4], such as First-In-First-Out (FIFO), Most Recently Used (MRU), and Least Recently Used (LRU). In reactive caching, contents may only be cached after being requested. Thus, if a content has not been requested before, there is no cached copy of this content. However, the high mobility of vehicles and complex vehicular environments cause highly dynamic content popularity. In this case, the previously requested contents may become obsolete soon, so the reactive caching scheme cannot satisfy strict performance requirements of users. In contrast, proactive caching predicts content popularity and caches predicted popular contents before the arrival of user requests. It can pre-fetch the popular contents, even these contents may have never been requested before. Thus, proactive caching is considered to be more suitable for the VEC scenarios. In proactive caching, Machine Learning (ML) is a powerful approach to predict content popularity for efficient caching. Some works focus on learning-based caching schemes in VNs by utilising reinforcement learning [5], [6], multilayer perceptron and convolutional neural networks [7], etc.

Although some progresses have been achieved in learning-based proactive caching, utilising ML techniques for edge caching in VNs still faces the following three challenges: 1) High mobility: Vehicles send requests to an RSU and go through its coverage area quickly, making the caching content easily to be out of date. To improve the cache performance, the caching scheme should be both context and mobility aware, making cache decisions based on the content popularity predictions and vehicles' mobility. 2) Privacy: Most ML algorithms train models in a centralised manner where the data generated by multiple vehicles must be sent to an edge server for analysis. These generated data may involve personal sensitive information used for various vehicular applications. Therefore, uploading and processing these data centrally may raise privacy and security concerns. 3) Scalability: As the number of connected vehicles grows, data generated by the vehicles increase. The centralised ML algorithms may find it difficult to handle such data due to the incurred high

Z. Yu, J. Hu, G. Min, W. Miao are with the Department of Computer Science, College of Engineering Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, U.K. Email: {zy246, J.Hu, G.Min, Wang.Miao}@exeter.ac.uk

Z. Zhao is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, 610051, China. Email: zzw@uestc.edu.cn

M. S. Hossain is with the Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia. E-mail: mshossain@ksu.edu.sa

Corresponding authors: Jia Hu and Geyong Min

computation and communication costs.

To deal with the above-mentioned challenges, a Mobility-aware Proactive Edge Caching Scheme based on Federated Learning (MPCF) is proposed. MPCF utilises Context-aware Adversarial AutoEncoder (C-AAE) to predict the content popularity and then caches the predicted popular contents in RSUs. Our proposed proactive caching scheme is based on the Federated Learning (FL) framework [8]. In the designed scheme, vehicles collect and store data for local training. A global model (*i.e.*, C-AAE) is updated at RSU by aggregating the locally trained models. Moreover, a mobility-aware cache replacement policy is developed to dynamically update cached contents according to the mobility and position information of vehicles.

The main contributions of this paper are summarized as follows.

1) We propose a mobility-aware federated learning scheme for edge caching in VNs, which can protect users' privacy, reduce communication costs, and support high mobility of vehicles. This new scheme includes four main components: content popularity prediction, vehicle selection, model aggregation, and cache replacement.
2) We utilise the C-AAE model to predict the popularity of contents, which adds the adversarial network to the AutoEnocoder (AE) architecture by turning an AE into a generative model. It helps to learn deep latent representations from users' historical requests and contextual information, and obtain implicit relationships between users and contents for improving prediction accuracy.
3) We design mobility-aware vehicle selection, model aggregation, and cache replacement policies with the aim of optimising the caching resource utilisation in VNs. Especially, the decision for selecting vehicles to participate in the FL training process and the value of weights for parameter aggregation depend on the positions and resources of connected vehicles. It can ensure that vehicles have enough time for training and the RSU can aggregate high-quality updated models. Meanwhile, the cache replacement policy dynamically updates the contents at RSUs in response to the preferences of their connected vehicles and predictions of content popularity.

The rest of this paper is organised as follows: Section II reviews the related work. The system architecture of the proposed cache scheme is presented in Section III. Section IV describes the detailed implementation of the MPCF. The performance evaluation and analysis of MPCF are provided in Section V. Section VI concludes this paper.

II. Related Work

Vehicle networks have drawn much research attention [9], [10], [11] with a wide range of works on resource allocation, routing, security, etc. Cheng *et al*. [9] presented a concept IoVB-net and a routing method for the Internet of Vehicles (IoV), which can realise interconnections among nodes. This solution will promote the deployment of the IoV in an urban scene and open a door to other related researches such as routing strategy, security and privacy. Zhang *et al*. [10] creatively proposed the Chinese remainder theorem based conditional privacy-preserving authentication protocol, which only needs realistic TPDs and greatly reduces the computational complexity. This will greatly promote the application of the authentication protocol in VNs.

Edge servers (*e.g.*, BSs and RSUs) and vehicular users with ample caching capacities can cache popular contents to increase the agility for service provisioning. A number of recent works [12], [13], [14], [15], [16] have been reported to investigate the content caching at RSUs in vehicular networks. Hu *et al*. [12] proposed a multi-object auction-based method to solve the competition of content providers caused by the limited storage resources of RSUs. Ding *et al*. [13] studied three methods (optimal, sub-optimal and greedy) to allocate contents on RSUs, aiming to minimise the average downloading time for requested contents. Su *et al*. [14] designed a cross-entropy based dynamic content caching scheme to optimise cache resources by utilising cooperation among RSUs and the request history of vehicles. High movement of vehicles results in unstable connectivity, and vehicles may not have enough time to download the entire requested content during the time staying in the area of one edge node. Thus, a mobility-aware probabilistic caching scheme was applied in [15] by considering the vehicle trajectories and content service time at the edge node. Mahmood *et al*. [16] developed a probabilistic caching scheme to store content chunks at edge nodes by considering both the historical statistics of achievable data rates and the time of vehicles staying in the area of edge nodes.

Connected vehicles, equipped with storage resources, can be exploited as caching entities to cache the popular contents locally, which brings the benefits of utilising their own resources. Kumar *et al*. [17] presented a peer-to-peer cooperative caching scheme for data dissemination that leverages a Markov chain model to share information among multiple vehicles and uses a probabilistic method to update the existing data. Fang *et al*. [18] provided a cooperative caching scheme for cluster-based VNs, which considers both the caching resource-constrained vehicles and caching status of vehicular clusters in a global view. Deng *et al*. [19] presented a distributed probabilistic caching scheme to make caching decisions. In this scheme, users' content requests, the importance of vehicles and their movement characteristics are all taken into consideration. A cooperative caching scheme was designed in [20]. It is based on the mobility prediction that estimates the probability of vehicles visiting hot spot areas. In order to minimise users' delay, [21] investigated a caching placement in both vehicular and RSU layers. Zhang *et al*. [22] developed a mobility-aware cooperative caching framework, where vehicles are as caching nodes to share contents tasks with BSs. Park *et al*. [23] proposed a distributed proactive caching scheme in VNs by distributing contents for RSUs, based on the movement of vehicles. Ainagar *et al*. [24] introduced a mobility-aware proactive caching scheme by taking the effect of the vehicle velocity into account.

Recent advances in ML have attracted extensive research interests with fruitful outcomes [25], [26], [27], [28]. ML has

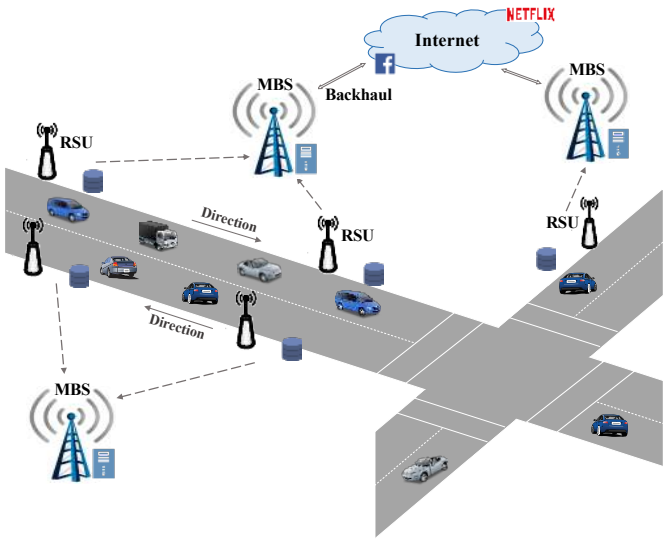


Fig. 1. Proactive edge caching for connected vehicles

also been widely used in content caching. Ndikumana *et al.* [7] studied a deep learning based proactive caching scheme by adopting a Multi-Layer Perceptron (MLP) approach to predict the popularity of contents within the coverage area of mobile edge computing (MEC) servers, and exploiting a Conventional Neural Network (CNN) to estimate the age and gender of passengers. Comparing the MLP's outputs with CNN's outputs, the contents which need to be downloaded from MEC servers to vehicles can be decided. MEC is one of the prospective technologies to enhance the performance of 5G. It brings the cloud computing and caching capabilities to network edges (*e.g.*, base stations, access points), thus various tasks can be executed at the edge rather than remote clouds. A Q-learning based proactive caching scheme was devised in [6] with the support of a long short-term memory neural network. A deep reinforcement learning based content caching scheme was exploited in [29] by optimising the content placement and content delivery to minimise content delivery latency.

Most existing caching schemes for conducting prediction are based on a centralised method that RSUs gather all vehicles' data. However, the contextual information of vehicles in uploading data contains personal sensitive data, which may cause privacy and security issues. Therefore, we propose a mobility-aware proactive edge caching for connected vehicles based on federated learning to improve cache performance as well as protect vehicles' privacy.

III. System Architecture

We consider a vehicular network in an urban scenario, consisting of several MBSs, RSUs and vehicles, as shown in Fig. 1. The MBSs are located in different locations at the edge of VNs. Within the coverage area of an MBS, a set of RSUs $S = \{S_1, S_2, S_3, ..., S_n\}$ are placed equidistantly with distance $\mathfrak{D}$ over both sides of the road, where n is the number of RSUs. Each RSU serves its connected vehicles. These vehicles are denoted by a set $V = \{V_1, V_2, V_3, ..., V_m\}$, where m is the number of connected vehicles. Vehicles traverse the coverage areas of several MBSs. The communication among vehicles, RSUs and MBSs are through wireless links, while MBSs connect to the Internet via a reliable backhaul link. Both MBSs and RSUs are equipped with cache-enabled edge servers. RSUs are used to cache contents likely to be requested by the vehicles nearby. MBSs store the lists of cached contents in connected RSUs and manage their cache resources.

The speeds of vehicles are assumed to be independent and identically distributed, forming a set $U = \{U_1, U_2, U_3, ..., U_m\}$. They are generated by a truncated Gaussian distribution. Compared to the normal Gaussian distribution or a fixed speed, the truncated Gaussian distribution is more feasible for modelling vehicles' speed because it limits the scope of vehicles' speed to a certain range. This assumption has also been widely used in many state-of-the-art works of vehicular networks [24], [30], [31]. Vehicles keep their assigned speeds invariable during each experiment. There is an entrance RSU on each side of the road. The number of arrived vehicles for entering each entrance during the period t is defined as $V_q(t)$. It follows a Poisson process with the parameter λ:

$$P\left(V_q(t) = g\right) = \frac{(\lambda t)^g}{g!} e^{-\lambda t}, \tag{1}$$

where g equals the number of vehicles generated in a period t.

These entered vehicles are interested in a set of popular contents. In the concerned scenario, each RSU can prefetch up to N contents from the Internet and cache these contents locally. The moving vehicles can connect to an RSU and send content requests to it, when vehicles are located within the area of the RSU. If the requested content is available in the current connected RSU (*i.e.*, a cache hit), the RSU can directly transmit this content to the vehicle. Otherwise, the RSU has to obtain the requested content from the Internet (*i.e.*, a cache miss).

Placing the popular caching contents at RSUs can effectively improve cache performance, which primarily depends on the knowledge of content popularity. The popularity of contents is influenced by many factors, including the contextual information of vehicular users (*e.g.*, age and gender) and the mobility pattern of vehicles. Thus, to enhance the cache performance, predicting the content popularity and deciding which contents to be cached in RSUs need to consider the above information.

To address the above challenge, we design a mobility-aware proactive content caching scheme for connected vehicles using FL. As shown in Fig. 1, it is a three-layered architecture. The bottom layer contains vehicles requesting for contents. The middle layer includes several RSUs equipped with cache-enabled edge servers. The top layer has a cache-enabled MBS. The multiple connected vehicles in an RSU collaboratively train a shared global learning model. The RSU firstly disseminates an initial global model to the connected vehicles. Based on the received model, vehicles utilise their local data to compute an updated model. Next, each vehicle sends the updates of global model back to the RSU. Finally, the RSU aggregates the updates from vehicles and builds an updated global model. The above steps are repeated until a satisfying

global model is achieved. The learning model in this work is specially developed to predict content popularity by learning data representation from the data of local vehicles. We rank all contents by their predicted popularity and select the top N popular contents as caching contents in the RSU. Meanwhile, the list of cached contents in RSU is stored at the MBS.

The federated deep learning model in RSU uses the data from current connected vehicles to predict the content popularity and prefetches their predicted results in the cache. However, the high mobility characteristic of vehicles may result in the following situation: Vehicles send content requests to the current RSU, but trying to fetch the requested contents from another RSU. Furthermore, due to the small coverage area of the RSU, vehicles may not have enough time to download the whole requested content. It may pass several RSUs to obtain the full content. Therefore, a mobility-aware cache replacement policy is developed to address these issues. Based on the prior knowledge of vehicle trajectories, predicted content popularity and lists of cached contents in MBS, the MBS dynamically updates the caching contents of each RSU. This policy enables the predicted contents of vehicles to serve themselves. When the vehicle is going to leave the current RSU and enter to the next RSU, the MBS will cache the popular contents for the vehicle in the neighbour RSU that it will enter. Due to the similar locations of these vehicles, cached contents in RSUs have less geographical features. Thus, the predicted popular contents for each RSU may not vary much so only a small number of contents need to be replaced during this cache replacement.

IV. Mobility-aware Federated Deep Learning For Edge Caching

This section elaborates on our proposed caching scheme. We first describe the mobility-aware federated deep learning framework which includes the connected vehicle selection, federated training process and weighted aggregation method. Then, we introduce the context-aware adversarial autoencoders based method to predict the popularity of contents. Finally, with the prediction of the content popularity and a coarse knowledge of vehicle trajectories, we explore a mobility-aware replacement cache policy. Table I lists the definition of notations in the MPCF.

A. Mobility-aware Federated Deep Learning

FL facilitates collaborative training of a deep neural network model among vehicles under the orchestration of a server in RSU by keeping the training data on vehicles. It significantly mitigates the privacy risk of vehicles and largely reduces communication costs, resulting from centralised ML [8]. FL is performed by multiple communication rounds (iterations). Based on the speed and position of vehicles, K vehicles are selected at each communication round to conduct model training, as shown in Fig. 2. The K vehicles are indexed by k. Then, each vehicle receives a global model from the RSU and trains this model from its local data. Following the local training of vehicles, the updated weights and gradients are sent back to the RSU. The RSU aggregates the collected models

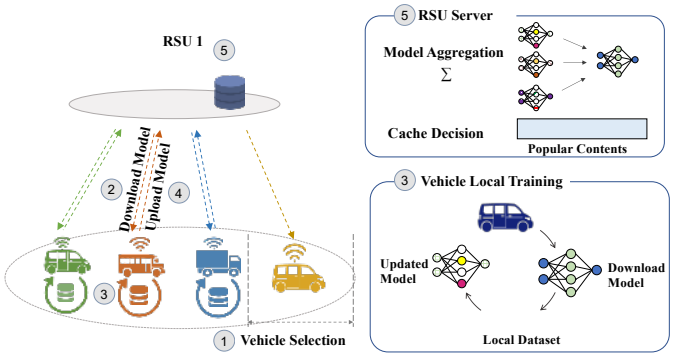


Fig. 2. Mobility-aware Federated Deep Learning

from vehicles to construct an updated global model. Finally, according to the predicted content popularity by the updated global model and the coarse knowledge of vehicle trajectories, the caching replacement strategy will decide where and which contents to be cached.

The details of our designed FL communication round consists of the following steps:

1) Vehicle Selection: Due to the small coverage area of an RSU, some vehicles with high-mobility may go though quickly and cannot finish the FL training. It leads to train an inefficient model and deteriorates the cache performance [32]. Aggregating high-quality updated models of vehicles on the RSU server can construct a more accurate global model. Thus, we design a mobility-aware vehicle selection method to cope with high-mobility training environment. Only the RSU located at the road entrance is chosen to execute the FL training. A set of its connected vehicles are selected as nodes performing computation on their local data to update the global model. The vehicle selecting process will consider the factors of good channel condition, unmetered wi-fi stable connectivity, sufficient local training data and a long standing time in the current RSU's coverage area [25]. Sufficient local training data guarantees to train a high-quality model. The standing time is the driving time of the vehicle staying in the area of RSU. It largely depends on the position and speed of connected vehicles. The long standing time in the coverage area promises that the training process can be completed and its results can be delivered.

U_k denotes the speed of the k-th connected, which is a constant with the lower and upper bounds ($U_{min} \leq U_k \leq U_{max}$) in the urban area. We suppose that U_k follows a truncated Gaussian distribution [24]:

$$f(U_k) = \begin{cases} \frac{e^{-\frac{1}{2\sigma^2}(U_k-\mu)^2}}{\sqrt{2\pi\sigma^2}\left(erf\left(\frac{U_{max}-\mu}{\sigma\sqrt{2}}\right)-erf\left(\frac{U_{min}-\mu}{\sigma\sqrt{2}}\right)\right)}, \\ \qquad\qquad U_{min} \leq U_k \leq U_{max}, \\ 0, \qquad\qquad \text{otherwise}, \end{cases} \tag{2}$$

where σ^2 is the variance and μ ($-\infty < \mu < \infty$) is the mean, $erf()$ is the Gauss error function, and P_k is the position of the k-th vehicle that represents the distance to the entrance. The diameter of coverage area of RSU s is $\mathfrak{D}$. Thus, for each vehicle, the standing time in the coverage area of current RSU

TABLE I
NOTATIONS DEFINITION

Notation	Definition
S	Set of RSUs
n	Number of RSUs
V	Set of connected vehicles for the RSU
m	Number of connected vehicles
U	Set of vehicles' speed
N	Maximum number of caching contents in RSU
λ	Rate parameter of Poisson process
K	Set of selected vehicles for FL training
k	Index of selected vehicles for FL training
H	Set of datasets stored in selected vehicles
d_k	Size of the local dataset in the connected vehicle k
d	Size of datasets among connected vehicles
P_k	Position of the connected vehicle k
$\mathfrak{D}$	Diameter of the coverage area of RSU (Distance between RSUs)
$T^k_{standing}$	Standing time of the connected vehicle k
r	Number of FL communication rounds
T_{round}	Average training time for each communication round
T_{inf}	Inference time
w	Parameters of model
w_r	Parameters of model in the r^{th} round
$L_k(w)$	FL loss function of the selected vehicle k
Q	Position of RSU
c	Number of contents
b	Local minibatch size
η	Learning rate
D	Discriminative model
G	Generative model
X	User-by-content request matrix
x	Sample from X
$\hat{a}$	Users' context matrix
$\tilde{X}$	Reconstructed user-by-content request matrix
z	Latent code
z'	Real input with the required distribution
$p(x)$	Model distribution
$p_d(x)$	Data distribution
$p(z)$	Prior distribution
$q(z \mid x)$	Encoding distribution
$p(x \mid z)$	Decoding distribution

is:

$$T^k_{standing} = (\mathfrak{D} - P_k)/U_k \tag{3}$$

We assume that the average training time for each communication round is T_{round} and the inference time is T_{inf}. (They depend on the size of dataset and the deep learning model. In this work, they are sampled from our experiments.) As shown in Fig. 2, step (1), if the $T^k_{standing} > T_{round} + T_{inf}$, the vehicle will meet the requirement for standing time and is chosen for FL training.

2) Model Download: A set of vehicles K_r are selected to participate in FL training for the r-th communication round. The next step in the typical FL training process is that selected vehicles download the global model from the RSU and train this model over their own local data, see Fig. 2, step (2). In self-driving scenario, the MBS has a coarse knowledge of vehicle trajectories. Thus, the MBS allows some vehicles to directly use their own models downloaded from the previous RSU to participate in the currently FL training. The previous connected RSU of these vehicles is located next to the current connected RSU with the same direction. Using the previous model brings the benefits to accelerate the training of a global shared model, as it trains based on a high-quality model and the training time can be greatly saved. In this way, it also considers preferences of the vehicles connected by the previous RSU. They may enter the current connected RSU with high probability. Considering the preferences of future coming vehicles can help to train a mobility-aware model. For other selected vehicles, they perform the typical FL training process. They download the parameters w_r of the global model from the RSU.

3) FL Model Training: The third step in our proposed FL is to train the model by utilising local data at vehicles, as shown in Fig. 2, step (3). Let $H = \{H_1, H_2, H_3, ..., H_k\}$ represent the datasets stored in selected vehicles. H_k represents the local dataset of the k-th vehicle with the length d_k, $d_k = |H_k|$. d is the size of the whole data among the selected vehicles. Similar to the typical FL, the goal of our proposed FL is to minimise the loss function $\ell(w)$:

$$\min_w \ \ell(w) = \sum_{k=1}^{K} \frac{d_k}{d} L_k(w) \quad \text{where} \quad L_k(w) = \frac{1}{n_k} \sum_{j \in H_k} \ell_j(w), \tag{4}$$

where $\ell_j(w)$ is the loss of the prediction on the j-th dataset in H with the parameters of model w. k is the index of total selected vehicles K. $L_k(w)$ represents the local loss function of vehicle k. Minimising the weighted average of local loss function $L_k(w)$ is equal to optimise the loss function $\ell(w)$ of FL.

4) Upload Updated Model: As shown in Fig. 2, step (4), the fourth step is to upload the local model w^k_{t+1} from vehicles to the RSU server. Compared to the computation costs, communication costs dominate in FL [26]. In order to reduce communication costs and save the upload time, the model can be compressed before being uploaded to the RSU, as the uplink speed is slower than the download speed [33].

5) Weighted Aggregation: After vehicles upload their models, the fifth step is to generate the new global model w_{r+1} by computing a weighted sum of all received local models w^k_{r+1}, as shown in Fig. 2, step (5). The new constructed global model is used for the next training round. r denotes the communication rounds in FL. Federated Averaging (FedAVG) algorithm is widely applied in FL. Compared to the typical federated stochastic gradient descent algorithm (FedSGD) [26], it increases local training epochs as well as decreases mini-batch sizes. In FedSGD, each vehicle k utilises its own data to locally compute the average gradient $\nabla L_k(w)$ on its global model w_r. The RSU server then aggregates these computed gradients by taking a weighted average sum and applies the update gradients:

$$w_{r+1} \leftarrow w_r - \eta \sum_{k=1}^{K} \frac{d_k}{d} w^k_{r+1}, \tag{5}$$

where η is the fixed learning rate. Whereas, in FedAVG, each vehicle adds more computation by iterating the local updates $w^k_r \leftarrow w^k_r - \eta \nabla L_k\left(w^k_r\right)$ multiple times before the averaging step in the RSU server. The weighted averaging algorithm is implemented to aggregate the model. Weights for parameter aggregation is dependent on the position of the connected

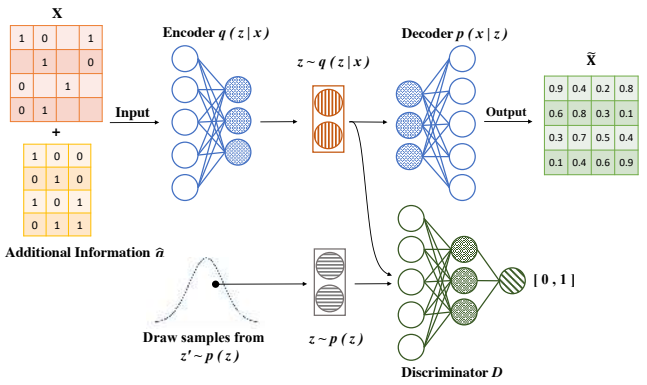


Fig. 3. C-AAE model architecture

vehicle, which is $\gamma_k = P_k/\mathfrak{D}$, and the number of local training data. Then, we can re-write the aggregate method as

$$w_{r+1} \leftarrow w_r - \eta \sum_{k=1}^{K} \gamma_k \frac{d_k}{d} w_{r+1}^k. \tag{6}$$

Selected vehicles with a longer available training time account for more contributions and are given greater weight in model aggregation.

Additionally, after training a shared global model, each RSU predicts its popular contents and then sends a list to the MBS. The MBS stores all lists of cached contents for future replacing contents use. The above process is repeated. The full algorithm is outlined in Algorithm 1.

B. Contextual-aware Adversarial Autoencoders For Content Popularity Prediction

The model we trained in the above FL framework is the Contextual-aware Adversarial Autoencoders (C-AAE) model. It is used to predict the popularity of contents for proactive edge caching purpose. Adversarial Autoencoders (AAE) is a probabilistic AutoEncoder (AE), combining Generative Adversarial Networks(GANs) and Variational Autoencoders (VAE) [34]. The architecture of a C-AAE is shown in Fig. 3. The top row is an AE. It is able to learn a latent code z to replicate its input X to its output $\tilde{X}$ in an unsupervised learning manner [35]. The bottom row is an adversarial network. It is mainly used to distinguish whether the sample generates from the specified distribution of the user or the sample comes from the latent code z of the AE. A user-by-content request matrix X, used as the input data, is content retrieval history by vehicular users. It consists of samples of variable x, where $X \in \mathbb{N}^{m \times c}$. m and c stand for the number of connected vehicles and contents, respectively. Moreover, which contents will be requested in the future may depend on the vehicular users' context. The contextual information of connected vehicular users $\hat{a}$ is used in our proposed method, in order to predict the popularity of context-specific contents. $\hat{a}$ is appended to X. The output of C-AAE is $\tilde{X}$, which is the reconstructed inputs filling in prediction values.

C-AAE adds the GAN to the AE architecture by turning an AE into a generative model. It trains an AE with an adversarial loss to adapt the distribution of latent space to an arbitrary prior. GANs build two neural networks: the generative model G and the discriminative model D. G uses a vector of random numbers as the inputs and generates the input at the output. D is utilised to differentiate between a sample is generated from the G and a sample is taken from the input data. The minmax game of GAN between G and D can be expressed as follows [34]:

$$\min_G \max_D \mathrm{E}_{x \sim p_d(x)} [\log D(x)] + \mathrm{E}_{z \sim p(z)} [\log (1 - D(G(z)))]. \tag{7}$$

The reconstruction phase and regularization phase are two phases in the training process of C-AAE. In the phase of reconstruction, an AE is to update the encoder and encoder is to minimise reconstruction error of input x. Firstly, z is generated by the generator network $q(z \mid x)$, an encoding distribution. Next, z is feed to the decoder and the output $\tilde{x}$ is reconstructed from z. The loss of reconstruction is calculated between x and $\tilde{x}$. In the phase of regularization, discriminator D is firstly updated by the adversarial network to distinguish true prior samples from generated samples. Then, in order to fool the discriminator D, the generator G is updated. The hidden code is essential to be considered by the discriminative network, which are distributed as the true prior distribution $p(z)$. This generator is also the encoder of the AE.

The output is imposed to the encoder by an adversarial network to follow the distribution of $p(z)$. z can be obtained by the discriminator. z' is drawn by $p(z)$. Backpropagation is applied to adjust the weights of discriminator. Meanwhile, the parameters of generator are updated. This process is repeated. The generative model is defined by the decoder of the AE, when the procedure of training is finished. The imposed prior of $p(z)$ is mapped to the data distribution $p_d(x)$. Thus, the regularisation of C-AAE can be achieved by matching $q(z)$ to $p(z)$, where $q(z)$ is the aggregated posterior and $p(z)$ is the arbitrary prior. The $q(z)$ is given by [34]:

$$q(z) = \int_x q(z \mid x) p_d(x) \, dx, \tag{8}$$

where $p(x)$ is the model distribution. The loss of the discriminator is

$$L_D = -\frac{1}{b} \sum_{a=1}^{b} log(D(z')) + log(1 - D(z)), \tag{9}$$

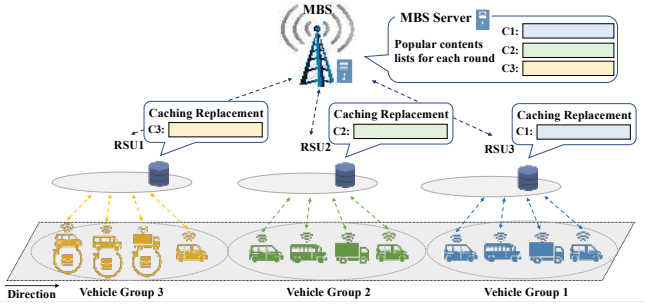
where b is the minibatch size. The adversarial generator we used is

$$L_G = -\frac{1}{b} \sum_{a=1}^{b} log(D(z)). \tag{10}$$

Additionally, C-AAE is inspired in VAE. VAE is a generative autoencoder, aiming to learn the data distribution $p(x)$. It attempts to minimize the KL-Divergence between latent codes distribution and the desired distribution (*i.e.*, Gaussian). A sample from the desired distribution is feed to the decoder, for reconstructing inputs. VAE can effectively cluster similar input data in the latent space [36]. However, one drawback of utilising the KL-divergence in VAE is to handle the functional

	RSU 1	*RSU 2*	*RSU 3*
Round 1	Contents 1	Contents 1	Contents 1
	Vehicle Group 1		
Round 2	Contents 2	Contents 1	Contents 1
	Vehicle Group 2	Vehicle Group 1	
Round 3	Contents 3	Contents 2	Contents 1
	Vehicle Group 3	Vehicle Group 2	Vehicle Group 1

(a) Summary



(b) Example: round 3

Fig. 4. Mobility-aware caching replacement process

form of $p(z)$. Instead, AAE provides a more flexibility way that is to sample from $p(z)$, in order to match latent codes distribution with the prior. As a result, more distributions can be used as prior for the latent code.

In our design, the input matrix is the user-by-content request matrix X, which is binary. 1 and 0 represent user's interested contents and uninterested contents, respectively. We mark the contents that users requested before as the interested contents. In fact, it is hard to identify the uninterested contents, since unknown contents and uninterested contents are mixed in the unrequested contents. Marking all unrequested contents as uninterested contents is a bias prediction. Thus, we marked unknown contents, corresponding to the missing entries by a random sampling mechanism. The probability of random sampling is related to the preference of clients for contents. We utilise the proposed C-AAE model to predict these missing values. The C-AAE learns the latent code Z from the input matrix X. Then, X can be recovered from Z to generate $\tilde{X}$. X has incomplete rows and columns. $\tilde{X}$ is a matrix with predicting the missing entries. We rank the predicted contents and the highest score contents in outputs will be chosen as the caching contents in the RSU. The complexity of the MPCF algorithm depends on the C-AAE model's complexity, which is $O(h)$. h is the size of a hidden layer in the C-AAE model.

C. Mobility-aware Cache Replacement Policy

As the vehicles move from one RSU to another, the requested contents cached at one RSU might become obsolete, whereas the other RSUs do not cache these previously requested contents for the coming vehicles. Such ineffective utilisation of cache resources motivates us to design a mobility-aware cache replacement policy. This policy enables the RSU to replace its caching contents, in response to the mobility of vehicles.

Fig. 4 shows the details about how the process of cache replacement is carried out between RSUs and vehicles. Fig. 4 (a) summarises the activities of each RSU in three rounds. To explain the dynamic process, Fig. 4 (b) illustrates the interactions amongst three RSUs, groups of vehicles and one MBS in the third round of the process. All RSUs are located within the coverage area of MBS. RSU 1 is located at the entrance to the road. It is different from other RSUs, because RSU 1 is the only RSU that executes the training of C-AAE model to predict popularity of contents for its connected vehicles. In self-driving scenario, MBS holds prior knowledge of its connected vehicles, including speed, position and destination. MBS has the capacity to estimate the arrival times of vehicles at each RSU.

In round 1, Vehicle Group 1 is moving through the area covered by RSU 1. At the same time, popularity of contents is predicted and a list of predicted popular contents (C1) for Vehicle Group 1 is instantly sent to MBS. MBS then sends the predicted contents (C1) to all RSUs to cache. In round 2, new vehicles (*i.e.*, Vehicle Group 2) move into the coverage area of RSU 1 and Vehicle Group 1 moves to the area covered by RSU 2. MBS then updates the caching contents for all RSUs. RSU 1 will cache the second round prediction (C2), RSU 2 and RSU 3 still store C1. In round 3, new vehicles (*i.e.*, Vehicle Group 3) enter the RSU1's coverage area. Similar to round 2, RSU 1 replaces caching contents to newly updated prediction (C3), depending on the requests made by the new vehicles. C2 is cached in RSU 2, as Vehicle Group 2 moves into the coverage area of RSU 2. RSU 3 still stores C1. Vehicles continue to move over time. In the fourth round, Vehicle Group 1 will leave the area covered by the current MBS and enter to the next MBS. Therefore, the current MBS shall remove C1 from the cache.

To sum up, the proposed cache replacement policy aims to effectively update the contents on RSU in response to its connected vehicles' prediction results. It is worth noting that the predict contents (*e.g.*, C1, C2, C3 *etc.*) can be similar and MBS only needs to replace the different contents at each round, which effectively reduces the processing time.

V. Performance Results and Analysis

The performance of the proposed MPCF under various environments in VEC is evaluated in this section.

A. Simulation Settings and Dataset

We simulate a VEC environment in an urban area consisting of an MBS, 3 RSUs and several vehicles located within the coverage areas of RSUs. A HP Z440 workstation with 64G memory is exploited as the MBS to store the lists of cached contents and manage cache resources. Two HP Z440 workstations, working as RSUs, aggregate the parameters of C-AAE model and cache the predicted popular contents. The

Algorithm 1 : ***Mobility-aware Proactive Edge Caching Scheme based on Federated Learning (MPCF)***, M is the set of vehicles connected to the RSU, where $m \in M$. S is the set of RSUs, where $s \in S$; Q is the position of RSU.

RSU Server Execution:

Initialise w_0
for each round r = 1, 2, ... **do**:
  K_r: A set of selected vehicles in r^{th} round
  for each vehicle $m \in M$ **in parallel do**:
    $T^m_{standing} = \left(\mathfrak{D} - P^i_m\right)/U_m$
    if $T^m_{standing} > T_{round} + T_{inf}$ **then**
      add m to K_r
    end if
  end for
  C_r: A set of caching contents in r^{th} round
  C_k: A set of predicted popular contents from vehicle k
  for each vehicle $k \in K_r$ in parallel **do**:
    if $Q^k_{s-1} == Q^k_s$:
      Use the previous model $w_r = w_{r-1}$
    else
      Download the current global model w_r
    end if else
    $w^k_{r+1}, C_k \leftarrow$**VehicleUpdate**$(w_r, k)$
    Add C_k to C_r
  end for
  $w^k_{r+1} \leftarrow \sum_{k=1}^{K} \frac{d_k}{d} w^k_{r+1}$
  Count C_r
  Cache Top N contents from C_r
  CacheReplace(C_r)**:**
end for
Return w_{r+1}

Vehicle Execution:

Input: X, w_r, P_k, $\mathfrak{D}$
VehicleUpdate(w, k)**:**
  for each local epoch e = 1, 2, ...**do**
    for each batch b **do**
      Compute parameters with gradient descent:
      $w^k_{r+1} \leftarrow w_r - \eta \nabla l\left(w_r; b\right)$
      $\gamma_k = P_k/\mathfrak{D}$
      $w^k_{r+1} \leftarrow \gamma_k w^k_{r+1}$
    end for
  end for
  Rank predicted contents C_k
Return w^k_{r+1}, C_k

Cache Replacement Execution:

CacheReplace(C)**:**
  for each round r = 1,2,... **do**:
    Compare C_{r-1} and C_r
    Update new contents from C_r
  end for

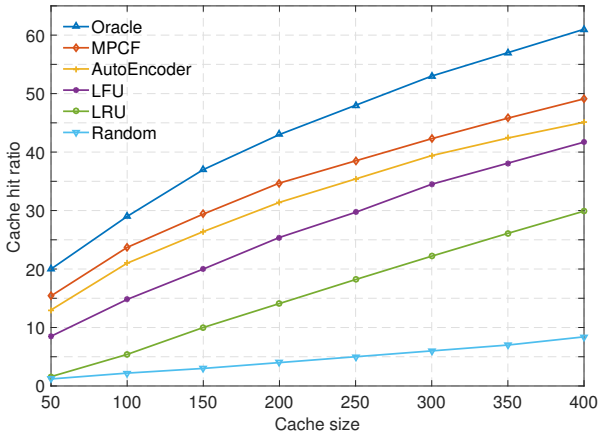


Fig. 5. Cache hit ratio with different cache sizes

number of vehicles under each RSU varies from 1 to 100. All vehicles have datasets to conduct local model training. Keras is employed as the Deep learning framework to implement C-AAE and FL, with TensorFlow as backend. The dataset we used in our experiments is MovieLens 1M dataset collected from the MovieLens website [37]. About 1 million ratings are contained in this dataset, which came from 6040 anonymized users on 3883 movies. The contextual information of users, *e.g.* gender, age, address and occupation, is also provided in the dataset. To simulate the process of vehicular users' requests, the rated movies are assumed to request contents from vehicles.

B. Performance Evaluation

Cache hit ratio is the performance metric we used to evaluate the proposed MPCF, which measures the effectiveness of a cache in fulfilling content requests. Cache hit ratio is calculated as follows: Cache hit ratio = cache hits/ (cache hits + cache misses). One cache hit is captured when the requested content is delivered by the cache, whereas a cache miss is captured when the requested content is not stored in the cache.

Our proposed caching scheme MPCF is compared to the following four baseline caching schemes, as shown in Fig. 5.

- Oracle: It has the prior knowledge of the exact future content requests from vehicles and provides the maximum of cache hit ratio.
- Random: The contents stored at cache are randomly selected by the RSU.
- Least Recently Used (LRU): When the limit of cache capacity is reached, it firstly removes the least recently used content in the cache.
- Least Frequently Used (LFU): In LFU, the least frequently used content in the cache is discarded whenever the cache capacity is full.
- AutoEncoder: It is a learning based caching scheme, using AutoEncoder model.

Fig. 5 depicts the cache hit ratio for varying cache sizes from 50 to 400 contents. The results demonstrate that our proposed MPCF outperforms other reference caching schemes. With the increase of cache size, the cache hit ratios of all caching

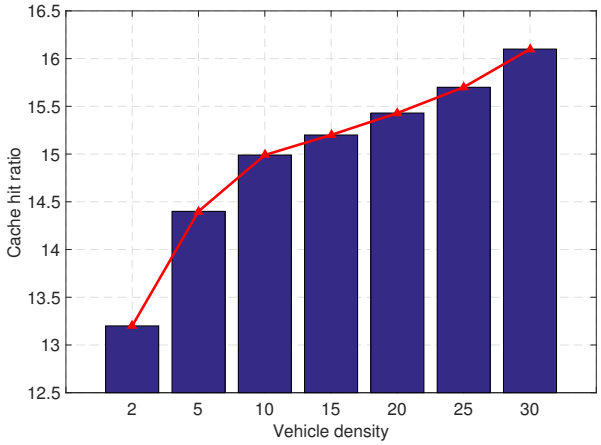


Fig. 6. Cache hit ratio vs Vehicle density

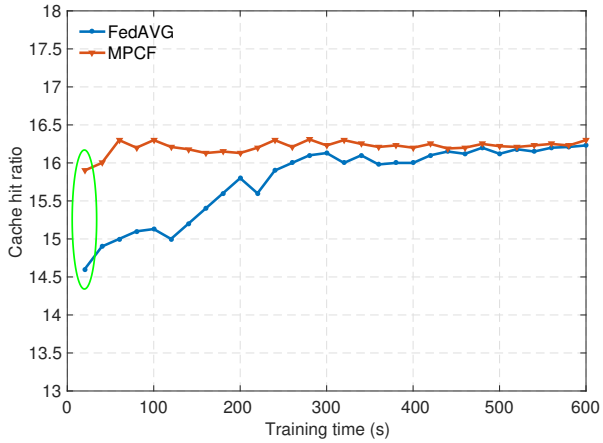


Fig. 8. FL training process (27 vehicles)

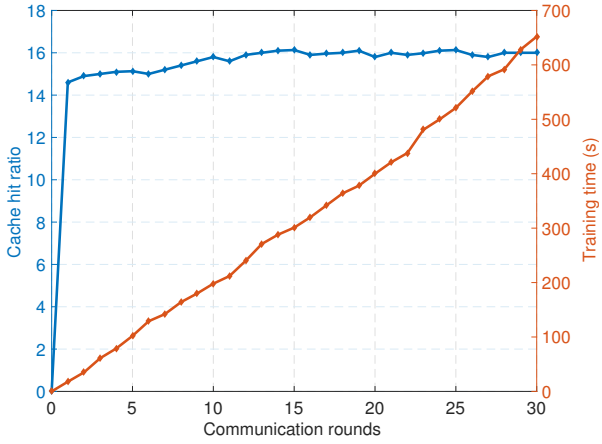


Fig. 7. Cache hit ratio and Training time against Communication rounds

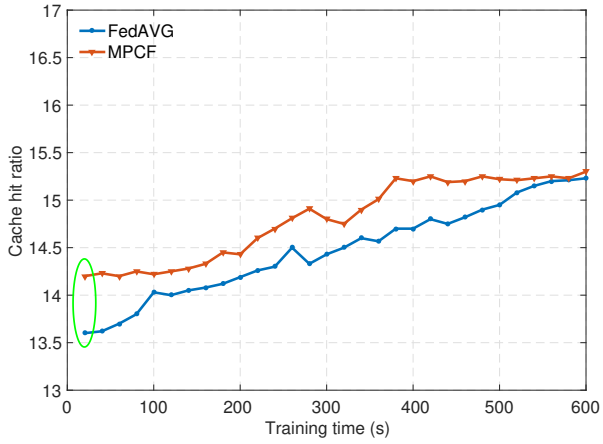


Fig. 9. FL training process (5 vehicles)

schemes rise. As our expected, the lowest cache hit ratio is presented by Random. The proposed MPCF and AutoEncoder outperform LFU, LRU and Random because they learn the latent representations and extract features from the content request history of connected vehicles to predict precise content popularity. LFU and LRU follow static rules, but dynamically changing content popularity is not considered. The MPCF shows a better performance compared to AutoEncoder. It is due to the fact that MPCF captures useful features from data and clusters the data in the latent space. Oracle provides the best cache hit ratio, because it has the prior knowledge of content requests from vehicles in the future.

Fig. 6 presents the influence of the vehicle density on the cache hit ratio. The cache size of RSU is fixed at 50 in this experiment and the density of vehicles varies from 2 to 30 vehicles/km. The results show that the cache hit ratio increases with the grown of vehicle density. When only two vehicles are moving in the coverage area of RSU, the cache hit ratio is 13.2%. However, when five vehicles connect to the RSU, the cache hit ratio increases to 14.4%. Along with more vehicles in the RSU's coverage area, more computation capacity and training data are offered by these vehicles. Correspondingly, more accurate prediction of content popularity can be obtained.

Fig. 7 shows the results of cache hit ratio, communication rounds versus training time. In this experiment, 10 vehicles collaboratively participate a global model training. In the first round, the cache hit ratio is 14.6 %, while it increases to 14.9% in the second round. When the communication round is 30, the cache hit ratio achieves above 16%. These show that the cache hit ratio rises with the increasing communication rounds. It comes with a price of training time. The training time for one round is about 18 seconds, compared to more than 600 seconds for 30 rounds. As can be seen in Fig. 7, the cache hit ratio changes not significant after 15 rounds. Thus, considering a trade-off between communication rounds, training time and cache hit ratio, training FL model for 15 rounds is the best choice to achieve the optimal cache hit ratio.

Fig. 8 and Fig. 9 show the difference between typical federated learning training process (FedAVG) [26] and our proposed federated learning method (MPCF), in respect of the training time and cache hit ratio. Both methods exhibit the same trend. With longer training time, the more accurate results can be achieved. In particular, after training for a while, both methods reach a similar cache hit ratio. However, compared with FedAVG in the first round, MPCF can obtain a higher cache hit ratio. As depicted in Fig. 8, when 27 vehicles participate in the FL training, the cache hit ratio of FedAVG is 14.3%. In comparison, the cache hit ratio of MPCF is 15.7%.

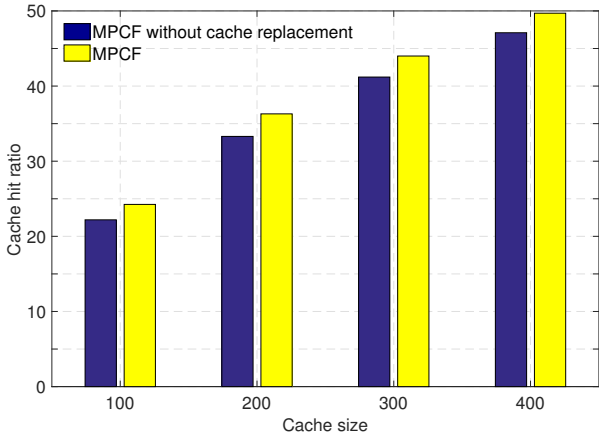


Fig. 10. Mobility-aware cache replacement

Fig. 9 (5 vehicles) also shows that the MPCF outperforms FedAVG at the beginning in terms of cache hit ratio, while the final results are similar. In VEC, vehicles are driving at high speed. They go through the coverage area of RSU quickly and result in the short training time. The results exhibit that the proposed MPCF can gain a desirable cache hit ratio in a shorter time than FedAVG. For example, Fig. 8 depicts the MPCF achieves the target cache hit ratio of 16% within 3 rounds, less than 60 seconds, while FedAVG needs more than 600 seconds. The reason is that the MPCF is mobility-aware and thus more suitable to VEC scenarios.

Fig. 10 demonstrates the effectiveness of the mobility-aware cache replacement policy. We compare the cache hit ratio of MPCF and MPCF without cache replacement. As shown in Fig. 10, the cache hit ratio for MPCF outperforms MPCF without cache replacement policy. For example, the cache hit ratio of MPCF is around 24.25% when the cache size is 100, while MPCF without cache replacement policy achieves the cache hit ratio of 22.2%. These experiment results demonstrate that the proposed mobility-aware cache replacement policy is able to further enhance the cache performance of FL learning-based caching schemes in VEC.

VI. Conclusion and Future Work

In this paper, we have proposed a new Mobility-aware Proactive edge Caching scheme on Federated learning, termed MPCF, to improve cache performance and protect vehicles' privacy. MPCF utilises a context-aware adversarial autoencoder model to estimate content popularity and then places predicted popular contents at the edge of vehicular networks to reduce latency. To maximise the cache hit ratio, a mobility-aware cache replacement policy is designed to dynamically update caching contents at RSUs, according to the mobility pattern of moving vehicles and their predictions of content popularity. Numerical results demonstrate that MPCF outperforms other baseline caching schemes on cache hit ratio. The FL training process of MPCF accelerates the training of a global shared model. Implementing the mobility-aware cache replacement policy further improves the cache hit ratio.

Due to the dynamic environment of vehicular networks and heterogeneous communication and computing capabilities of vehicles, some vehicles may go offline or fall behind during the federated learning process. Thus, synchronized federated learning can be very slow in vehicle networks. In our future work, we intend to design a proactive content caching scheme based on fully asynchronous federated learning to better cope with the highly dynamic vehicular environments.

Acknowledgment

This work was supported in part by the UK EPSRC project EP/M013936/2 and the Researchers Supporting Project number (RSP-2020/32), King Saud University, Riyadh, Saudi Arabia.

References

[1] S. Garg, K. Kaur, G. Kaddoum, S. H. Ahmed, and D. N. K. Jayakody, "SDN-based secure and privacy-preserving scheme for vehicular networks: A 5G perspective," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8421–8434, 2019.

[2] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *arXiv preprint arXiv:1908.06849*, 2019.

[3] S. Wang, Z. Zhang, R. Yu, and Y. Zhang, "Low-latency caching with auction game in vehicular edge computing," in *Proc. of 2017 ICCC*. IEEE, 2017, pp. 1–6.

[4] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "Deepcache: A deep learning based framework for content caching," in *Proc. of 2018 SIGCOMM*. ACM, 2018, pp. 48–53.

[5] Z. Ning, K. Zhang, X. Wang, M. S. Obaidat, L. Guo, X. Hu, B. Hu, Y. Guo, B. Sadoun, and R. Y. Kwok, "Joint computing and caching in 5G-envisioned internet of vehicles: A deep reinforcement learning-based traffic control system," *IEEE Transactions on Intelligent Transportation Systems*, doi:10.1109/TITS.2020.2970276, 2020.

[6] L. Hou, L. Lei, K. Zheng, and X. Wang, "A Q-learning-based proactive caching strategy for non-safety related services in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4512–4520, 2018.

[7] A. Ndikumana, N. H. Tran, K. T. Kim, C. S. Hong *et al.*, "Deep learning based caching for self-driving cars in multi-access edge computing," *IEEE Transactions on Intelligent Transportation Systems*, doi:10.1109/TITS.2020.2976572, 2020.

[8] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[9] J. Cheng, G. Yuan, M. Zhou, S. Gao, C. Liu, H. Duan, and Q. Zeng, "Accessibility analysis and modeling for iov in an urban scene," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4246–4256, 2020.

[10] J. Zhang, J. Cui, H. Zhong, Z. Chen, and L. Liu, "Pa-crt: Chinese remainder theorem based conditional privacy-preserving authentication scheme in vehicular ad-hoc networks," *IEEE Transactions on Dependable and Secure Computing*, doi: 10.1109/TDSC.2019.2904274, 2019.

[11] Y. Cao, S. Yang, G. Min, X. Zhang, H. Song, O. Kaiwartya, and N. Aslam, "A cost-efficient communication framework for battery-switch-based electric vehicle charging," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 162–169, 2017.

[12] Z. Hu, Z. Zheng, T. Wang, L. Song, and X. Li, "Roadside unit caching: Auction-based storage allocation for multiple content providers," *IEEE Transactions on Wireless Communications*, vol. 16, no. 10, pp. 6321–6334, 2017.

[13] R. Ding, T. Wang, L. Song, Z. Han, and J. Wu, "Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery," in *Proc. of 2015 WCNC*. IEEE, 2015, pp. 1207–1212.

[14] Z. Su, Y. Hui, Q. Xu, T. Yang, J. Liu, and Y. Jia, "An edge caching scheme to distribute content in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5346–5356, 2018.

[15] A. Mahmood, C. Casetti, C.-F. Chiasserini, P. Giaccone, and J. Harri, "Mobility-aware edge caching for connected cars," in *Proc. of 2016 WONS*. IEEE, 2016, pp. 1–8.

[16] A. Mahmood, C. E. Casetti, C. F. Chiasserini, P. Giaccone, and J. Härri, "The rich prefetching in edge caches for in-order delivery to connected cars," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 4–18, 2018.

[17] N. Kumar and J.-H. Lee, "Peer-to-peer cooperative caching for data dissemination in urban vehicular communications," *IEEE Systems Journal*, vol. 8, no. 4, pp. 1136–1144, 2013.
[18] S. Fang and P. Fan, "A cooperative caching algorithm for cluster-based vehicular content networks with vehicular caches," in *Proc. of 2017 Globecom GC Wkshps*. IEEE, 2017, pp. 1–6.
[19] G. Deng, L. Wang, F. Li, and R. Li, "Distributed probabilistic caching strategy in vanets through named data networking," in *Proc. of 2016 INFOCOM WKSHPS*. IEEE, 2016, pp. 314–319.
[20] L. Yao, A. Chen, J. Deng, J. Wang, and G. Wu, "A cooperative caching scheme based on mobility prediction in vehicular content centric networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5435–5444, 2017.
[21] J. Ma, J. Wang, G. Liu, and P. Fan, "Low latency caching placement policy for cloud-based vanet with both vehicle caches and rsu caches," in *Proc. of 2017 IEEE Globecom GC Wkshps*. IEEE, 2017, pp. 1–6.
[22] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, 2018.
[23] S. Park, S. Oh, Y. Nam, J. Bang, and E. Lee, "Mobility-aware distributed proactive caching in content-centric vehicular networks," in *Proc. of 2019 WMNC*. IEEE, 2019, pp. 175–180.
[24] Y. AlNagar, S. Hosny, and A. A. El-Sherif, "Towards mobility-aware proactive caching for vehicular ad hoc networks," in *Proc. of 2019 WCNCW*. IEEE, 2019, pp. 1–6.
[25] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. of 2019 ICC*. IEEE, 2019, pp. 1–7.
[26] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *Proc. of 2016 AISTATS*, pp. 1–10.
[27] X. Zhou, W. Liang, I. Kevin, K. Wang, and S. Shimizu, "Multi-modality behavioral influence analysis for personalized recommendations in health social media environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 888–897, 2019.
[28] X. Zhou, W. Liang, I. Kevin, K. Wang, H. Wang, L. T. Yang, and Q. Jin, "Deep learning enhanced human activity recognition for internet of healthcare things," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6429–6438, 2020.
[29] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 246–257, 2019.
[30] S. M. Abuelenin and A. Y. Abul-Magd, "Empirical study of traffic velocity distribution and its effect on vanets connectivity," in *Proc. of 2014 ICCVE*. IEEE, 2014, pp. 391–395.
[31] S. Yousefi, E. Altman, R. El-Azouzi, and M. Fathy, "Analytical model for connectivity in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 6, pp. 3341–3356, 2008.
[32] S. Garg, K. Kaur, S. H. Ahmed, A. Bradai, G. Kaddoum, and M. Atiquzzaman, "Mobqos: Mobility-aware and QoS-driven SDN framework for autonomous vehicles," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 12–20, 2019.
[33] T. Leighton, "Improving performance on the internet," *Communications of the ACM*, vol. 52, no. 2, pp. 44–51, 2009.
[34] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *Proc. of ICLR*, pp. 1–10, 2016.
[35] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes, Stanford University, USA*, vol. 72, no. 2011, pp. 1–19, 2011.
[36] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," *Proc. of 2017 IJCAI*, pp. 1965–1972, 2017.
[37] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, p. 19, 2016.

Zhengxin Yu is a Ph.D. student in the Department of Computer Science within College of Engineering, Maths and Physical Science at the University of Exeter, UK. She received an MSc in Information Technology Management for Business from the University of Exeter in 2016. Her research interests focus on deep learning, federated learning and mobile edge computing.

Jia Hu is a Senior Lecturer in Computer Science at the University of Exeter. He received his Ph.D. degree in Computer Science from the University of Bradford, UK, in 2010, and M.Eng. and B.Eng. degrees in Electronic Engineering from Huazhong University of Science and Technology, China, in 2006 and 2004, respectively. His research interests include edge-cloud computing, resource optimization, applied machine learning, and network security.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems, Modelling and Performance Engineering.

Zhiwei Zhao received his PhD degree at the College of Computer Science, Zhejiang University in 2015. He is currently an associate professor at the College of Computer Science and Engineering in University of Electronic Science and Technology of China. His research interests focus on edge computing for IoT, low power wireless, and mobile computing. He is a member of IEEE, ACM and CCF.

Wang Miao received his Ph.D. degree in Computer Science from the University of Exeter, United Kingdom in 2017. He is currently a Postdoctoral Research Associate at the College of Engineering, Mathematics, and Physical Sciences of the University of Exeter. His research interests focus on Network Function Virtualization, Software Defined Networking, Unmanned Aerial Networks, Wireless Communication Networks, Wireless Sensor Networks, and Performance Modelling and Analysis.

M. Shamim Hossain (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Ottawa, Canada, in 2019. He is currently a Professor at the Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He is also an Adjunct Professor at the School of Electrical Engineering and Computer Science, University of Ottawa, Canada. His research interests include cloud networking, smart environment (smart city and smart health), AI, deep learning, edge computing, the Internet of Things (IoT), multimedia for health care, and multimedia big data.