2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA) | 979-8-3503-4588-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SERA57763.2023.10197801

Split and Federated Learning with Mobility in Vehicular Edge Computing

Sungwon Moon
dept. of IT Engineering
Sookmyung Women's University
Seoul 04310, Korea
sungwon268@sookmyung.ac.kr

Yujin Lim
dept. of IT Engineering
Sookmyung Women's University
Seoul 04310, Korea
yujin91@sookmyung.ac.kr

***Abstract*—Vehicular edge computing (VEC) is a promising technology to support vehicular applications that leverage machine learning (ML) technology. Due to limited resources of the vehicle, the vehicle uses Split learning (SL) to split the computation of the ML model and offload it to the VEC server (VECS). Federated learning (FL) is also used for data privacy and parallel training of the vehicles. Therefore, SplitFed learning, which combines SL and FL, enables parallel processing, which is an advantage of FL, and reduces the computational burden on the vehicle through ML model split, which is an advantage of SL. However, the SplitFed learning does not consider the mobility of device/vehicle. Therefore, we propose a SplitFed learning with mobility method to minimize the training time of the model. SplitFed learning with mobility method is a migration method of the ML model when the vehicle moves from the current serving VECS to the target VECS. Through simulations, compared with conventional SplitFed learning where the vehicle travels after 50% and 80% of training is completed, the proposed method can reduce training time by about 19-33% for LeNet and by about 22-44% for VGG16, respectively, and does not degrade accuracy of model.**

Keywords—Split learning, Federated learning, Vehicular edge computing, Mobility

I. INTRODUCTION

The rapid development of vehicular network and communication technology has expanded to the Internet of Vehicles (IoV) [1]. IoV leverages machine learning (ML) technology to facilitate vehicular applications that provide various intelligent services, such as autonomous driving and intelligent navigation, and path planning [2]. It utilizes data generated or collected from vehicle and resources to train ML model used in vehicular applications. However, it is impractical to train complex ML models in vehicle terminal, given that limited computing and networking capabilities of vehicle are not sufficient.

Therefore, the vehicular edge computing (VEC) paradigm [3] is a promising solution to support vehicular applications. The vehicular applications leverage computational resources provided by the VEC server (VECS), which has cloud-like capability and is placed at the edge of the network near vehicles. For performance efficiency, the vehicle splits the computation of the ML model and offloads them to the VECS.

Split Learning (SL) [4] splits the ML model that is a deep learning model into multiple parts, which are processed and calculated by devices and servers separately. On the device side, only a portion of the network is processed, reducing the processing load. This has a significant impact on ML calculations for resource-constrained devices. However, the sequential round-robin based training in SL significantly increases training overhead because only one device can engage with the server at one time. Also, data collected from devices often contains personal information, thereby data privacy and security should be considered.

Therefore, we use combined SL and Federated Learning (FL), SplitFed learning [5]. FL [6] is a privacy technique that is collaboratively trained among devices without sharing data. FL is reluctant to send raw data to a central server for data security and privacy. Each device train a network model with their local data, and send the updated parameters to a central server. The server then combines the parameters from the different devices to generate the global parameters. However, each device needs to run a full ML model, which is bound to be a burden on devices with resource-constrained.

The synergy of SL and FL alleviates the limitations of SL and FL. Therefore, SplitFed learning not only allows parallel processing among devices with FL, but also alleviates the computational burden that is processed on the device by splitting the ML model with SL. It also improves the data privacy and security. Recently, SplitFed learning has become an interesting research topic, so several papers on SplitFed learning have been published as shown in [5, 7–9]. However, many studies have usually been conducted for the efficiency of communication or computation, and have not addressed the mobility of the device during training.

In VEC system, it is important to consider the mobility of device/vehicle. The vehicles move across the coverage area of VECS because of its high mobility. If the vehicle is moving outside the coverage area of the VECS currently in service, the migration from the current serving VECS to the target VECS have to be carried out. If the vehicle moves from the serving VECS to the target VECS without migrating the ML model that comes with it, the vehicle has to restart the training from the beginning on the target VECS. This is because the target VECS does not have a copy of the training model. This results in inefficient results, increasing the total training time. Therefore, in this study, we propose a method to consider mobility using SplitFed learning in VEC system.

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ICAN(ICT Challenge and Advanced Network of HRD) program(IITP-2023-RS-2022-00156299) supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation)

II. System Model

As shown in Fig.1, the VEC system proposed in this study consists of three layers: the vehicle, VECS, and VECS controller. The full ML model/network of each vehicle is split into multiple smaller network parts and offloaded from the vehicle to the VECS. The split ML models are then trained separately using their own local data on the vehicles and VECS.

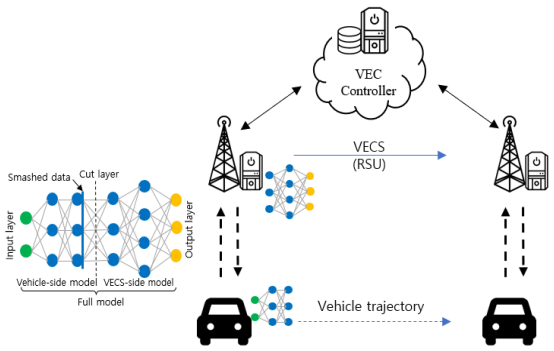


Fig. 1. The architecture of the proposed VEC system

The vehicle n, $n \in N$, moves on the road at a speed of v_n. The VECS m, $m \in M$, is attached to the roadside unit (RSU) and placed on the road. The VECSs communicate among each other through backhaul links. The RSU is connected to a base station equipped with a VECS controller. Each vehicle is connected to the VECS with the best signal, and offload the split model to it.

Let W_n be the size of the full model of vehicle n, and β_n be the proportion of the model size allocated to the vehicle n. In other word, the vehicle n and VECS m execute the proportion of the workload as $\beta_n W_n$ and $(1-\beta_n)W_n$, respectively. Let D_n be the size of local dataset of vehicle n, q be the size of the smashed layer, and P_n and P_m be the computing powers of the vehicle n and VECS m. In addition, we define R as the network bandwidth between the vehicle and VECS. Therefore, the training time of vehicle n for one global epoch is as follows [5]:

$$T_n = \frac{\beta_n D_n W_n}{P_n} + \frac{(1-\beta_n) D_n W_n}{P_m} + \frac{2qD_n + 2\beta W_n}{R} \quad (1)$$

where the training time for updating the model on the vehicle n and VECS m is $\frac{\beta_n D_n W_n}{P_n}$ and $\frac{(1-\beta_n) D_n W_n}{P_m}$, respectively. And the communication time for uploading and downloading model updates before and after training is $\frac{2qD+2\beta W_n}{R}$.

The objective of the proposed method is to reduce the model update training time calculated by (1) by allowing the training to continue on the target VECS instead of restarting the training from the beginning when the vehicle is moved.

III. Proposed Method

In this section, we propose a SplitFed learning considering mobility method in the VEC system. The objective is to minimize the total training time of the vehicle by preventing unnecessary repetitive model updating training due to the high mobility of the vehicle. First, we provide the background of SL, FL and SplitFed learning, and then detail the proposed method.

A. Background

SL splits the ML model into two parts: a device-side network and a server-side network, respectively. The device trains the device-side network where the model has up to the cut or split layer, which is the layer after split. The device then transmits the smashed data, i.e., the activations of the cut layer, to the server. After the server received the gradients of the smashed data, the server leverages it to train the remaining layers of the model, server-side network. The server trains backward propagation up to the cut layer, and transmits the smashed data to the device. The device leverages the smashed data to train backward propagation on the remaining model.

In FL, the server distributes the parameters of the initialized a global model to all devices. Each device then trains the entire ML model in parallel with their own local data, and transmits the parameters of the updated local model to the server without sending the raw data. The server forms the global model with the parameters of the updated local model received from each device, and the parameters of the updated global model are transmitted to all connected devices. The server also synchronizes the device-side model at each round. This process continues until the method converges.

SplitFed learning combines the strength of SL, which is split the network into device-side network and server-side network during training, and the strength of FL, which is parallel processing among devices. Therefore, unlike SL, SplitFed learning performs processing in parallel among devices. Compared to FL, SplitFed learning performs the ML model by splitting, thereby alleviating the computational burden of the device. However, there is a limitation that SplitFed learning does not consider the mobility of device. Therefore, we propose a SplitFed learning considering the mobility of device/vehicle method (SFLM) in the VEC system. Table 1 shows a comparison of SL, FL, SplitFed, learning and SFLM.

TABLE I. Comparison of SL, FL, SFL, and SPFLM

	SL	**FL**	**SplitFed**	**SFLM**
Access to raw data	No	No	No	No
Model privacy	Yes	No	Yes	Yes
Model aggregation	No	Yes	Yes	Yes
Parallel training	No	Yes	Yes	Yes
Mobility consideration	No	No	No	Yes

B. Split and federated learning with mobility in VEC system

Due to the high mobility of the vehicle, the vehicle can be move during training. As the vehicle moves, the migration from the current VECS to the target VECS is required. If the migration is not performed, training is restarted from the beginning because the target VECS does not have a copy of the trained model. This results in an increase in training time. Therefore, we propose a SplitFed learning with mobility method in the VEC system by transferring model data from the current VECS to target VECS.

The detailed workflow of SplitFed learning with mobility method is following:

1) Initialization of global network: The VECS controller initializes the parameters of the global model and sends them to all connected vehicles. The vehicles receive the parameters of the global model from the VECS controller.

2) Split the ML model: Once the ML model on the vehicle is initialized, it splits into vehicle-side and VECS-side networks. (Considering the VEC system, the device-side and server-side networks are called vehicle-side and VECS-side networks, respectively)

3) Forward propagation on the vehicles: The vehicles train the vehicle-side model, i.e., forward propagation, in parallel, and transmits their smashed data, i.e., the activations of the cut layer, to the VECSs.

4) Forward and backward propagation on the VECSs: The VECSs use smashed data from each vehicle to train VECS-side models, i.e., forward propagation and backward propagation. Then, the gradient of the smashed data is transmitted to each vehicle to perform backward propagation on its vehicle-side model.

5) Backward propagation on the vehicles: After receiving the gradients of its smashed data, each vehicle trains their vehicle-side local model, i.e., backward propagation, and computes its gradients.

6) Global model aggregation: For global model aggregation, the parameters of the local model of each vehicle are sent to the VECS controller. The VECS controller updates a global model with the local models received from each device, and send the parameters of the updated global model to all connected vehicles.

If vehicle moves from the current VECS to the target VECS, the ML model processed by the current VECS is migrated to the target VECS. When the model has been migrated to the target VECS and the vehicle is connected to the target VECS, training begins after the training point before moving. The VECSs are connected by backhaul, so there is negligible overhead to migrate the data between VECSs.

IV. Simulation

In this section, the proposed method is evaluated compared with conventional method to verify the performance of it. The simulation was implemented on Python 3.6 using Pytorch 1.4. We randomly distributed 20 VECSs, which are deployed in RSUs, with a communication radius of 250m within the VEC system. In an area of 2.5 × 1.5 km, 20 vehicles travel at a speed of 10–20m/s using vehicular mobility trace dataset [10].

In simulations, we consider two popular architectures with respect to ML models. These two architectures belong to the convolutional neural network (CNN) architecture, Lenet5 and VGG16, and are summarized in Table 2. We use CIFAR10, a public image dataset. The CIFAR10 dataset consists of 10 classes and includes 50K training and 10K testing samples. The CIFAR10 is used as an input of 32 x 32 size, and a batch size of 256 is used in this simulations.

The FedAvg method [6] is used for model aggregation in FL, and stochastic gradient descent (SGD) is used for updating the parameters of the model, and a learning rate and momentum are used in the 200 global epochs as 0.01, 0.0001, and 0.9, respectively.

TABLE II. Model Architecture

Architecture	Number of parameters	Layers	Kernel size
LeNet	60 thousands	5	(5×5), (2×2)
VGG16	138 million	16	(3×3)

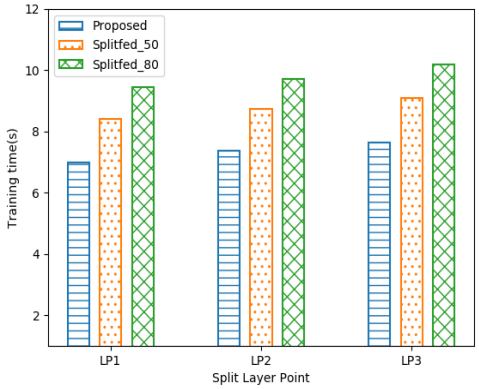
SplitFed learning, which combines split and federated learning, does not consider the mobility of device/vehicle. Once the vehicle has been moved, the target VECS does not have a copy of the training model, thereby the training has to be restarted from the beginning. This will increase the overall model update training time. In addition, it affects the model update training time, depending on how much of the training step has been processed before the vehicle is moved. For example, if migration is not considered, 50% of the training is completed in current VECS, and then if vehicle moves to the target VECS, the 50% of training is lost, and the training is restarted in the target VECS. Therefore, we compared *Proposed method*, SplitFed learning with mobility, with the existing SplitFed learning, *Splitfed_50* and *Splitfed_80*, which are the methods by which the vehicle travels after completing 50% and 80% of the model update training step.

Fig. 2 shows the training time for *Proposed*, *Splitfed_50* and *Splitfed_80* methods depending on the splitting at different layer points (LP). The training time is calculated by (1). In Fig. 2(a), Lenet5 network model splits into three different LPs, and in Fig. 2(b), VGG16 network model splits into five different LPs. The network layer of Lenet5 is split at the following LPs: after the second layer (LP1), after the fourth layer (LP2), and after the last layer (LP3). The network layer of VGG16 is split at the following LPs: after the first 2D MaxPool layer (LP1), after second 2D MaxPool layer (LP2), after third 2D MaxPool layer (LP3), after fourth 2D MaxPool layer (LP4), and after fifth 2D MaxPool layer (LP5).

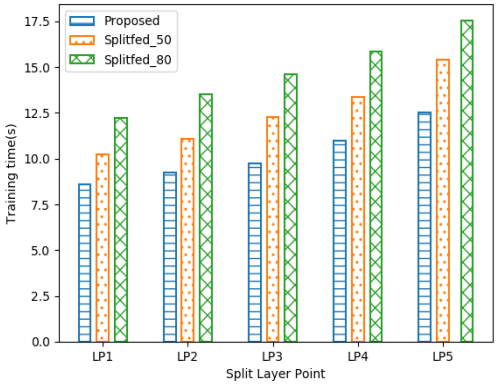
It can be seen that the training time increases as LP increases in the direction of increasing the number of layers of the vehicle. This indicates that the training time is longer if more processes are performed on the vehicle than the VECS. Therefore, the system performance in terms of training time is affected by where LP is split. Comparing *SplitFed_50* and *SplitFed_80* methods, it can be confirmed that the mobility of the vehicle also affects the training time. The training time depends on the amount of model update training completed before the vehicle is moved. The training time is more spent when the vehicle travels after 80% of training is completed than when the vehicle travels after 50% of training is completed. This is because no matter how much the model was trained before the vehicle is moved, the model has to restart training from the beginning after the vehicle is moved.

The *Proposed* method is to migrate the model considering the mobility of the vehicle, thereby there is no need to restart the model training on the target VECS. Therefore, the *Proposed*

method can reduce the training time by about 19-33% for LeNet, and by about 22-44% for VGG16, compared to *SplitFed_50* and *SplitFed_80* methods.



(a) LeNet5



(b) VGG16

Fig. 2. Training time for the vehicle per global round at different spllit layer points

TABLE III. TEST ACCURACY

Model Architecture	Method		
	Proposed	**SplitFed_50**	**SplitFed_80**
LeNet	61.52%	60.64%	61.40%
VGG16	57.42%	57.58	57.26%

Table 3 shows the accuracy of the model for *Proposed*, *Splitfed_50* and *Splitfed_80* methods, and also shows the accuracy according to the model architecture. Comparing the two model architectures, LeNet and VGG16, both models converge, but do not show high accuracy. We assume that it is related to the adjustment of experimental factors such as hyperparameters or data distribution, which is not the scope of this paper.

In the VEC system, a migration frequency of approximately 47% occurs when 20 vehicles travelling at a speed of 10–20 m/s. In this system, it can be seen that there is no significant difference in the accuracy performance when comparing *Proposed* method with *Splitfed_50* and *Splitfed_80* methods. *Proposed* method can be similar to *Splitfed_50* and *Splitfed_80* methods in terms of accuracy because training continues on the target VECS by migrating the model to the target VECS without restarting the training. Moreover, *Splitfed_50* and *Splitfed_80* methods do not degrade accuracy performance because they could receive the parameters of the updated global model from the VECS controller and restart the training on the target VECS.

V. CONCLUSION

In this study, we proposed a SplitFed learning considering the mobility of vehicle method in the VEC system. The objective of the proposed method is to minimize the model update training time of the vehicle by allowing the training to continue on the target VECS instead of restarting on the target VECS. Therefore, when the vehicle moves to the target VECS, it migrate the model that is being trained on the current VECS to the target VECS so that it does not need to be restarted unnecessarily on the target VECS.

Through the simulations, we found that the proposed method can reduce the training time by about 20-39% when the vehicle travels after 50% and 80% of the model update training is completed, respectively, and there is no degradation in accuracy performance. Therefore, there is a big difference in performance in terms of training time, but there is no big difference in terms of accuracy. In the future, we will conduct further experimental evaluations on various model architectures and datasets. In addition, this paper restricts the experiments by fixing the split layer. Therefore, we will conduct further study to determine the optimal split layer.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Commun. Surv. Tutor, vol. 17, no. 4, pp. 2347-2376, Jun. 2015.

[2] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang and W. Shi, "Edge Computing for Autonomous Driving: Opportunities and Challenges," Proc. IEEE, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[3] S. Moon, Y. Lim, "Federated Deep Reinforcement Learning Based Task Offloading with Power Control in Vehicular Edge Computing," Sensors 2022, vol. 22, no. 24, Dec. 2022.

[4] O. Gupta, R. Raskar, "Distributed Learning of Deep Neural Network over Multiple Agents," J. Netw. Comput. Appl. vol. 116, pp. 1–8, Aug. 2018.

[5] C. Thapa, M.A.P. Chamikara, S. Camtepe, L. Sun, "SplitFed: When Federated Learning meets Split Learning," arXiv 2021, Apr. 2020.

[6] H.B. McMahan, E. Moore, Ramage, D.; S. Hampson, B.A.Y. Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data. in Proc. AISTATS 2017, Lauderdale, FL, USA, 20-22 April 2017, pp. 1273–1282.

[7] Q. Duan, S. Hu, R.Deng, Lu, Z. "Combined Federated and Split Learning in Edge Computing for Ubiquitous Intelligence in Internet of Things: State-of-the-Art and Future Directions," Sensors 2022, vol. 22, no. 16, Aug. 2022.

[8] T. A. Khoa, D. V. Nguyen, M. S. Dao and K. Zettsu, "SplitDyn: Federated Split Neural Network for Distributed Edge AI Applications," 2022 IEEE Big Data, Osaka, Japan, 17-20 Dec 2022, pp. 6066-6073.

[9] J. Karjee, P. S. Naik and N. Srinidhi, "Split Federated Learning and Reinforcement based Codec Switching in Edge Platform," 2023 IEEE ICCE, Las Vegas, NV, USA, 06-08 Jan 2023, pp. 1-6.

[10] F.H. Kumbhar, "Vehicular Mobility Trace at Seoul, South Korea," IEEE Dataport, 2020.