Overcoming Environmental Challenges in CAVs through MEC-based Federated Learning

Zekun Wang*, Jin Nakazato†, Muhammad Asad†, Ehsan Javanmardi†, Manabu Tsukada†
*KTH Royal Institute of Technology, Brinellvägen 8, 114 28 Stockholm, Sweden
†The University of Tokyo, 1-1-1, Yayoi, Bunkyo-ku, Tokyo, 113-8657 Japan
Email: *zekunw@kth.se, †{jin-nakazato,asad,ejavanmardi,mtsukada}@g.ecc.u-tokyo.ac.jp

***Abstract*— Connected autonomous vehicles (CAVs), through vehicle-to-everything communication and computing resources, enable the vital exchange of information. Although deep learning is crucial in this landscape, it requires extensive and intricate datasets covering all potential scenarios. Furthermore, this situation poses a hazard, as the likelihood of accidents associated with imbalanced datasets increases, particularly in scenarios where processing analysis is compromised due to fluctuating weather conditions. We propose a Federated Learning (FL) framework undergirded by Multi-Access Edge Computing (MEC) to counter these challenges. This local device-focused framework enhances task-specific models' caching and continual updating across various conditions. In a more specific sense, edge nodes (ENs) operate as MEC, each caching multiple dedicated models and serving as the aggregator as part of the FL process. Additionally, we have engineered two innovative algorithms that categorize various states into multiple classes, thereby ensuring the efficient utilization of computing resources in ENs. Simulation results substantiate the effectiveness of our approach, showing that the proposed dedicated model consistently outperforms a general model designed for all situations.**

***Index Terms*—Multi-access Edge Computing, federated learning, deep learning, connected autonomous vehicles**

I. Introduction

As one specific variant of autonomous driving, connected autonomous vehicles (CAVs) are equipped with sensors, local computing systems, and wireless communication technologies that allow them to communicate with other devices [1]. One important aspect of CAVs is deep learning (DL), which is extensively employed for tasks such as motion control, path planning, and driving scene perception [2]. However, traditional DL model training requires a comprehensive dataset that encapsulates all possible conditions to gain a high accuracy in complex environments [3], [4]. Given that CAVs may function in a variety of locations and times, each presenting an array of situations and weather conditions, it is challenging to create such a complete dataset and train a single model that performs accurately in diverse environmental conditions [5]. Furthermore, if the dataset is augmented later on, the previously trained model would need to be fine-tuned or even entirely retrained, complicating keeping all CAVs models up to date. Another issue is the detrimental impact of imbalanced datasets. For instance, if the images captured during snowy weather are significantly fewer than those taken in clear weather conditions, the influence of these minority images may be overshadowed by the majority, resulting in relatively poorer performance on these minority images. This could lead the model to underperform in rare conditions, increasing the risk of traffic accidents. There needs to be more research regarding this problem of uneven distribution in datasets.

To relieve this problem, one possible idea is to allow CAVs to use dedicated models for different conditions instead of a general model trained on a centralized dataset [6]. This approach means that a specialized model does not need to manage all conditions but just a subset. For example, if the CAVs require a model for drivable area detection, it could be feasible to train a series of dedicated models, such as a model for highway scene in snowy weather, etc. Each CAV would select the dedicated model that best suits its current conditions. By distributing tasks across different models based on conditions, the issue of dataset distribution becomes separate from the training process [7]. It also makes it easy to expand the functionality by training one new dedicated model directly for new conditions. However, under this arrangement, the CAVs would need to retrieve dedicated models frequently, and it also needs one mechanism to train each dedicated model efficiently. Multi-access Edge Computing (MEC) is a technology that facilitates cloud computing at the network's edge by deploying storage and computing resources closer to the end-users [8]. It is an excellent solution to cache the dedicated models in the ENs, which are deployed in the base stations [9]. This ENs provides low latency service to CAVs, enabling them to retrieve the necessary models quickly. Since one EN only handles the requests from one particular area and the storage resource is limited [10], it only caches the models for conditions likely to occur within its service area. Meanwhile, all the dedicated models can be stored in the cloud, which, although assumed to have unlimited resources, needs faster and more reliable communication [11].

As discussed before, it is hard to collect comprehensive data. One possible solution is to leverage the data collected from CAVs. However, transmitting this data to a central server for model training is neither efficient nor privacy-conscious [5]. Instead of this centralized training, a more flexible approach would be to train the model collaboratively by federated learning (FL) [12]. In FL framework, each participant trains a model on its local data. An aggregator server then collects the local models from participants and aggregates them into a single global model [13]. FL enables a collaborative model training on distributed data while retaining data privacy, as the

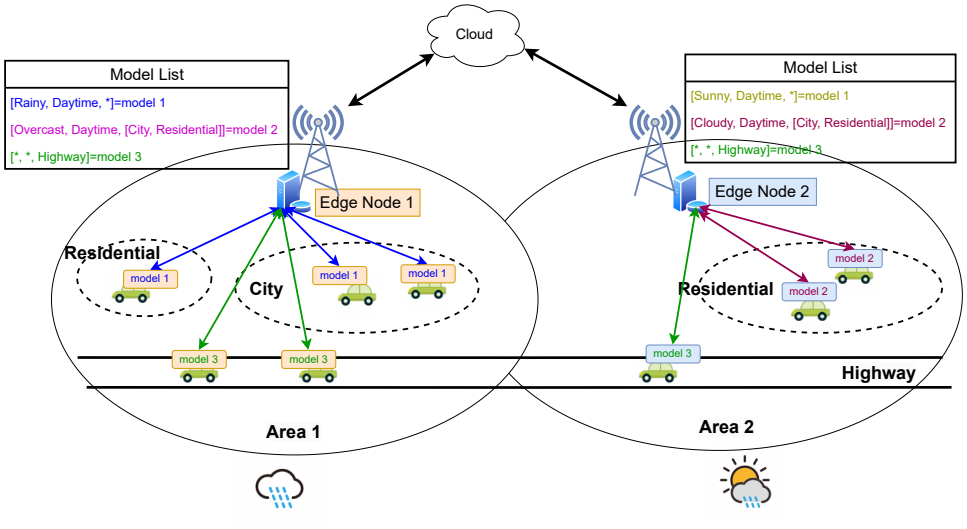

Fig. 1. The overview of the proposed framework. In this example, there are two areas with different weather. Each area has one EN and contains multiple kinds of road scenes such as city, residential, and highway. The model list of one EN stores the condition class and dedicated model pairs.

training data remains local.

In this paper, we aim to design a FL framework based on MEC, thus enabling the caching and updating of dedicated models for various conditions. In our proposed framework, the EN performs as one aggregator and assigns a unique FL process for each dedicated model cached on it. All the CAVs in its area collaborate to train and update these dedicated models. To save the storage resources of ENs, we design two algorithms for categorizing various conditions into multiple condition classes. This allows the EN to cache a single dedicated model applicable to several conditions. We have validated this framework by simulating and assessing model performance on a specific drivable area detection problem. The results demonstrate that the dedicated model performs better than the general model in corresponding conditions, particularly in instances where these conditions are rare. The key contributions of this study are summarized below.

- We propose a MEC-based model caching framework to decouple the condition complexity from the model training process by training dedicated models for different conditions.
- We design two algorithms to classify many conditions into several distinct categories, thus conserving the storage capacity of ENs and reducing the risk of overfitting.
- We simulate to compare the performance of dedicated and general models in the context of a drivable area detection problem.

The remainder of this paper is structured as follows. Section II offers a detailed description of the proposed system model. Subsequently, Section III provides two algorithms of condition classes. Section IV presents the comparative simulation results, followed by the conclusion in Section V.

II. System Model

This section first introduces a proposal framework overview and components, then clarify the FL process conducted by the EN and vehicles.

A. Framework Structure

The framework follows a hierarchical structure with three layers. Figure 1 illustrates the overview of the system that consists of two areas with three condition attributes: weather, time of day, and road scene. The cloud layer represents the cloud computation and storage resources that are distant. This research assumes no limitation of cloud resources, and all the dedicated models could be pre-trained, stored, and synchronized on the cloud platforms. The second layer is the ENs, which is the core component of this framework. Since there are too many conditions for different weather, scene, and time of day, assigning one model for each condition is not feasible. The data of only one condition might not be enough to train a robust model. Therefore, each EN categorized the conditions into a limited number of condition classes. Each condition class is assigned one dedicated model, and the pairs are cached in one model list. The EN also performs as the aggregator in the FL process. The bottom layer includes CAVs operating in different conditions. As shown in Fig. 1, the vehicles operating in multiple conditions might fetch different dedicated models according to the area's model list of the EN. Each vehicle fetches one model and participates in the corresponding FL process to update the model by refining the local data collected by itself. The CAVs can communicate with the EN, and each EN has one stable connection to the cloud.

B. Federated Learning Process

According to the framework design, for each EN there might be multiple active FL processes at the same time since, in an area, multiple conditions could happen concurrently. However,

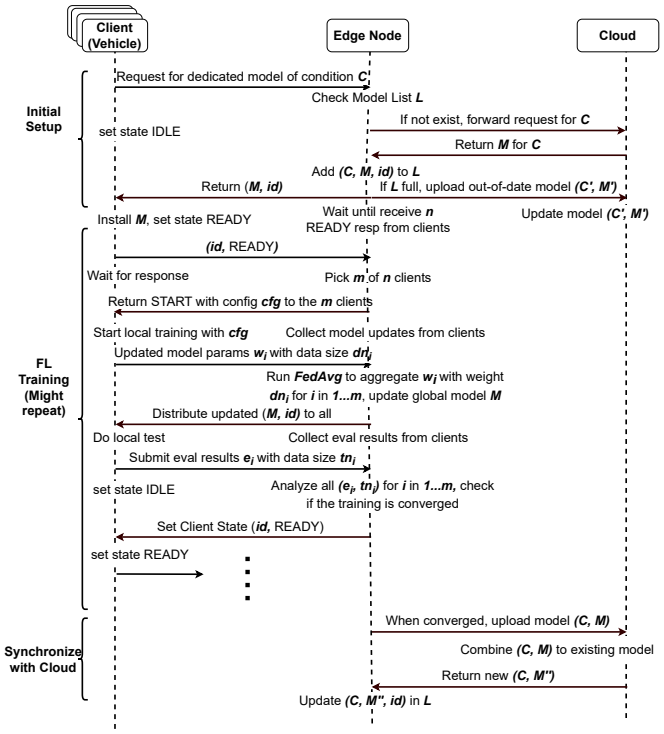

Fig. 2. View of the FL process, mainly three stages: Setup, FL training, and Synchronization. FL training could repeat multiple rounds

they all follow the same procedure, and this section only considers one process. In this research, we use the FedAvg [12] algorithm to aggregate the model updates, but it is possible to implement any other feasible FL algorithm. Figure 2 shows how the framework conducts one FL process to update one dedicated model and interacts with vehicles and the cloud.

As an EN has limited storage resources, it only caches the dedicated models that have been recently used. If the node receives a request for a dedicated model not currently in its cache, it will retrieve it from the cloud and update the model list. This could involve appending it to the end of the list or replacing the least frequently used model with the new one if the cache is full. It is important to note that any replaced model will be uploaded back to the cloud, ensuring that the dedicated models stored in the cloud remain updated.

The FL training follows the classic FL steps. However, after aggregating local updates in each round, the resultant global model is first disseminated and evaluated before commencing the next round. The goal is to enable EN to determine if the training has already converged. Upon convergence, the EN is directed to upload the model to the cloud and subsequently receive the combined model to synchronize its local cache with the cloud.

III. Algorithms of Condition Classes Management

A. Judge and Split Condition Class

As previously discussed, to conserve the storage resources in EN, multiple conditions are categorized into a single condition class, which connects to a dedicated model. This approach allows similar conditions to be grouped within a single condition class, thereby enlarging the training dataset, enhancing model robustness, and reducing the likelihood of overfitting. We design an algorithm to split the conditions based on the evaluation results of the dedicated model, as illustrated in Algorithm 1. This algorithm is based on the observation that the model usually has approximately the same accuracy for similar conditions. To judge and split condition class C, EN first collect evaluation results (e_c^i, n_c^i) from total V vehicle clients representing the metric score e and sample number n of vehicle i in condition c. Then calculate the standard deviation σ by the following equation.

Algorithm 1 Judge and split a single condition class

Input $C, V, \theta, e_c^i, n_c^i, c \in C, i \in V$
Output C_0, C_1 or $None$

```
calculate σ following the Equation 1
if σ > θ then
    Initialize empty list D
    for c in C do
        e_c = Σ_{i∈V} e_c^i × n_c^i
        Append (c, e_c) into D
    end for
    Standardize D   ▷ Standardize data of different format into Scaler.
    D_0, D_1 = kmeans(D, k = 2)   ▷ Run k-means algorithm with k=2.
    Extract C_0, C_1 from D_0, D_1
    Return C_0, C_1
else
    return None
end if
```

$$\begin{cases} N = \sum_{c,i} n_c^i \\ e_c = \frac{\sum_i e_c^i}{N} \\ \mu = \frac{\sum_c e_c}{N} \\ \sigma = \sqrt{\frac{\sum_c (e_c - \mu)^2}{N}} \end{cases} \quad c \in C, i \in V \tag{1}$$

where N represents the total number of evaluation samples, while μ denotes the population mean. If the σ is larger than a preset threshold θ, the condition class C is divided into two smaller ones C_0 and C_1. Currently, we use the mean Intersection over Union (mIoU) metric [14], while the clustering algorithm is the classic k-means algorithm [15] where $k = 2$.

B. Judge and Merge Condition Classes

More than having the strategy to split large condition classes is required, as it results in an increasing number of condition classes, which can be equivalent to the number of conditions. Consequently, it is necessary to design a strategy to merge similar condition classes to achieve a balance. However, determining the similarity of models solely based on their parameters is challenging, and it is impractical to have vehicles attempt models that do not match their respective conditions [16]. Thus, the most practical approach is to provide the

Algorithm 2 Judge and merge two condition classes

Input $C_0, C_1, \theta, \mathcal{E}, model\mathcal{M}_0, \mathcal{M}_1$
Output C or $None$

Extract data samples $E = \mathcal{E}(C_0|C_1)$ from $\mathcal{E}$
$N =$ size of E
total metric score $s = 0$
for j in 1...N **do**
  $e_{j,0}, e_{j,1} = \mathcal{M}_0(E_j), \mathcal{M}_1(E_j)$
  calculate $e_j = mIoU(e_{j,0}, e_{j,1})$ ▷ Here we use mIoU to represent the similarity of the two outputs. Other metrics might also work.
  $s = s + e_j$
end for
calculate mean score $\overline{s} = \frac{s}{N}$
if $\overline{s} > \theta$ **then**
  return $C_0|C_1$ as C
else
  return $None$
end if

TABLE I
BDD100K IMAGE CONDITION ATTRIBUTES

Attribute	Arguments	Amount
weather	rainy, snowy, clear, overcast, undefined, partly cloudy, foggy	7
scene	tunnel, residential, parking lot, undefined, city street, gas stations, highway	7
timeofday	daytime, night, dawn/dusk, undefined	4

models with some data and calculate the similarity based on their outputs. Fortunately, through experiments, we have found that it only requires a small piece of data to distinguish the performance of models in different conditions. Based on this observation, we have designed a preliminary Algorithm 2 to judge and merge condition classes. One trade-off is that the algorithm requires a small *eval dataset* $\mathcal{E}$ stored in EN for calculating the similarity. However, since $\mathcal{E}$ contains only a limited number of samples for several conditions, it does not consume substantial storage resources, especially compared to the storage consumed by model caching.

IV. SIMULATION

A. Simulation Setup

To evaluate the performance of the proposed framework, we conducted the simulation using the BDD100K dataset, focusing on the drivable area detection task [17]. The BDD100K is a large-scale dataset comprising images captured under diverse conditions, with annotations available for various tasks. Each image in the dataset has three condition attributes, which are shown in Table I, enabling us to split the dataset based on these conditions. To ensure evaluation accuracy evaluation, we excluded conditions with a limited number of images, i.e., specifically conditions with fewer than 30. For the simulation, we utilized the Flower FL framework [18], which is a user-friendly Python library that supports flexible customization and simulated FL on a single machine. The setup consisted of one EN and 10 CAVs operating under random conditions. The

TABLE II
SIMULATION PARAMETER

Parameter	Description	Value
lr	Learning rate	0.0001
n	Number of vehicles	10
m	Active vehicles for one round	5
R	Maximum number of rounds	10
θ_1	Threshold of Algorithm1	0.05
θ_2	Threshold of Algorithm2	0.8

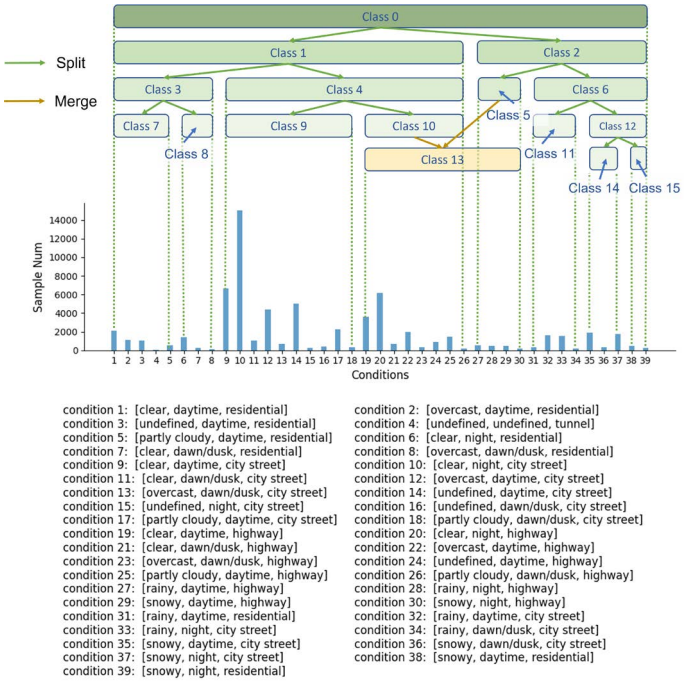

Fig. 3. The training data distribution and the range of each condition class

EN functioned as the Flower server, serving as the aggregator, while each vehicle ran the Flower client to build a connection with the server. Every vehicle is assigned 300 BDD100K images of a particular condition class in every round. These images in each vehicle were randomly selected from all the samples of one specific condition, imitating that the vehicle operates in this condition. We used the deeplabv3 model [19], [20], while the benchmark was a general model of the same type pre-trained on the whole BDD100K dataset.

The specific simulation parameters are detailed in Table II. The simulation follows the process illustrated in Fig. 2, where evaluation results are cached. Algorithm 1 is executed to attempt splitting the condition class once the corresponding dedicated model reaches convergence. In this simulation, when the number of condition classes achieves 8, the EN ran the Algorithm 2 to endeavor to merge the condition classes.

B. Simulation Results

Figure 3 shows the training data distribution across all conditions, which illustrates the process of splitting and merging condition classes step by step. Analyzing the number of training samples of various conditions makes it easy to find that the data is unevenly distributed. The conditions of the city

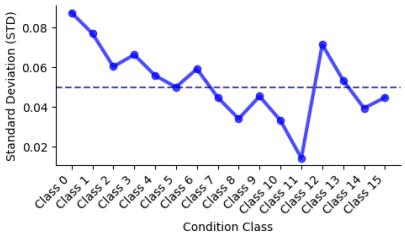

Fig. 4. Standard deviations following the Equation 1 on all condition classes

TABLE III
SIMILARITY BETWEEN DEDICATED MODELS

Model 1	Model 2	Similarity (mIoU)
Class 5	Class10	0.82
Class 10	Class 11	0.70
Class 10	Class 14	0.70
Class 8	Class 9	0.69
...	...	...
Class 7	Class 14	0.50

street scene (condition 9 – condition 18) dominate most of the dataset. However, the amount of samples is much smaller for some relatively rare weather, such as rainy weather or snowy weather (condition 27 – condition 39).

We designed the Algorithm 1 and established a preset threshold of $\theta_1 = 0.05$ to facilitate the reasonable splitting of conditions. Figure 4 depicts the standard deviation (STD) values of all condition classes. Classes with STD values exceeding θ_1 are divided into two smaller classes. The splitting process is illustrated in Fig. 3, where the green arrows represent the splitting operation.

More than splitting conditions with a strategy for merging condition classes is required. Therefore we propose the Algorithm 2 for balancing. The threshold is $\theta_2 = 0.8$, meaning that one condition class pair with a similarity higher than θ_2 should be merged. Table III illustrates the rank of similarities between two condition classes. The similarity of condition classes 5 and 10 is 0.82, which is much higher than all the other condition class pairs and exceeds θ_2. Thus they are merged into condition class 13.

After fully splitting and merging, the conditions are categorized into seven classes (classes 7, 8, 9, 11, 13, 14, 15). Note that the sample quantities of condition classes also differ significantly. However, since they use different dedicated models, this variety of data does not affect the training. By leveraging the dedicated model, the negative effect of uneven data distribution is relieved to a certain extent. In practice, the conditions with high similarity are expected to categorize into one condition class since one dedicated model can work well on all these conditions with a low accuracy divergence. In the simulation, condition class 9 finally covers almost all the images taken in city streets, except those images taken in rainy or snowy weather. Moreover, the various rare conditions are also divided into several condition classes, such as class 7,8,14,15. Therefore, by leveraging the two algorithms for splitting and merging, the framework can categorize conditions reasonably and significantly reduce the number of dedicated

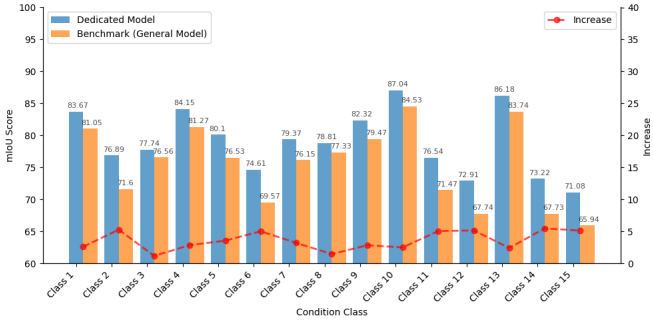

Fig. 5. Comparision of the performance between the pre-trained general model and the dedicated model on every condition class

models required to cover all the conditions, saving the cache resource of ENs.

Figure 5 shows the performance of each dedicated model on its corresponding condition class and compares it with the benchmark, which is the pre-trained general model on the whole BDD100K dataset [17]. The results indicate that the dedicated model performs better on specific condition classes than the general model, especially on those rare condition classes. In classes 11, 14, and 15, the dedicated models outperform approximately 0.05 higher than the benchmark, significantly increasing the accuracy in these conditions. The reason is that these condition classes do not include much data compared to other majority classes, such as class 9 or condition 10, according to the data distribution shown in Fig. 3. Therefore, they are not considered much when training a general model on the whole dataset since the loss items on these data do not contribute much to the average loss overall. In our real life, these conditions, such as snowy/rainy in the dawn in residential area, happen relatively more rarely, too. Another observation is that the accuracy of condition classes also differs. It might be because the image complexity of different conditions varies a lot. It could be much harder for the model to correctly recognize one image of [snowy, night, residual] than an image of [clear, daytime, highway]. The amount of training data could also impact performance. One frequent-used and intuitive method for optimization is to build a larger dataset with more data. However, in this research, we do not dive too much into this aspect since we find that the quantity is not the main factor, considering that some classes, such as classes 7 and 8, also do not include many examples but still retain the performance well.

C. Discussion and Limitation

The simulation results prove the feasibility of the proposed framework. However, this research still has limitations and open problems, which will be briefly discussed in this section.

While the splitting and merging strategies effectively control the number of dedicated models, the parameters θ_1 and θ_2 must be appropriately set. Currently, We lack a dynamic and automatic method for adjusting them. Another limitation is that in current simulation we only tested the drivable area detection scenario, but in this scenario the merging does not happen frequently. In future more scenarios will be considered

to find possible weaknesses of the strategies and optimize them accordingly. One possible problem is the dead loop of splitting and merging, i.e., the condition class just merged from two subclasses is judged to be split. There could be some additional strategy to link the two strategies to synchronize the settings, but we leave it as our future work.

Another area for improvement is efficiency. After analyzing the model performance shown in Fig. 5, we found that the first round of splitting could significantly improve the accuracy already. The further splitting does not continuously improve much, i.e., class 1 is split into class 3 and class 4, but the dedicated model only achieves a finite score increase. So there is a trade-off between efficiency and dedicated model splitting.

Furthermore, in current stage we have not developed much on the cloud side, only using it for dedicated model backup. However, it is very promising to leverage the cloud resources for more complex and diverse functionalities.

V. Conclusion

In this research, we introduced a unified MEC-based model caching framework with FL to enable a more flexible and robust model training system for CAVs. The proposed framework enables the caching of dedicated models in ENs, allowing vehicles to utilize particular models for their current conditions. In order to prevent the creation of an excessive number of dedicated models, we designed two algorithms that help conserve storage resources and enhance training efficiency by logically splitting and merging the conditions. To evaluate the performance and demonstrate its operational flow, we tested the drivable area detection task based on BDD100K dataset. We simulated it via the Flower FL framework. The simulation results show that the proposed framework can outperform the general model, which is fully trained in the whole dataset without considering the uneven distribution of conditions, especially in rare conditions, such as rainy or snowy weather, where the dedicated models achieve a larger score increase. Therefore, our framework promises to provide robust and high-quality models for uncommon geographical scenes or weather conditions. By leveraging the designed algorithms, the number of dedicated models is significantly reduced without sacrificing the model performance.

In future work, we will conduct more comprehensive simulations in different scenarios and optimize the algorithms toward the shortcomings we have already found.

Acknowledgements

These research results were obtained from the commissioned research by the National Institute of Information and Communications Technology (NICT), JAPAN.

References

[1] T. Zeng, O. Semiari, M. Chen, W. Saad, and M. Bennis, "Federated Learning on the Road Autonomous Controller Design for Connected and Autonomous Vehicles," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 10 407–10 423, 2022.

[2] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[3] D. Chen, C. S. Hong, L. Wang, Y. Zha, Y. Zhang, X. Liu, and Z. Han, "Matching-theory-based low-latency scheme for multitask federated learning in mec networks," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 415–11 426, 2021.

[4] M. Asad, S. Shaukat, E. Javanmardi, J. Nakazato, and M. Tsukada, "A Comprehensive Survey on Privacy-Preserving Techniques in Federated Recommendation Systems," *Applied Sciences*, vol. 13, no. 10, 2023.

[5] L. Fantauzzo, E. Fanì, D. Caldarola, A. Tavera, F. Cermelli, M. Ciccone, and B. Caputo, "FedDrive: Generalizing Federated Learning to Semantic Segmentation in Autonomous Driving," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 11 504–11 511.

[6] M. Ahmed, M. A. Mirza, S. Raza, H. Ahmad, F. Xu, W. U. Khan, Q. Lin, and Z. Han, "Vehicular Communication Network Enabled CAV Data Offloading: A Review," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[7] S. Sharma, L. Dabbiru, T. Hannis, G. Mason, D. W. Carruth, M. Doude, C. Goodin, C. Hudson, S. Ozier, J. E. Ball *et al.*, "Cat: Cavs Traversability Dataset for Off-Road Autonomous Driving," *IEEE Access*, vol. 10, pp. 24 759–24 768, 2022.

[8] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[9] J. Nakazato, M. Nakamura, T. Yu, Z. Li, K. Maruta, G. K. Tran, and K. Sakaguchi, "Market Analysis of MEC-Assisted Beyond 5G Ecosystem," *IEEE Access*, vol. 9, pp. 53 996–54 008, 2021.

[10] J. Nakazato, M. Kuchitsu, A. Pawar, S. Masuko, K. Tokugawa, K. Kubota, K. Maruta, and K. Sakaguchi, "Proof-of-Concept of Distributed Optimization of Micro-Services on Edge Computing for Beyond 5G," in *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, 2022, pp. 1–6.

[11] A. Daoud, G. Picard, H. Alqasir, P. Gianessi, and F. Balbo, "Communication-wise Comparison of the Online Resource Allocation Methods in CAV Fleets," *Procedia Computer Science*, vol. 220, pp. 299–306, 2023.

[12] B. McMahan, E. Moore, D. Ramage, and others, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Artif. Intell.*, 2017.

[13] E. the Communication Efficiency in Federated Learning Algorithms, "Evaluating the communication efficiency in federated learning algorithms," in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2021, pp. 552–557.

[14] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation." [Online]. Available: http://arxiv.org/abs/1704.06857

[15] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[16] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 713–10 722.

[17] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100k: A Diverse Driving Dataset for Heterogeneous Multitask Learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2633–2642.

[18] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, "Flower: A Friendly Federated Learning Research Framework."

[19] C. Chen, C. Wang, B. Liu, C. He, L. Cong, and S. Wan, "Edge Intelligence Empowered Vehicle Detection and Image Segmentation for Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2023.

[20] S. Das, A. A. Fime, N. Siddique, and M. M. A. Hashem, "Estimation of Road Boundary for Intelligent Vehicles Based on DeepLabV3+ Architecture," *IEEE Access*, vol. 9, pp. 121 060–121 075, 2021.