A Hierarchical Federated Continual Learning Framework for Dynamic and Heterogeneous IoV

Yiming Chen, *Student Member, IEEE*, Celimuge Wu, *Senior Member, IEEE*, Lei Zhong, *Member, IEEE*, Yangfei Lin, *Member, IEEE*, Zhaoyang Du, *Member, IEEE*, Wugedele Bao, and Peter Han Joo Chong, *Senior Member, IEEE*

***Abstract*—Traditional federated learning (FL) architectures face challenges in handling heterogeneous data and dynamic tasks, often resulting in catastrophic forgetting when new training tasks are continuously introduced. Federated continual learning (FCL) integrates the privacy-preserving capabilities of FL with the knowledge retention and incremental update mechanisms of continual learning, effectively mitigating catastrophic forgetting and protecting user privacy. However, existing FCL solutions largely overlook the unique requirements of Internet of Vehicles (IoV) scenarios, such as data heterogeneity and dynamic task management. To address these challenges, we propose a novel framework, hierarchical federated continual learning (Hier-FCL), which incorporates local continual learning via optimized experience replay and meta-knowledge distillation, along with dynamic client clustering to tackle data heterogeneity. Additionally, a hierarchical aggregation mechanism is employed to enhance scalability and adaptability in diverse IoV scenarios. Experiments conducted in mixed-task environments using multiple datasets demonstrate that Hier-FCL outperforms baseline algorithms in terms of retained accuracy and backward transfer impact, validating its effectiveness in mitigating catastrophic forgetting and managing heterogeneous client data.**

***Index Terms*—Continual learning (CL), federated learning (FL), Internet of Vehicles (IoV), meta-learning.**

Received 31 December 2024; revised 21 May 2025 and 15 October 2025; accepted 19 December 2025. This work was supported in part by the Inner Mongolia Autonomous Region Key Research & Development and Achievement Transformation Project under Grant 2025YFHH0215; in part by JST ASPIRE under Grant JPMJAP2325; in part by JSPS KAKENHI under Grant 24K02937; and in part by collaborative research with Toyota Motor Corporation. *(Corresponding author: Celimuge Wu.)*

Yiming Chen, Yangfei Lin, and Zhaoyang Du are with the Department of Computer and Network Engineering, The University of Electro-Communications, Tokyo 182-8585, Japan (e-mail: c2341005@gl.cc.uec.ac.jp; linyangfei@uec.ac.jp; duzhaoyang@uec.ac.jp).

Celimuge Wu is with the Meta-Networking Research Center, The University of Electro-Communications, Tokyo 182-8585, Japan (e-mail: celimuge@uec.ac.jp).

Lei Zhong is with Toyota Motor Corporation, Tokyo 100-0004, Japan (e-mail: lei.zhong@toyota.global).

Wugedele Bao is with the Trustworthy Big Data & AI Academician Workstation, Hohhot Minzu College, Hohhot, 010051, China (e-mail: wugedele.imnc@gmail.com).

Peter Han Joo Chong is with the Department of Electrical and Electronic Engineering, Auckland University of Technology, Auckland, 1142, New Zealand (e-mail: peter.chong@aut.ac.nz).

Digital Object Identifier 10.1109/TCSS.2025.3649065

I. Introduction

As an important technical support for modern travel, the Internet of Vehicles (IoV) has greatly improved the travel experience through real-time navigation, personalized services, and autonomous driving. These are driven by the massive data generated by vehicle sensors, personal devices, and transportation infrastructure. However, these data often contain sensitive and private information, making privacy protection and security critical concerns. To address these concerns, federated learning (FL) offers a distributed learning paradigm where models are trained locally on devices, requiring only the exchange of model parameters rather than raw data. This approach facilitates multiparty data sharing while safeguarding privacy. Additionally, recent advancements in networking architectures [1], [2] have improved communication efficiency and security performance, making FL a compelling choice for IoV applications. Despite these advantages, IoV environments are characterized by task dynamics and data heterogeneity, which exacerbate issues such as catastrophic forgetting. Traditional FL methods struggle to meet these demands, necessitating a solution that can continuously acquire new knowledge while preserving previously learned knowledge.

Continual learning (CL) presents a viable solution, inspired by the way humans learn incrementally. CL enables devices to update models using new data without storing or retraining on all historical data, thereby reducing storage and computation requirements. Integrating CL into FL frameworks can effectively address challenges arising from task dynamics and data heterogeneity in IoV scenarios.

Several federated continual learning (FCL) approaches have been proposed to solve practical problems. Li et al. [3] proposed an FCL framework called cTD-αMAML to solve the problem of electroencephalogram-based biometrics. Usmanova et al. [4] proposed a distillation-based FCL framework to solve human activity recognition classification. Although these methods effectively tackle generalization and catastrophic forgetting in their respective fields, they fall short in highly dynamic environments such as IoV. Traditional FL architectures cannot adequately accommodate IoV, which includes sensors, vehicles, base stations, and cloud servers, and even involves cross-organizational systems. Hierarchical federated learning (HFL) [5] can achieve model aggregation and updating between

different layers, improving the flexibility and scalability of the system. It offers a more suitable alternative by enabling model aggregation and updates across layers. By utilizing flexible edge servers to manage and aggregate regional models, HFL reduces reliance on cloud servers and enhances cross-organizational collaboration.

To address the challenges of privacy protection, data heterogeneity, and dynamic task management in IoV, we propose a hierarchical federated continual learning (Hier-FCL) framework. Designed for complex IoV environments, this framework is tailored to handle highly dynamic and heterogeneous data. The main contributions of this article are as follows.

1) We propose a Hier-FCL framework consisting of a three-layer architecture of cloud server, edge servers, and clients, aiming to apply FCL to complex IoV environments.
2) To manage dynamic changes in client tasks, we propose a client grouping strategy based on dynamic clustering and enhance model performance through hierarchical aggregation.
3) To address task heterogeneity in IoV scenarios, we use improved experience replay and meta-knowledge distillation, optimizing local continual learning for heterogeneous data.
4) We extend the traditional single dataset task division approach by designing a heterogeneous task division method based on multiple datasets, enabling more realistic and representative evaluations.
5) Experimental results show that Hier-FCL significantly outperforms the baseline algorithm in terms of retained accuracy and backward transfer impact, and exhibits significant robustness and performance advantages when dealing with multitask heterogeneous scenarios, providing strong support for solving similar problems.

The remainder of this article is organized as follows. Section II reviews related work on HFL, CL, and FCL. Section III introduces the proposed Hier-FCL framework, detailing its architecture, key components, and mechanisms. Section IV first describes the experimental setup, including datasets, task configurations, and evaluation metrics. Then, we present the experimental results and provide a comprehensive analysis of the framework's performance compared with baseline algorithms. Finally, Section V concludes the article and discusses potential future research directions.

II. Related Work

A. HFL

FL is a distributed learning paradigm that protects data privacy by aggregating local model updates instead of raw data. Classic strategies such as FedAvg [6] use weighted averaging of local updates. Numerous studies aim to improve FL by addressing client selection [7], [8], communication security [9], and personalization [10].

However, traditional FL architectures struggle in dynamic, multilayer settings such as IoV, which involve hierarchical and heterogeneous data sources. To address this, Liu et al. [5] proposed HierFAVG by introducing edge servers between clients and the cloud. Building on this concept, researchers have proposed enhancements to HFL. For example, Zhou et al. [11] used Gaussian mixture models for user grouping in medical IoT. Wu et al. [12] designed an asynchronous HFL framework, and De Rango et al. [13] proposed HED-FL with adaptive learning rounds to reduce communication cost.

Despite advancements in privacy protection and communication optimization, these FL methods remain insufficient for highly dynamic IoV scenarios with strong data heterogeneity. Enhancing adaptability and model performance in such environments remains an open challenge.

B. Continual Learning

Continual learning enables models to adapt to new tasks while preserving knowledge from previous tasks. The primary challenge in CL is mitigating catastrophic forgetting and reducing negative transfer between tasks. At present, CL methods can be mainly divided into five categories [14]: 1) regularization; 2) parameter isolation; 3) experience replay; 4) knowledge distillation; and 5) meta-learning. Regularization [15] generally adds additional regularization terms to the loss function to limit the change of model parameters, which is suitable for retaining old knowledge under stable data distribution; Parameter isolation [16] sets a dedicated parameter interval to isolate the parameters of different tasks, which is mainly suitable for scenarios with large task differences; Experience replay [17] saves old task samples and repeats training while training new tasks. This can effectively deal with dynamic data distribution and scenarios with large data distribution; knowledge distillation methods [18] regard the old task model as the teacher model and the new task model as the student model. By guiding students through teachers, they retain experience while learning new knowledge; Meta-learning-based methods [19] are a new CL scheme that has emerged in recent years. It can quickly adapt to new tasks through small samples or a small number of training steps, and achieve gradient alignment between multiple tasks [19] to reduce interference between tasks. Among these, experience replay and meta-learning are particularly effective in dynamic and heterogeneous scenarios, making them suitable for IoV environments.

C. FCL

FCL combines the advantages of FL and CL, aiming to protect data privacy while maintaining compatibility with both new and old tasks. In FCL, clients manage local task sequences and perform continual learning locally before sharing their models for federated aggregation.

Several FCL frameworks have been proposed to tackle specific challenges. Inspired by MER [19], Li et al. [3] proposed a personalized FCL framework called cTD-αMAML to achieve adaptive learning of the model through meta-learning and knowledge distillation. To reduce the communication cost of

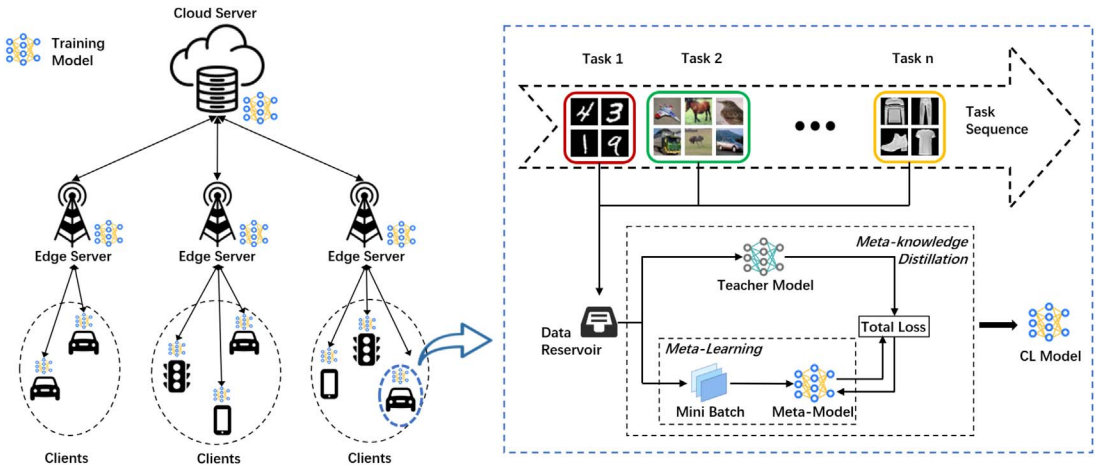

Fig. 1. Hier-FCL framework.

model transmission, FedWeIT [20] decomposed model parameters into global shared parameters and sparse task-specific parameters, reducing the interference of irrelevant tasks. For asynchronous FL scenarios, Fed-Cprompt [21] used asynchronous prompt learning and contrastive loss to improve knowledge transfer efficiency without replay. To meet the needs of replay-free learning, Zhang et al. [22] proposed a sample-free distillation method based on knowledge distillation and global distribution data generation to retain old task knowledge without relying on sample storage.

While FCL has demonstrated effectiveness in multitask environments, existing methods face challenges such as simplistic task generation, balancing knowledge retention with new learning, and adapting to multilayer architectures. These limitations highlight the need for an HFL framework that integrates continual learning to address the unique demands of IoV scenarios.

III. Proposed Hier-FCL Architecture

This section provides an in-depth analysis of how to perform FCL in a multilayer IoV environment. First, we introduce the framework of Hier-FCL, followed by a comprehensive explanation of its technical components and implementation details.

A. Overall Framework

To achieve efficient cooperative training and effectively reduce the interference and catastrophic forgetting problems between tasks while protecting the privacy of customer data, we propose a Hier-FCL based on improved experience replay and meta-knowledge distillation.

As illustrated in Fig. 1, the framework consists of three layers: 1) a cloud server; 2) multiple edge servers; and 3) a large number of clients.

1) Client Layer: This layer comprises various IoT devices such as vehicles, traffic lights, and smartphones, which act as clients. Continual learning at the client layer is realized through a combination of experience replay and meta-knowledge distillation strategies, as shown on the right side of Fig. 1. Key samples from previous tasks are stored in the data reservoir and replayed to reduce catastrophic forgetting. Mini batches, derived from the data reservoir, are used in meta-learning to facilitate the rapid adaptation and generalization of new tasks. The teacher model, which is the model trained on previous tasks, is employed in the process of meta-knowledge distillation to integrate old knowledge with new learning and compute the total loss.

2) Edge Server Layer: Edge servers, such as base stations, dynamically cluster clients with similar data distributions for localized training, improving model aggregation quality. To preserve client-specific knowledge, the model is partitioned into two components: 1) *Private Layer:* Trained exclusively on clients and edge servers, retaining personalized knowledge; and 2) *Global Shared Layer:* Aggregated across all clients and edge servers to create a universal representation. Only the shared layer participates in global aggregation, ensuring data privacy and model personalization.

3) Cloud Server Layer: At the cloud layer, global aggregation is performed across all edge servers, leveraging their aggregated models. This hierarchical architecture achieves effective knowledge sharing and collaborative training, minimizes catastrophic forgetting from task dynamics, and ensures robust privacy protection.

Through this HFL architecture, we can not only achieve effective knowledge sharing and collaborative training, but also minimize the catastrophic forgetting problem caused by dynamic changes in tasks. Moreover, by limiting frequent global communication and aggregating updates at the edge layer, the architecture helps reduce communication overhead in bandwidth-constrained IoV environments [5], [12]. In the following sections, we detail the problem formulation, optimization objectives, and technical methodologies underlying Hier-FCL.

B. Problem Definition

In the context of continual learning, the goal is to train a model on a sequential stream of tasks $\{T_1, T_2, \ldots, T_{t_n}\}$, where

each task is represented by a dataset $D_t = \{X_t, Y_t\}$, representing the input data X_t and corresponding labels Y_t. At any given time step, the model is exposed to a single task T_t, and access to data from previous tasks $T_{1:t-1}$ is typically restricted. Consequently, the challenge lies in retaining the knowledge acquired from earlier tasks while effectively learning new tasks, without relying on storing all past task data. The optimization goals of continual learning can be formulated as follows:

$$\min_{\theta_t} \mathcal{L}(\theta_t; \theta_{1:t-1}, D_t) \tag{1}$$

where θ_t is the model parameter of the current task, $\theta_{1:t-1}$ is the model parameter of all previous tasks, and D_t is the current task data. When the model trains the current task T_t, it must minimize the loss of the current task while taking into account the model parameters of all previous tasks.

When applied to an HFL framework, continual learning must account for the multilayered structure consisting of clients, edge servers, and a cloud server [5]. Each layer operates as follows.

1) Client Layer: Each client k processes its local task sequence $\{T_k^1, T_k^2, \ldots, T_k^{t_n}\}$ and trains a local continual learning model θ_k. The client's training focuses on adapting to new tasks while retaining knowledge from earlier ones, subject to the constraints of limited data and computational resources.

2) Edge Server Layer: Edge servers aggregate the models from clients with similar data distributions, generating a region-specific intermediate model. These intermediate models balance local personalization with broader generalization, and are subsequently forwarded to the cloud server.

3) Cloud Server Layer: The cloud server aggregates the intermediate models received from edge servers to construct a global model θ. This hierarchical aggregation process allows for scalable and efficient training across multiple layers.

The global optimization objective for HFL can be expressed as

$$\min_{\theta} \sum_{e=1}^{E} \sum_{k=1}^{K} \frac{N_k}{N} \mathcal{L}_k(\theta) \tag{2}$$

where E is the number of edge servers, K is the number of clients, N is the total global data volume, N_k is the data volume of the kth client, θ is the model parameter, and $\mathcal{L}_k(\theta)$ is the local loss function on the kth client.

C. Continual Learning Based on Experience Replay and Meta-Knowledge Distillation

According to the definition of continual learning, the data of the local client is not obtained all at once, but is continuously obtained in the form of a task stream. Therefore, traditional training methods often result in catastrophic forgetting, where parameters of new tasks overwrite those of prior tasks, leading to degraded performance on earlier tasks. The objective of continual learning is to balance the acquisition of new knowledge with the retention of old knowledge. This article employs two core strategies to achieve this: 1) experience replay; and 2) meta-knowledge distillation.

1) Experience Replay Strategy: As mentioned earlier, common continual learning methods include regularization, parameter isolation, experience replay, knowledge distillation, and meta-learning [14]. Experience replay allows the model to relearn old knowledge later by retaining old samples, which is more applicable when the task heterogeneity is significant. In addition, it can be combined with other methods, such as meta-knowledge distillation mentioned later, to further improve the performance of continual learning.

In this article, we use improved reservoir sampling to implement experience replay. The core idea of reservoir sampling is to manage a storage pool M with a maximum capacity of M_{max} for experience replay. This pool stores samples observed in the past and replaces them appropriately when new samples are observed to ensure that both new and old samples can be trained. In traditional reservoir sampling, the probability of each sample being selected is equal. However, simple random sampling cannot fully utilize the key information in the data, especially when the sample quality varies greatly. Therefore, we modify the replacement rules for new and old samples and introduce sample quality and category diversity as important considerations.

Specifically, we calculate a comprehensive score for each sample, which combines two key dimensions: 1) sample quality; and 2) category representativeness. Sample quality reflects the importance of the sample in the model. For an image classification problem, we can use a convolutional neural network to extract the feature vector of the image, and then calculate the cosine similarity between the sample and the average feature of its category, that is, the class center, and use the similarity as the sample quality score $Score_Q$. Category representativeness is used to ensure that samples of different categories are retained evenly, and can be calculated by the inverse of the number of samples of this category in the pool, that is, $Score_C = 1/\textit{remaining samples}$. Therefore, the replacement score calculation formula of new and old samples is as follows:

$$Score_{old} = w_1 \cdot Score_{Q\_old} + w_2 \cdot Score_{C\_old} \tag{3}$$

$$Score_{new} = w_3 \cdot Score_{Q\_new} + w_4 \cdot Score_{C\_new} \tag{4}$$

where $Score_{Q\_old}$ and $Score_{Q\_new}$ are the quality scores of the old and new samples, respectively. $Score_{C\_old}$ and $Score_{C\_new}$ are the category scores of the old and new samples, respectively. Based on the replacement scores of the new and old samples mentioned above, through a probability function such as the sigmoid function, we dynamically decide whether to replace the old samples in the pool.

The specific process of improving reservoir sampling is shown in Algorithm 1. When the reservoir is not full, the new sample is added directly. When the reservoir is full, the algorithm first checks whether the number of samples in the same category of the new sample is lower than the minimum threshold. In such a case, the old samples of the category with more samples are replaced first. Otherwise, we calculate the quality scores of the new and old samples. Then, based on the difference in their scores, the replacement probability is calculated by the sigmoid function to decide whether to replace.

Algorithm 1: Improved Reservoir Sampling.

Require: Old reservoir memory array M^{t-1}, maximum reservoir size M_{max}, current reservoir memory size S_M, new sample data x, new sample label y, minimum number of samples in a label C_{min}.

Ensure: New reservoir memory M^t.

if $S_M < M_{max}$ **then**
  $M^{t-1}[S_M] \leftarrow (x, y)$
else
  if the number of new sample label is less than minimum number **then**
    Randomly select a j whose number of samples in this label is greater than C_{min}.
    $M^{t-1}[j] \leftarrow (x, y)$
  else
    Randomly select j from $(0, S_M)$.
    Calculate the old sample score $Score_{old} = w_1 \cdot Score_{Q\_old} + w_2 \cdot Score_{C\_old}$
    Calculate the new sample score $Score_{new} = w_3 \cdot Score_{Q\_new} + w_4 \cdot Score_{C\_new}$
    Calculate replace probability
    $p = \frac{1}{1+e^{-(Score_{new} - Score_{old})}}$
    if $random() < p$ **then**
      $M^{t-1}[j] \leftarrow (x, y)$
    end if
  end if
end if

This method calculates the replacement probability based on the difference in the scores of the new and old samples, making the replacement decision more flexible. The computational cost of scoring is $O(M_{max}d)$ per update, where d is the feature dimension, and it operates entirely on local data.

2) Meta-Knowledge Distillation: In continual learning, experience replay strategies have been able to effectively preserve old knowledge. However, due to limited storage space, the total number of stored samples remains small, even if they are of high quality and diversity. Therefore, to quickly adapt and learn on limited samples, we introduced a meta-learning method when performing continual learning on the client. The core idea of meta-learning is learning how to learn, that is, by training the model on small sample data, it can quickly adapt and update when facing new tasks. Model-agnostic meta-learning algorithm (MAML) [23] is a classic meta-learning method, especially suitable for small sample scenarios and environments that quickly adapt to task changes. In this article, we use FOMAML [23], a first-order variant of MAML. Compared with MAML, FOMAML ignores the calculation of second-order derivatives, which greatly improves the computational efficiency, and can still achieve the same accuracy as MAML on many tasks. The training process of FOMAML is divided into two stages: 1) inner training; and 2) outer training.

a) Inner training: The goal of the inner training stage is to perform fast training on the meta-task, allowing the model to learn features of the current meta-task. We randomly sample a mini-batch of data $D_k^i = \{X_k^i, Y_k^i\}$ from the current client's reservoir memory M_k^t, which serves as a meta-task τ_k^i. Then, based on the current model θ_k, we compute the inner loss $\mathcal{L}_{\text{inner}}(D_k^i; \theta_k)$ on the meta-task. Next, we use one step of gradient descent to update the model parameters

$$\theta_k^{i'} = \theta_k^i - \alpha \nabla_\theta \mathcal{L}_{\text{inner}}(D_k^i; \theta_k) \tag{5}$$

where α is the learning rate for inner training. θ_k^i represents the model parameters of client k on the current meta-task τ_k^i. After N_{meta} gradient updates epochs, the inner training stage ends.

b) Outer training: The outer training is trained between meta-tasks, with the goal of enabling the model to quickly adapt to new tasks. Therefore, we first sample multiple meta-tasks for inner training, then compute the outer loss $\mathcal{L}_{\text{outer}}(\theta_k; D_k)$ based on the loss values of all meta-tasks. Using this loss, the client's model parameters are updated with the final gradient to obtain the new model parameters

$$\begin{aligned}\theta_k' &= \theta_k - \beta \nabla_\theta \mathcal{L}_{\text{outer}}(D_k; \theta_k) \\ &= \theta_k - \frac{1}{N_\tau} \beta \sum_{i=1}^{N_\tau} \nabla_\theta \mathcal{L}_{\text{inner}}(D_k^i; \theta_k)\end{aligned} \tag{6}$$

where β is the learning rate for outer training. N_τ is the number of all meta-tasks. At this point, a complete client-side meta-update has been performed. After several meta-updates, clients can upload model for the next round of aggregation.

To further reduce the loss of knowledge from old tasks during meta-updates, we introduce an attention-based knowledge distillation method. Knowledge distillation is an algorithm that transfers knowledge using a "teacher–student model" structure. Specifically, we treat the model of the old tasks as the teacher model, and the model of the new tasks as the student model. Teacher model computes the teacher logit outputs $y_k^{t-1} = f(X_k; \theta_k^{t-1})$ for each sample based on the current meta-task data $\{D_k^1, D_k^2, \ldots, D_k^{N_\tau}\}$. Similarly, the student model computes the student logit outputs $y_k^t = f(X_k; \theta_k^t)$ for each sample. After temperature scaling, the predicted outputs of the teacher model and student model are as follows:

$$\hat{y}_{k,i}^{t-1} = \frac{e^{(y_{k,i}^{t-1})/\alpha_T}}{\sum_{j=1}^{C} e^{(y_{k,j}^{t-1})/\alpha_T}} \tag{7}$$

$$\hat{y}_{k,i}^{t} = \frac{e^{(y_{k,i}^{t})/\alpha_T}}{\sum_{j=1}^{C} e^{(y_{k,j}^{t})/\alpha_T}} \tag{8}$$

where C is the number of sample categories, and α_T is the temperature parameter. $y_{k,i}^{t-1}$ and $y_{k,i}^t$ are the logit outputs of the teacher model and the student model for the ith class of the tth task on the kth client. The distillation loss for each sample is expressed as

$$\mathcal{L}_{\text{KD}}(D_k; \theta_k^t, \theta_k^{t-1}) = -\sum_{i=1}^{C} \hat{y}_{k,i}^{t-1} \log \hat{y}_{k,i}^t. \tag{9}$$

During the distillation process, to emphasize important knowledge from previous tasks while learning new ones, we introduce an attention-based weighting mechanism. Each sample in the reservoir is assigned an attention weight to reweight

its contribution to the distillation loss. These attention weights are not learned parameters, but are statically derived from the sample quality scores $Score_Q$ computed during the reservoir sampling process. Specifically, for each sample m in the local reservoir, the attention weight ω_m is computed by linear normalizing $Score_Q$ across the reservoir

$$\omega_m = \frac{Score_{Q\_m}}{\sum_{m=1}^{|D_k|} Score_{Q\_m}} \tag{10}$$

where $|D_k|$ denotes the number of samples in the reservoir. As new samples are added or replaced over time, the attention weights are updated accordingly across meta-tasks, reflecting the evolving quality of the stored data. The final distillation loss is then calculated as the weighted average of per-sample teacher losses using these attention weights

$$\mathcal{L}_{KD}(D_k; \theta_k^t, \theta_k^{t-1}) = -\sum_{m=1}^{|D_k|} \sum_{i=1}^{C} \omega_m \cdot \hat{y}_{k,i}^{t-1} \log \hat{y}_{k,i}^{t}. \tag{11}$$

It should be noted that, unlike traditional distillation FL, the distillation loss is added during the outer training update, not the inner training. This is because we prefer to ensure a macro balance between new and old knowledge, rather than focusing too much on the current task in each small batch. This method can achieve overall coordination between new and old tasks and retain global knowledge. Therefore, the total loss consists of the outer loss of the student model and the distillation loss. The total outer loss function is

$$\begin{aligned}\mathcal{L}'_{\text{outer}}(\theta_k; D_k) = \gamma \mathcal{L}_{KD}(D_k; \theta_k^t, \theta_k^{t-1}) \\ + (1-\gamma)\mathcal{L}_{\text{outer}}(\theta_k; D_k)\end{aligned} \tag{12}$$

where γ is the distillation loss weight. Algorithm 2 shows the specific process of a meta-knowledge distillation update. Concurrently, a complete meta-update is also a local update on the client. By combining meta-training with knowledge distillation based on the attention mechanism, the model can not only quickly adapt to small sample data when training new tasks, but also effectively retain the core knowledge of old tasks. This allows Hier-FCL to significantly mitigate catastrophic forgetting while efficiently learning new tasks. Since the attention weights are derived from static quality scores and applied only in a forward pass, the additional complexity is $O(M_{max})$ per update and does not require gradient computation, making it computationally lightweight.

Algorithm 2: Meta-Knowledge Distillation.

Require: Reservoir memory M_k^t, number of meta tasks N_τ, inner update epoch N_{meta}, inner learning rate α, outer learning rate β, distillation loss weight γ, current model θ_k^t and old model θ_k^{t-1}.

Ensure: New model θ'_k after a meta update on client k.

```
Divide M_k^t into N_τ mini batches {D_k^1, D_k^2, ..., D_k^{N_τ}}.
// Outer training
for each D_k^i in {D_k^1, D_k^2, ..., D_k^{N_τ}} do
    Divide D_k^i into a support set D_{k,S}^i and a query set D_{k,Q}^i.
    // Inner training
    for n = 1, 2, ..., N_meta do
        for each {X_{k,S}^i, Y_{k,S}^i} in D_{k,S}^i do
            θ_k^{i'} = θ_k^i − α∇_θ L_inner(D_{k,S}^i; θ_k)
        end for
    end for
    Calculate batch loss L_outer^i(θ_k^{i'}; D_{k,Q}^i) on the query set.
    Calculate batch distillation loss L_KD^i(D_{k,Q}^i; θ_k^{i'}, θ_k^{t-1}) on the query set.
end for
Accumulate outer loss: L_outer(θ_k; D_k) = Σ_{i=1}^{N_τ} L_outer^i(θ_k^{i'}; D_{k,Q}^i)
Accumulate distillation loss: L_KD(D_k; θ_k^t, θ_k^{t-1}) = Σ_{i=1}^{N_τ} L_KD^i(D_{k,Q}^i; θ_k^{i'}, θ_k^{t-1})
Calculate total loss L'_outer(θ_k; D_k) = γL_KD(D_k; θ_k^t, θ_k^{t-1}) + (1 − γ)L_outer(θ_k; D_k)
Update model parameters θ'_k = θ_k − β∇_θ L'_outer(θ_k; D_k)
```

D. Dynamic Clustering Based on Data Similarity

In dynamic and heterogeneous IoV environments, vehicles and devices continuously move and collect diverse data. As a result, the similarity among clients changes over time, making static clustering ineffective. To ensure that clients with similar data distributions collaborate efficiently during each training round, we design a dynamic clustering mechanism that periodically reorganizes client groups according to current data characteristics.

Before performing dynamic clustering, it is necessary to calculate the data similarity between each client. Assume that the data features of client k can be represented as the feature vector G_k. We use cosine similarity as the similarity measure, and the data similarity between client k and client k' can be expressed as

$$S_{k,k'} = \frac{G_k \cdot G_{k'}}{\|G_k\| \|G_{k'}\|} \tag{13}$$

where $G_k \cdot G_{k'}$ represents the dot product of the two feature vectors, and $\|G_k\|$ and $\|G_{k'}\|$ are the magnitudes of the two vectors, respectively.

After obtaining the client similarity matrix, we can then dynamically cluster the clients based on the similarity matrix. We use the K-means [24] clustering algorithm to group clients based on the similarity of the data. The goal of K-means clustering is to minimize the distance within the group

$$\min \sum_{q=1}^{Q} \sum_{G_i \in C_q} \|G_i - \mu_q\|^2 \tag{14}$$

where Q represents the number of clusters, which is also the number of edge servers. C_q denotes the qth cluster. μ_q is the center of the qth cluster, which can be obtained by averaging the feature vectors of the client data. By minimizing the intracluster distance, clients are assigned to different edge servers.

As client data distributions may shift significantly with the arrival of new tasks, it is essential to trigger reclustering at appropriate times to ensure effective model aggregation. Because

data variation in IoV may occur in different forms and time scales, we design three adaptive trigger strategies to flexibly respond to both abrupt task transitions and gradual distribution drifts in continual learning.

1) *Task Switching Control:* Recluster when a new task arrives. This strategy suits scenarios with clearly defined task boundaries and allows recalculating client similarity only when necessary, reducing redundant computations while maintaining stability.
2) *Data Distribution Drift Detection:* Trigger reclustering when a significant shift in data distribution is detected. For instance, if the distance between the new task's feature vector and the current cluster center exceeds a threshold, clustering is updated. This approach is useful in settings with implicit task transitions or high heterogeneity.
3) *Data Volume Threshold Control:* Perform reclustering once the volume of newly acquired data surpasses a predefined threshold. This is suitable for incremental learning scenarios where data arrives continuously.

In summary, in Hier-FCL, dynamic clustering based on data similarity helps address client heterogeneity and enhances the accuracy and stability of model aggregation. By appropriately selecting clustering trigger conditions, the system can adapt client grouping in response to data shifts or task transitions, maintaining a balance between old and new knowledge. While cosine similarity is used in our current image-based experiments, the similarity metric is modular and can be replaced with alternatives (e.g., Euclidean distance or dynamic time warping) depending on the data modality and feature representation, ensuring broad applicability across diverse IoV scenarios. The similarity matrix construction involves $O(K^2d)$ complexity per clustering event, but re-clustering is performed only at task transitions and can be efficiently parallelized.

E. Hierarchical Aggregation and Loading of Models

To enhance model quality while preserving personalized knowledge, the model is divided into global shared layer and local private layer. The global shared layer, designed to extract common features, participates in global aggregation to facilitate knowledge sharing while retaining client-specific information. Conversely, the local private layer, trained exclusively on client and edge server data, prevents interference with unique client characteristics. Notably, the local private layer is deployed on the edge server rather than directly on the client, effectively mitigating overfitting issues associated with limited and noisy client data. By managing models from dynamically clustered clients, edge servers achieve a balance between personalization and generalization within the local environment. Therefore, the model aggregation and loading process of different layers in Hier-FCL are as follows.

Client Layer: Each client first loads the complete model from the edge server as the initial model for the next training. After training on its local data, the global shared layer and local private layer parameters are updated at the same time and uploaded to the edge server.

Edge Server Layer: The edge server locally aggregates the overall model uploaded by the client. After aggregation, only the global shared layer parameters are uploaded to the cloud server, and the local private layer does not participate in the global aggregation. Simultaneously, when loading the cloud server global model, only the global shared layer parameters are loaded.

Cloud Server Layer: The cloud server aggregates the models uploaded by the edge server into a global model. Then, the aggregated global shared layer parameters are distributed to the edge servers one by one for the next round of local training.

It is worth noting that although Hier-FCL differs from conventional FL in training and aggregation strategies, the transmission of model parameters and intermediate representations during communication may still pose potential privacy risks, similar to traditional FL settings. While the main focus of this work is on mitigating catastrophic forgetting and improving learning efficiency, the framework, such as traditional FL, can be easily combined with privacy-preserving methods [25] to further protect user data during communication.

F. Workflow of Hier-FCL

The workflow of the Hier-FCL framework is illustrated in Algorithm 3. It follows a bottom-up training process among three layers. Each layer updates its parameters according to its role and contributes to global optimization.

Each client begins by retaining representative and category-diverse samples using an improved reservoir sampling algorithm. These samples are then used alongside meta-knowledge distillation to perform local continual learning. This combination helps balance the retention of old knowledge and the adaptation to new tasks. At the edge server layer, clients are clustered dynamically based on the similarity of their data, with the clustering structure updated in real time to accommodate shifts in data distribution.

The hierarchical aggregation mechanism operates between the edge servers and the cloud server. Edge servers first aggregate the models of clients within their respective clusters and then forward these aggregated models to the cloud server. The cloud server then aggregates shared parameters from all edge servers to update the global model and redistributes it for the next training round.

Through this iterative hierarchical process, Hier-FCL achieves efficient knowledge sharing while maintaining personalization in dynamic IoV environments.

IV. Experiments and Evaluations

To verify the effectiveness of the Hier-FCL framework, we conduct experiments simulating dynamic IoV task environments. These experiments analyze Hier-FCL's performance under heterogeneous data distribution and dynamic task changes, including comparisons with classic federated and continual learning methods, as well as ablation and parameter sensitivity analyses to evaluate each module's contribution.

Algorithm 3: Hier-FCL.

Require: Global epoch E_g, Edge aggregation epoch E_e, Local train epoch E_c, Edge server set S_e, Client set S_c.
Ensure: Trained model θ.

Initialize global model θ_g
for each k in $\{S_c\}$ **do**
  Initialize local task sequence $\{T_k^1, T_k^2, \ldots, T_k^{t_n}\}$.
end for
for $round = 1, 2, \ldots, E_g$ **do**
  if re-clustering is needed **then**
    Execute k-means to cluster the clients and obtain the client set $S_{c|e}$ of the edge server.
  end if
  for each edge server l in $\{S_e\}$ **do**
    $\theta_l' \leftarrow$ **EDGEUPDATE**$(\theta_g, E_e, S_{c|e_l})$
    Upload θ_l' to the cloud server.
  end for
  Aggregate global shared layer parameters: $\theta_g' = \frac{1}{\|S_e\|} \sum_{l=1}^{\|S_e\|} \theta_{l,\text{shared}}'$
end for

Function EDGEUPDATE$(\theta_g, E_e, S_{c|e_l})$
Load θ_g global shared parameters to edge model θ_l.
for $j = 1, 2, \ldots, E_e$ **do**
  for each client k in $\{S_{c|e_l}\}$ **do**
    $\theta_k' \leftarrow$ **CLIENTUPDATE**(θ_l, E_c)
    Upload θ_k' to the edge server
  end for
  Aggregate parameters: $\theta_l' = \frac{1}{\|S_{c|e_l}\|} \sum_{k=1}^{\|S_{c|e_l}\|} \theta_k'$
end for
Return θ_l'

Function CLIENTUPDATE(θ_l, E_c)
if get new task T_k^t **then**
  $M^t \leftarrow$ **Algorithm 1: Improved Reservoir Sampling**
end if
Load θ_l to client model θ_k
for $i = 1, 2, \ldots, E_c$ **do**
  $\theta_k' \leftarrow$ **Algorithm 2: Meta-knowledge Distillation**
end for
Return θ_k'

A. Experiment Setup

1) Dataset and Task Generation: We refer to the task division scheme of most existing works [3], [17], [22], that is, generating the task sequence of clients by class increment. This method can effectively simulate the process of the client receiving different tasks in continual learning.

However, we further consider that although the task division of a single dataset can verify the continual learning ability of the model, it may not fully reflect the diversity and dynamics of tasks in IoV scenarios. Therefore, we extend the traditional task generation method from a single dataset division to a division across multiple datasets to be closer to the multitask learning needs in IoV scenarios. Specifically, our dataset uses five datasets: 1) MNIST [26]; 2) FASHIONMNIST [27]; 3) CIFAR10 [28]; 4) CIFAR100 [28]; and 5) Tiny-ImageNet [29]. Each client contains a task sequence consisting of five tasks, and each task selects samples from different datasets. Concurrently, the data distribution still adopts the Non-iid method. Specifically, each task contains two categories, and 100 samples are randomly sampled from each category.

2) Experiment Parameters: The experimental system is set up with one cloud server, five edge servers, and 20 clients. The local training epoch E_c for clients is set to 4, and the edge aggregation epoch E_e is set to 6. Each client contains a task sequence of five tasks, with 1800 epochs of training for each task. The maximum capacity of the reservoir M_{max} is set equal to the data size of a single task, which is 200. In the sample replacement formula of the reservoir, w_1 is set to 0.5, w_2 to 0.5, w_3 to 0.8, and w_4 to 0.8. The experiment uses a ResNet [30] encoder pre-trained on local data to extract feature representations for each sample. These features are then used to compute cosine similarity with class centers and client-side clustering. During meta-learning, the batch size of meta-tasks is 16, the inner training learning rate α is 0.0001, and the outer training learning rate β is 0.0001. For knowledge distillation, the temperature α_T is set to 2.0, and the distillation loss weight γ is set to 0.5. When clustering each client, task switching control strategy is adopted to recalculate the data similarity between clients and perform dynamic clustering when switching tasks. The model used for training consists of two parts: 1) global shared layer; and 2) local private layer. The global shared layer includes a three-layer convolutional network, which is mainly responsible for extracting common features and shared by all clients; the local private layer consists of a two-layer fully connected network, which is responsible for classifying specific task features.

3) Baseline: To verify the effectiveness of our proposed method, we selected several representative continual learning schemes as baselines. All baseline methods were implemented within the same hierarchical architecture (client–edge–cloud) as our proposed framework to ensure structural consistency and fair comparison. The following is a detailed description of these baselines.

1) *FedAvg [6]:* One of the most commonly used FL methods, which achieves global model aggregation by weighted averaging client models. In the experiment, FedAvg learns each task on the client sequentially.
2) *FedEWC [15]:* EWC is a classic regularization-based continual learning method that introduces regularization terms in the loss function to limit the changes in key parameters and retain the knowledge of old tasks. FedEWC is its FL version.
3) *FedWeIT [20]:* A classic FCL method that achieves selective knowledge transfer by decomposing model parameters into global shared parameters and sparse task adaptive parameters.
4) *FedLwF [4]:* An algorithm that combines FL and knowledge distillation. It guides new task learning through old task models to achieve positive knowledge transfer.

TABLE I
PERFORMANCE COMPARISON ACROSS ALGORITHMS

Algorithm	RA					LA	BTI
	Task1	Task2	Task3	Task4	Task5/Final		
Hier-FCL	60.45%	**45.82%**	**38.08%**	**48.12%**	**59.72%**	65.45%	**−8.41%**
FedAvg	58.95%	38.62%	29.43%	32.46%	28.57%	77.61%	−61.30%
FedEWC	60.60%	41.15%	30.46%	34.41%	31.14%	76.22%	−56.35%
FedWeIT	**61.55%**	39.75%	27.66%	31.23%	29.20%	**78.61%**	−61.76%
FedLwF	61.15%	40.90%	29.93%	34.80%	27.44%	69.82%	−52.98%
FedMER	50.00%	32.97%	25.95%	35.57%	36.84%	51.38%	−18.18%

Note: Bold entries indicate the best performance among all compared methods for the corresponding metrics.

5) *FedMER [19]:* MER is a continual learning method that combines meta-learning and experience replay. It optimizes the inner loop on small batches and updates the outer loop on the global experience pool, maximizing the transfer between tasks by using gradient alignment. FedMER is the FL version of MER.

4) Evaluation Metrics: To objectively and comprehensively evaluate the performance of the proposed algorithm, we adopted widely recognized evaluation metrics commonly used in continual learning studies [3], [4], [19], [22].

a) Retained accuracy (RA): RA is used to measure the model's ability to retain knowledge of old tasks while learning new tasks. It calculates the average accuracy of the model on all previous tasks to reflect the knowledge retention effect. A higher RA value indicates less knowledge forgetting when tasks change dynamically and better retention performance. The formula is

$$RA = \frac{1}{t}\sum_{i=1}^{t} Acc_{t,i} \tag{15}$$

where t represents the current task, and $Acc_{t,i}$ indicates the model's accuracy on the old task i while training on task t.

b) Learning accuracy (LA): LA measures the model's adaptability to the current new task. It is defined as the model's accuracy $Acc_{t,t}$ on the current task t.

c) Backward transfer impact (BTI): BTI is used to quantify the degree of influence of the model on old tasks when learning new tasks. It is an important measure of catastrophic forgetting or positive transfer. If $BTI \geq 0$, the model's performance on old tasks improves after learning the new task, indicating positive transfer. Conversely, a $BTI < 0$ indicates that the performance on the old task becomes worse after learning the new task, which is negative interference. The formula is

$$BTI = \frac{1}{t-1}\sum_{i=1}^{t-1}(Acc_{t,i} - Acc_{i,i}) \tag{16}$$

where $Acc_{t,i}$ represents the model's accuracy on the ith task after completing task t, and $Acc_{i,i}$ indicates the model's accuracy on the ith task after completing the ith task.

B. Performance Analysis

1) Performance Comparison With Baselines: We conduct comparative experiments under the task division setting based on multiple datasets. As shown in Table I, the experimental results of Hier-FCL and the five baseline methods demonstrate that Hier-FCL consistently achieves superior performance. To more comprehensively analyze the dynamic changes of knowledge retention in the task sequence, in addition to the final RA, LA, and BTI, we also provide the RA after each task learning. As can be seen from the table, our proposed algorithm is far superior to other algorithms in RA and BTI, and RA has always maintained an advantage over other algorithms since the second task. This indicates that Hier-FCL has the ability to continuously retain knowledge in the process of learning task sequences and can effectively remember the learned tasks.

In addition, we notice that FedMER, which also relies on meta-learning and experience replay, has a higher RA than other baseline methods. This demonstrates that when it comes to task division of multiple datasets, the data heterogeneity between tasks is greater, and the continual learning method using meta-learning plus experience replay may be more suitable than the method based on regularization or knowledge distillation. Because when the heterogeneity between tasks is large, the parameters of the regularization method may conflict between different tasks and are difficult to unify. The method of knowledge distillation alone relies on the transfer of old knowledge, and also faces the problem that the old knowledge is not suitable for new tasks and the distillation effect is poor. Experience replay can explicitly strengthen the knowledge of old tasks while learning new tasks by replaying samples of old tasks. This direct memory mechanism is particularly important when there is large heterogeneity between tasks, because the data features of old tasks are difficult to retain indirectly in new tasks through parameters or distillation. Finally, we observe that Hier-FCL shows lower performance in the LA indicator, indicating that the learning of new tasks is somewhat interfered with by old tasks. However, this aligns with our expectations, as achieving both high retained accuracy for old tasks and high accuracy for new tasks is challenging, especially between tasks with large heterogeneity. Hier-FCL pays more attention to the knowledge retention of historical tasks, and may compromise the adaptability to current tasks.

Therefore, we examine whether Hier-FCL can maintain a balance between new and old tasks during the learning process. Fig. 2 shows the RA curves of all tasks during the training of different algorithms. As can be seen from the figure, Hier-FCL has a significant advantage in the retained accuracy of all tasks, that is, Hier-FCL can better maintain a balance between

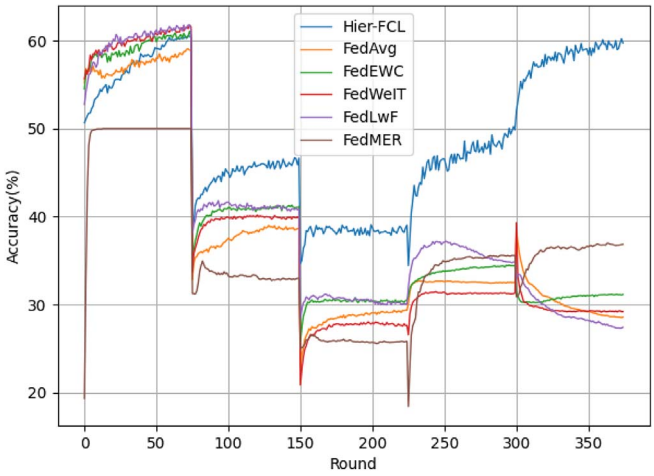

Fig. 2. Comparison of RA across different algorithms.

TABLE II
PERFORMANCE OF DIFFERENT ALGORITHMS UNDER DIFFERENT TASK ORDERS

Algorithm	RA	LA	BTI
Hier-FCL	**55.48%**	61.43%	**−8.69%**
FedAvg	24.67%	**77.02%**	−65.29%
FedEWC	16.00%	59.47%	−54.33%
FedWeIT	28.14%	76.37%	−60.25%
FedLwF	26.40%	72.88%	−58.10%
FedMER	34.88%	46.59%	−14.65%

Note: Bold entries indicate the best performance among all compared methods for the corresponding metrics.

new and old tasks, so that the model has good performance in both new and old tasks. Simultaneously, the results reveal that, except for Hier-FCL and FedMER, other methods exhibit a downward trend in performance on the last task. This is because the data of the last task is quite different from the previous tasks, which leads to obvious negative transfer in the process of learning new tasks. This further demonstrates that continual learning methods based on experience replay offer significant advantages over regularization and knowledge distillation approaches in scenarios with high task heterogeneity.

2) Impact of Task Order: It is worth noting that changes in the temporal order of tasks on the client can have a significant impact on the results. Certain task orders can introduce higher interference between tasks, leading to differences in knowledge retention [31]. Table II shows the average results of ten experiments with different random task sequences. Concurrently, Fig. 3 shows the comparison of RA, LA, and BTI. The results indicate that Hier-FCL is better than other baseline algorithms in terms of RA and BTI, with RA reaching 55.48% and BTI only –8.69%, indicating that it has better knowledge retention ability when facing different task sequences, not limited to specific task sequences. FedMER performs relatively well in RA and BTI, with RA of 34.88% and BTI of –14.65%, indicating that the combination of meta-learning and experience replay exhibits robustness in dealing with changes in task order. In

TABLE III
COMPARISON OF PERFORMANCE UNDER DIFFERENT MODULE COMBINATIONS

Improved Sampling	Dynamic Clustering	Meta-Knowledge Distillation	RA	LA	BTI
✓	✗	✗	55.60%	61.59%	−7.74%
✗	✓	✗	52.30%	62.33%	−12.04%
✗	✗	✓	54.41%	65.76%	−14.19%
✓	✓	✗	58.50%	65.06%	−8.95%
✓	✗	✓	59.07%	64.10%	−7.04%
✗	✓	✓	57.68%	64.84%	−9.45%
✗	✗	✗	50.21%	62.98%	−15.21%
✓	✓	✓	59.72%	65.45%	−8.41%

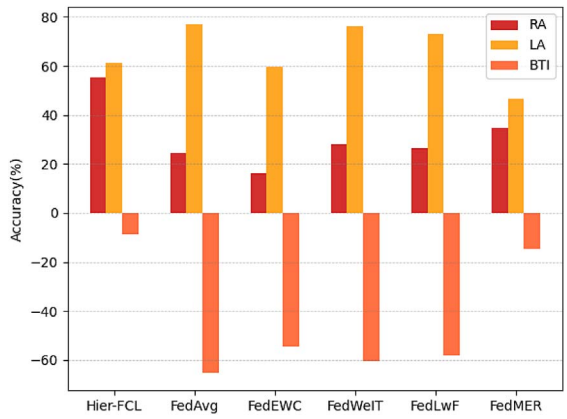

Fig. 3. Comparison of performance under different task orders.

contrast, FedAvg, FedEWC, FedWeIT, and FedLwF have lower RA, 24.67%, 16.00%, 28.14%, and 26.40%, respectively, and their BTI is much lower than Hier-FCL, indicating that they are also prone to knowledge forgetting under different task orders. Among them, FedEWC exhibits the largest fluctuation and is most seriously affected by the task order.

C. Ablation Experiments

This section analyzes the contributions of the three core modules in Hier-FCL through ablation experiments: 1) improved sampling; 2) dynamic clustering; and 3) meta-knowledge distillation. Table III presents the RA, LA, and BTI results under different module combinations. The improved sampling module contributes most significantly to RA. When used alone, it achieves 55.60% RA, outperforming the other individual modules. This is attributed to its ability to select high-quality and diverse samples, improving experience replay and mitigating forgetting. The dynamic clustering module enhances performance under task heterogeneity by grouping clients with similar data, but shows limited effectiveness in preserving old knowledge, as reflected in lower RA. The meta-knowledge distillation module boosts LA by transferring prior knowledge into new task learning, achieving 65.76% LA and 54.41% RA when used independently, though it results in higher BTI. Importantly, the full Hier-FCL framework, integrating all three modules, achieves the best overall performance in both RA and LA, while significantly reducing BTI. This confirms that the modules are

TABLE IV
COMPARISON OF PERFORMANCE ACROSS DIFFERENT COMBINATIONS OF E_c AND E_e

Parameters	RA	LA	BTI
$E_c = 4,\ E_e = 6$	59.72%	65.45%	−8.41%
$E_c = 6,\ E_e = 4$	59.59%	66.58%	−9.99%
$E_c = 3,\ E_e = 8$	60.54%	65.82%	−7.85%
$E_c = 8,\ E_e = 3$	54.90%	63.73%	−12.29%
$E_c = 2,\ E_e = 12$	59.54%	63.20%	−5.82%
$E_c = 12,\ E_e = 2$	41.08%	58.72%	−23.30%

not only individually beneficial but also synergetic when combined, validating the effectiveness of our integrated design.

D. Key Parameter Analysis

In this section, we conduct experiments on aggregation interval, reservoir size, and the number of task epochs, discussing their effects on RA, LA, and BTI of Hier-FCL.

1) Aggregation Interval: First, we examine the two key aggregation interval parameters in Hier-FCL: 1) client training rounds E_c; and 2) edge aggregation rounds E_e. E_c is the number of rounds of local training for each client, which is used to adapt the model to local data. E_e is the number of rounds of edge aggregation before global aggregation by the edge server, which is used to adapt the model to the feature distribution within the client group. To compare the effect of a global communication more fairly, we fix the value of $E_c \times E_e$ to compare the impact of different parameter values on the algorithms. Table IV shows the results of RA, LA, and BTI under different parameters, and Fig. 4 shows in detail the changes of RA during the training process. The results show that with the same parameter combination, a higher E_e improves RA and LA. Increasing E_e enables more edge sharing and allows the model to learn group-specific knowledge more effectively. Simultaneously, because each client includes local old knowledge when sharing the model, increasing E_e can also improve the ability to learn old knowledge. When E_c is higher, the model has better performance in the early stage of the task, because more local training rounds allow the model to adapt to local data faster. However, in the later stage of training, too many local training rounds may cause the model to fall into overfitting. Moreover, the limited parameter sharing among clients restricts the ability to acquire sufficient global knowledge.

2) Reservoir Size: Next, we analyze the impact of changes in the reservoir size on the algorithm performance to evaluate the robustness of the algorithm under resource constraints. We adjust the capacity of the reservoir to 25%, 50%, and 75% of the original size, corresponding to capacities of 50, 100, and 150. Table V shows the final RA, LA, and BTI metrics under different reservoir sizes, and Fig. 5 intuitively compares the RA and LA results of different sizes. The results indicate that as the size of the reservoir increases, RA and LA show an upward trend, and BTI gradually decreases. This means that a larger reservoir can store more comprehensive information about new and old tasks, which can enhance the model's learning effect on all tasks. Concurrently, even when the reservoir size is small

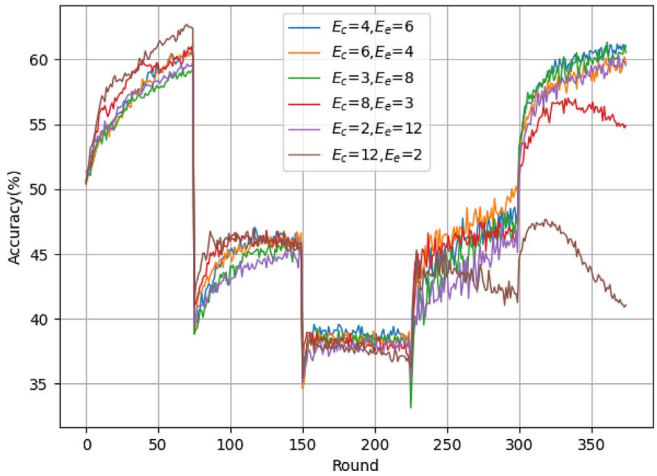

Fig. 4. Comparison of RA across different combinations of E_c and E_e.

TABLE V
COMPARISON OF PERFORMANCE ACROSS DIFFERENT RESERVOIR SIZES

Parameters	RA	LA	BTI
pool_size = 50	52.72%	58.28%	−13.20%
pool_size = 100	56.03%	62.93%	−9.88%
pool_size = 150	58.54%	64.37%	−8.54%
pool_size = 200	59.72%	65.45%	−8.41%

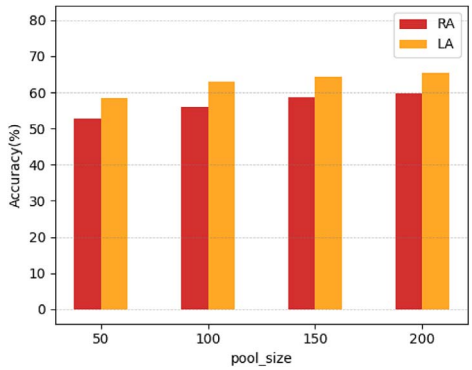

Fig. 5. Comparison of RA and LA across different reservoir sizes.

(such as a capacity of 50 or 100), the model still shows strong robustness, and RA and LA remain at a high level. This result shows that Hier-FCL can still perform well under resource-constrained conditions. The reason is that our method stores more important and diverse samples during experience replay, making full use of the limited reservoir capacity.

3) Number of Task Epochs: Finally, we compare the impact of the number of task epochs trained on each task on the client. Table VI shows the final RA, LA, and BTI under different task epochs. Fig. 6 illustrates the variations of different metrics with task epochs. The results indicate that when the task epoch is low (such as 720 or 1080), RA and LA are both maintained at a low level, and the difference between the two is not large,

TABLE VI
COMPARISON OF PERFORMANCE ACROSS DIFFERENT NUMBERS OF TASK EPOCHS

Parameters	RA	LA	BTI
task_epochs = 720	52.73%	51.48%	0.31%
task_epochs = 1080	55.64%	53.63%	**1.26%**
task_epochs = 1440	56.57%	58.53%	−3.70%
task_epochs = 1800	**59.72%**	**65.45%**	−8.41%
task_epochs = 2160	57.45%	61.45%	−6.25%
task_epochs = 2520	56.64%	61.26%	−7.03%

Note: Bold entries indicate the best performance among all compared methods for the corresponding metrics.

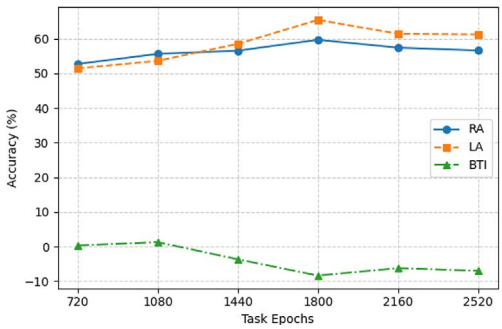

Fig. 6. Trend of performance across different numbers of task epochs.

and even BTI is positive, which indicates that the old task has a positive transfer effect on the new task. At this time, the model has insufficient training time on the new task and fails to fully learn the task features. As the task epoch increases, RA and LA will gradually increase within a certain range, and BTI will gradually decrease. When the epoch number increases to 1800, RA and LA reach a peak of 59.72% and 65.45%, respectively, indicating that the model has achieved a good balance between the new and old tasks. However, as the number of task epochs continues to increase, the performance begins to decline. This may be attributed to several factors. On the one hand, excessive training on new tasks may lead to overfitting and exacerbate forgetting of old tasks. On the other hand, we find that LA did not increase like simple overfitting, but decrease instead. This suggests that prolonged training may cause the model to emphasize features of the current task, which conflict with those of old tasks. When trying to balance these conflicts, the model may not be able to adapt well to all tasks, resulting in a decrease in overall performance. Therefore, in task scenarios involving multiple datasets, it is crucial to choose the right number of task epochs. Both excessively long and overly short training durations can negatively impact the model's ability to balance old and new tasks, ultimately degrading its overall performance.

V. CONCLUSION

This article proposes Hier-FCL, a Hier-FCL framework designed to address the challenges of multi-layer architectures and task heterogeneity in dynamic IoV scenarios. By employing dynamic client clustering and hierarchical model aggregation, Hier-FCL mitigates data heterogeneity while enabling efficient collaboration across clients, edge servers, and the cloud server. The framework leverages an improved reservoir sampling strategy and meta-knowledge distillation to balance the retention of knowledge from previous tasks with adaptation to new tasks, effectively mitigating catastrophic forgetting. Extensive experiments on multiple datasets demonstrate that Hier-FCL significantly outperforms baseline algorithms in retention accuracy and backward transfer impact, highlighting its robustness and adaptability. Our future research will focus on enhancing the scalability of Hier-FCL under extreme data imbalance, addressing latency challenges in real-time scenarios, improving its practicality for edge environments with limited resources, and evaluating communication efficiency under different network conditions and mobility patterns.

REFERENCES

[1] Y. Lin, C. Wu, J. Wu, L. Zhong, X. Chen, and Y. Ji, "Meta-networking: Beyond the Shannon limit with multi-faceted information," *IEEE Netw.*, vol. 37, no. 4, pp. 256–264, Jul./Aug. 2023.
[2] X. Zhou et al., "Reconstructed graph neural network with knowledge distillation for lightweight anomaly detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 11817–11828, Sep. 2024.
[3] D. Li, N. Huang, Z. Wang, and H. Yang, "Personalized federated continual learning for task-incremental biometrics," *IEEE Internet Things J.*, vol. 10, no. 23, pp. 20776–20788, Dec. 2023.
[4] A. Usmanova, F. Portet, P. Lalanda, and G. Vega, "A distillation-based approach integrating continual learning and federated learning for pervasive services," 2021, *arXiv:2109.04197*.
[5] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 1–6.
[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, PMLR, 2017, pp. 1273–1282.
[7] X. Zhou, W. Liang, A. Kawai, K. Fueda, J. She, and K. I.-K. Wang, "Adaptive segmentation enhanced asynchronous federated learning for sustainable intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 7, pp. 6658–6666, Jul. 2024.
[8] Z. Du, C. Wu, T. Yoshinage, L. Zhong, and Y. Ji, "On-device federated learning with fuzzy logic based client selection," in *Proc. Conf. Res. Adaptive Convergent Syst.*, 2022, pp. 64–70.
[9] X. Zhou et al., "Federated distillation and blockchain empowered secure knowledge sharing for internet of medical things," *Inf. Sci.*, vol. 662, 2024, Art. no. 120217.
[10] X. Zhou et al., "Personalized federated learning with model-contrastive learning for multi-modal user modeling in human-centric metaverse," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 4, pp. 817–831, 2024.
[11] X. Zhou et al., "Hierarchical federated learning with social context clustering-based participant selection for internet of medical things applications," *IEEE Trans. Computat. Social Syst.*, vol. 10, no. 4, pp. 1742–1751, Aug. 2023.
[12] Q. Wu et al., "HiFlash: Communication-efficient hierarchical federated learning with adaptive staleness control and heterogeneity-aware client-edge association," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 5, pp. 1560–1579, May 2023.
[13] F. De Rango, A. Guerrieri, P. Raimondo, and G. Spezzano, "Hed-FL: A hierarchical, energy efficient, and dynamic approach for edge federated learning," *Pervasive Mob. Comput.*, vol. 92, 2023, Art. no. 101804.
[14] X. Yang, H. Yu, X. Gao, H. Wang, J. Zhang, and T. Li, "Federated continual learning via knowledge fusion: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 8, pp. 3832–3850, Aug. 2024.
[15] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017.
[16] P. Zhang et al., "Continual learning on dynamic graphs via parameter isolation," in *Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2023, pp. 601–611.
[17] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICARL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2001–2010.

[18] X. Li, S. Wang, J. Sun, and Z. Xu, "Variational data-free knowledge distillation for continual learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12618–12634, Oct. 2023.
[19] M. Riemer et al., "Learning to learn without forgetting by maximizing transfer and minimizing interference," 2018, *arXiv:1810.11910*.
[20] J. Yoon, W. Jeong, G. Lee, E. Yang, and S. J. Hwang, "Federated continual learning with weighted inter-client transfer," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2021, pp. 12073–12086.
[21] G. Bagwe, X. Yuan, M. Pan, and L. Zhang, "Fed-Cprompt: Contrastive prompt for rehearsal-free federated continual learning," 2023, *arXiv:2307.04869*.
[22] J. Zhang, C. Chen, W. Zhuang, and L. Lyu, "Target: Federated class-continual learning via exemplar-free distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 4782–4793.
[23] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2017, pp. 1126–1135.
[24] J. MacQueen et al., "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Mathematical Stat. Probability*, vol. 1, no. 14, Oakland, CA, USA, 1967, pp. 281–297.
[25] X. Zhou et al., "Decentralized federated graph learning with lightweight zero trust architecture for next-generation networking security," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 6, pp. 1908–1922, Jun. 2025.
[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[27] H. Xiao, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[28] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[29] Y. Le and X. Yang, "Tiny ImageNet visual recognition challenge," *CS 231n*, vol. 7, no. 7, p. 3, 2015.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR)*, 2016, pp. 770–778.
[31] S. J. Bell and N. D. Lawrence, "The effect of task ordering in continual learning," 2022, *arXiv:2205.13323*.

Yiming Chen (Student Member, IEEE) received the B.E. degree in computer science and technology from Bohai University, Jinzhou, China, in 2017, and the M.E. degree in software engineering from Nanjing University of Information Science and Technology, Nanjing, China, in 2020. He is currently working toward the Ph.D. degree in computer and network engineering with The University of Electro-Communications, Tokyo, Japan.

His research interests include Internet of Vehicles, federated learning, continual learning, and intelligent traffic management.

Celimuge Wu (Senior Member, IEEE) received the Ph.D. degree in engineering from The University of Electro-Communications, Tokyo, Japan.

He is currently a Professor and the Director of Meta-Networking Research Center, The University of Electro-Communications; and a core member of Advanced Wireless and Communication Research Center, The University of Electro-Communications. His research interests include vehicular networks, edge computing, IoT, and AI for wireless networking and computing.

Dr. Wu serves as an Associate Editor of IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, and IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He is the Vice Chair (Asia Pacific) of *IEEE Technical Committee on Big Data (TCBD)*. He is a Recipient of 2021 IEEE Communications Society Outstanding Paper Award, 2021 *IEEE Internet of Things Journal* Best Paper Award, IEEE Computer Society 2020 Best Paper Award, and IEEE Computer Society 2019 Best Paper Award Runner-Up. He is an IEEE Vehicular Technology Society Distinguished Lecturer.

Lei Zhong (Member, IEEE) received the Ph.D. degree in informatics from the Graduate University for Advanced Studies, Hayama, Japan, in 2011.

He is currently a Principal Researcher and a Project Manager in the InfoTech Division with Toyota Motor Corporation, where he leads research and standardization activities in intelligent connected vehicles. He serves as Toyota's chief delegate to major standardization organizations, including 3GPP, IEEE, and the Wi-Fi Alliance, and chairs the PoC and Liaison Strategy committees within the Automotive Edge Computing Consortium (AECC). His research interests include wireless networking, edge computing, machine learning, connected vehicles, and big data analytics. He has coauthored more than 80 technical publications and holds over 40 patents in these areas.

Dr. Zhong also actively contributes to the academic community as Vice-Chair of the IEEE Technical Committee on Green Communications and Computing Special Interest Group on Green Internet of Vehicles and serves on the editorial boards of *Elsevier Vehicular Communications*, *IEEE Internet Computing*, and the IEEE Open Journal of the Computer Society.

Yangfei Lin (Member, IEEE) received the Ph.D. degree in computer science from the University of Tsukuba, Tsukuba, Japan, in 2022.

She is currently a Postdoctoral Researcher with the University of Electro-Communications, Tokyo, Japan. Her research interests include cloud computing security, privacy preserving, and semantic communications.

Zhaoyang Du (Member, IEEE) received the Ph.D. degree in engineering from The University of Electro-Communications, Tokyo, Japan, in 2021.

He is currently a Postdoctoral Researcher with The University of Electro-Communications. His research interests include delay tolerant networks, vehicular ad hoc networks, and IoT.

Wugedele Bao received the Ph.D. degree in linguistics and applied linguistics from Minzu University of China, Beijing, China, in 2018.

He is currently a Professor with the School of Computer Science and Information Engineering and a member of the Trustworthy Big Data and AI Academician Workstation, Hohhot Minzu College, Hohhot, China. His research interests include delay-tolerant networks, vehicular ad hoc networks, and IoT.

Peter Han Joo Chong (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of British Columbia, British Columbia, Canada, in 2000.

He is currently a Professor and an Associate Head of School (Research) with the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand. He was previously an Associate Professor (tenured) with Nanyang Technological University, Singapore. His research interests include MANETs/VANETs, V2X, Internet of Things/Vehicles, artificial intelligence for wireless networks, and 5G networks.