Resource-efficient Heterogenous Federated Continual Learning on Edge

Zhao Yang[1], Shengbing Zhang[2], Chuxi Li[2], Haoyang Wang[2] and Meng Zhang[2*]
[1]College of Future Transportation, Chang'an University, Xi'an, China
[2]School of Computer Science, Northwestern Polytechnical University, Xi'an, China

***Abstract*—Federated learning (FL) has been widely deployed on edge devices. In practical, the data collected by edge devices exhibits temporal variations. This leads to catastrophic forgetting issue. Continual learning methods can be used to address this problem. However, when deploying these methods in FL on edge devices, it is challenging to adapt to the limited resources and heterogeneous data of the deployed devices, which reduces the efficiency and effectiveness of federated continual learning (FCL). Therefore, this article proposes a resource-efficient heterogeneous FCL framework. This framework divides the global model into an adaptation part for new knowledge and a preservation part for old knowledge. The preservation part is used to address the catastrophic forgetting problem. Only the adaptation part is trained when learning new knowledge on a new task, reducing resource consumption. Additionally, the framework mitigates the impact of heterogeneous data through an aggregation method based on feature representation. Experimental results show that our method performs well in mitigating catastrophic forgetting in a resource-efficient manner.**

I. Introduction

Federated Learning (FL) [1]–[3] has emerged as a novel paradigm of distributed machine learning. However, the practical implementation of FL presents challenges stemming from the evolving data distribution over time. This leads to the problem of catastrophic forgetting, where the global model adapts to new tasks but fails to retain knowledge from previous tasks. To address this issue, continual learning (CL) [4]–[7] has been proposed to enhance the model's adaptability.

To preserve knowledge of old tasks, data replay [4] involves storing data from old tasks to mitigate the problem of catastrophic forgetting. Model progress [5] introduces new structures to the existing model to enable adaptation to new tasks. Knowledge distillation [6] extracts knowledge from public datasets to enhance the model's adaptability. Parameter regularization [7], on the other hand, constrains the updating process by penalizing updates on crucial parameters. However, deploying them in FL on edge devices presents additional challenges.

Firstly, edge devices typically have limited storage, computing, and communication capabilities. Storing data from old tasks consumes additional storage space, potentially displacing new task data. Reducing the amount of training samples can adversely affect the model's ability to adapt to new tasks. The progressed model requires more storage for larger structures, and the increased number of parameters leads to a higher computational burden during the training process. Uploading more parameters for model aggregation further exacerbates communication bottlenecks. Additionally, training distilled networks on edge devices adds additional model training tasks, resulting in conflicts between training overheads and limited computational resources. While these methods can mitigate the catastrophic forgetting in FL, they can severely impact the efficiency and effectiveness of FL on edge devices.

Secondly, the wide distribution and mobility of deployed edge devices result in the presence of heterogeneous data, known as Non-Independent and Identically Distributed (Non-IID) data. Non-IID data presents two challenges. First, it introduces the impact of old knowledge in existing models on the training process for new tasks. Second, it leads to interference from other devices, arising from the sharing of uncorrelated knowledge, which may hinder the training of the target task. These challenges affect the effectiveness of FL in extracting knowledge for new tasks. The reduced training convergence may bring more computational burdens.

To address the challenges posed by limited resources and heterogeneous data during federated continual learning (FCL) on edge, this paper introduces a novel resource-efficient heterogeneous FCL framework. The proposed framework determines and divides the global model into two distinct functional modules: the adaptation module for new tasks (Adapter) and the knowledge retention module for old tasks (Retainer). Each device only trains the Adapter, reducing the local training cost and minimizing the interference of historical knowledge on new task training. Additionally, aggregating partial models eases the communication bottlenecks of edge devices. Furthermore, a federated heterogeneous aggregation method is employed during model aggregation to decrease the impact of uncorrelated knowledge on the training process across different devices. Moreover, dynamic updates to the Retainer for old tasks effectively mitigate the catastrophic forgetting while preventing excessive memory occupancy associated with storing old task models and data. The main contributions are summarized as:

- We propose a resource-efficient heterogeneous FCL framework based on model function decomposition. This framework addresses catastrophic forgetting issue in a low-cost manner while ensuring the performance of FL.
- We propose an FCL training strategy for new tasks, which involves adapting a specific portion of the model structure dedicated to the new task. This approach allows to conserve computing and communication resources while mitigating the impact of old knowledge. Furthermore, we ensure that the aggregation process is not influenced by

*Meng Zhang is the corresponding author.

979-8-3503-4859-0/DATE24/© 2024 EDAA

uncorrelated knowledge between nodes.

- We propose an updating method for preserving old knowledge, which addresses the issue of catastrophic forgetting in a continuous manner. This method effectively mitigates this problem while minimizing the additional storage costs associated with models and data.

II. Related Works

FL [1]–[3] is a distributed learning paradigm where multiple devices collaboratively participate in model training. However, in practical scenarios, the following phenomena may occur: participants in the current training round may be unable to participate in the next round due to network connectivity issues, low battery levels, or relocation, leading to data variations between consecutive training rounds. Additionally, edge devices continuously collect new data, and minimizing the loss on the new data may increase the loss on old data, resulting in catastrophic forgetting. Therefore, FL on edge requires CL [4]–[7], enabling the global model to continuously learn from new data while retaining the knowledge from previous data.

Several works have been proposed to address FCL. For instance, [8] explores continual edge learning by leveraging knowledge transfer from previous tasks, either from the cloud or other edge nodes. They devise an ADMM-based federated meta-learning algorithm to train a meta model that enables fast adaptation of the target node model. Another work, [9], proposes an FCL framework called FedWeIT. In FedWeIT, each client receives selective knowledge from other clients through a weighted combination of their task-specific parameters. However, it requires high local storage of task-specific parameters on terminal devices, making it unsuitable for resource-constrained edge devices. Additionally, [10] presents a distillation-based approach to address catastrophic forgetting in the FL scenario based on the CL method LwF [11]. Furthermore, when learning a new task, [12] utilizes the method of Progressive Neural Network [5] to incorporate a new neural network (a column) into the existing network structure. It is evident that the aforementioned methods impose additional computational overhead or storage requirements to learn new knowledge while preserving old knowledge. These factors pose significant challenges for FL on edge devices, particularly due to their limited resources and the presence of heterogeneous data.

III. Problem Formulation and Solutions

A. Problem Formulation

In FL, a central server collaborates with a group of K participating devices to train a global model M for a given task $\mathcal{T}$. The optimization goal of FL is as follows:

$$\min_{\theta} \sum_{k=1}^{K} \frac{s_k}{S} \mathcal{L}\left(\mathcal{T}^k; \theta\right), \tag{1}$$

$\mathcal{T}^k$ denotes the collection of training samples s_k available at the k-th device. The total number of local training samples across all devices is represented by S. The global model M is parameterized by θ, and $\mathcal{L}$ represents the loss function.

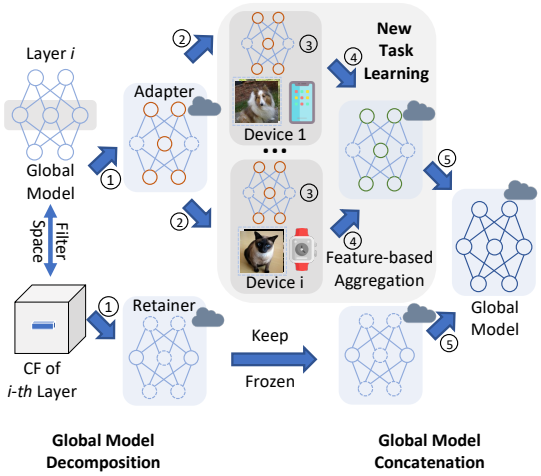

Fig. 1. The overview of our proposed FCL method.

Furthermore, the objective of FCL is to train a global model on a series of tasks $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, ...$ The data for these tasks are acquired from various local devices. When the t-th task arrives at time t, the local devices are required to collect data specific to the new task. At this stage, the global model inherits the model parameters θ_{t-1} from the preceding task. The new task is denoted as $\mathcal{T}_t = \bigcup_{k=1}^{K} \mathcal{T}_t^k$, where $\mathcal{T}_t^k$ is the data collected on the k-th device.

The objective of training the global model is to minimize the loss not only on the current task $\mathcal{T}_t$, but also on the preceding tasks $\mathcal{T}_1, \ldots, \mathcal{T}_{t-1}$. This optimization goal is achieved by iteratively engaging in server-client communication to minimize the losses of K local devices across all local tasks up to time t. The global model parameters can be obtained:

$$\theta_t = \arg\min_{\theta} \sum_{k=1}^{K} \frac{s_k}{S} \sum_{i=1}^{t} \mathcal{L}\left(\mathcal{T}_i^k; \theta\right). \tag{2}$$

During the training process of FCL, it is essential for the global model at task $\mathcal{T}_t$ to ensure that the loss incurred on historical tasks does not exceed that of the previous time step $(t-1)$. This can be expressed by the following metric:

$$\sum_{i=1}^{t-1} \mathcal{L}\left(\mathcal{T}_i; \theta_t\right) \leq \sum_{i=1}^{t-1} \mathcal{L}\left(\mathcal{T}_i; \theta_{t-1}\right). \tag{3}$$

In addition to accomplishing the aforementioned training objectives, it is equally important to minimize additional storage costs and reduce computational expenses on local devices. This consideration ensures that the FCL process aligns well with the resource constraints commonly found in edge devices.

B. Method Overview

Our framework addresses the challenge of managing diverse data at various time steps across different devices within the FL setup while minimizing storage, training, and communication overheads. The proposed FCL framework consists of three vital components: (1) functional decomposition of the global model, (2) FCL training process tailored for new tasks, and (3) preservation and update of prior knowledge to mitigate catastrophic forgetting.

Upon the arrival of the t-th task, as shown in Fig. 1, in Step 1, the global model is partitioned into two distinct components: the adaptation part, Adapter, which facilitates the integration of new knowledge, and the preservation part, Retainer, which retains the existing knowledge. This partitioning aims to reduce the local training overheads associated with the new task while minimizing the influence of parameters associated with prior knowledge on the acquisition of new knowledge. Additionally, this partitioning enables the preservation of crucial aspects of prior knowledge, mitigating catastrophic forgetting.

Subsequently, in Step 2, the server exclusively transmits the Adapter to each device for local training. Following the completion of local training (Step 3), the locally trained Adapters are sent back to the server for model aggregation. The server then undertakes the aggregation of local models. In Step 4, we introduce a dynamic feature-based aggregation method specifically designed for heterogeneous FL. This method aims to mitigate the influence of heterogeneous knowledge on different devices.

Lastly, the parameters acquired from the training of the Adapters are combined with the frozen parameters in the Retainer, resulting in the formation of a new global model (Step 5). Furthermore, when a new task arrives, the Adapter and Retainer are updated in Step 1.

IV. Function Decomposition of Global Model

To facilitate the continual adaptation of the global model to new tasks while retaining knowledge from previous tasks, we propose dividing the global model into two separate components: an Adapter A responsible for adapting to new task data, and a Retainer R designed to preserve old knowledge, as shown in Fig. 2. Consequently, the global model M is formed by concatenating A and R, such that $M = [A, R]$. During the training process of a new task, each device exclusively trains the model parameters associated with the Adapter A, while the model parameters of the Retainer R remain frozen on the server. Once the training of the Adapter A is completed, the resulting adapted component A' is concatenated with the Retainer R, yielding the updated global model $M' = [A', R]$.

Determining the appropriate combination of the Adapter A and the Retainer R presents a notable challenge. The selection of A must strike a balance between capturing knowledge from new tasks while mitigating the influence of previous knowledge during the training process on new tasks. Simultaneously, the Retainer R must preserve vital information from the old tasks to mitigate the issue of catastrophic forgetting. As shown in Fig. 2, we select the complete structures of both deep and shallow layers in the network to constitute the part of the Adapter. Furthermore, to enhance overall performance and maintain the integrity of the original knowledge, we employ an additional partitioning of the intermediate part of the model's structure. This division results in an additional portion allocated to the Adapter and another portion dedicated to the Retainer.

The identification of essential knowledge from previous tasks to be preserved in the Retainer presents another challenge. To address this challenge, a feature representation-based method can be employed. We use CNN as an example. In order to extract essential knowledge from previous tasks, it is crucial to

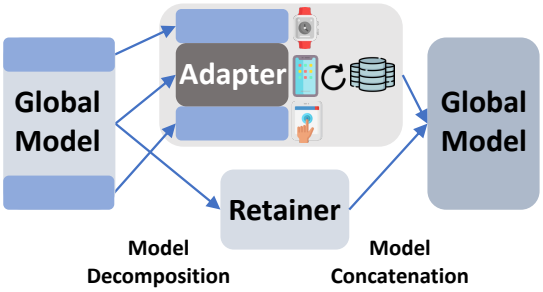

Fig. 2. The global model is divided into the Adapter A and the Retainer R.

comprehend the functional significance of different parameters, specifically the feature representations. To achieve this, we leverage the common feature embedded in the global model as an intermediary and criterion for identifying the feature representation of different parameters. In this section, our initial objective is to identify the most predominant feature within the global model. This feature serves as a foundation for further analysis and selection processes aimed at extracting important knowledge from the old tasks.

Given the diverse functions and scopes of individual layers within a neural network, it is imperative to examine common feature in each layer independently. In this pursuit, we adopt the CNN's filter as the unit of feature representation. To accomplish this, we identify the filter with the highest correlation across all other filters within a layer. To initiate this process, we conduct a comparison of the correlation among filters in each layer.

In the case of a convolutional layer with filters of shape $K \times K \times M \times N$, where K denotes the kernel size, M represents the input channel number, and N signifies the output channel number, each filter can be regarded as a collection of $K \times K$ independent nodes. To facilitate the computation process, we organize the filter tensors into $K \times K$ sets of nodes, simplifying the correlation calculations. Then, we determine the pairwise correlations among the nodes within each set. Finally, we obtain the filter correlation by averaging the node correlations across the $K \times K$ nodes associated with the specific filter:

$$sim\left(F_m^l, F_n^l\right) = \frac{1}{K^2} \sum_{i}^{K} \sum_{j}^{K} corr\left(\vec{\omega}_{i,j,m}, \vec{\omega}_{i,j,n}\right), \tag{4}$$

where, F_m^l and F_n^l are the m-th and n-th filter in the l-th layer, $\vec{\omega}_{i,j,m}$ is the vector $[W_{i,j,m,1}, W_{i,j,m,2}, \cdots, W_{i,j,m,n}]$ of weights W of the l-th layer. Calculating $corr\left(\vec{\omega}_{i,j,m}, \vec{\omega}_{i,j,n}\right)$, Pearson correlation [13] is applied for high dimension data:

$$corr\left(\vec{\omega}_{i,j,m}, \vec{\omega}_{i,j,n}\right) = \frac{E\left[\left(\vec{\omega}_{i,j,m} - \mu_{\vec{\omega}_{i,j,m}}\right)\left(\vec{\omega}_{i,j,n} - \mu_{\vec{\omega}_{i,j,n}}\right)\right]}{\sigma_{\vec{\omega}_{i,j,m}} \sigma_{\vec{\omega}_{i,j,n}}}, \tag{5}$$

where, $\mu_{\vec{\omega}_{i,j,m}}$ is the mean of $\vec{\omega}_{i,j,m}$, $\sigma_{\vec{\omega}_{i,j,m}}$ is the standard deviation (std) of $\vec{\omega}_{i,j,m}$. E is the expectation function.

Having established the correlations between each pair of filters, it is necessary to transform the pairwise correlations into the relationships of a specific filter with all other filters in the same layer. In order to determine the significance of each filter in containing the common information, we define

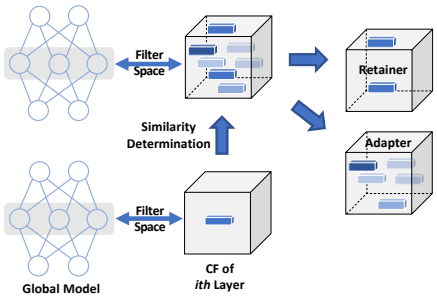

Fig. 3. The details of the model decomposition for the Adapter and Retainer.

an importance coefficient. To calculate it, we employ a max-normalization approach using its top-k highest correlations:

$$Imp\left(F_m^l\right) = 1 - \frac{1}{k}\sum^{k} top\text{-}k\left(\frac{sim\left(F_m^l, F_n^l\right)}{\max_{p\neq q}\left(sim\left(F_p^l, F_q^l\right)\right)}\right), \quad (6)$$
$$p, q, n \in [1, M] \text{ and } n \neq m.$$

After obtaining the normalized importance for all filters in each layer, we select the filter with the highest value as the common feature F_{CF}^l for that specific l-th layer, as shown in Fig. 3. This filter encompasses the most significant and shared features within the layer. For each layer, we proceed to compare the different filters with the obtained common features to assess their ability to capture the functionality of that specific layer. To measure this, we calculate the distance $D_m^l = \left\|F_m^l - F_{CF}^l\right\|_2$ between each filter and the common feature in each layer.

When a filter is in close proximity to the common feature, it indicates that the features extracted by that particular filter are more akin to the common feature extracted within the layer. Consequently, such filters are deemed essential and should be preserved in the Retainer. By preserving these filters, we can effectively mitigate the problem of catastrophic forgetting.

Therefore, we rank the filters within each layer based on their proximity to the common feature. Subsequently, we select the top-k filters, deemed to have the highest similarity, for inclusion in the Retainer. These selected filters are then frozen on the server, ensuring their preservation throughout the training process. In parallel, the remaining portion of the network structure, forms the Adapter, consisting of both deep and shallow layers. Each node then undergoes training on the new task using this Adapter structure. It should be noted that this model decomposition process occurs on the server without imposing additional computational burdens on edge devices.

V. Heterogeneous Federated Continual Learning

After the training process on the new task's data with Adapter A, the training parameters are returned to the server for model aggregation. Given the heterogeneous characteristics of the data on different devices and the presence of strong local preferences in the training results, it becomes crucial to optimize the aggregation process. This optimization is necessary to bolster the global model's capacity to effectively acquire new knowledge while mitigating the impact of heterogeneous data. The conventional FL approach aggregates parameters located at the same coordinates across various local models. However, when

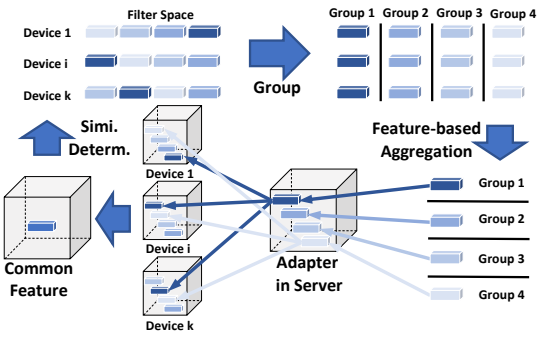

Fig. 4. The feature-based heterogeneous model aggregation process.

such a model aggregation method is applied to heterogeneous FL, the gradient direction can become unpredictable [14].

Therefore, we propose a novel heterogeneous aggregation strategy based on feature representation. In this approach, we also leverage the common feature of filters in the global model as the evaluation criterion. As shown in Fig. 4, after the server collects the parameters uploaded by different nodes, we proceed with a comparison of the different filters against the common feature of the global model to assess their similarity. Based on this comparison, the filters are ranked and grouped according to their similarities. Filters within the same group exhibit a higher degree of similarity in their feature representations. The filters within each group are then aggregated, resulting in a new aggregated Adapter. Subsequently, the aggregated parameters are sent back to each node. Meanwhile, the server updates and stores the common feature of the new aggregated Adapter in the same way presented in the preceding section.

In the aforementioned process, it is important to note that the common feature of the global model undergoes dynamic changes after each round. This adaptive aggregation method effectively caters to the evolving nature of the training process within the same task while also accommodating the inherent feature variations present in new data across different tasks. Moreover, by incorporating historical information from the previous round as an evaluation criterion, we can alleviate the impact of training result fluctuations resulting from the heterogeneous data on the aggregation of the global model.

Once the training of the Adapter is completed, the Adapter parameters are concatenated with the frozen Retainer parameters, thereby forming the updated global model. In the event of a further new task, utilizing the newly obtained global model parameters, we proceed to determine the Adapter and Retainer structures. By updating the Retainer, it can retain the old knowledge from the previous time steps while incorporating the new knowledge from the current time step. The Retainer ensures that valuable information acquired in previous tasks is preserved, allowing the global model to retain its generalizations across multiple tasks. This capability helps alleviate the problem of catastrophic forgetting.

VI. Experimental Setup

A. Datesets and Baselines

Split-Caltech-256 [15]: To construct a Non-IID dataset, we selected 250 classes from the Caltech-256 dataset and organized

TABLE I
PERFORMANCE COMPARISON WITH SIX BASELINES. A_5 IS AVERAGE ACCURACY(%) ON 5 TASKS, F IS AVERAGE FORGETTING(%) ON 5 TASKS.

Method	Split-Caltech		Split-MNIST		Split-CIFAR-100		Split-Traffic Sign		IID-CIFAR-100		Params	FLOPs
	A_5	F	A_5	F	A_5	F	A_5	F	A_5	F	(MB)	(M)
STL	50.63	/	62.38	/	42.31	/	53.43	/	47.62	/	57.15	/
FedAvg	28.69	18.75	39.32	13.47	19.73	38.21	35.74	20.73	27.59	25.14	11.43	21.91
FedEWC	17.90	23.81	38.75	16.51	18.22	40.32	33.85	22.45	25.92	29.36	22.86	21.91
FedLwF	53.28	3.98	64.37	2.65	38.73	5.39	52.49	4.02	43.34	4.97	15.35	30.56
CHFL	57.21	4.02	65.96	3.34	40.79	5.88	55.13	4.82	45.52	5.23	28.68	58.91
FedWeIT	64.21	3.79	67.24	1.98	43.25	6.93	57.27	2.18	51.17	4.45	3660	266.8
Ours	66.58	4.83	70.45	3.58	44.41	8.72	60.45	3.77	52.24	6.14	6.63	11.54

them into 50 Non-IID subtasks. Split-MNIST series: Including MNIST [16], FashionMNIST [17], and Not-MNIST [18]. We divide a total of 30 classes from the three datasets into 6 Non-IID tasks. Split-CIFAR-100 [19]: We partition the dataset into 20 Non-IID tasks, with each task consisting of 5 distinct classes. Split-Traffic Sign [20]: We divide this dataset into 8 Non-IID tasks, with each task consisting of 5 classes. IID-CIFAR-100: The total of 100 image classes is divided into disjoint 10-class datasets. Each dataset is further divided into 6 IID sub-datasets, each consisting of 10 classes.

In our experimental setup, tasks belonging to the same dataset are randomly allocated to various local devices. Each device receives a new task corresponding to the dataset when it enters a new time step. We use AlexNet as the backbone structure for each dataset. And 6 baselines are selected for comparison, including Single-task learning (STL), FedAvg [3], and 4 FCL methods: FedEWC [7], FedLwF [10], CHFL [12], and FedWeIT [9].

B. Evaluation Metrics

In accordance with standard practices in CL, we present the average accuracy and average forgetting metrics [4], [21]. Average Accuracy: This metric quantifies the model's performance after training on a sequence of consecutive tasks:

$$A_t = \frac{1}{t}\sum_{i=1}^{t} a_{t,i}, \tag{7}$$

where $a_{t,i}$ refers to the model performance on task i after being trained on task t.

Average Forgetting: This metric measures the accuracy decline for each task by comparing the highest accuracy achieved during training to the final accuracy obtained when the model training is completed. It is calculated as follows:

$$F = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{1,\ldots,T-1} \left(a_{t,i} - a_{T,i}\right). \tag{8}$$

VII. EXPERIMENTAL RESULTS

A. Performance Evaluation

We conduct a comparative analysis across multiple datasets (see Table I) simulating FL with 10 edge devices learning 5 consecutive tasks. Our approach partitions the network into Adapter and Retainer components, excluding the one deep and one shallow layers, each allocated 50% of the structure.

TABLE II
ACCURACY(%) OF LOCAL MODELS ON SPLIT-CIFAR100 DATASET.

Device Index	**FedAvg**	**FedLwF**	**CHFL**	**FedWeIT**	**Ours**
0	58.14	68.23	72.34	76.23	79.54
1	47.36	71.34	74.13	67.34	77.13
2	52.17	59.12	61.86	62.01	71.67
3	51.85	62.71	62.51	68.24	75.27
4	44.46	63.56	67.59	71.46	73.41
5	46.81	68.23	75.76	72.65	75.59
6	51.67	60.56	70.34	63.91	76.56
7	55.75	66.41	63.46	69.45	77.18
8	52.58	68.58	62.81	72.85	79.92
9	46.51	70.34	67.34	70.23	78.19
Avg.	50.73	65.91	67.81	69.44	76.45
std	4.37	4.19	5.14	4.27	2.58

Results highlight FedAvg's inferior performance with heterogeneous data and uncorrelated tasks. FedEWC also falls short due to global aggregation and its task alignment nature. Our method outperforms FedLwF and competes favorably with CHFL and FedWeIT, excelling in average accuracy. While the latter two excel in average forgetting, they suffer in average accuracy due to retained old knowledge and slower convergence.

Experimental findings show substantial improvements in our method's average accuracy across datasets (2.37%, 3.21%, 1.16%, 3.18%, and 1.07%) and significantly lower average forgetting rates, showcasing an effective balance between acquiring new knowledge and preserving past learning results.

B. Resource Usage

We conduct a comparative analysis of different methods concerning model complexity and FLOPs. The results are shown in the last two columns in Table I. CHFL increases storage and computational demands with continuously increasing the model scale. Similarly, FedWeIT requires substantial storage for task-specific parameters, making it impractical for edge deployment. In contrast, our method's training of Adapter reduces storage and computational requirements, making it more suitable for resource-limited edge devices. Furthermore, with less training parameters, our method efficiently addresses communication bottlenecks during parameter aggregation.

C. Effectiveness for Heterogeneous Data

In this section, we validate our method's efficacy in handling heterogeneous data without considering the CL problem. We

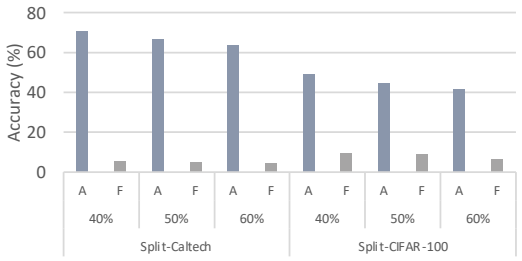

Fig. 5. The average accuracy and forgetting with Retainer in different scales.

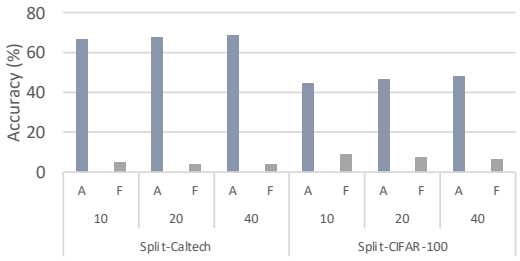

Fig. 6. The scalability evaluation with more participants.

conduct experiments in a single task under a highly Non-IID setting that each device only has 5 classes of images, comparing model accuracy across devices. The results in Table II demonstrate our method's accuracy, revealing its capability to mitigate the impact of heterogeneous data during model training. Notably, our method yields a reduction in accuracy std by 40.96%, 38.42%, 49.81%, and 39.58%. The reduced fluctuations in accuracy across devices signify its effectiveness in handling data heterogeneity.

D. Impact of Retainer Scale

In the previous section, we allocated 50% of the model structure to the Retainer, excluding shallow and deep layers. In this experiment, we explore the impact of different Retainer scales $\{40\%, 50\%, 60\%\}$ on FCL performance, as shown in Fig. 5. Increasing the Retainer size aids in preserving old knowledge but hinders new knowledge acquisition. Conversely, reducing the Retainer size allows more parameters for training new tasks, improving average accuracy but negatively impacting average forgetting. This empirical investigation guides the selection of an appropriate division ratio.

E. Scalability Evaluation

The number of participating devices in FL significantly impacts training accuracy. We investigate this by conducting experiments with 20 and 40 participants, as shown in Fig. 6. As the number of devices increases, available training samples grow, but so does data heterogeneity. Our approach exhibits scalability, achieving improved accuracy while effectively handling data heterogeneity. However, practical scenarios involve contention for limited communication bandwidth as participants increase, exacerbating communication issues. For instance, with a rise from 10 to 40 participants, communication quadruples, but accuracy gain is less than 5%. Therefore, increasing participants is inefficient when considering the trade-off between communication overheads and accuracy improvements.

VIII. Conclusions

This article tackles catastrophic forgetting in FL by introducing a resource-efficient approach for heterogeneous FCL on edge. We divide the global model into two components: an Adapter for new knowledge and a Retainer for old knowledge. Training a portion of the network structure reduces computation and communication costs. Adapter training employs feature-based aggregation to mitigate data heterogeneity's impact. Extensive experiments showcase our method's superior balance between learning new and retaining old knowledge.

Acknowledgment

This work was supported by National Key R&D Program of China (No. 2022YFB2901100).

References

[1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, and et al., "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1-2, pp. 1–210, 2021.

[2] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE TKDE*, 2021.

[3] H. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[4] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-GEM," in *ICLR*, 2019.

[5] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," in *arXiv preprint arXiv:1606.04671*, 2016.

[6] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, "Class-incremental learning via deep model consolidation," in *WACV*, 2020.

[7] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, and et al., "Overcoming catastrophic forgetting in neural networks," in *Proceedings of the national academy of sciences*, vol. 114, no. 13, 2017, p. 35213526.

[8] S. Yue, J. Ren, J. Xin, S. Lin, and J. Zhang, "Inexact-admm based federated meta-learning for fast and continual edge learning," in *Mobihoc*, 2021.

[9] J. Yoon, W. Jeong, G. Lee, E. Yang, and S. J. Hwang, "Federated continual learning with weighted inter-client transfer," in *ICML*, 2021.

[10] A. Usmanova, F. Portet, P. Lalanda, and G. Vega, "A distillation based approach integrating continual learning and federated learning for pervasive services," in *CML-IOT IJCAI workshop*, 2021.

[11] Z. Li and D. Hoiem, "Learning without forgetting," in *IEEE TPAMI*, vol. 40, no. 12, 2017, pp. 2935–2947.

[12] J. Mori, I. Teranishi, and R. Furukawa, "Continual horizontal federated learning for heterogeneous data," in *IJCNN*, 2022.

[13] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, 2009.

[14] F. Yu, W. Zhang, Z. Qin, Z. Xu, D. Wang, C. Liu, Z. Tian, and X. Chen, "Fed2: Feature-aligned federated learning," in *SIGKDD*, 2021.

[15] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[16] Y. LeCun. (1998) The mnist database of handwritten digits. [Online]. Available: https:// goo.gl/t6gTEy

[17] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," in *arXiv preprint arXiv:1708.07747*, 2017.

[18] Y. Bulatov. (2011) Not-mnist dataset. [Online]. Available: https://goo.gl/t6gTEy

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Tech. Rep.*, 2019.

[20] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

[21] S. I. Mirzadeh, M. Farajtabar, R. Pascanu, and H. Ghasemzadeh, "Understanding the role of training regimes in continual learning," *NeurIPS*, 2020.