Cluster-Based Heterogeneous Federated Foundation Model Adaptation and Fine-Tuning

Xianda Wang[1,2,*], **Yaqi Qiao**[1,*], **Duo Wu**[3,*], **Chenrui Wu**[4,5], **Fangxin Wang**[2,1,†]

[1]Future Network of Intelligence Institute (FNii), The Chinese University of Hong Kong, Shenzhen
[2]School of Science and Engineering (SSE), The Chinese University of Hong Kong, Shenzhen
[3]Shenzhen International Graduate School, Tsinghua University
[4]School of Computing Science, Simon Fraser University
[5]College of Computer Science, Zhejiang University
{xiandawang@link., yaqiqiao@link., wangfangxin@}cuhk.edu.cn, wu-d24@mails.tsinghua.edu.cn, chenrui_wu@sfu.ca

Abstract

In recent years, the distributed training of foundation models (FMs) has seen a surge in popularity. In particular, federated learning enables collaborative model training among edge clients while safeguarding the privacy of their data. However, federated training of FMs across resource-constrained and highly heterogeneous edge devices encounter several challenges. These include the difficulty of deploying FMs on clients with limited computational resources and the high computation and communication costs associated with fine-tuning and collaborative training. To address these challenges, we propose `FedCKMS`, a **C**luster-Aware Framework with **K**nowledge-Aware **M**odel **S**earch. Specifically, `FedCKMS` incorporates three key components. The first component is multi-factor heterogeneity-aware clustering, which groups clients based on both data distribution and resource limitations and selects an appropriate model for each cluster. The second component is knowledge-aware model architecture search, which enables each client to identify the optimal sub-model from the cluster model, facilitating adaptive deployment that accommodates highly heterogeneous computational resources across clients. The final component is cluster-aware knowledge transfer, which facilitates knowledge sharing between clusters and the server, addressing model heterogeneity, and reducing communication overhead. Extensive experiments demonstrate that `FedCKMS` outperforms state-of-the-art baselines by 3-10% in accuracy.

Introduction

Recent advances in foundation models (FMs), driven by transformers, increased computational power, and large-scale training data, have enabled exceptional performance across various tasks and diverse applications when fine-tuned on specific datasets (Chernyavskiy et al. 2021; Naveed et al. 2023; Le Scao et al. 2023). However, their development faces challenges in some real-world scenarios owing to the imperative of maintaining private data privacy. In order to solve this problem, federated learning (FL) is a feasible solution, enabling decentralized clients to collaboratively train models without sharing raw data (Chen et al. 2023; McMahan et al. 2017; Zhang et al. 2024; Wu et al. 2024).

While federated learning has seen successful applications, fine-tuning FMs through cloud-edge collaborative training still faces the following challenges due to the deployment and training demands of FMs: **a) Deploying appropriate FMs to heterogeneous clients with varying resource capacities is challenging.** Popular FMs often contain hundreds of billions of parameters that require training (Brown et al. 2020). However, the computational resources of edge devices are limited, such as those with 2-4 GB of RAM (Li et al. 2020). Therefore, edge clients are unable to support local FM training. Thus, traditional federated algorithms, such as FedAvg (McMahan et al. 2017) and its variants (Karimireddy et al. 2020), are unsuitable, as they require that all client models share the same structure as the server model. **b) Collaborative FMs training entails large computation and communication overhead.** Exchanging FM parameters or gradients frequently in federated learning can result in significant communication and computation overhead, especially with limited bandwidth (Mao et al. 2022). **c) Heterogeneous data and resource distribution lead to imbalanced training, slow convergence, and poor performance.** FMs raise much higher demand for the quality of training data so that practical data and resource heterogeneity can have a serious impact (Sun et al. 2024).

To address these challenges, adopting different model architectures across clients is a common approach. This involves transferring knowledge from various clients or aggregating different structured models into the server without leaking private data (Ye et al. 2023). In previous works, knowledge distillation (KD) and partial training (PT) are two solutions to model heterogeneous scenarios. The goal of KD is to use the client model (teacher model) to guide the server model (student model) to achieve comparable performance (Li et al. 2024). Integrating KD into FL improves model performance across heterogeneous client devices (Cheng et al. 2021; Cho et al. 2022; Deng et al. 2023; Fan et al. 2024). However, directly using KD to train the FM on the server is

[*]These authors contributed equally.
[†]Corresponding author

not practical. As KD is constrained by performance of the teacher model, the terrible performance of models located on resource-limited or data-biased clients will negatively affect FM training. PT-based approaches deploy sub-models from the server model (full model) to clients for training, updating the server model by aggregating these trained sub-models from clients (Wu et al. 2023b). However, regardless of the method used to extract sub-models, such as random selection or rolling extraction, the sub-model structures are still constrained by the server model. Although recent work (Jiang et al. 2022) proposes to leverage model pruning techniques to address resource limitations and model heterogeneity, this approach also faces the challenge of client models being constrained by the structure of the server model, and the pruning ratio can significantly impact the accuracy. Therefore, we delve into the research of heterogeneous federated learning to address the following key problem: **how can effective federated FM training be achieved across resource-constrained and highly heterogeneous edge devices, with strong data heterogeneity?**

In this paper, we propose `FedCKMS`, which is a cluster-based framework that utilizes a PT-based method and a KD-based method. To be specific, `FedCKMS` incorporates the following three key building blocks:

- **Multi-factor heterogeneity-aware clustering (MHAC).** Initially, `FedCKMS` clusters clients based on data distribution and computational resources and selects one client as the leader node within each cluster based on its computational capacity. A smaller FM (compared to the server FM) will be deployed in the leader node as the base model for this cluster. By utilizing MHAC, the challenges posed by system heterogeneity can be effectively addressed.
- **Knowledge-aware model architecture search (KAMAS) for intra-cluster aggregation.** We design the KAMAS algorithm for transformer architectures inspired by NASWOT (Mellor et al. 2021), which can search for the optimal sub-models without any training. Different clients obtain sub-models by searching from the cluster base model according to their own computing resource limitations and then aggregate the corresponding model parts after local training. KAMAS achieves dynamic FM deployment based on resource constraints, thereby addressing the challenge of FM deployment in resource-limited and dynamic edge environments. In addition, to further reduce computational overhead, our framework applies the parameter-efficient fine-tuning (PEFT) technique Low-Rank adaptation (LoRA) (Hu et al. 2022).
- **Cluster-aware knowledge transfer (CAKT) for inter-cluster aggregation.** After cluster training, domain knowledge from different clusters is transferred to the server using a weighted knowledge distillation approach, which is based on the quality of each cluster's knowledge. Using CAKT significantly reduces communication overhead while simultaneously addressing knowledge transfer between different model architectures.

Our main contributions to this paper are as follows.

- We propose `FedCKMS`, a novel hierarchical federated framework that supports FM partial training aggregation and knowledge transfer to address model heterogeneity in a resource-constrained scenario. Additionally, it effectively reduces computational and communication overhead without compromising accuracy.
- We implement a Knowledge-Aware Model Architecture Search (KAMAS) algorithm to extract the optimal sub-model from a FM without requiring any training, enabling `FedCKMS` to fully utilize the diverse heterogeneous resources across numerous edge devices.
- We implement `FedCKMS` and compare it against various methods under the assumption of limited resources. Extensive experimental results show that `FedCKMS` outperforms baselines in both IID and non-IID scenarios with an improvement in accuracy of 3-10%.

Related Work

Model Heterogeneity

To address model heterogeneity, there are two main approaches: knowledge distillation (KD) based methods and partial training (PT) based methods.

KD-based methods. KD-based methods address model heterogeneity by transferring knowledge between models without exchanging parameters. Nevertheless, existing methods still face several limitations. For example, FedDF (Lin et al. 2020) heavily relies on the performance of the teacher model, making the overall performance dependent on the accuracy of each client model. Fed-ET (Cho et al. 2022) proposes a dual distillation mechanism to prevent catastrophic forgetting, which, however, increases communication overhead. FedGen (Zhu, Hong, and Zhou 2021) introduces a lightweight generator to facilitate knowledge exchange among clients, but it requires extensive communication and coordination, which can be challenging in practice. FedHKT (Deng et al. 2023) transfers specific knowledge from clusters to the server, but it uses FedAvg (McMahan et al. 2017) for parameter aggregation within clusters. This approach can lead to high communication overhead and edge resource limitations, similar to traditional federated learning. In this paper, we introduce `FedCKMS`, which significantly reduces computation and communication requirements and minimizes reliance on the teacher model, thereby mitigating the impact of data heterogeneity.

PT-based methods. PT-based methods tackle model heterogeneity by training sub-models that are subsets of the server model and then aggregating these sub-models. For instance, Federated Dropout (Wen, Jeon, and Huang 2022) and HeteroFL (Diao, Ding, and Tarokh 2020) apply random and specific portions of the server model, respectively, leading to the inability to fully train the entire server model. Although FedRolex (Alam et al. 2022) employs a rolling selection of sub-models, it trains the whole model without prioritizing the selection of parameters that significantly impact performance. Furthermore, a key drawback of PT-based methods is that the model structure deployed on the clients depends on the server's. In contrast, we design `FedCKMS` to allow base models in different clusters to be entirely independent of the server model, enabling heterogeneous model structures.

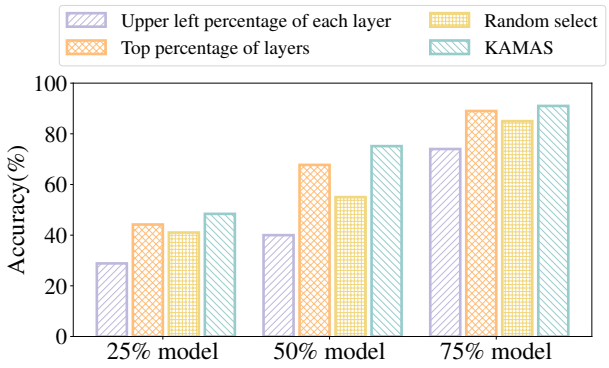

Figure 1: Illustration of the effectiveness of KAMAS by Comparison of 25%, 50% and 75% density CLIP-large models extracted by different methods.

Cluster Federated Learning

In traditional federated learning, with non-IID data on each client, the distributed training process converges slowly and incurs a large communication overhead. To tackle these issues, clustered federated learning (CFL) emerges, which divides clients into multiple clusters, and selects a representative subset of each cluster as the client model aggregator (Sattler et al. 2020; Ghosh et al. 2019; Zhang et al. 2023). There are two popular clustering methods that improve the performance of FL training. The first method is grouping distributed clients with similar data distribution into the same cluster during static clustering (Deng et al. 2021; Huang et al. 2023). The other method is reasonably selecting partial update clients during dynamic updating (Wang et al. 2021; Ni and Hashemi 2023). However, these approaches are not directly applicable in the scenario where distributed clients collaboratively fine-tune an FM. Given the varying sizes of FMs assigned to each cluster, edge device resources must be considered in the cluster partitioning process, which is neglected in previous works.

Design of FedCKMS

Preliminaries

In this paper, we consider a system with heterogeneity, where clients have varying resources and model architectures. Specifically, we have K clients, which are grouped into M clusters. Each client k has a local, private labeled training dataset $\mathcal{D}_k$, which is defined as $\mathcal{D}_k = \{\mathbf{X}^k, \mathbf{Y}^k\}$, where $\mathbf{X}^k = \{x_1^k, x_2^k, \ldots, x_{N^k}^k\}$ and $\mathbf{Y}^k = \{y_1^k, y_2^k, \ldots, y_{N^k}^k\}$ denote the data samples and corresponding labels, respectively. Here, N^k is the total number of samples. For KD-based knowledge transfer, we assume the availability of an unlabeled public dataset $\hat{\mathcal{D}}$, denoted as $\hat{\mathcal{D}} = \{\hat{\mathbf{X}}\}$, where $\hat{\mathbf{X}} = \{\hat{x}, \hat{x}_2, \ldots, \hat{x}_N\}$, which is accessible to the leader node of each cluster and the cloud server. In this highly heterogeneous system, the resources of each client are different, denoted as $R = \{R_1, R_2, \ldots, R_k\}$. Each client k utilizes its private dataset to develop a local model, represented as θ_k. If the client is the leader node of m-th cluster, its model is denoted as $\theta_{(m)}$. In addition to these local models, there is a larger server model θ that will be trained at the cloud server.

Overview

We design FedCKMS, an efficient and scalable fine-tuning framework that supports heterogeneous model aggregation and knowledge transfer, as illustrated in Figure 2. Firstly, Multi-factor Heterogeneity-aware Clustering (MHAC) groups clients with similar data distribution and comparable computational resources into a single cluster. From each cluster, the client with the most abundant computational resources is selected as the leader node, where partial parameter aggregation within the cluster is conducted. Before training, the leader node will select an appropriate FM as the representative of the cluster, based on the available computational resources. Secondly, Knowledge-Aware Model Architecture Search (KAMAS) is implemented to perform adaptive model searches, selecting the optimal sub-model for clients to conduct local training under resource-constrained conditions. After the activated cluster clients complete local training, KAMAS-Driven Aggregation (KDA) will aggregate the corresponding model parameters from each client. Afterwards, the knowledge is transferred to the server through the cluster's leader node using Cluster-Aware Knowledge Transfer (CAKT), utilizing an unlabeled public dataset. Ultimately, via reverse knowledge distillation, the server model's knowledge is transferred and merged into the leader node of the cluster, and subsequently distributed to clients through a parameter-based method.

Multi-factor Heterogeneity-aware Clustering

In scenarios where clustering is done directly based on the attributes of the edge devices, previous work often focuses on data distribution because the edge models are isomorphic (Deng et al. 2021; Wu et al. 2023a; Huang et al. 2023). In Model-heterogeneity-aware, in addition to data distribution, the computational power of each edge device measured by video memory is considered.

Inspired by previous work (Xia et al. 2020), we use the K-means algorithm to achieve the goal of clustering based on data distribution and computational power in a concise and fast way, and implement differential privacy to protect the data of the edge devices from being leaked. The leader node in each cluster is then selected based on recourse.

First, to protect data privacy on edge devices, we adopt the method of local differential privacy, that is, each client is required to add interference information before uploading data. In the specific implementation, we add Laplace noise to the data on each device:

$$\tilde{\mathcal{D}}_k = \mathcal{D}_k + Laplace(0, \frac{s}{\epsilon}), \tag{1}$$

where s and ϵ are hyper-parameters. The data with added noise is denoted by $\tilde{\mathcal{D}}_k$, which shows the distribution characteristics of the data D_k without leaking the initial data.

Furthermore, with R_k denoting the video memory of the client device, the features, $\tilde{\mathcal{D}}_k$ and R_k are normalized and then stitched together to form a composite feature vector $\mathcal{X}_k$.

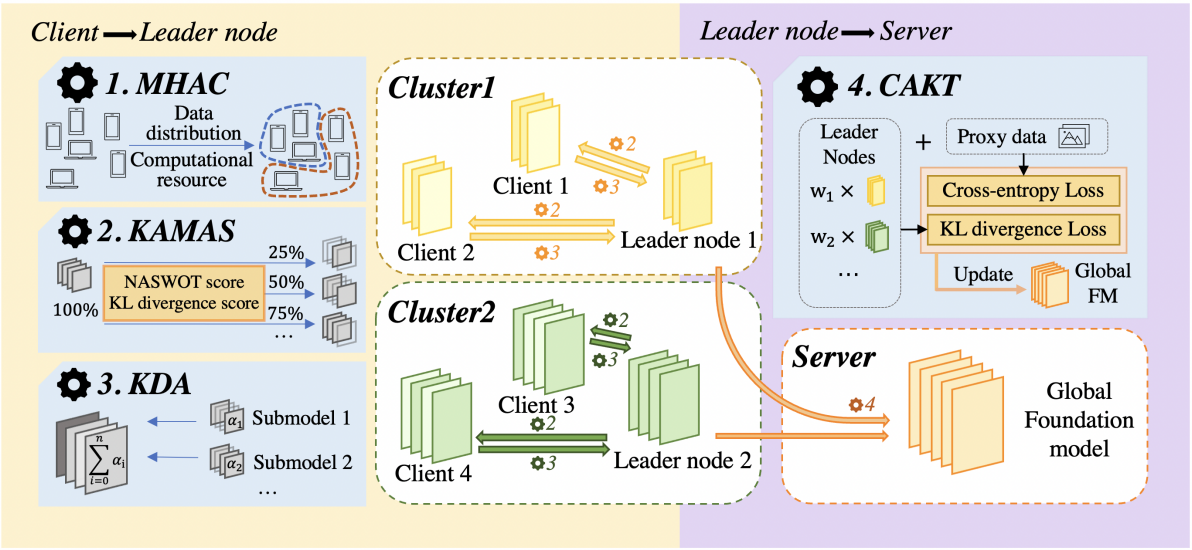

Figure 2: The overview of FedCKMS framework. First, MHAC groups clients by data distribution and computational resources, and selects the leader node. Second, using the KAMAS algorithm, each client in the cluster searches the optimal sub-model of the cluster model to match its computational resource constraints. After local training, KDA will aggregate corresponding model parameters from each client. Finally, CAKT enables inter-cluster knowledge transfer to update the server model. After each training round, global knowledge is distilled back to the leader nodes, which then update the corresponding client parameters.

Next, we use the K-means algorithm to group $\mathcal{X}_k$ to obtain M clusters. To promote balanced and effective contributions from each cluster in the subsequent training phase, it is essential to establish a lower bound on the number of clients within each cluster. Therefore, we set a stopping condition for multiple rounds of the K-means algorithm: stop clustering when all clusters contain at least $\frac{K}{2M}$ clients. Finally, the client with the most abundant resources in each cluster is chosen as the leader node, where a complete and appropriately suited FM is deployed.

Model Search and Partial Aggregation

Owing to the substantial heterogeneity between the clients, even following the clustering process, there are still a number of clients whose computational resources are limited compared to the leader node. Due to the parameter aggregation approach adopted within the cluster, employing traditional federated learning algorithms such as FedAvg could lead to a scenario where certain clients are unable to deploy the model of the leader node. Partial training is an approach to tackle model heterogeneity and the limitations of client-side resources. By extracting a sub-model, the overhead associated with training is significantly reduced.

However, in extracting submodels, existing methods, such as HeteroFL and FedRolex, select the top k% of nodes in each layer rather than performing targeted model extraction. Depth pruning, which removes entire layers or Transformer blocks, can effectively reduce memory requirements while preserving model performance as much as possible (Kim et al. 2024). As shown in Figure 1, our experiments confirm that in transformers, retaining entire layers while reducing the number of layers through structured pruning is more beneficial for the potential performance of sub-model. Therefore, clients obtain sub-models by selectively extracting certain layers from the lead node model for each cluster. To identify which layers are critical to sub-model performance, we design Knowledge-Aware Model Architecture Search (KAMAS), an algorithm based on Neural Architecture Search without Training (NASWOT) (Mellor et al. 2021) and Kullback-Leibler (KL) Divergence score.

We design two metrics that can be used to evaluate the potential performance of a sub-model without any training. The first is a score based on NASWOT calculations, and the second is the KL divergence between the logits produced by the sub-model and those of the dense model. The NASWOT score leverages the initial activation patterns of the activation function within an untrained network to estimate its performance. The kernel matrix $\mathcal{K}$ is constructed by computing the Hamming distances between binary codes that represent the activation states of mini-batch data points $\mathbf{X} = \{x_i\}_{i=1}^n$. The score, derived from the logarithm of the value of the determinant of $\mathcal{K}$, is given by

$$\mathcal{K} = (N_A - d_H(c_i, c_j))_{i,j=1}^n, \quad s = \log|\mathcal{K}|, \tag{2}$$

where N_A is the number of activation function, c is the activation codes, and d_H denotes the hamming distance. This score quantifies the potential of untrained networks, offering an insightful correlation with the model's accuracy once trained. Another metric is KL divergence score, denoted as d, can be calculated as:

$$d = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{T}\sum p(x_i)\cdot\log\left(\frac{p(x_i)}{q(x_i)}\right)\right)^{\frac{1}{T}}, \tag{3}$$

where p is the logits of the original model, q is the logits of the sub-model, and T is a temperature parameter. We then

Algorithm 1: Knowledge-Aware Model Architecture Search

Input: Crossover rate p_c, Mutation rate p_m, Population size
Output: Sub-model structure

Initialize population with random sub-model structures.
Evaluate the fitness of each sub-model in the population.
for each generation **do**
    Randomly select two sub-model structures from the population, A and B.
    Compute $\mathcal{F}_A$ and $\mathcal{F}_B$, the fitness of A and B, using Eq. 2, 3, 4
    $winner, loser \leftarrow argmax(\mathcal{F}_A, \mathcal{F}_B)$
    if random number $r < p_c$ **then**
        Crossover $\{winner$'s structure, $loser$'s structure$\}$.
    end if
    if random number $r < p_m$ **then**
        Mutation $loser$'s structure.
    end if
    Re-evaluate $\mathcal{F}$ of the mutated loser, using Eq. 2, 3, 4
    Replace the loser with the mutated sub-model.
end for
return The sub-model structure with the highest fitness.

consider both the NASWOT score and the KL divergence score to implement a simple yet effective genetic algorithm. The fitness $\mathcal{F}$ of a sub-model is defined as:

$$\mathcal{F} = s - d. \tag{4}$$

By applying a simple genetic algorithm, we can perform crossover or mutation operations on the models' structures to effectively search for the sub-model structure with the highest fitness in the population, thereby identifying crucial parameters that significantly impact accuracy, without any additional training. The detailed algorithm is illustrated in Algorithm 1. Assuming that computational requirements are directly proportional to the number of model parameters, KAMAS search the optimal sub-model under computational constraints. This approach ensures that the selected parameters are the most critical to the model's potential performance, allowing for better knowledge retention. Shown in Figure 1, the sub-model obtained through the KAMAS search outperformed those obtained by directly selecting the top k% layers or by random selection after training.

Within each cluster, clients select a search ratio $\beta \in (0, 1]$ based on their computational constraints and use the KAMAS algorithm to search for the optimal sub-model, enabling adaptive deployment. After local training, clients upload sub-models to the cluster's leader node. The leader node aggregates the corresponding layers of all uploaded sub-model parameters, as shown in Figure 3. Once aggregation is complete, the leader redistributes the parameters to each client according to their sub-model structure. This process is repeated in a cycle, referred to as cluster training.

Cluster-Aware Knowledge Transfer

After cluster training, the knowledge from different clusters must be aggregated on the server to finalize the server model training. Given the varying capabilities of leader nodes across clusters, we introduce a cluster-aware knowledge distillation algorithm. This algorithm assigns aggregation weights based on the accuracy of each leader node evaluated on its test data with n samples, ensuring that the server model benefits the most from high-performing clusters.

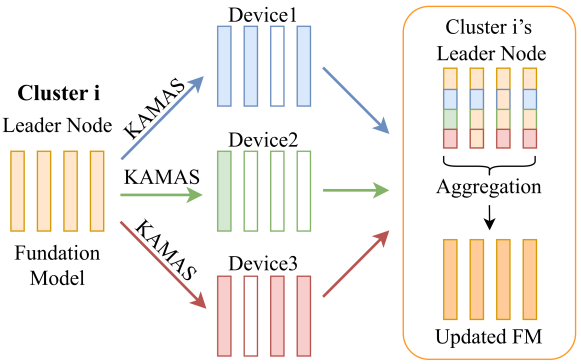

Figure 3: Illustration of KDA by using 3 devices as an example. In KDA process, the leader node will aggregate the corresponding layers (Transformer blocks) from the clients.

$$w_m = \frac{\sum_{i=1}^{n} I(f(\theta_{(m)}; x_i^k) = y_i^k)}{\sum_{m=1}^{M} \left(\sum_{i=1}^{n} I(f(\theta_{(m)}; x_i^k) = y_i^k)\right)}. \tag{5}$$

Since the used public dataset is unlabeled, pseudo-labels are generated using the cluster leader node model $\theta_{(m)}$ for each sample $\hat{x}_i \in \hat{\mathcal{D}}$ in the public dataset:

$$\hat{y}_{m,i} = \text{argmax}_{c \in \{1,2,\cdots,C\}} f(\theta_{(m)}; \hat{x}_i). \tag{6}$$

Afterwards, we can adopt cross-entropy loss to update the server model θ:

$$\mathcal{L}_{CE}^{m} = w_m \times \mathbb{E}_{\hat{x} \sim \hat{\mathcal{D}}} \left[L_{\text{CE}} \left(\theta; (\hat{x}, \hat{y}_m) \right) \right], \tag{7}$$

where $\hat{y}_m$ denotes the pseudo-labels generated by the m-th cluster. Meanwhile, KL divergence loss can be calculate by:

$$\mathcal{L}_{KL}^{m} = w_m \times \mathbb{E}_{\hat{x} \sim \hat{\mathcal{D}}} \left[D_{KL}(\sigma(f(\theta; \hat{x})) \parallel \sigma(f(\theta_{(m)}; \hat{x}))) \right]. \tag{8}$$

Finally, the server model is finally trained by minimizing the following loss function:

$$\mathcal{L} = \sum_{m=1}^{M} \left(\gamma \times \mathcal{L}_{CE}^{m} + (1 - \gamma) \times \mathcal{L}_{KL}^{m} \right). \tag{9}$$

Experimental Evaluation

Experimental Setup

Datasets Setting. We evaluate our method, `FedCKMS`, using three datasets: CIFAR-10, CIFAR-100 (Krizhevsky, Hinton et al. 2009), and Tiny-ImageNet (Le and Yang 2015). The CIFAR-10 dataset contains 10 classes with a total of 50,000 training images, while CIFAR-100, with 100 classes, presents a more challenging task. Tiny-ImageNet includes 200 classes, each with 500 images. For our experiments, we randomly select about 5000 images from the training data, remove their labels, and create an unlabeled public dataset

Dataset	IID/non-IID	DS-FL	FedDF	FedHKT	HeteroFL	FedRolex	FedRolex$*$	**FedCKMS**
CIFAR-10	IID	87.46%	89.25%	89.43%	92.3%	80.1%	92.57%	91.93%
	Dir(0.5)	79.13%	80.19%	88.39%	83.29%	77.09%	84.68%	89.97%
	Dir(0.1)	74.08%	75.36%	81.51%	75.4%	76.92%	77.89%	86.8%
	Average	80.22%	81.6%	86.44%	83.66%	78.03%	85.04%	89.56%
CIFAR-100	IID	67.29%	66.82%	66.86%	72.81%	46.94%	74.44%	71.12%
	Dir(0.5)	62.42%	60.66%	66.83%	66.32%	41.41%	67.45%	69.03%
	Dir(0.1)	54.74%	51.07%	57.3%	57.03%	40.24%	58.4%	61.82%
	Average	61.48%	59.51%	63.66%	65.38%	42.86%	66.76%	67.32%
Tiny-Imagenet	IID	59.72%	59.02%	57.7%	59.18%	38.64%	66.12%	63.68%
	Dir(0.5)	54.6%	52.16%	52.2%	56.95%	39.3%	56.52%	60.67%
	Dir(0.1)	42.36%	45.98%	53.85%	51.12%	40.19%	51.88%	58.59%
	Average	52.22%	52.38%	54.58%	55.75%	39.37%	58.17%	60.98%

$*$: The second FedRolex experiment is in an ideal scenario where at least one client's recourse is enough to deploy a model exactly the same as the server's FM.

Table 1: Test accuracy on three different datasets under different degrees of data heterogeneity

for use in knowledge distillation. We consider both IID and non-IID data distributions. In the IID setting, data is evenly distributed among clients in terms of both quantity and class distribution. In the non-IID setting, we use a Dirichlet distribution to allocate data across clients, with $\alpha = \{0.1, 0.5\}$.

Models Setting. We have selected CLIP (Radford et al. 2021) as the backbone model for our experiments. To simulate system heterogeneity, we assume eight different levels of computational resource constraints, denoted as $C = \{C_1, C_2, \ldots, C_8\}$. Different versions of CLIP are chosen to accommodate these constraints. The server model is CLIP-2.5B (Schuhmann et al. 2022), which has 2.5 billion parameters. Each cluster's leader node can choose the CLIP-base or CLIP-large model, based on its resource limitations, with 150 million and 430 million parameters respectively.

Baselines. We compared our methods with two groups of methods.1) KD-based FL: DS-FL (Itahara et al. 2021), FedDF (Lin et al. 2020) and FedHKT (Deng et al. 2023). 2) PT-based FL: HeteroFL (Diao, Ding, and Tarokh 2020), FedRolex (Alam et al. 2022). To preserve fairness, we applied LoRA (Hu et al. 2022) to both `FedCKMS` and the baselines.

Implementation details. For `FedCKMS`, with the model search algorithm KAMAS, we can extract sub-models with varying sizes. The search ratio $\beta \in \{1.0, 0.75, 0.5, 0.25\}$ corresponds to searched sub-models a, b, c, and d by using KAMAS. Sub-model a retains the full original model, while d means that only about 25% of the original model's parameters are preserved. Other clients within each cluster can then select an appropriate sub-model of the cluster's representative model based on their individual resource constraints. For the baselines, DS-FL, FedDF, and FedHKT, clients can choose between the CLIP-base and CLIP-large models based on their resource constraints. However, FedHKT requires homogeneous models within each cluster, so the model selection is determined by the most resource-constrained client in the cluster. For HeteroFL and FedRolex, we consider an ideal scenario where at least one client's model is sub-model a that matches the server model, while the other clients choose from sub-models b, c, and d, which correspond to 75%, 50%, and 25% of the server model's parameters, respectively. For both `FedCKMS` and all baselines, the server model is consistently the CLIP-2.5B model.

Hyper-parameters. For both `FedCKMS` and the baselines, we adopt the Adam optimizer with a learning rate of $\eta = 0.0005$ and a batch size of 8. For lora rank, we chose 64 as the standard for our experiments. We set the local training epochs to 3 and the communication rounds to 10, which were sufficient to achieve convergence.

Performance Results and Analysis

Performance Results. We compare the performance of `FedCKMS` against baseline methods, including DS-FL, FedDF, FedHKT, HeteroFL, and FedRolex, using the CIFAR-10, CIFAR-100, and Tiny-ImageNet datasets with 10 clients under both IID and non-IID settings. As shown in Table 1, `FedCKMS` outperforms these baselines, with an accuracy improvement of 3-10% in most scenarios. For both FedRolex and HeteroFL, we considered an ideal scenario where at least one client's resources are sufficient to deploy a model exactly the same as the server's FM, a condition that is difficult to achieve in real-world scenarios. In such settings, `FedCKMS` may slightly underperform compared to FedRolex. When we tested a more stringent scenario where no client could deploy a model identical to the server's, FedRolex's performance declined significantly, underperforming compared to `FedCKMS`.

Sensitivity Analysis. As shown in Table 1, under IID data distributions, methods based on partial training generally outperform those based on knowledge distillation. However, knowledge distillation demonstrates superior knowledge transfer capabilities in non-IID settings, especially with a Dirichlet distribution $\alpha = 0.1$, which indicates significant class imbalance across clients. `FedCKMS` performs exceptionally well in both IID and non-IID scenarios, closely approaching FedRolex's performance in the ideal IID case,

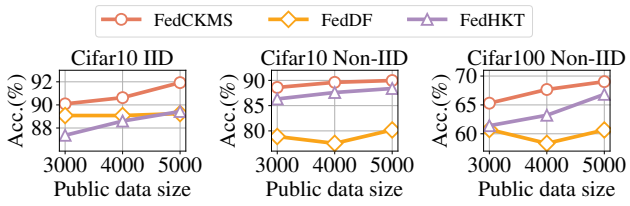

Figure 4: Performance comparison of FedCKMS and baselines across different public dataset sizes

by leveraging multi-factor heterogeneity-aware clustering with KAMAS-driven partial aggregation within clusters and knowledge distillation across clusters. Additionally, FedCKMS effectively utilizes clients' resources, even under stringent constraints on clients. Meanwhile, CAKT avoids transmitting parameters to the server, significantly reducing communication overhead compared to PT-based baselines.

Impact of public dataset size. To assess the effect of the public dataset, we adjust its sample size from 3,000 to 5,000 and compare the performance of FedCKMS with baseline methods that also utilize a public dataset. As shown in Figure 4, as the size of unlabeled public data increases, FedCKMS gains more knowledge, especially on more challenging tasks like CIFAR-100. Unlike FedDF, where increasing the public data size may sometimes decrease accuracy, FedCKMS better leverages this additional knowledge because CAKT utilizes pseudo labels to provide more information and employs weight distillation to enable high-performance leader nodes to share more knowledge.

Ablation Study

Effect of KAMAS for partial aggregation. To validate the effectiveness of the KAMAS-Driven partial aggregation within intra-cluster aggregation in FedCKMS, we compared it with the PT-based baseline, FedRolex. For simplicity and to clearly demonstrate its effectiveness, we use the CLIP-large model as the server model and conducted experiments on the CIFAR10 and CIFAR100 datasets. As shown in Table 2, different clients select sub-models with varying proportions, where a, b, c, and d represent 100%, 75%, 50%, and 25% of the server's sub-model, respectively. The relationship between the average search ratio and model combinations of all clients is detailed in the table. The results presented in Figure 5 illustrate that aggregating sub-models obtained through KAMAS results in higher accuracy for the final server model compared to FedRolex, especially when clients cannot deploy the same model as the server (i.e., the 100% model).

Effect of Cluster-Aware Knowledge Transfer. To validate the effectiveness of CAKT, we separately evaluate its performance by removing the weights assigned to each cluster in the knowledge distillation process and by considering only the KL loss from Eq. 9. As shown in Figure 6, removing the weights from the CAKT component results in varying degrees of accuracy degradation, as weight distillation enhances the contribution of better-performing teacher models, which in turn improves the accuracy of the server model.

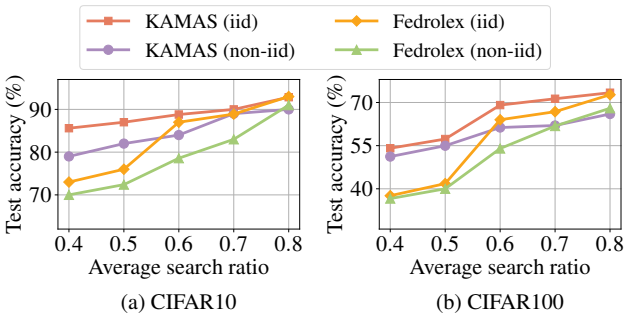

Figure 5: The model accuracy of FedRolex and FedCKMS with different search ratios (β) under IID and non-IID settings on the CIFAR-10 and CIFAR-100 datasets.

search ratio β	Average parameters p	Sub-model
0.4	$p = \lceil 0.4P \rceil$	b-d
0.5	$p = \lceil 0.5P \rceil$	b-c-d
0.6	$p = \lceil 0.6P \rceil$	a-b-c-d
0.7	$p = \lceil 0.7P \rceil$	a-b-c
0.8	$p = \lceil 0.8P \rceil$	a-b

Table 2: Average search ratio and corresponding sub-model combinations across all clients.

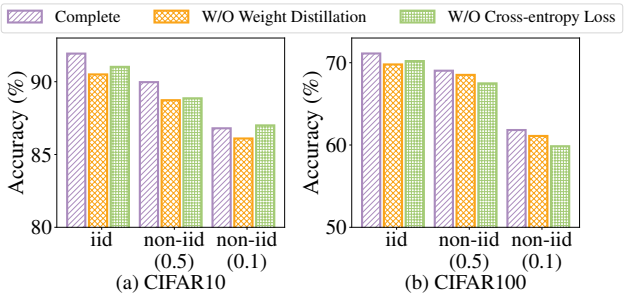

Figure 6: Comparison of accuracy after removing weights and cross-entropy Loss from CAKT.

Considering only the KL loss can lead to different levels of performance decline, as adding cross-entropy loss through the generation of pseudo-labels enhances robustness, particularly in complex tasks like CIFAR-100.

Conclusion

In this paper, we have proposed FedCKMS for federated FM training on highly heterogeneous and resource-constrained edge devices, in order to reduce computation and communication overhead and while still maintaining satisfying model accuracy. FedCKMS leverages CAKT, a cluster-aware weight distillation method, to significantly reduce communication overhead and facilitates effective knowledge transfer across different model architectures. Additionally, KAMAS and KDA enable the adaptation of deployment for highly resource heterogeneous clients, significantly reducing computational costs.

Acknowledgments

The work was supported in part by NSFC with Grant No. 62471423, the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, NSFC with Grant No. 62293482, the Shenzhen Science and Technology Program with Grant No. JCYJ20230807114204010, the Shenzhen Outstanding Talents Training Fund 202002, the Guangdong Research Projects No. 2019CX01X104, the Young Elite Scientists Sponsorship Program of CAST (Grant No. 2022QNRC001), the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001) and the Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. ZDSYS201707251409055).

References

Alam, S.; Liu, L.; Yan, M.; and Zhang, M. 2022. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *Advances in neural information processing systems*, 35: 29677–29690.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, C.; Feng, X.; Zhou, J.; Yin, J.; and Zheng, X. 2023. Federated large language model: A position paper. *arXiv preprint arXiv:2307.08925*.

Cheng, S.; Wu, J.; Xiao, Y.; and Liu, Y. 2021. Fedgems: Federated learning of larger server models via selective knowledge fusion. *arXiv preprint arXiv:2110.11027*.

Chernyavskiy, A.; et al. 2021. Transformers:"the end of history" for natural language processing? In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III 21*, 677–693. Springer.

Cho, Y. J.; Manoel, A.; Joshi, G.; Sim, R.; and Dimitriadis, D. 2022. Heterogeneous ensemble knowledge transfer for training large models in federated learning. *arXiv preprint arXiv:2204.12703*.

Deng, Y.; Lyu, F.; Ren, J.; Zhang, Y.; Zhou, Y.; Zhang, Y.; and Yang, Y. 2021. SHARE: Shaping data distribution at edge for communication-efficient hierarchical federated learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 24–34. IEEE.

Deng, Y.; Ren, J.; Tang, C.; Lyu, F.; Liu, Y.; and Zhang, Y. 2023. A hierarchical knowledge transfer framework for heterogeneous federated learning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, 1–10. IEEE.

Diao, E.; Ding, J.; and Tarokh, V. 2020. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*.

Fan, B.; Wu, C.; Su, X.; and Hui, P. 2024. FedTSA: A Cluster-based Two-Stage Aggregation Method for Model-heterogeneous Federated Learning. In *The 18th European Conference on Computer Vision (ECCV)*.

Ghosh, A.; Hong, J.; Yin, D.; and Ramchandran, K. 2019. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Huang, H.; Shi, W.; Feng, Y.; Niu, C.; Cheng, G.; Huang, J.; and Liu, Z. 2023. Active client selection for clustered federated learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Itahara, S.; Nishio, T.; Koda, Y.; Morikura, M.; and Yamamoto, K. 2021. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22(1): 191–205.

Jiang, Z.; Xu, Y.; Xu, H.; Wang, Z.; Qiao, C.; and Zhao, Y. 2022. Fedmp: Federated learning through adaptive model pruning in heterogeneous edge computing. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 767–779. IEEE.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, 5132–5143. PMLR.

Kim, B.-K.; Kim, G.; Kim, T.-H.; Castells, T.; Choi, S.; Shin, J.; and Song, H.-K. 2024. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Le, Y.; and Yang, X. 2015. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7): 3.

Le Scao, T.; Fan, A.; Akiki, C.; Pavlick, E.; Ilić, S.; Hesslow, D.; Castagné, R.; Luccioni, A. S.; Yvon, F.; Gallé, M.; et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Li, L.; Gou, J.; Yu, B.; Du, L.; and Tao, Z. Y. D. 2024. Federated Distillation: A Survey. *arXiv preprint arXiv:2404.08564*.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3): 50–60.

Lin, T.; Kong, L.; Stich, S. U.; and Jaggi, M. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems*, 33: 2351–2363.

Mao, Y.; Zhao, Z.; Yan, G.; Liu, Y.; Lan, T.; Song, L.; and Ding, W. 2022. Communication-efficient federated learning with adaptive quantization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4): 1–26.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Mellor, J.; Turner, J.; Storkey, A.; and Crowley, E. J. 2021. Neural architecture search without training. In *International conference on machine learning*, 7588–7598. PMLR.

Naveed, H.; Khan, A. U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Barnes, N.; and Mian, A. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

Ni, Z.; and Hashemi, M. 2023. Efficient Cluster Selection for Personalized Federated Learning: A Multi-Armed Bandit Approach. In *2023 IEEE Virtual Conference on Communications (VCC)*, 115–120. IEEE.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Sattler, F.; et al. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8): 3710–3722.

Schuhmann, C.; Beaumont, R.; Vencu, R.; Gordon, C. W.; Wightman, R.; Cherti, M.; Coombes, T.; Katta, A.; Mullis, C.; Wortsman, M.; Schramowski, P.; Kundurthy, S. R.; Crowson, K.; Schmidt, L.; Kaczmarczyk, R.; and Jitsev, J. 2022. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Sun, J.; Mei, C.; Wei, L.; Zheng, K.; Liu, N.; Cui, M.; and Li, T. 2024. Dial-insight: Fine-tuning Large Language Models with High-Quality Domain-Specific Data Preventing Capability Collapse. *arXiv preprint arXiv:2403.09167*.

Wang, Z.; Xu, H.; Liu, J.; Huang, H.; Qiao, C.; and Zhao, Y. 2021. Resource-efficient federated learning with hierarchical aggregation in edge computing. In *IEEE INFOCOM 2021-IEEE conference on computer communications*, 1–10. IEEE.

Wen, D.; Jeon, K.-J.; and Huang, K. 2022. Federated dropout—A simple approach for enabling federated learning on resource constrained devices. *IEEE wireless communications letters*, 11(5): 923–927.

Wu, C.; Wang, H.; Zhang, X.; Fang, Z.; and Bu, J. 2024. Spatio-temporal Heterogeneous Federated Learning for Time Series Classification with Multi-view Orthogonal Training. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 2613–2622.

Wu, C.; Zhu, Y.; Zhang, R.; Chen, Y.; Wang, F.; and Cui, S. 2023a. FedAB: Truthful Federated Learning With Auction-Based Combinatorial Multi-Armed Bandit. *IEEE Internet of Things Journal*, 10(17): 15159–15170.

Wu, H.; et al. 2023b. Model-heterogeneous federated learning with partial model training. In *2023 IEEE/CIC International Conference on Communications in China (ICCC)*, 1–6. IEEE.

Xia, C.; Hua, J.; Tong, W.; and Zhong, S. 2020. Distributed K-Means clustering guaranteeing local differential privacy. *Computers & Security*, 90: 101699.

Ye, M.; Fang, X.; Du, B.; Yuen, P. C.; and Tao, D. 2023. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3): 1–44.

Zhang, R.; Chen, Y.; Wu, C.; and Wang, F. 2023. Cluster-driven GNN-based Federated Recommendation with Biased Message Dropout. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, 594–599.

Zhang, R.; Chen, Y.; Wu, C.; Wang, F.; and Li, B. 2024. Multi-Level Personalized Federated Learning on Heterogeneous and Long-Tailed Data. *IEEE Transactions on Mobile Computing*, 23(12): 12396–12409.

Zhu, Z.; Hong, J.; and Zhou, J. 2021. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, 12878–12889. PMLR.