META KNOWLEDGE CONDENSATION FOR FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing federated learning paradigms usually extensively exchange distributed models at a central solver to achieve a more powerful model. However, this would incur severe communication burden between a server and multiple clients especially when data distributions are heterogeneous. As a result, current federated learning methods often require a large number of communication rounds in training. Unlike existing paradigms, we introduce an alternative perspective to significantly decrease the communication cost in federate learning. In this work, we first introduce a meta knowledge representation method that extracts meta knowledge from distributed clients. The extracted meta knowledge encodes essential information that can be used to improve the current model. As the training progresses, the contributions of training samples to a federated model also vary. Thus, we introduce a dynamic weight assignment mechanism that enables samples to contribute adaptively to the current model update. Then, informative meta knowledge from all active clients is sent to the server for model update. Training a model on the combined meta knowledge without exposing original data among different clients can significantly mitigate the heterogeneity issues. Moreover, to further ameliorate data heterogeneity, we also exchange meta knowledge among clients as conditional initialization for local meta knowledge extraction. Extensive experiments demonstrate the effectiveness and efficiency of our proposed method. Remarkably, our method outperforms the state-of-the-art by a large margin (from 74.07% to 92.95%) on MNIST with a restricted communication budget (*i.e.*, 10 rounds).

1 INTRODUCTION

Most deep learning-based models are trained in a data-centralized manner. However, in some cases, data might be distributed among different clients and cannot be shared. To address this issue, Federated Learning (FL) (Yang et al., 2019b;a; Kairouz et al., 2021) has been proposed to learn a powerful model without sharing private original data among clients. In general, most prior FL works often require frequent model communications to exchange models between local clients and a global server, resulting in heavy communications burden caused by the decentralized data (Wu & Wang, 2021; Chencheng et al., 2022). Therefore, it is highly desirable to obtain a powerful federated model with only a few communication rounds.

In this work, we propose a new meta knowledge-driven federated learning paradigm to achieve an effective yet communication-efficient model, thus significantly reducing communication costs. Unlike prior works, we formulate federated learning in a new perspective, where representative information will be distilled from original data and sent to the server for model training. On the client side, we extract representative information of original data and condense it into a tiny set of highly-compressed synthetic data, namely *meta knowledge*. Furthermore, we develop two mechanisms, *i.e.*, dynamic weight assignment and meta knowledge sharing, in the condensation process to mitigate the data heterogeneity issue widely existing in decentralized data. On the server side, we train our global model with meta knowledge uploaded from clients rather than simply averaging client models.

Specifically, we firstly distill the task-specific knowledge from private data on local clients and condense it as meta knowledge. The meta knowledge condensation process is modeled as a bi-level optimization procedure under the federated learning setting: the inner-loop minimizes the training loss on meta knowledge to update a model; and the outer-loop minimizes the training loss on original data to update meta knowledge based on the updated model. In the optimization process, we assign

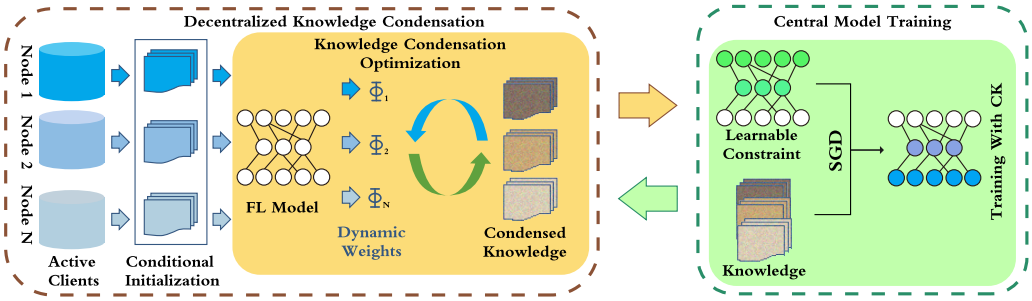

Figure 1: Illustration of our pipeline, in which only three active clients are shown. The local clients conduct meta knowledge condensation from local private data, and the server utilizes the uploaded meta knowledge for training a global model. The local meta knowledge condensation and central model training are conducted in an iterative manner. For meta knowledge extraction on clients, we design two mechanisms, *i.e.*, meta knowledge sharing, and dynamic weight assignment. For server-side central model training, we introduce a learnable constraint.

dynamic weights to each sample based on its training loss. By dynamically adjusting the weight of each sample in training, we empower each sample to contribute adaptively to the current model. Besides, to further mitigate heterogeneous data distributions among different clients, we design a meta knowledge sharing mechanism. The meta knowledge sharing mechanism provides *conditional* initialization to our meta knowledge optimization procedure, thus promoting knowledge exchange among clients.

Since meta knowledge from various clients has been uploaded to the server, our model can be trained with such knowledge that better describes the overall distribution. In contrast, previous methods that average local models on the server do not have such a merit. Additionally, we impose a learnable conditional generator on the central model training to improve the training stability. Specifically, the generator models the statistical distribution of the uploaded meta knowledge. We generate synthetic samples by the modeled distribution, providing historical information to model update. Note that, as meta knowledge contains the essential information of the original data as well as the corresponding class information, it can be used as normal training data for model training. Then, our global model is trained with the *uploaded* and *generated* meta knowledge on the server side. Consequently, we significantly reduce the impacts of data heterogeneity and decrease the communication rounds.

We conduct extensive experiments on various datasets, including MNIST (LeCun et al., 2010), SVHN (Netzer et al., 2011), CIFAR10 (Krizhevsky & Hinton, 2009), and CIFAR100 (Krizhevsky & Hinton, 2009). The experimental results demonstrate the effectiveness and efficiency of our proposed method. Specifically, our method outperforms competing works by a remarkable margin on all the datasets, especially when facing limited communication budgets (*i.e.*, 10 communication rounds). Overall, our key contributions are summarized as follows:

- We propose a new meta knowledge driven federated learning paradigm. We transmit highly condensed meta knowledge extracted from various local clients to the global server for model training, thus significantly mitigating data heterogeneity issues in federated learning.
- We design a federated meta knowledge extraction method that can effectively encode local data for global model training. Moreover, we propose a dynamic weight assignment scheme to promote informativeness of meta knowledge, and a knowledge sharing strategy to exchange meta knowledge among clients without leaking original data.
- We introduce a server-side conditional generator that models the statistical distribution of uploaded meta knowledge to stabilize the training process. Benefiting from the extracted meta knowledge and learned statistical distribution, our model requires fewer communication rounds compared to competing methods while achieving superior performance.

2 Methodology

2.1 Problem Definition

Federated Learning (FL) trains a model across a set of decentralized devices, *i.e.*, a set of clients and a server. Suppose there is data $\mathcal{D}$ distributed on C clients, each of which has a local private training dataset $\mathcal{D}^c = \{x_i^c, y_i^c\}, 1 \leq i \leq n^c$ and a weight value p^c. It is noted that $\mathcal{D} = \cup \mathcal{D}^c$ and $\sum_{c=1}^{C} p^c = 1$.

Without loss of generality, we discuss a multi-class classification problem under a federated learning setting. The learning target is formulated as follows:

$$\min_{\mathbf{w}}\{\mathcal{L}(\mathbf{w},\mathcal{D}) \triangleq \sum_{c=1}^{C} p^c \mathcal{L}^c(\mathbf{w},\mathcal{D}^c)\}, \tag{1}$$

where $\mathcal{L}^c(\cdot)$ is a local objective function that is optimized on the c-th client. The loss function $\mathcal{L}^c(\cdot)$ is formulated as follows:

$$\mathcal{L}^c(\mathbf{w},\mathcal{D}^c) \triangleq \frac{1}{n^c}\sum_{i=1}^{n^c} \ell(\mathbf{w}, x_i^c, y_i^c), \tag{2}$$

where $\ell(\cdot)$ is a user-defined loss function (*e.g.*, cross-entropy loss), and $\mathbf{w}$ denotes model parameters. As most FL algorithms need to exchange locally trained models multiple times, communication burden is often non-negligible. Though one-shot FL (Zhou et al., 2020) has been proposed to reduce communication cost, it suffers performance degradation.

2.2 Federated Learning via Meta Knowledge

We propose a new FL paradigm to solve the aforementioned limitations. Specially, as shown in Figure 1, our method conducts federated meta knowledge extraction (FMKE) on local clients and server-side central model training (CMT). To mitigate the data heterogeneity issue in FMKE, we design two mechanisms, *i.e.*, dynamic weight assignment, and meta knowledge sharing. To stabilize the central model training, we introduce a learnable constraint modeled by a conditional generator. The technical details are provided in the following subsections.

2.2.1 Federated Meta Knowledge Extraction on Clients

We design Federated Meta Knowledge Extraction (FMKE) to extract key information from decentralized data. In the decentralized scenario, the original data $\mathcal{D}$ is distributed on a set of clients. Each client has its private data $\mathcal{D}^c$ ($\mathcal{D} = \cup\mathcal{D}^c$, $1 \leq c \leq C$, where C is the client number), and a model downloaded from a server. For simplifying the following discussion, we denote the downloaded model on the client c as $\mathbf{w}^c$.

On each active local client c, FKME distills key information from corresponding local private data $\mathcal{D}^c$. The distilled information is condensed as meta knowledge $\hat{\mathcal{D}}^c$, which will be used to replace the original data $\mathcal{D}^c$ in global model training *on the server* [1]. The cardinality of $\hat{\mathcal{D}}$, *i.e.*, the size of extracted meta knowledge from all active clients, is much less than that of $\mathcal{D}$. The condensed meta knowledge $\hat{\mathcal{D}}$ is highly compressed and representative. To make the models trained on the original data and the meta knowledge achieve the comparable performance, we formulate the target function of FMKE as[2]:

$$\hat{\mathcal{D}}^{c,*} = \arg\min_{\hat{\mathcal{D}}^c} \mathcal{L}^c(\mathbf{w}^*,\mathcal{D}^c) = \arg\min_{\hat{\mathcal{D}}^c} \mathcal{L}^c(\mathbf{w}^c - \eta \bigtriangledown \mathcal{L}^c(\mathbf{w}^c,\hat{\mathcal{D}}^c),\mathcal{D}^c), \tag{3}$$

where η denotes a learning rate for updating $\mathbf{w}^c$ by stochastic gradient descent (SGD).

To solve the target function, we employ a bi-level optimization solution (Rajeswaran et al., 2019; Wang et al., 2018). To be specific, the meta knowledge extraction is formulated as a nested optimization process: in the inner-loop, based on an initialized meta knowledge, a model is updated to minimize the training loss over the meta knowledge; in the outer-loop, given the updated model, the meta knowledge is renewed by minimizing the training loss over the original data. The loopy formulation of our bi-level optimization process on client c is:

$$\hat{\mathcal{D}}^{c,*} = \arg\min_{\hat{\mathcal{D}}^c} \mathcal{L}^c(\mathbf{w}^*,\mathcal{D}^c) \quad s.t. \quad \mathbf{w}^* = \arg\min_{\mathbf{w}^c} \mathcal{L}^c(\mathbf{w}^c,\hat{\mathcal{D}}^c), \tag{4}$$

where $\mathcal{L}^c(\cdot,\mathcal{D}^c)$ denotes a loss function over the original data $\mathcal{D}^c$ on client c, $\mathcal{L}^c(\cdot,\hat{\mathcal{D}}^c)$ denotes a loss function over the meta knowledge $\hat{\mathcal{D}}^c$ on client c.

The inner-loop and outer-loop are implemented alternatingly and stochastic gradient descent is employed to update the model and meta knowledge. At first, on an active client c, we update the model parameter by the following formulation:

$$\mathbf{w}^c \leftarrow \mathbf{w}^c - \eta \bigtriangledown \mathcal{L}^c(\mathbf{w}^c,\hat{\mathcal{D}}^c), \tag{5}$$

[1] The dimension of synthesized $\hat{\mathcal{D}}_i^c$ is the same as that of original data $\mathcal{D}_i^c$.

[2] Assume one gradient descent step is conducted to achieve $\mathbf{w}^*$ here.

where η denotes a learning rate for the inner-loop. An updated model $\mathbf{w}^*$ is obtained in this inner-loop.

Secondly, given the updated model $\mathbf{w}^*$, we evaluate it on original data $\mathcal{D}^c$ and calculate the loss $\mathcal{L}^c(\mathbf{w}^*, \mathcal{D}^c)$. Then the condensed meta knowledge $\hat{\mathcal{D}}^c$ can be updated by:

$$\hat{\mathcal{D}}^c \leftarrow \hat{\mathcal{D}}^c - \alpha \bigtriangledown \mathcal{L}^c(\mathbf{w}^*, \mathcal{D}^c), \tag{6}$$

where α denotes a learning rate for the outer-loop, and the meta knowledge $\hat{\mathcal{D}}^c$ is initialized based on a uniform distribution (*i.e.*, $\hat{\mathcal{D}}^c_{ini} \sim U[-1, +1]$). An updated meta knowledge $\hat{\mathcal{D}}^c$ is obtained in this outer-loop. The inner-loop and outer-loop are conducted in an alternatingly manner.

Due to the data heterogeneity issue widely existing in FL, the extracted meta knowledge on each client might be biased. To mitigate the heterogeneity issue, we design two effective mechanisms in FMKE, namely Dynamic Weight Assignment, and Meta Knowledge Sharing.

Dynamic Weight Assignment: Concretely, we dynamically assign weights for each training sample in $\mathcal{D}$. As training progresses, the prediction confidence of each sample varies, making the samples contribute dynamically to the meta knowledge extraction. Thus, it is beneficial to treat those samples in an adaptive manner, *i.e.*, based on their prediction output, empowering the samples with different confidence to make different contributions in the current training. Specifically, we calculate a dynamic weight value ϕ^c_i for each sample $\mathcal{D}^c_i$, *e.g.*, (x^c_i, y^c_i), based on its prediction loss $\ell(\mathbf{w}^c, x^c_i, y^c_i)$. The formulation is defined as:

$$\phi^c_i = \frac{1}{1 + \exp\left(-\tau * \ell(\mathbf{w}^c, x^c_i, y^c_i)\right.}, \tag{7}$$

where τ is a hyper-parameter to smooth the result, $\mathcal{D}^c = \{\mathcal{D}^c_i\}, 1 \leq i \leq N^c$, and N^c denotes original data number on client c. We assign the dynamic weight to each sample to update meta knowledge:

$$\hat{\mathcal{D}}^c \leftarrow \hat{\mathcal{D}}^c - \alpha \bigtriangledown \mathcal{L}^c(\mathbf{w}^c, \Phi^c, \mathcal{D}^c), \tag{8}$$

where $\mathcal{L}^c(\mathbf{w}^c, \Phi^c, \mathcal{D}^c) \triangleq \frac{1}{N^c} \sum_{i=1}^{N^c} \phi^c_i \cdot \ell(\mathbf{w}^c, x^c_i, y^c_i)$.

Meta Knowledge Sharing: As an optimization process, an appropriate initialization for $\hat{\mathcal{D}}^c$ in Eq. 5 is crucial to obtain a good final result. In prior bi-level optimization works (Rajeswaran et al., 2019; Wang et al., 2018), the initialization value is randomly sampled from a constant distribution (*i.e.*, $\hat{\mathcal{D}}^c_{ini} \sim U[-1, +1]$), namely unconditional initialization. Due to the heterogeneity issue in FL, the extracted meta knowledge might become biased to the corresponding local data. To solve those limitations, we implement a *conditional* initialization in Eq. 8 with a meta knowledge sharing mechanism. Conditional initialization (Wang et al., 2020; Denevi et al., 2020) requires the initialization value is obtained from data characteristics rather than random generation. To achieve conditional initialization, we design a simple yet effective strategy in extracting meta knowledge $\hat{\mathcal{D}}^c$ for client c at the current round t. Concretely, in the process of initializing client c, we *randomly* select client c' and use its meta knowledge $\hat{\mathcal{D}}^{c'}_{t-1}$ extracted in the previous round $t-1$ as an initial value in Eq. 8. Correspondingly, the initialization for $\hat{\mathcal{D}}^c$ changes from an unconditional manner to a conditional manner: $\hat{\mathcal{D}}^c_{ini} \leftarrow \hat{\mathcal{D}}^{c'}_{t-1}, c' \sim randint[1, C]$. In this manner, the meta knowledge condensation for client c is mainly determined by the local data on client c as well as the knowledge extracted on another client c', mitigating the heterogeneity issue significantly.

2.2.2 Server-side Central Model Training

After conducting FMKE, we upload the condensed meta knowledge $\hat{\mathcal{D}}$ from clients to a server. On the server, the uploaded meta knowledge is used as normal training data to train a global model $\mathbf{W}_G$:

$$\mathcal{L}(\mathbf{W}_G, \hat{\mathcal{D}}) = \frac{1}{|\hat{\mathcal{D}}|} \sum_{\hat{x}_i, \hat{y}_i \in \hat{\mathcal{D}}} \ell(\mathbf{W}_G, \hat{x}_i, \hat{y}_i), \tag{9}$$

where $\ell(.)$ is a cross-entropy loss function as in Eq. 2, and $\hat{\mathcal{D}} = \cup \hat{\mathcal{D}}^c, 1 \leq c \leq C$.

To further ameliorate data biases among diverse clients, we introduce additional synthetic training samples into the central model training. Those introduced training samples are from the same distribution of upload meta knowledge $\hat{\mathcal{D}}$. Specifically, at first, we model the statistical distribution of

Algorithm 1: FedMK

Input: Original data $\mathcal{D}$; global parameters $\mathbf{W}_G$; generator parameter $\mathbf{w}^{\mathcal{G}}$; the communication budget.
Output: Optimal $\mathbf{W}_G^*$

```
while not over the communication budget do
    the server selects active clients C uniformly at random, broadcasts W_G to selected clients C.
    ▷Federated Meta Knowledge Extraction on selected clients C:
    for all user c ∈ C in parallel do
        w^c ← W_G;
        for t = 0, ..., #Round do
            if t == 0: D̂^c_ini ~ U[−1, +1];
            elif t == 1: conduct the conditional initialization, i.e., D̂^c_ini ← D̂^{c'}_{t−1}, c' ~ randint[1, C];
            calculate dynamic weights by Eq. 7;
            generate D̂^c_t by Eq. 4;
        end
        send the D̂^c_t to the server.
    end
    ▷Global Model Training on the server:
    update generator parameter w^G by Eq. 10;
    generate D̂^pseu by the updated generator G;
    update global parameter W_G by Eq. 11.
end
return W_G as W*_G;
```

uploaded meta knowledge $\hat{\mathcal{D}}$ via a conditional generator, and then we sample additional data points based on the learned distribution. Thus, sampled data would share the same distribution as $\hat{\mathcal{D}}$. After the introduction of sampled synthetic data, we not only stabilize our training procedure but also achieve better performance.

To facilitate the discussion, we divide the model $\mathbf{W}_G$ into a feature extractor $\mathcal{F}$ with parameter $\mathbf{W}_G^{\mathcal{F}}$ and a classifier $\mathcal{C}$ with parameter $\mathbf{W}_G^{\mathcal{C}}$, in which $\mathbf{W}_G = (\mathbf{W}_G^{\mathcal{F}}, \mathbf{W}_G^{\mathcal{C}})$. Accordingly, we denote a latent representation as $z = \mathcal{F}(\mathbf{W}_G^{\mathcal{F}}, x)$ and a final prediction as $y^{'} = \mathcal{C}(\mathbf{W}_G^{\mathcal{C}}, z)$. The conditional generator $\mathcal{G}$ maps a label y into a latent representation $z \sim \mathcal{G}(y, \mathbf{w}^{\mathcal{G}})$, and $\mathcal{G}$ is optimized by the objective:

$$\mathcal{G}^* = \arg\max_{\mathcal{G}:y\to z} \mathbb{E}_{y\sim p(y)} \mathbb{E}_{z\sim\mathcal{G}(y,\mathbf{w}^{\mathcal{G}}))} \log p(y|z, \mathbf{W}_G^{\mathcal{C}}), \tag{10}$$

where $\mathbf{w}^{\mathcal{G}}$ denotes the parameter of $\mathcal{G}$.

The trained generator $\mathcal{G}$ models the distribution of uploaded meta knowledge $\hat{\mathcal{D}}$. By sampling data from the distribution, we obtain a set of "pseudo" meta knowledge $\hat{\mathcal{D}}^{pseu}$ with corresponding labels. The generated "pseudo" meta knowledge $\hat{\mathcal{D}}^{pseu}$ as well as uploaded $\hat{\mathcal{D}}$ are utilized to train the global model by minimizing the following function:

$$\mathcal{L}_{overall}(\mathbf{W}_G, \{\hat{\mathcal{D}}, \hat{\mathcal{D}}^{pseu}\}) = \mathcal{L}(\mathbf{W}_G, \hat{\mathcal{D}}) + \beta\mathcal{L}(\mathbf{W}_G, \hat{\mathcal{D}}^{pseu}), \tag{11}$$

where β is a parameter and determined by the cardinality fraction $\frac{|\hat{\mathcal{D}}^{pseu}|}{|\hat{\mathcal{D}}|}$.

Iterative Symbiosis Paradigm: After central model training, we broadcast the obtained global $\mathbf{W}_G$ and meta knowledge $\hat{\mathcal{D}}$ to clients. On each active client, the broadcasted model $\mathbf{W}_G$ as well as meta knowledge $\hat{\mathcal{D}}$ are used for a new round of FMKE. FMKE and CMT collaborate with each other in an iterative symbiosis paradigm, benefiting each other increasingly as the learning continues. The algorithm of FedMK training is illustrated in Alg. 1. When the training finishes, the trained global model is directly used for inference.

Computational Complexity: FedMK includes two parts: federated meta knowledge extraction on clients and global model training on the server. On clients, our method adopts a bi-level optimization to extract the meta-knowledge. The bi-level optimization has a running-time complexity of $O(N \times n)$ (Fallah et al., 2020a), in which n denotes the meta knowledge size, N denotes the number of samples on the client. It should be noted that the selected nodes can conduct FMKE in parallel. On the server, the global model training in our method has a running-time complexity of $O(n)$. In total, the overall running-time complexity of our method is $O(N \times n)$.

Table 1: The classification accuracy with limited communication rounds (*i.e.*, 10).

Setting	FedAvg	FedProx	FedDistill	FedEnsem	FedGen	FedMK
MNIST						
α=0.50	74.61±2.79%	73.56±3.84%	75.04±2.32%	75.39±1.44%	74.07±1.20%	**92.95±3.81%**
α=0.75	73.49±3.62%	73.13±2.68%	76.21±1.59%	74.28±1.65%	74.57±3.98%	**92.86±0.28%**
α=1.0	74.10±1.84%	73.35±0.10%	76.19±1.19%	74.45±2.25%	73.97±1.69%	**93.63±0.40%**
SVHN						
α=0.50	29.55±2.02%	28.52±2.44%	26.92±2.02%	29.12±2.99%	28.94±2.62%	**74.11±1.00%**
α=0.75	31.71±0.90%	25.78±2.76%	25.77±3.27%	29.53±1.91%	30.40±2.86%	**74.90±0.44%**
α=1.0	30.87±1.41%	29.59±1.26%	25.84±1.21%	32.67±2.07%	33.62±3.28%	**74.84±0.32%**
CIFAR10						
α=0.50	26.63±0.95%	26.21±0.98%	24.38±2.12%	27.56±0.70%	25.42±2.56%	**47.33±1.08%**
α=0.75	25.42±1.17%	24.85±1.41%	24.18±1.25%	26.42±0.59%	26.25±0.33%	**49.04±1.15%**
α=1.0	26.80±0.89%	26.66±0.37%	25.83±0.79%	26.74±0.87%	25.36±1.32%	**50.32±0.69%**
CIFAR100						
α=0.50	11.66±1.22%	12.09±0.72%	10.76±0.77%	13.20±0.44%	10.34±0.65%	**26.74±0.60%**
α=0.75	12.11±0.45%	11.65±0.36%	11.55±0.79%	13.15±0.20%	10.21±1.02%	**27.43±0.54%**
α=1.0	12.31±0.60%	11.34±0.48%	11.50±0.66%	13.31±0.54%	11.19±0.75%	**28.20±0.10%**

3 EXPERIMENTS

3.1 DATASETS

We evaluate our algorithm and compare to the key related works on four benchmarks: MNIST (LeCun et al., 2010), SVHN (Netzer et al., 2011), CIFAR10 (Krizhevsky & Hinton, 2009), and CIFAR100 (Krizhevsky & Hinton, 2009). MNIST is a database of handwritten digits (0-9). In MNIST, there are $50,000$ images in the training set and $10,000$ images in the test set, and their size is $1 \times 28 \times 28$ pixels ($c \times w \times h$). There are 10 classes in MNIST dataset. SVHN is a real-world image dataset, in which there are $600,000$ color images collected from house numbers in Google Street View images. In SVHN, each image is of size $3 \times 32 \times 32$. The class number of SVHN is as the same as MNIST. CIFAR10 dataset consists of $60,000$ color images, each of which has a size of $3 \times 32 \times 32$. There are $50,000$ images in the training set and $10,000$ images in the testing set. There are 10 classes in CIFAR10. CIFAR100 dataset consists of $60,000$ color images, each of which has a size of $3 \times 32 \times 32$. There are $50,000$ images in the training set and $10,000$ images in the testing set. For each image in CIFAR100, there are two kinds of labels, *i.e.*, fine label and coarse label. We choose coarse labels and therefore we have 20 classes in the experiment on CIFAR100.

3.2 IMPLEMENTATION DETAILS

We set the user number to 20, and the active-user number to 10. We use 50% of the training set and distribute it on all clients. All testing data is utilized for evaluation. We use LeNet (LeCun et al., 1989) as the backbone for all methods: FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020), FedDistill (Seo et al., 2020), FedEnsemble (Zhu et al., 2021), FedGen (Zhu et al., 2021), and FedMK. Dirichlet distribution $Dir(\alpha)$ is used to model data distributions. Specifically, we test three different α values: 0.5, 0.75, and 1.0, respectively. We set the communication round number to 10. Learning with a small communication round number (*e.g.*, 10) denotes learning under a limited communication budget. For the methods conducting local model training on clients, *i.e.*, FedAvg, FedProx, FedDistill, FedEnsemble, and FedGen, we set the local updating number to 20, and the batch size number to 32. In our method, we set meta knowledge size for each datasets based on their different characteristics (*e.g.*, the class number, sample size, etc): 20 per class for MNIST; 100 per class for SVHN; 100 per class for CIFAR10; 40 per class for CIFAR100. We run three trials and report the mean accuracy performance (MAP).

3.3 COMPARATIVE STUDIES

Compared with Prior Works: We run all methods under limited communication budgets (10 rounds) on four datasets and report the results in Table 1. As shown in Table 1, when communication budgets are limited, all prior works fail to construct a model with a good performance; on the contrary, our method can learn a model outperforming competing works by a remarkable margin. For example, on MNIST, our method achieves 93.63% ($\alpha = 1$) MAP, outperforming the competing works significantly. On SVHN, our method achieves 75.85% ($\alpha = 1$) MAP in 10 rounds, while all competing works have not converged yet. We believe that the superior performance of our method comes from those aspects: 1) unlike competing methods conducting local model training with local private data, our FedMK trains a global model based on meta-knowledge from all active clients. Training on knowledge from all active clients makes our method less biased; 2) the designed two mechanisms, *i.e.*, dynamic weight assignment and meta knowledge sharing, make the meta knowledge extraction more stable, leading to a better performance than competing methods [3].

Convergence rate comparisons: We compare the convergence rates of all methods and show the performance curves with respect to communication rounds in Figure 2. As expected, our method

[3] A more detailed comparison with FedGen can be found in the supplementary document.

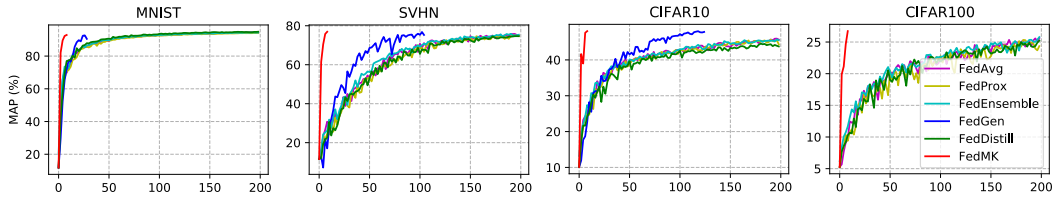

Figure 2: Convergence rate comparisons on MNIST, SVHN, CIFAR10, and CIFAR100. $\alpha = 1.0$. The x-axis represents communication round numbers.

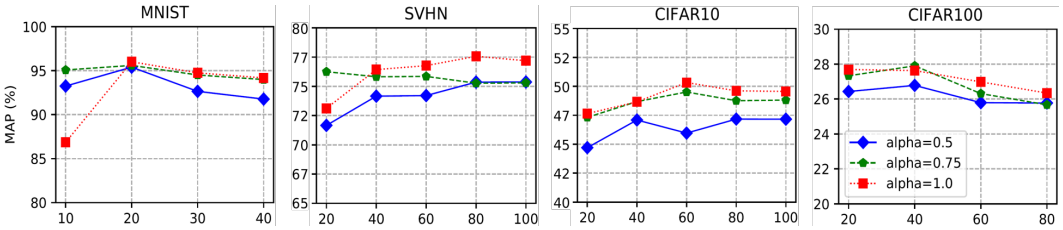

Figure 3: Impact of meta knowledge size on final performance. #round=10. The x-axis represents meta knowledge sizes.

Table 2: Impact of each designed mechanism.

	w/o Iter	w/o Sharing	w/o pseudo knowledge	w/o dynamic weights	Ours
CIFAR10	28.15%	45.79%	46.71%	47.16%	**47.33%**
CIFAR100	18.82%	25.56%	26.10%	26.43%	**26.74%**

achieves a high convergence speed compared to all competing methods. On all datasets, our method achieves satisfactory performance in much fewer communication rounds. The results shown in both Table 1 and Figure 2 demonstrate the effectiveness and efficiency of our proposed method.

Communication cost comparisons: We analyze the communication cost of our method on MNIST. In each round of FedMK, we need to upload and download the meta knowledge, and download a trained model. In the experiment, the meta image size is 20 per class, each of which is 4 bytes $\times$ 28 $\times$ 28 (sizeof(f32)*w*h). Thus, in each round, the uploading cost of FedMK is 4 bytes $\times$ 28 $\times$ 28 $\times$ 20 $\times$10=627.2K (meta knowledge), and the downloading cost is 627.2K + 105K $\times$10= 1677.2K (meta knowledge and a model). In total, the communication cost for FedMK is (627.2K+$1,677.2$K) $\times$10 (# rounds)=16M. As shown in Figure 2, to achieve comparable performance on MNIST, FedAvg has to run around 200 rounds. In this case, the communication cost for FedAvg is $105K \times 10 \times 2 \times 200$ (model size $\times$ #active node $\times$ 2 $\times$ #communication)=420M. This indicates that to obtain models with good performance, FedAvg and its variants require much higher communication costs.

Impact of the meta knowledge size: We explore the impact of different meta knowledge sizes on four datasets. We draw the performance comparisons in Figure 3. It can be seen that the final performance changes as the meta knowledge size varies. For example, on CIFAR10, the MAP score improves as the meta knowledge size increases. Another finding is that the optimal meta knowledge sizes between different datasets might be different. For example, we achieve the highest MAP score when setting the meta knowledge size to 20 on MNIST; while on the other three datasets, we obtain the best MAP scores using different meta knowledge sizes. More discussions about why utilizing meta knowledge decreases the required communication round, and why increasing meta knowledge size will not necessarily improve final performance, can be found in the supplementary material.

Impact of active client numbers: We explore the different active client numbers. We set the number to 5, 7, and 10, respectively. The performance curves are shown in Figure 4. As shown in the figure, our method outperforms all competing works significantly with all different active client numbers.

Impact of communication rounds: We conduct an experiment on CIFAR10 to analyze the impact of communication rounds. Specifically, we set the communication round number to 20, 30, and 40 for comparisons. The comparison results are shown in Figure 5. We demonstrate that our method consistently outperforms competing works by a remarkable margin.

Impact of node numbers: We conduct experiments on SVHN to analyze the impact of node numbers. Specifically, we set the node number to 200 for comparisons. Intuitively, when the number of user becomes larger, the number of data on each client will be sparse. And thus, the learning will become

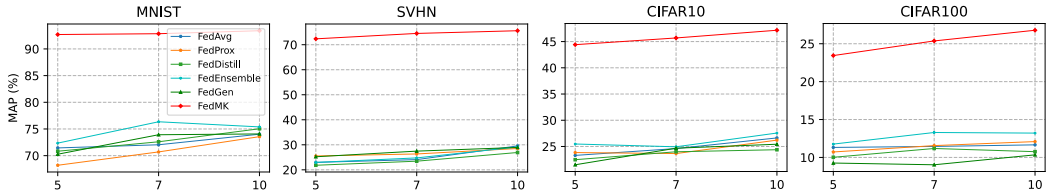

Figure 4: The performance curve w.r.t. the active client number. α = 0.5, #node = 20, #round = 10. The x-axis represents the active client numbers

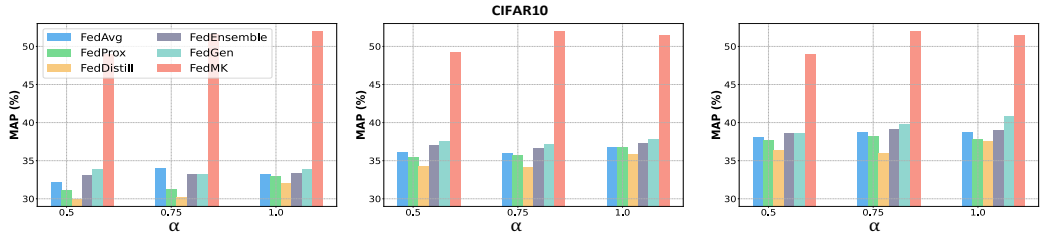

Figure 5: Performance comparisons w.r.t. different round numbers. Left column: 20 rounds; middle column: 30 rounds; right column: 40 rounds. The x-axis represents different α values.

Table 3: Comparisons on MNIST under different weight assignment mechanisms.

Method	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 1.0$
Balance Class Weight	92.17%	91.84%	91.69%
Our method	**92.95%**	**92.86%**	**93.63%**

difficult, leading to a performance drop. In our experiment, we find that when we increase the node number to 200, our method (70.51%) still outperforms the FedGen (66.31%) by a remarkable margin.

Impact of designed mechanisms: We conduct a study on CIFAR10 and CIFAR100 to explore the impact of each designed mechanism. We report the performance in Table 2, where "w/o Iter" means that we conduct local meta knowledge condensation and global model update only once [4], "w/o Sharing" means that we do not adopt meta knowledge sharing between clients, "w/o pseudo meta knowledge" means that there is no learnable constraint on the server, "w/o dynamic weights" means that we update $\hat{\mathcal{D}}$ by treating all samples in $\mathcal{D}$ equally.

As shown in Table 2, there is a drastic performance degradation when there is no iterative collaboration between meta knowledge extraction and central model training ("w/o Iter"). Making the meta knowledge condensation and central model training running in an iterative manner improves the performance significantly, *e.g.*, on CIFAR10, MAP score increases from 28.15% to 45.79%. Moreover, the two designed mechanisms, *i.e.*, the meta knowledge sharing between clients, dynamic weights assignment, significantly boost the MAP score.

Impact of the weight assignment mechanism: To explore the impact of different weight assignment mechanisms, we tested another weight assignment strategy targeting ameliorating the class imbalance problem. This strategy has been utilized in prior works (Tian et al., 2022a; Shao et al., 2021). To simplify the discussion, we call this strategy as Class Balance Weight. Concretely, in the Class Balance Weight strategy, the weight of each sample during the training is determined by the occurrence rate of the corresponding class. We replace the Dynamic Weight Assignment with the Balance Class Weight, and conduct an experiment on MNIST. We report the results in Table 3. As shown in the Table, we demonstrate that our designed Dynamic Weight Assignment outperforms the Class Balance Weight method significantly.

Evaluation on the pathological non-iid setting: We conduct an experiment on MNIST under the pathological non-iid setting (Huang et al., 2021). In the experiment, we set the node number to 20, the active node number to 10, the number of classes on each client to 5, α to 1.0. We compare all methods under a limited budget communication (*i.e.*, 10 rounds). As seen in Table 4, our method achieves better performance compared to the competing methods.

[4] "w/o Iter" can be regarded as one-shot FL like Zhou et al. (2020)

Table 4: Result comparisons on MNIST under the pathological non-iid setting.

	FedAvg	FedProx	FedEnsem	FedDistill	FedGen	Ours
MAP (%)	75.11%	72.99%	76.43%	71.19%	86.44%	**88.85**%

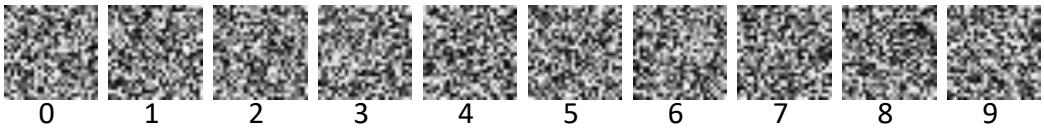

Figure 6: The visualization of condensed meta knowledge for MNIST. Each image corresponds to the condensed meta knowledge encoded from images of a specific class in MNIST.

Visualization of condensed meta knowledge We visualize the condensed meta knowledge coded from MNIST dataset. As shown in Figure 6, it is difficult to infer the semantic information from the meta knowledge. Furthermore, we conduct an experiment based on Deep Leakage (Zhu & Han, 2020) to explore the possibility of restoring original data from meta knowledge. We report the results in the supplementary document, in which we demonstrate that it is hard to construct correspondence between entries in restored data and original data.

4 Related Work

Federated Learning. As a privacy-preserving solution, federated learning (Li et al., 2019) provides a new training manner to learn models over a collection of distributed devices. In federated learning, the data is located on distributed nodes and not shared. Learning a model without exchanging local data between nodes minimizes the risk of data leakage but increases the difficulty of training. As a representative method, FedAvg (McMahan et al., 2017) is proposed to obtain a global model by aggregating local models trained on active clients. To improve the local training, works such as (Li et al., 2020; Zhu et al., 2021; Li et al., 2021; Tian et al., 2022b) design various strategies in their solutions. All those methods require a large number of communication round to construct a model with satisfactory performance, resulting in heavy communication burden.

Recently, several methods (Fallah et al., 2020b; Dinh et al., 2020; Seo et al., 2020; Li et al., 2021; Zhu et al., 2021; Fan & Huang, 2021; Gou et al., 2021; Dai et al., 2022; Zhang et al., 2022) have been proposed to handle the data heterogeneity issue, which can be categorized into personalization-based methods and knowledge distillation-based methods. The personalized FL methods (Fallah et al., 2020b; Dinh et al., 2020; Dai et al., 2022) aim to learn client-specific models while our goal is to obtain a global model across different clients. Knowledge distillation based FL methods (Seo et al., 2020; Zhu et al., 2021; Zhang et al., 2022) propose to share knowledge between nodes, and thus the local training is regularized by local private data and knowledge on other clients.

Compact Data Representation. Generally, prior works compressing a large scale data into a small set can be categorized into two main branches: data selection and data compression. Data selection methods (Rebuffi et al., 2017; Castro et al., 2018; Aljundi et al., 2019; Sener & Savarese, 2018) select the most representative samples from the original data based on predefined criteria. The selected samples are utilized to replace the original data for model training. However, how to select an appropriate criterion based on the given data and task is not a trivial issue. To overcome the aforementioned limitations, synthesizing new samples rather than selecting existing samples becomes a more preferable solution. (Wang et al., 2018; Zhao et al., 2021; Zhao & Bilen, 2021; Cazenavette et al., 2022) design different solutions for generating synthetic data based on given datasets. In those methods, the generated data can replace the original data in the model construction process. However, those prior synthetic data generation works require data to be localized in a centralized manner. It is not trivial to directly apply those methods under a federated learning setting.

5 Conclusion

In this paper, we present a new federated learning paradigm driven by meta knowledge, dubbed FedMK, to obtain an effective and fast-converging model. With the help of the proposed paradigm, FedMK can train a powerful model even under a limited communication budget (*e.g.*, 10 communication rounds), decreasing the communication cost significantly. Moreover, our designed mechanisms, *i.e.*, meta knowledge sharing, dynamic weight assignment, and a learned constraint, collectively facilitate the central model training, benefiting FedMK outperforming all competing methods.

REFERENCES

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, 2019.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *CVPR*, 2022.

Xu Chencheng, Hong Zhiwei, Huang Minlie, and Jiang Tao. Acceleration of Federated Learning with Alleviated Forgetting in Local Training. In *ICLR*, 2022.

Rong Dai, Li Shen, Fengxiang He, Xinmei Tian, and Dacheng Tao. DisPFL: Towards communication-efficient personalized federated learning via decentralized sparse training. In *ICML*, 2022.

Giulia Denevi, Massimiliano Pontil, and Carlo Ciliberto. The Advantage of Conditional Meta-Learning for Biased Regularization and Fine Tuning. In *NeurIPS*, 2020.

Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *NeurIPS*, 2020.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *AISTATS*, 2020a.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *NeurIPS*, 2020b.

Chenyou Fan and Jianwei Huang. Federated few-shot learning with adversarial learning. In *WiOpt*, 2021.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021.

Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *AAAI*, 2021.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 2021.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

Yann LeCun, Corinna Cortes, and Chris Burges. Mnist handwritten digit database, 2010.

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *arXiv*, 2019.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *CVPR*, 2021.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS-Workshop*, 2011.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018.

Hyowoon Seo, Jihong Park, Seungeun Oh, Mehdi Bennis, and Seong-Lyun Kim. Federated knowledge distillation. *arXiv preprint arXiv:2011.02367*, 2020.

Zhiwen Shao, Jianfei Cai, Tat-Jen Cham, Xuequan Lu, and Lizhuang Ma. Unconstrained facial action unit detection via latent feature domain. *IEEE Transactions on Affective Computing*, 2021.

Junjiao Tian, Niluthpol Chowdhury Mithun, Zachary Seymour, Han-Pang Chiu, and Zsolt Kira. Striking the right balance: Recall loss for semantic segmentation. In *ICRA*, 2022a.

Junjiao Tian, James Seale Smith, and Zsolt Kira. FedFOR: Stateless Heterogeneous Federated Learning with First-Order Regularization. *arXiv*, 2022b.

Ruohan Wang, Yiannis Demiris, and Carlo Ciliberto. Structured prediction for conditional meta-learning. In *NeurIPS*, 2020.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv*, 2018.

Hongda Wu and Ping Wang. Node selection toward faster convergence for federated learning on non-iid data. *arXiv preprint arXiv:2105.07066*, 2021.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 2019a.

Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2019b.

Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *CVPR*, 2022.

Bo Zhao and Hakan Bilen. Dataset Condensation with Differentiable Siamese Augmentation. In *ICML*, 2021.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *ICLR*, 2021.

Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

Ligeng Zhu and Song Han. Deep leakage from gradients. In *NeurIPS*, 2020.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *ICML*, 2021.