MEMORY-ADAPTIVE DEPTH-WISE HETEROGENOUS FEDERATED LEARNING

Anonymous Authors[1]

ABSTRACT

Federated learning is a promising paradigm that allows multiple clients to collaboratively train a model without sharing the local data. However, the presence of heterogeneous devices in federated learning, such as mobile phones and IoT devices with varying memory capabilities, would limit the scale and hence the performance of the model could be trained. The mainstream approaches to address memory limitations focus on width-slimming techniques, where different clients train subnetworks with reduced widths locally and then the server aggregates the subnetworks. The global model produced from these methods suffers from performance degradation due to the negative impact of the actions taken to handle the varying subnetwork widths in the aggregation phase. In this paper, we introduce a memory-adaptive depth-wise learning solution in FL called FEDEPTH, which adaptively decomposes the full model into blocks according to the memory budgets of each client and trains blocks sequentially to obtain a full inference model. Our method outperforms state-of-the-art approaches, achieving 5% and more than 10% improvements in top-1 accuracy on CIFAR-10 and CIFAR-100, respectively. We also demonstrate the effectiveness of depth-wise fine-tuning on ViT. Our findings highlight the importance of memory-aware techniques for federated learning with heterogeneous devices and the success of depth-wise training strategy in improving the global model's performance.

1 INTRODUCTION

Federated Learning (FL) is a popular distributed learning paradigm that can address decentralized data and privacy-preserving challenges by collaboratively training a model among multiple local clients without centralizing their private data (McMahan et al., 2017; Kairouz et al., 2021). FL has gained widespread interest and has been applied in numerous applications, such as healthcare (Du Terrail et al., 2022), anomaly detection (Zhang et al., 2021a), recommendation system (Lin et al., 2020b), and knowledge graph completion (Zhang et al., 2022). However, a defining trait of FL is the presence of heterogeneity — 1) data heterogeneity, where each client may hold data according to a distinct distribution, leading to a sharp drop in accuracy of FL (Zhao et al., 2018), and 2) heterogeneous clients, which are equipped with a wide range of computation and communication capabilities — challenges the underlying assumption of conventional FL setting that local models have to share the same architecture as the global model (Diao et al., 2021). In the last five years, data heterogeneity has been largely explored in many studies (Karimireddy et al., 2019; Lin et al., 2020a; Li et al., 2020a; Seo et al., 2020; Acar et al., 2021; Zhu et al., 2021; Li et al., 2021; Tan et al., 2022). However, only a few works aim to address the problem of heterogeneous clients, particularly memory heterogeneity in FL (Diao et al., 2021; Hong et al., 2022).

One solution to heterogeneous clients is to use the smallest model that all clients can train, but this can severely impact FL performance as larger models tend to perform better (Frankle & Carbin, 2019; Neyshabur et al., 2019; Bubeck & Sellke, 2021). Another approach is to prune channels of the global model for each client based on their memory budgets and average the resulting local models to produce a full-size global model (Diao et al., 2021; Hong et al., 2022; Horvath et al., 2021). However, such approaches suffer from the issue of under-expression of small-size models, since the reduction in the width of local models can significantly degrade their performance due to fewer parameters (Frankle & Carbin, 2019). The negative impact of aggregating small-size models in FL is also verified by our case studies in Section 3.2.

Considering the exceptional performance of the full-size model, we aim to provide an algorithmic solution to enable each client to train the same full-size model and acquire adequate global information in FL. Specifically, we propose memory-adaptive depth-wise learning, where each client sequentially trains blocks of a neural network based on the local memory budget until the full-size model is updated. To ensure the classifier layer's supervised signal can be utilized for training each block, we propose two learning strategies: 1) incorporating a skip connection between training blocks and the classifier, and 2) introducing auxiliary classifiers. Our method is suitable for memory-constrained settings as it does not require storing the full intermediate activation and computing full intermediate gradients. Additionally, it can

be seamlessly integrated with most FL algorithms, e.g., FedAvg (McMahan et al., 2017) and FedProx (Li et al., 2020a). Apart from providing adaptive strategies for low-memory local training, we investigate the potential of mutual knowledge distillation (Hinton et al.; Zhang et al., 2018) to address on-the-fly device upgrades or participation of new devices with increased memory capacity. Lastly, we consider devices with extremely limited memory budgets such that some blocks resulting from the finest network decomposition cannot be trained. We propose a partial training strategy, where some blocks that are close to the input sides are never trained throughout. The main contributions of our work are summarized as follows.

1. Through comprehensive analysis of memory consumption, we develop two memory-efficient training paradigms that empower each client to train a full-size model for improving the global model's performance.
2. Our framework is model- and optimizer-agnostic. The flexibility allows for deployment in real-world cross-device applications, accommodating clients with varying memory budgets on the fly.
3. Our proposed approach is not sensitive to client participation resulting from unstable communication because we learn a unified model instead of different local models as in prior works.
4. Experimental results demonstrate that the performance of the proposed methods is better than other FL baselines regarding top-1 accuracy in scenarios with heterogeneous memory constraints and diverse non-IID data distributions. We also show the negative impact of sub-networks using width-slimming techniques.

2 Related Work

Federated Learning. FL emerges as an important paradigm for learning jointly among clients' decentralized data (Konečnỳ et al., 2016; McMahan et al., 2017; Li et al., 2020a; Kairouz et al., 2021; Wang et al., 2021a). One major motivation for FL is to protect users' data privacy, where users' raw data are never disclosed to the server and any other participating users (Abadi et al., 2016; Bonawitz et al., 2017; Sun et al., 2019). Partly opened by the *federated averaging* (FedAvg) (McMahan et al., 2017), a line of work tackles FL as a distributed optimization problem where the global objective is defined by a weighted combination of clients' local objectives (Mohri et al., 2019; Li et al., 2020a; Reddi et al., 2020; Wang et al., 2020b). The federated learning paradigm of FedAvg has been extended and modified to support different global model aggregation methods and different local optimization objectives and optimizers (Yurochkin et al., 2019; Reddi et al., 2020; Wang et al., 2021a;b; 2020a). Theoretical analysis has been conducted, which demonstrated that federated optimization enjoys convergence guarantees under certain assumptions (Li et al., 2020b; Wang et al., 2021a).

System-efficiency of Federated Learning. In this realm, we concentrate on *cross-device FL*, where participating users are typically edge devices, e.g., mobile phones, embedded systems, and etc. (Kairouz et al., 2021). Such edge devices are typically resource-constrained, e.g., the computing, communication, and memory capacities are limited. Several research efforts have been conducted to enhance the computation and communication efficiency of cross-device FL via model updates sparsification, quantization, and low-rank factorization (Konečnỳ et al., 2016; Wang et al., 2018; Yao et al., 2021; Hyeon-Woo et al., 2021). Training deep neural networks requires high memory consumption (Sohoni et al., 2019). To reduce it, prior works focus on deploying methods, like gradient accumulation, activation materialization, and partial model training (Huang et al., 2019; Sohoni et al., 2019; Chen et al., 2016). However, these approaches do not fundamentally address the memory limitation and typically suffer from overfitting to the local data and not fully capturing the global patterns in the data.

Device Heterogeneity in Federated Learning. Especially for cross-device FL, it is a natural setting that client devices are with heterogeneous computation power, communication bandwidth, and/or memory capacity. A few research efforts have been paid to designing memory heterogeneity-aware FL algorithms. HeteroFL (Diao et al., 2021) and FjORD (Horvath et al., 2021) allows model architecture heterogeneity among participating clients via varying model widths. The method bears similarity to previously proposed *slimmable neural network* (Yu et al., 2018; Yu & Huang, 2019) where sub-networks with various widths and shared weights are jointly trained with self-distillation (Zhang et al., 2021b). SplitMix (Hong et al., 2022) tackles the same device heterogeneity problem via learning a set of base sub-networks of different sizes among clients based on their hardware capacities, which are later aggregated on-demand according to inference requirements. Some recent works adopt the concept of layer-wise training. Specifically, InclusiveFL (Liu et al., 2022) and DepthFL (Kim et al., 2023) assign models of different sizes to clients regarding the on-device capability by adjusting the depth of a network. In summary, prior research efforts mainly focus on partitioning the global model weights among participating users given their hardware resource constraints, which leads each local user only to see part of the global model. In this work, however, we seek a holistic approach to handling device heterogeneity in FL without the need for partitioning model weights. Instead, our approach allows each device to train a full-size model in a sequential manner.

3 EMPIRICAL STUDY

3.1 Preliminaries

This section briefly reviews prior works that aim to address heterogeneous clients in FL, including HeteroFL (Diao et al., 2021) and SplitMix (Hong et al., 2022). We then conduct an extensive analysis of memory consumption of training a neural network, which has not been explored thoroughly in the FL community.

Prior works on FL for heterogeneous clients. Existing works such as HeteroFL (Diao et al., 2021) and SplitMix (Hong et al., 2022) address memory heterogeneity by pruning a single global model in terms of channels, creating heterogeneous local models. HeteroFL is the first work in FL that tackles memory heterogeneity via the width-scaling approach but still produces a full-size global model. However, HeteroFL suffers from two major limitations: 1) partial model parameters are under-trained because only partial clients and data are accessible for training the full-size model; 2) small models' information tends to be ignored because of their small-scale parameters. SplitMix was then proposed to address these two issues, which first splits a wide neural network into several base sub-networks for increasing accessible training data, then boosts accuracy by mixing base sub-networks.

Memory consumption analysis. Training a neural network with backpropagation consists of feedforward and backward passes (Rumelhart et al., 1986). A feed-forward pass over each block of a neural network generates an activation or output. These intermediate activations are stored in the memory for the backward pass to update the neural network. Although several works of literature (Sohoni et al., 2019; Gomez et al., 2017; Raihan & Aamodt, 2020; Chen et al., 2021) demonstrate that activations usually consume most of the memory in standard training of a neural network as shown in Figure 1, HeteroFL and SplitMix merely consider the number of model parameters as the memory budget in their experiments. Specifically, they divide clients into groups that are capable of different widths, e.g., a $\frac{1}{8}$-width neural network, which costs approximately $\frac{1}{8}$ activations but only around $\frac{1}{8^2}$ model parameters compared to the full-size neural network.

3.2 Behaviors of Sub-networks in Prior Works

In this section, we analyze the behaviors and influences of sub-networks in HeteroFL and SplitMix with respect to the performance of the global model.

Experimental setting. We follow the configuration on the CIFAR-10 dataset in SplitMix (Hong et al., 2022), where 10 out of 100 clients participate in training in any given communication round, and each client has three classes of the data. In our case studies, we divide clients into four groups with $\{\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1\}$-width sub-networks in HeteroFL, and into two groups with $\{r, 1\}$ in SplitMix, where $r = \{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}\}$. Our observations are summarized below.

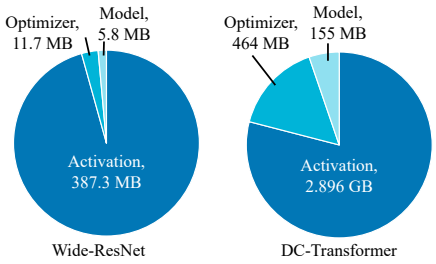

Figure 1. Training memory consumption for **left**: WideResNet on CIFAR-10 and **right**: DC-Transformer on IWSLT'14 German to English. Data source: (Sohoni et al., 2019).

1. Small sub-networks make negative contributions in HeteroFL. Figure 2 presents typical examples of HeteroFL (Diao et al., 2021) under non-IID settings. The orange line represents the default setting of HeteroFL, where all sub-networks of different widths are aggregated. The other lines indicate specific size of sub-networks that are not aggregated. For example, the green line indicates that the smallest ($\frac{1}{8}$-width) sub-networks do not participate in aggregation. We observe that the global model obtained via aggregating small sub-networks consistently has worse performance than the global model obtained via only aggregating the full-size neural networks, indicating that small size sub-networks make negative contributions.

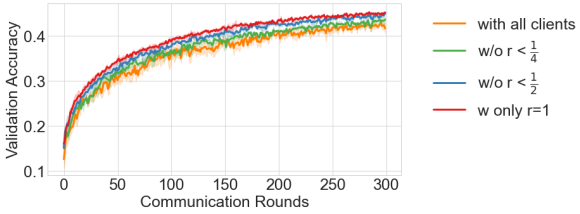

Figure 2. Performance of the global model in HeteroFL with different sub-networks aggregation strategy.

2. Small sub-networks limit global performance in SplitMix. Figure 3 depicts the prediction performance of the global model in SplitMix (Hong et al., 2022) by mixing base neural networks with different-width. It clearly illustrated that slimmer base neural networks produce a less powerful global model. Intuitive reasoning is that combining very weak learners leads to an ensemble model with worse generalization.

3. The full-size net makes a difference. Inadequate presence of the full-size models incurs degradation of validation accuracy as shown in Figure 2 and 3. In real-world FL sys-

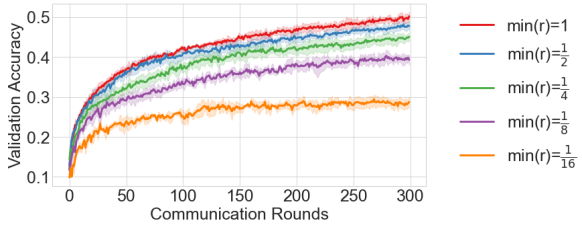

Figure 3. Performance of the global model in SplitMix with the varying-width base model.

tems, communication can be unstable, and clients with the largest memory budgets may not be available in each round of communication (Bonawitz et al., 2017). This constraint limits the practicality of both HeteroFL and SplitMix.

4 METHODOLOGY

Inspired by the observation in the previous section, we introduce a memory-efficient framework FEDEPTH in section 4.1 to train full-size neural networks with memory budget constraints in the FL setting. FEDEPTH aims to empirically solve the optimization problem $\min_{\mathcal{W}} F(\mathcal{W}) := \sum_{k=1}^{K} p_k F_k(\mathcal{W})$. Here, F_k represents the loss function on the kth clients. $p_k > 0$ for all k and $\sum_{k=1}^{K} p_k = 1$. FEDEPTH features memory-adaptive decomposition, where a neural network is decomposed into blocks based on the memory consumption and local clients' memory budgets. An implict assumption made is that all blocks can be trained locally after the decomposition. in Section 4.1, to further address extreme case that some blocks still cannot fit into the memeory even after the finnest decomposition FEDEPTH integrates the partial training strategy into the local training stage. In Section 4.3, we consider the possibilty that some clients with rich memory budgets may suffer from memory underutilization, hence a variant of FEDEPTH is propsed to use mutual knowledge distillation (Zhang et al., 2018) to boost the performance and fully exploit the local memory of clients.

4.1 FEDEPTH and Its Variants

Memory-adaptive neural network decomposition. Since various clients could have drastically different memory budgets, FEDEPTH conducts local training in a memory-adaptive manner. Specifically, for the k-th client the full model $\mathcal{W}$ is decomposed to into $J_k + 1$ blocks, i.e., $\mathcal{W} = \{\theta_{k,1}, \cdots, \theta_{k,J_k}, \phi\}$, where $\{\theta_{k,j}\}_{j=1}^{J_k}$ and ϕ denote body and head of the neural network, respectively. Note that $\theta_{k,j}$ can be different from $\theta_{k',j}$ for any (k, k', j) triple, and the number of parameters contained in $\theta_{k,j}$ is solely determined by the kth client's memory budget, hence FEDEPTH is memory-adaptive. In practice, the model decomposition can be determined for each client before training via the estimating memory consumption (Gao et al., 2020). See Figure 4 for an illustration. Suppose the full-size model is composed of 6 layers, where each of layer costs memory of $\{3, 2, 1, 0.5, 0.5, 0.5\}$ GB, respectively. Assume the kth and k'th client has 3 GB and 5 GB memory budget, respectively. Then, client k has $J_k = 3$ and client k' has $J_{k'} = 2$ trainable blocks, respectively. That is, client k' will start with training the first two blocks, then the remaining four blocks.

Algorithm 1 FEDEPTH

Require: Total number of clients K; participation rate γ; number of communication rounds R.
Initialization: Model parameter $\mathcal{W}^0$.
for $t = 0, \ldots, R-1$ communication rounds **do**
  Sample a subset $\mathcal{S}^t$ of clients with $|\mathcal{S}^t| = \lceil \gamma K \rceil$.
  Broadcast $\mathcal{W}^t$ to clients $k \in \mathcal{S}^t$.
  for each client $k \in \mathcal{S}^t$ **in parallel do**
    $\mathcal{W}_k^{t+1} \leftarrow \text{ClientUpdate}(\mathcal{W}^t, k)$.
  end for
  Aggregate as $\mathcal{W}^{t+1} = \sum_{k \in \mathcal{S}^t} \frac{p_k}{\sum_{k' \in \mathcal{S}^t} p_{k'}} \mathcal{W}_k^{t+1}$.
end for
procedure CLIENTUPDATE($\mathcal{W}^t, k$)
  for $j = 1, \cdots, J_k$ **do**
    Approximately solve the problem (1).
  end for
  Set $\phi_k^{t+1} = \phi_J^{t+1}$
  Return $\mathcal{W}_k^t = \{\theta_{k,1}^{t+1}, \cdots, \theta_{k,J_k}^{t+1}, \phi_k^{t+1}\}$.
end procedure

Depth-wise sequential learning. Once the decomposition is determined, the k-th client at the t-th round, locally solves J_k subproblems in a block-wise fashion, i.e., for all $j \in \{1, \ldots, J_k\}$,

$$(\theta_{k,j}^{t+1}, \phi_j^{t+1}) \in \arg\min_{\theta_{k,j}, \phi} \mathcal{L}(\theta_{k,j}, \phi; \{z_{j-1,i}^{t+1}, y_i\}_{i=1}^{n_k}), \quad (1)$$

where $\mathcal{L}$ is a loss function, e.g, cross-entropy; $\{z_{j-1,i}^{t+1}\}_{i=1}^{n_k}$ are activations obtained after the local training samples forward-passed through the first j blocks, i.e., $z_{j-1,i}^{t+1} = f(x_i; \{\theta_{k,\ell}^{t+1}\}_{\ell=1}^{j-1})$ for $i \in [n_k]$ and $f(\cdot; \{\theta_{k,\ell}^{t+1}\}_{\ell=1}^{j-1})$ is the neural network up to the first $j-1$ blocks; n_k is the total number of training samples. Specifically, $z_{0,i}^{t+1} = x_i$ for all $i \in [n_k]$. Problem (1) can be solved by any stochastic gradient-based method. When solving for the j-th subproblem at the tth round, to fully leverage the global model $\mathcal{W}^t$'s body and the locally newly updated head, we use $(\theta_j^t, \phi_{j-1}^{t+1})$ as the initial point. See the data flow in Figure 5 when performing the training for the jth block. We remark that when the memory budget permits, the activation $\{z_{j-1,i}^{t+1}\}_{i=1}^{n_k}$, which no longer requires the gradient, could be buffered to compute $\{z_{j,i}^{t+1}\}_{i=1}^{n_k}$ once the $\theta_{k,j}^{t+1}$ is obtained, hence saving the redundant forward-pass from the first block to the jth block. Also, the number computation required

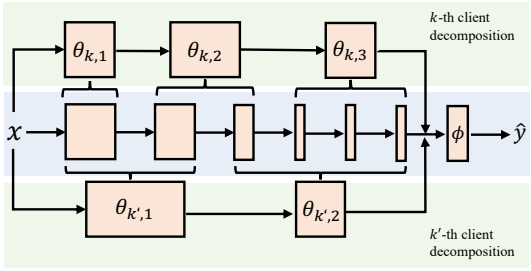

Figure 4. Memory-adaptive neural network decomposition. The second row represents the full neural network with the block size indicating the memory consumption. The first and third rows explain the neural network decomposition based on different clients' memory budgets.

for approximately solving J_k subproblems in the form of (1) should be similar to approximately solving the problem $\min_{\mathcal{W}} F_k(\mathcal{W})$ if we perform the same number of local updates, and ignore the negligible computation overhead in updating in the head ϕ. This is because the amount of computation required by one gradient evaluation $\nabla_{\mathcal{W}} F_k$ is equivalent to that of the summation of gradient evaluation of $\{\nabla_{\theta_j} \mathcal{L}\}_{j=1}^{J_k}$.

Attentive readers may notice that the outputs dimension of the jth block may not be consistent with the dimension of the ϕ, hence we use zero-padding on the activations outputted by the jth block to get the consistent dimension without introducing extra parameters. Other strategies, like, convolution filters are also applicable. To account for the potential noise introduced by the padding, we also develop a variant called m-FEDEPTH, which adds additional auxiliary heads $\{\phi_{k,j}\}_{j=1}^{J-1}$ in the local training phase. Specifically, the jth subproblem in (1) is modified as

$$(\theta_{k,j}^{t+1}, \phi_{k,j}^{t+1}) \in \arg\min_{\theta_{k,j}, \phi_{k,j}} \mathcal{L}(\theta_{k,j}, \phi_{k,j}; \{z_{j-1,i}^{t+1}, y_i\}_{i=1}^{n_k}).$$

and we set $\theta_k^{t+1} = \phi_{k,J}^{t+1}$.

Memory-efficient Inference. Depth-wise inference follows the similar logic of the *frozen-then-pass* forward in depth-wise training. Specifically, for each input x, we store the activation z_j in the hard drive and discard the predecessor activation z_{j-1}. Then we can reload z_j into memory as the input and get the activation z_{j+1}. The procedure is repeated until the prediction $\hat{y}$ is obtained.

We end this section by giving the detailed algorithmic description in Algorihtm 1.

4.2 Handle Extrem Memory Constraints with Partial Tranining

According to the memory consumption analysis in Section 3.2, the memory bottleneck of training a neural network is related to the block with the largest activations, which some devices may still not afford. These "large" blocks are usually the layers close to the input side. To tackle this issue, we borrow the idea of partial training in FEDEPTH, where we skip the first few blocks that are too large to fit even after the finnest blocks decomposition. This mechanism would not incur significant performance degradation because the input-side layers learn similar representations on different datasets (Kornblith et al., 2019), and clients with sufficient memory will provide model parameters of these input-side layers in the FL aggregation phase. Figure 6 emeprically validates such a strategy. We train a customized 14-layer ResNet (13 convolution layers and one classifier) on MNIST with 20 clients under non-IID distribution, respectively and measure the similarity of neural network representations, using both canonical correlation analysis (CCA) and centered kernel alignment (CKA) (Kornblith et al., 2019).

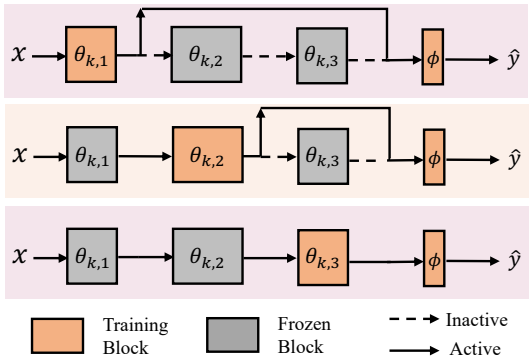

Figure 5. An example of depth-wise sequential learning. There are three training steps: 1) training the first block and the classifier with the skip connection (He et al., 2016a); 2) freezing the updated first block and using its activation to train the second block and the classifier with the skip connection; 3) freezing the updated first two blocks and using the activation of the second block to train the third block and the classifier.

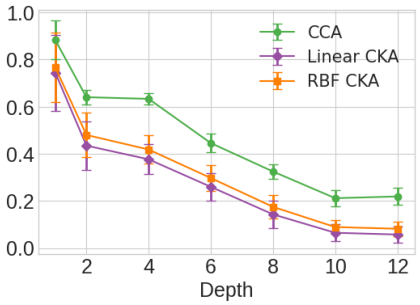

Figure 6. Correspondences between layers of different local neural networks trained from private datasets in FL under non-IID distribution. We observe that early layers, but not later layers, learn similar representations.

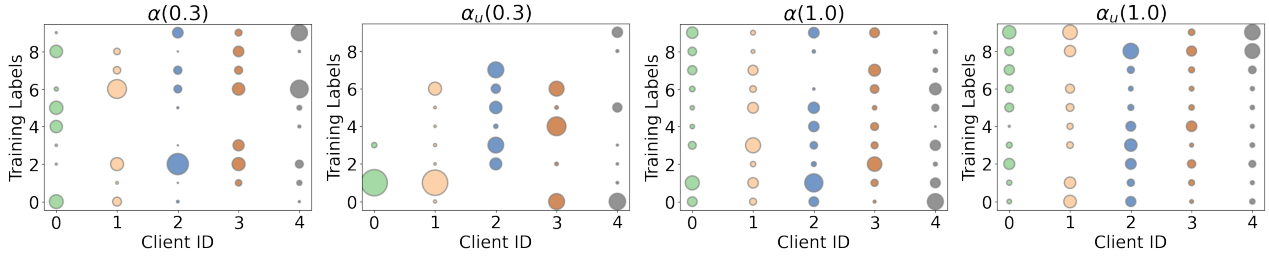

Figure 7. Visualization of statistical heterogeneity among partial clients on CIFAR-10 dataset, where the x-axis indicates client IDs, the y-axis indicates class labels, and the size of scattered points indicates the number of training samples for a label available to the client.

4.3 Exploit Sufficient Memory with Mutual Knowledge Distillation

Previous works on heterogeneous FL ignore the situation where some clients with rich computing resources may participate in the federated training on the fly. Their sufficient memory budget could be potentially utilized to improve the performance of FL via regularizing local model training (Mendieta et al., 2022; Li et al., 2020a). Ensembling multiple models is an effective way to improve generalization and reduce variance (Shi et al., 2021; Nam et al., 2021). However, considering each model is independently trained in ensemble learning methods, we have to upload/download all of these models in FL settings leading to a significant communication burden. Therefore, we design a new training and aggregation method based on mutual knowledge distillation (MKD) (Hinton et al.; Zhang et al., 2018), where all student neural networks learn collaboratively and teach each other. Therefore, clients with sufficient memory only need to upload one of the local models to the server for aggregation because the knowledge consensus achieved among all models through distillation. Formally, assume the kth client has a rich memory budget to train $M > 1$ models. Then locally it solves

$$\min_{\{\mathcal{W}_k^1,\cdots,\mathcal{W}_k^M\}} \frac{1}{M}\sum_{m=1}^{M} F_k(\mathcal{W}_k^m) + \frac{1}{M-1}\sum_{m'\neq m}^{M} \mathbf{KL}\left(\mathbf{h}^{m'} \| \mathbf{h}^{m}\right),$$

where $\mathbf{h}^m$ are logits calculated from the model $\mathcal{W}_m$ over the local training set and $\mathbf{KL}$ is the Kullback Leibler Divergence. More concretely, $\mathbf{KL}\left(\mathbf{h}^{m'} \| \mathbf{h}^{m}\right) = \frac{1}{n_k}\sum_{i=1}^{n_k} \mathbf{KL}(h_i^{m'} \| h_i^m)$, where h_i^m is the logits of the ith sample computed over model $\mathcal{W}_m$.

5 Experiments

In this section, we conduct extensive experiments to validate the effectiveness of FeDepth by image classification tasks. In Section 5.1, we compare the model performance of FeDepth and m-FeDepth with existing methods. Section 5.3 is an extra study about the robustness of FL algorithms against an imbalanced number of local data samples. In Section 5.4, we test the depth-wise finetuning strategy in FL scenarios.

5.1 Experimental Setups

1. Datasets and data partition. Our experiments are conducted on CIFAR-10 and CIFAR100 (LeCun et al., 1998; Krizhevsky & Hinton, 2009). To simulate the non-IID setting with class imbalance, we follow (Yurochkin et al., 2019; Acar et al., 2021; Gao et al., 2022) to distribute each class to clients using the Dirichlet distribution with $\alpha(\lambda)$, where $\lambda = \{0.3, 1.0\}$. Besides, we adopt pathological non-IID data partition $\beta(\Lambda)$ that is used by the selected baselines – HeteroFL and SplitMix (Diao et al., 2021; Hong et al., 2022), where each device has unique Λ labels with $\Lambda = \{2, 5\}$ for CIFAR-10 while $\Lambda = \{10, 30\}$ for CIFAR-100. We note that the ***balanced*** data partition is applied by default, which makes each client holds the same number of examples. The ***unbalanced*** $\alpha_u(\lambda)$ non-IID, where clients may have a different amount of samples with different feature distribution skew, is also used to evaluate the stability of FeDepth. We consider 100 clients in all experiments, and in Figure 7, we show label distributions of 5 out of 100 clients under balanced $\alpha(0.3), \alpha(1.0)$ and unbalanced $\alpha_u(0.3), \alpha_u(1.0)$ splits of CIFAR-10.

Depth	Memory	Width	Memory
$B_{1\sim3}$	20.02	$\times\frac{1}{8}$	14.51
B_4	14.05	$\times\frac{1}{6}$	19.34
$B_{5\sim6}$	10.07	$\times\frac{1}{3}$	38.68
B_7	7.21	$\times\frac{1}{2}$	58.02
$B_{8\sim9}$	5.28	$\times 1$	116.04

Table 1. Memory cost (in MB) with respect to depth and width of PreResNet-20. Each block consists of 2 convolution layers. $B_{1\sim3}$ indicates Block 1, 2 and 3 in PreResNet-20 have the same memory cost of 20.02 MB. The values are estimated by *pytorch-summary*[2].

2. Memory budgets. Using Pre-Activation of ResNet-20 (PreResNet-20) (He et al., 2016b) as an example, we show the relation of the memory cost of each block between width-wise and depth-wise training in Table 1. We can see that if clients afford to train $\frac{1}{6}$-width PreResNet-20, they can train the full-size neural network via depth-wise training. The training order is $\{ B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_{5,6} \rightarrow B_{7,8,9} \}$. Inspired by this example, we simulate three memory budget scenarios.

- **(Fair).** The memory budgets depend on the hidden channel shrinkage ratio, $r = \{\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, 1\}$, and are uniformly distributed into clients. It means 1/4 of clients can train a PreResNet-20 model within a maximal width of $\frac{1}{6}$-, $\frac{1}{3}$-, $\frac{1}{2}$- and full width, respectively.
- **(Lack).** $r = \{\frac{1}{8}, \frac{1}{6}, \frac{1}{2}, 1\}$. 1/4 of clients with limited memory adopt a partial training strategy.
- **(Surplus).** $r = \{\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, 2\}$. 1/4 of clients with sufficient memory can apply MKD.

[2]https://github.com/sksq96/pytorch-summary

Budget	Method	CIFAR-10				CIFAR-100			
		$\alpha(0.3)$	$\alpha(1.0)$	$\beta(2)$	$\beta(5)$	$\alpha(0.3)$	$\alpha(1.0)$	$\beta(10)$	$\beta(30)$
Unrealistic, $r=\{1,1,1,1\}$	~~FedAvg ($\times 1$)~~	~~61.00~~	~~66.33~~	~~32.72~~	~~69.95~~	~~28.44~~	~~29.58~~	~~31.86~~	~~43.84~~
Fair, $r=\{\frac{1}{6},\frac{1}{3},\frac{1}{2},1\}$	FedAvg ($\times\frac{1}{6}$)	48.82	54.57	24.94	52.69	15.52	16.50	12.60	18.48
	HeteroFL	52.08	55.53	26.68	56.71	16.76	18.84	19.79	30.26
	SplitMix	54.80	57.61	28.88	57.11	22.10	24.25	19.69	25.00
	FeDepth	59.79	63.39	29.14	60.43	24.51	25.90	22.74	33.49
	m-FeDepth	58.89	62.12	30.50	62.26	30.10	31.26	23.41	38.69
Lack, $r=\{\frac{1}{8},\frac{1}{6},\frac{1}{2},1\}$	FedAvg ($\times\frac{1}{8}$)	46.04	52.86	23.78	47.26	14.25	14.48	11.82	15.93
	HeteroFL	52.21	55.50	26.71	56.01	16.60	18.99	18.90	29.53
	SplitMix	50.34	53.22	25.35	52.73	21.40	22.80	17.91	22.78
	FeDepth	56.79	63.30	28.89	59.09	23.51	26.17	23.15	34.91
	m-FeDepth	57.72	61.08	28.22	59.06	30.59	31.28	23.78	37.54
Surplus, $r=\{\frac{1}{6},\frac{1}{3},\frac{1}{2},2\}$	FeDepth	59.82	**65.26**	30.79	63.27	25.16	27.33	24.21	35.81
	m-FeDepth	**60.76**	64.76	**31.34**	**65.63**	**32.07**	**36.52**	**25.67**	**39.90**

Table 2. Test results (top-1 accuracy) under balanced non-IID data partitions using PreResNet-20. Grey texts indicate that the training cannot conform to the pre-defined budget constraint. If not specified, FedAvg denotes the results with $\times \min(r)$ *-width* network. We highlight the best results with Blue Shadow, Red Shadow, and **Bold** in the scenarios including clients equipped with fairly sufficient, insufficient and abundant memory, respectively.

3. Implementation and evaluation. We compare FEDEPTH and its variants with several methods, including FedAvg (McMahan et al., 2017), HeteroFL (Diao et al., 2021), and SplitMix (Hong et al., 2022) in terms of Top-1 Accuracy. The memory budgets are uniformly distributed to 100 clients. All experiments perform 500 communication rounds with a learning rate of 0.1, local epochs of 10, batch size of 128, SGD optimizer, and a cosine learning rate scheduler.

5.2 Global Model Evaluation

Results in the *Fair Budget* scenario. In Table 2, we compare test results on a global test dataset (10,000 samples) considering a variety of balanced non-IID data partition and memory constraints. We highlight the best results under different scenarios. In all cases, HeteroFL, SplitMix, and FEDEPTH family outperform vanilla FedAvg, showing their system designs' effectiveness under balanced data distribution. Among all methods, our proposed FEDEPTH and m-FEDEPTH achieve the best performance with significant improvements. For example, on CIFAR10, under *Fair Budget*, FEDEPTH gains $5.44 \pm 2.39\%$ average improvement compared to HeteroFL while gains $3.59 \pm 2.12\%$ average improvement compared to SplitMix. m-FEDEPTH gains $5.69 \pm 1.18\%$ average improvement compared to HeteroFL while gains $3.84 \pm 1.33\%$ average improvement compared to SplitMix. Figure 8 shows convergence curves of FEDEPTH on non-IID CIFAR-10 dataset.

Results in the *Lack Budget* scenario. We observe that HeteroFL has slight accuracy drops or increases compared to the fair budget scenario. The explanation could be deducted from the behaviors of sub-networks discussed in Section 3.2 and Figure 2 that small sub-networks slightly influence the global performance because the small number of model parameters provides limited knowledge of the global model in the aggregation phase of FL. In contrast, SplitMix has an apparent performance degradation of an average of $2.86 \pm 1.41\%$ due to the weaker base learner. The FEDEPTH and m-FEDEPTH are relatively stable algorithms against insufficient memory budget, showing $0.45 \pm 1.25\%$ and $0.99 \pm 1.21\%$ degradation, respectively.

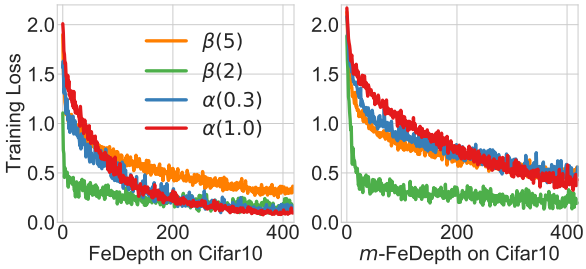

Figure 8. Convergence of FEDEPTH family on Cifar10.

Results in the *Surplus Budget* scenario. We let the new clients with rich resources $r=2$ join in FL and replace the clients with $r=1$. Prior works, like HeteroFL and SplitMix, did not consider such dynamics, and the clients with more memory budgets still train $\times 1$ neural networks. An alternative way is to train a new large base model from scratch and discard previously trained $\times 1$ neural networks, hence wasting computing resources. From Table 2, we can observe that MKD indeed makes sense for improving the performance of the global model (still $\times 1$-width). Furthermore, we note that combining depth-wise training and MKD is a flexible solution to simultaneously solve dynamic partition, device upgrade, and memory constraints. For example, when a new client with $r=\frac{7}{6}$ enters into the federated learning, the client can locally learn two models via regular and depth-wise sequential training, respectively, and then perform MKD while maintaining an original-size model for aggregation.

Comparison between FEDEPTH and its variant. As shown in Table 2, for CIFAR-10, FEDEPTH and m-FEDEPTH can achieve similar prediction accuracy. However, for CIFAR-100, m-FEDEPTH always outperforms FEDEPTH. It is worth recalling the design of FEDEPTH, which introduces zero paddings to match the dimension between two skip-connected blocks. This may inject negligible noise for the training on more complex data. Replacing the zero paddings with other modules, such as convolutions, may result in a better model. However, this usually comes at the cost of

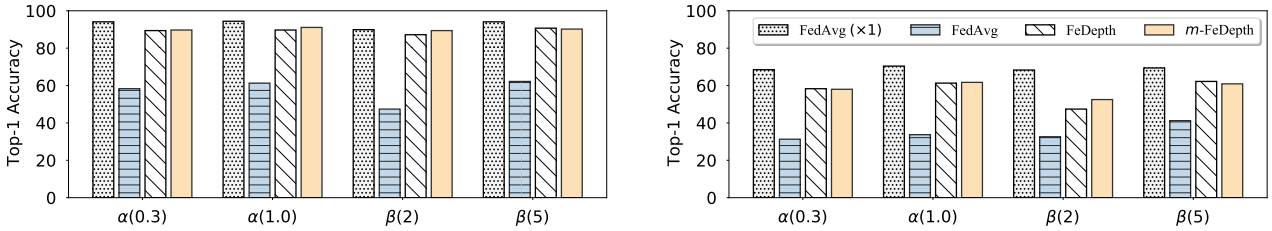

Figure 9. Fine-tuning ViT-T/16 on CIFAR-10 (**left**) and CIFAR-100 (**right**) under balanced non-IID data partitions with FedAvg FeDepth, and m-FeDepth. FedAvg $(\times 1)$ assumes each client can afford to train the full-size model with 12 identical encoder blocks, while FedAvg $(\times \frac{1}{6})$ assumes each client trains a $\frac{1}{6}$-width model, whose memory consumption is equal to train two encoder blocks.

Method	CIFAR-10		CIFAR-100	
	$\alpha_u(0.3)$	$\alpha_u(1.0)$	$\alpha_u(0.3)$	$\alpha_u(1.0)$
FedAvg $(\times \frac{1}{6})$	46.46	52.02	15.62	17.99
HeteroFL	46.14	52.20	16.02	18.36
SplitMix	31.23	44.70	22.68	25.28
FeDepth	**52.61**	**58.55**	23.25	26.16
m-FeDepth	51.58	57.91	**27.86**	**29.56**

Table 3. Experimental results on unbalanced Dirichlet partitions. Because of the relatively limited number of samples per class in CIFAR-100, an unbalanced Dirichlet partition outputs similar statistical distribution according to the number of local data examples.

extra memory consumption because of the new activations and parameters, which is usually intolerable to resource-constrained devices.

5.3 Influence of Unbalanced Non-IID Distribution

Table 3 shows the prediction results on distributed datasets in FL from the unbalanced Dirichlet partition (*Fair Budget*). We note that HeteroFL and SplitMix were not evaluated on such an unbalanced distribution. Overall, the higher skewed distribution leads to worse performance for FL, which can be observed by comparing results on Table 2 and Table 3.

Since the number of samples per class in CIFAR-100 is limited (there are 500 samples for each class), $\alpha_u(\lambda)$ and $\alpha(\lambda)$ will output similar statistical distribution according to the number of samples on each client in FL. Therefore, we obtain similar CIFAR-100 results on both balanced and unbalanced non-IID data partitions. Specifically, $\alpha(\lambda)$ always outputs 400 training samples per client on average. For CIFAR-100, $\alpha_u(0.3)$ outputs 399.40 ± 34.53 training samples per client, $\alpha_u(1.0)$ outputs 399.34 ± 17.74. For CIFAR-10, $\alpha_u(0.3)$ outputs 399.44 ± 150.60 training samples per client, $\alpha_u(1.0)$ outputs 399.39 ± 77.37.

Regarding CIFAR-10 results, we observe that HeteroFL and SplitMix cannot achieve comparable predictions or generalization ability compared to FedAvg. SplitMix even performs worse than training with the smallest models in FL. This result indicates that SplitMix is not robust to unbalanced distribution. One reason for this phenomenon is that small base models cannot capture representative features due to the significant weight divergence between local clients stemming from a highly skewed distribution (Frankle & Carbin, 2019; Li et al., 2022). For HeteroFL, as mentioned in the case study in Section 3.2, the full-size neural networks on resource-sufficient devices provide the fundamental ability but small sub-networks trained with unbalanced distribution indeed affect the global performance. In contrast to HeteroFL and SplitMix, our proposed FeDepth and m-FeDepth gain substantial improvements of 6.15% and 6.53% on CIFAR-10, and of 12.24% and 11.57% on CIFAR-100 compared to FedAvg.

5.4 Depth-wise Fine-tuning on ViT

Foundation models or Transformer architectures (Vaswani et al., 2017; Zhou et al., 2023), such as Vision Transformer (ViT) (Dosovitskiy et al., 2020), has shown robustness to distribution shifts (Bhojanapalli et al., 2021). Recent work has demonstrated that replacing a convolutional network with a pre-trained ViT can greatly accelerate convergence and result in better global models in FL(Qu et al., 2022). Inspired by this finding, we hypothesize that fine-tuning ViT with depth-wise learning will still produce a better global model because 1) decomposing blocks in a depth-wise manner maintains the knowledge learned from pretraining, and 2) memory consumptions of activations in each ViT's block are identical, which indicates that skip connection for handling resource constraints does not introduce any noises and extra parameters. The memory budgets in terms of the width shrinkage ratio $r = \{\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, 1\}$ are uniformly allocated to 100 clients as the same setting of PreResNet-20 in the scenario of *Fair Budget*.

For fine-tuning, we choose a learning rate of 5×10^4 and a training epoch of 100. Figure 9 shows the test results of ViT-T/16 (Qu et al., 2022) under balanced Dirichlet data partitions, on which we observe that exploiting FeDepth and m-FeDepth can produce good global models. Specifically, FeDepth-ViT significantly outperforms FeDepth-PreResNet-20 with $36.06 \pm 12.79\%$ and $30.64 \pm 4.24\%$ improvements on CIFAR-10 and CIFAR-100 on average, respectively. m-FeDepth-ViT significantly outperforms m-FeDepth-PreResNet-20 with $36.66 \pm 12.88\%$ and $27.41 \pm 3.13\%$ improvements on CIFAR-10 and CIFAR-100 on average, respectively. We also observe that although local ViTs are fine-tuned on varying distribution data, we obtain global models with similar performance. It indicates that ViT is more robust to distribution shifts and hence improves FL over heterogeneous data.

6 Conclusions

Despite the recent progress in FL, memory heterogeneity still remains largely underexplored. Unlike previous methods based on width-scaling strategies, we propose depth-wise learning for handling varying memory capabilities. The experimental results demonstrate our proposed FeDepth family outperform the state-of-the-art algorithms including HeteroFL and SplitMix, and are robust to data heterogeneity and client participation. FeDepth is a flexible and scalable framework that can be compatible with most FL algorithms and is reliable to be deployed in practical FL

systems and applications.

REFERENCES

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., and Veit, A. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10231–10241, 2021.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Bubeck, S. and Sellke, M. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34:28811–28822, 2021.

Chen, J., Zheng, L., Yao, Z., Wang, D., Stoica, I., Mahoney, M., and Gonzalez, J. Actnn: Reducing training memory footprint via 2-bit activation compressed training. In *International Conference on Machine Learning*, pp. 1803–1813. PMLR, 2021.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Diao, E., Ding, J., and Tarokh, V. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Du Terrail, J. O., Ayed, S.-S., Cyffers, E., Grimberg, F., He, C., Loeb, R., Mangold, P., Marchand, T., Marfoq, O., Mushtaq, E., et al. Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings. In *NeurIPS, Datasets and Benchmarks Track*, 2022.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Gao, L., Fu, H., Li, L., Chen, Y., Xu, M., and Xu, C.-Z. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10112–10121, 2022.

Gao, Y., Liu, Y., Zhang, H., Li, Z., Zhu, Y., Lin, H., and Yang, M. Estimating gpu memory consumption of deep learning models. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1342–1352, 2020.

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Hinton, G., Vinyals, O., Dean, J., et al. Distilling the knowledge in a neural network.

Hong, J., Wang, H., Wang, Z., and Zhou, J. Efficient split-mix federated learning for on-demand and in-situ customization. In *International Conference on Learning Representations*, 2022.

Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., and Lane, N. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

Hyeon-Woo, N., Ye-Bin, M., and Oh, T.-H. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14 (1-2):1–210, 2021.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. Scaffold: Stochastic controlled averaging for on-device federated learning. 2019.

Kim, M., Yu, S., Kim, S., and Moon, S.-M. DepthFL : Depthwise federated learning for heterogeneous clients. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=pf8RIZTMU58.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021.

Li, Q., Diao, Y., Chen, Q., and He, B. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 965–978. IEEE, 2022.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020a.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2020b.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020a.

Lin, Y., Ren, P., Chen, Z., Ren, Z., Yu, D., Ma, J., Rijke, M. d., and Cheng, X. Meta matrix factorization for federated rating predictions. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 981–990, 2020b.

Liu, R., Wu, F., Wu, C., Wang, Y., Lyu, L., Chen, H., and Xie, X. No one left behind: Inclusive federated learning over heterogeneous devices. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3398–3406, 2022.

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

Mendieta, M., Yang, T., Wang, P., Lee, M., Ding, Z., and Chen, C. Local learning matters: Rethinking data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8397–8406, 2022.

Mohri, M., Sivek, G., and Suresh, A. T. Agnostic federated learning. In *International Conference on Machine Learning*, pp. 4615–4625. PMLR, 2019.

Nam, G., Yoon, J., Lee, Y., and Lee, J. Diversity matters when learning from ensembles. *Advances in Neural Information Processing Systems*, 34:8367–8377, 2021.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019.

Qu, L., Zhou, Y., Liang, P. P., Xia, Y., Wang, F., Adeli, E., Fei-Fei, L., and Rubin, D. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10061–10071, 2022.

Raihan, M. A. and Aamodt, T. Sparse weight activation training. *Advances in Neural Information Processing Systems*, 33:15625–15638, 2020.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *International Conference on Learning Representations*, 2020.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088): 533–536, 1986.

Seo, H., Park, J., Oh, S., Bennis, M., and Kim, S.-L. Federated knowledge distillation. *arXiv preprint arXiv:2011.02367*, 2020.

Shi, N., Lai, F., Kontar, R. A., and Chowdhury, M. Fed-ensemble: Improving generalization through model ensembling in federated learning. *arXiv preprint arXiv:2107.10663*, 2021.

Sohoni, N. S., Aberger, C. R., Leszczynski, M., Zhang, J., and Ré, C. Low-memory neural network training: A technical report. *arXiv preprint arXiv:1904.10631*, 2019.

Sun, Z., Kairouz, P., Suresh, A. T., and McMahan, H. B. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient learning via atomic sparsification. *Advances in Neural Information Processing Systems*, 31, 2018.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020a.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020b.

Wang, J., Charles, Z., Xu, Z., Joshi, G., McMahan, H. B., Al-Shedivat, M., Andrew, G., Avestimehr, S., Daly, K., Data, D., et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021a.

Wang, J., Xu, Z., Garrett, Z., Charles, Z., Liu, L., and Joshi, G. Local adaptivity in federated learning: Convergence and consistency. *arXiv preprint arXiv:2106.02305*, 2021b.

Yao, D., Pan, W., Wan, Y., Jin, H., and Sun, L. Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization. *arXiv preprint arXiv:2111.14655*, 2021.

Yu, J. and Huang, T. S. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1803–1811, 2019.

Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. Slimmable neural networks. In *International Conference on Learning Representations*, 2018.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pp. 7252–7261. PMLR, 2019.

Zhang, K., Jiang, Y., Seversky, L., Xu, C., Liu, D., and Song, H. Federated variational learning for anomaly detection in multivariate time series. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 1–9. IEEE, 2021a.

Zhang, K., Wang, Y., Wang, H., Huang, L., Yang, C., Chen, X., and Sun, L. Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 613–621, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics.

Zhang, L., Bao, C., and Ma, K. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4388–4403, 2021b.

Zhang, Y., Xiang, T., Hospedales, T. M., and Lu, H. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4320–4328, 2018.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Zhang, K., Ji, C., Yan, Q., He, L., et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

Zhu, Z., Hong, J., and Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 12878–12889. PMLR, 2021.