FedRolex: Model-Heterogeneous Federated Learning with Rolling Sub-model Extraction

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated learning (FL) is a collaborative machine learning paradigm to train models from decentralized private data. Most FL research focuses on model-homogeneous setting where models deployed across all the participating clients and the server are required to be identical. However, in real-world scenarios, such a requirement acts as a constraint that restricts the outreach to clients with heterogeneous device resources and unfairly excludes users with low-end devices who would otherwise benefit from FL. In this work, we propose a simple yet effective model-heterogeneous FL method named `FedRolex` to tackle this constraint. Unlike the model-homogeneous scenario, the fundamental challenge of model heterogeneity in FL is that different parameters of the global model are trained on heterogeneous data distributions. Our method addresses this challenge by rolling the sub-model in each federated iteration so that the parameters of the global model are evenly trained on the global data distribution across all devices, making it more akin to model-homogeneous training. Our experiments show that `FedRolex` outperforms other state-of-the-art model-heterogeneous FL methods, especially under high data-heterogeneity scenarios. We have conducted ablation studies to show that submodel rolling is an effective technique to reduce the gap between model-heterogeneous and standard model-homogeneous settings. Lastly, we consider the distribution of client capabilities that is similar to real-world income distribution instead of the uniform distribution used in existing works. Our results show a consistent improvement in the accuracies on low-end devices, which enhances the inclusiveness of federated learning.

1 Introduction

Cross-device federated learning (FL) was originally proposed by McMahan et al. (2017) as a privacy-preserving machine learning paradigm to train a machine learning model on a federation of resource-constrained edge devices without accessing their data. This idea has far-reaching applications, which is made urgent by modern demands for privacy. Indeed, the goal of strong data privacy is a central motivation for FL. By storing data locally, instead of replicating them on a remote server, the attack surface of the system is decreased (Jere et al., 2020), and by using focused ephemeral updates and early aggregation, the communication cost is also minimized (Nishio and Yonetani, 2019).

The majority of existing cross-device FL studies focus on *model-homogeneous* setting, where models deployed across all the participating clients and the server are required to be identical. In practice, device heterogeneity is a more realistic constraint to consider for deploying in real-world environments. Different edge devices have different on-device resources and, hence, are only capable of training models with capacities that match their on-device resources. Having the same model on all the devices forces a limitation that restricts the outreach to clients with heterogeneous device resources and unfairly excludes users with low-end devices who would otherwise benefit from FL.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1: Comparison of `FedRolex` with model-homogeneous and model-heterogeneous FL methods.

	Model Heterogeneity	Need of Public Data	Scaling to Large Federations	Server Model Size
FedAvg McMahan et al. (2017)	No	No	Scalable	= Client Model
FedProx Li et al. (2020)	No	No	Scalable	= Client Model
SCAFFOLD Karimireddy et al. (2020)	No	No	Scalable	= Client Model
FedBE Chen and Chao (2020)	No	Unlabeled	Computationally Expensive	= Client Model
FedDF Lin et al. (2020)	Yes	Unlabeled	Computationally Expensive	= Largest Client Model
FedGKT He et al. (2020)	Yes	No	Computationally Expensive	$\geq$ Largest Client Model
FedGEMS Cheng et al. (2021)	Yes	Labeled	Computationally Expensive	$\geq$ Largest Client Model
Fed-ET Cho et al. (2022)	Yes	Unlabeled	Computationally Expensive	$\geq$ Largest Client Model
Federated Dropout Caldas et al. (2018)	Yes	No	Large Federation Required	$\geq$ Largest Client Model
HeteroFL Diao et al. (2020)	Yes	No	Scalable	= Largest Client Model
FjORD Horvath et al. (2021)	Yes	No	Scalable	= Largest Client Model
FedRolex	**Yes**	**No**	**Scalable**	**$\geq$ Largest Client Model**

To relax this fundamental constraint in standard FL, recent works proposed *model-heterogeneous* FL, where models with different capacities are trained on edge devices with heterogeneous resources during the federated training process. One fundamental challenge in model-heterogeneous FL is the aggregation of heterogeneous client models. Lopes et al. (2017); He et al. (2020); Cho et al. (2022) proposed to use knowledge distillation (KD) (Hinton et al., 2015), where the client models serve as teachers and the server ensembles the knowledge distilled from the individual client models. However, KD-based methods generally require access to public data to achieve competitive model accuracy, where the accessibility of high-quality public data is not always practical. Moreover, KD becomes computationally expensive when the cohort size in each FL round scales up. To remove the dependency on public data, another set of studies focuses on designing partial training (PT) based techniques such as federated dropout (Caldas et al., 2018), ordered dropout (Horvath et al., 2021), and static sub-model extraction (Diao et al., 2020), where each client trains a smaller sub-model of the larger global model on the server. However, the fundamental issue of existing PT-based methods is that the sub-models are extracted in ways such that the parameters of the global model obtained by aggregating the extracted sub-models are *not* evenly trained, making the global model vulnerable to a phenomenon called *client drift* (Wang et al., 2021) where it is used to describe the error in gradient estimation due to the differences in data distributions between clients. However, this estimation error can originate from any discrepancy across the federation. For instance, in model heterogeneous federated learning, client drift will also occur due to the inconsistency between the clients' individual model architectures in addition to the difference in private data distributions. This also becomes more prominent when the cohort size in each communication round is small or data heterogeneity across clients becomes more severe, resulting in lower global model accuracy.

In this work, we propose a simple yet effective PT-based model-heterogeneous FL method named `FedRolex` that tackles the fundamental issue of existing PT-based methods. Compared to existing PT-based methods, `FedRolex` proposes a *rolling sub-model extraction* scheme, where the sub-model is extracted from the global model using a rolling window that advances in each communication round. As such, sub-models are extracted from different parts of the global model in different rounds. Since the extraction window is rolling in each round, parameters of the global model are *evenly trained* to minimize the client drift. Throughout communication rounds, all the parameters of the global model are updated over the entire data distribution. As we show in §4.2, such a rolling sub-model extraction strategy significantly reduces the gap between model-heterogeneous and model-homogeneous settings.

We evaluate the performance of `FedRolex` on three datasets – CIFAR-10, CIFAR-100 (Krizhevsky, 2009) and StackOverflow (tff) – and compare it against state-of-the-art KD-based model-heterogeneous FL methods FedDF (Lin et al., 2020), DS-FL (Itahara et al., 2020) and Fed-ET (Cho et al., 2021) as well as PT-based model-heterogeneous FL methods HeteroFL (Diao et al., 2020) and Federated Dropout (Caldas et al., 2018)[1]. Our results show that `FedRolex` consistently outperforms state-of-the-art PT-based methods in both high and low data heterogeneity scenarios. `FedRolex` also outperforms state-of-the-art KD-based methods in low data heterogeneity scenarios as well as on the more challenging CIFAR-100 dataset in high data heterogeneity scenarios without public data. Furthermore, we show that our proposed rolling sub-model extraction scheme is flexible to train a global model that can be much larger than the largest client model. Lastly, we conducted an experiment that

[1]We did not compare with FjORD because its code is not open-source and we could not reproduce their results following the paper.

uses real-world household income distribution to emulate real-world device distribution. Our results show that `FedRolex` effectively boosts the model accuracy of low-end devices, which enhances the inclusiveness of federated learning. The code of `FedRolex` will be publicly available on Github.

2 Related Work

Our work focuses on model-heterogeneous FL. Existing methods can be generally grouped into two categories: knowledge distillation (KD) based and partial training (PT) based methods.

Knowledge Distillation (KD) based Model-Heterogeneous FL. The key idea of knowledge distillation (Hinton et al., 2015; Mirzadeh et al., 2020; Liu et al., 2019) is to compress a large pre-trained model by teaching a smaller network, step by step, exactly what to do using this larger pre-trained network. This mechanism has been adopted in model-heterogeneous FL to train the server model from a federation of client models with different architectures. For example, FedDF (Lin et al., 2020) distills knowledge from a set of classifiers trained with private data from a federation of client devices. The logit outputs of each classifier against a public dataset are then used to train a student model at the server with KD. Similarly, DS-FL (Itahara et al., 2020) keeps an unlabeled public dataset at the server, aggregates, and averages the logit outputs, and broadcasts the updated model back to the clients. Local models now train on the additional pseudo-labeled data, and performance is enhanced owing to the data augmentation effect. Fed-ET (Cho et al., 2022) considers a weighted consensus distillation approach with diversity regularization that enables training of a large server model with smaller models at the clients. These models have several drawbacks. KD is generally a computationally expensive task and so the costs blow up as the system scales up for larger models and more clients. These are also typically restricted to classification tasks as they depend on logits for implementing KD. The methods are also likely to face deployment issues, especially for tasks with a large number of possible output classes.

Partial Training (PT) based Model-Heterogeneous FL. There have been several studies employing different methods to train models of different sizes. These generally follow PT. HeteroFL (Diao et al., 2020) uses this concept to aggregate models of different sizes. Fjord (Horvath et al., 2021) uses ordered dropout, which similarly samples submodels from a global model, trains them, and then does a weighted aggregation. PruneFL (Jiang et al., 2022) uses model pruning to extract submodels. These algorithms are easily scalable and have minimum overheads on the server. However, these methods have a significant drawback in the sense that different parts of the model are restricted to seeing updates from a fixed pool of client data. This limits the performance as the global knowledge is concentrated on a smaller portion of the model. Furthermore, the server model capability is restricted to the same capability as the largest client model. Servers are usually much more computationally capable and so should be able to use a larger model. In comparison, Federated Dropout (Caldas et al., 2018; Guliani et al., 2022; Ro et al., 2022) can train on larger global models and use its entire capability, especially when the dataset and the number of clients are large. Here, a portion of the parameters in each layer of the global model is dropped, leaving a smaller submodel. Submodels are trained on the client devices according to their capacities and the updates are aggregated via weighted averaging similar to traditional dropout. However, as reported by Bouacida et al. (2020), the approach works well only with less heterogeneous data and large client pools. This is because of high variance in heterogeneous data when only a small subset of the client pool is selected.

3 Our Method

3.1 Problem Formulation of Model-Heterogeneous FL

Consider N different client devices, $\mathcal{N}$, with different compute capacities $\{C_1, C_2, ..., C_N\}$ and private data $\{D_1, D_2, ..., D_N\}$, each with different distributions. The objective is to train a large global model characterized by the set of trainable model parameters θ. Therefore, model-heterogeneous FL can be formulated as the following distributed optimization problem:

$$\min_{\theta} \left\{ F(\theta) \triangleq \sum_{n=1}^{N} p_n F_n(\theta) \right\} \tag{1}$$

where $F(\theta) = \mathbf{E}_{\xi \sim D} l(\theta; \xi)$, $F_n(\cdot)$ is the local objective function on the client dataset $D_n = \{d_{n,1}, d_{n,2}, d_{n,3} ... d_{n,m_n}\}$ with respect to the user defined loss function $l(\cdot;\cdot)$ and p_n is the corre-

sponding weight of the nth client such that $p_n \geq 0$ and $\sum_{n=1}^{N} p_n = 1$. The nth device contains m_n data points and has model parameters θ_n (this can be changed from one iteration to another). The size of θ_n depends on the client's model capacity C_n and are extracted from the global model i.e., $\theta_n \subset \theta$. The local objective function $F_n(\cdot)$ is thereby rewritten as:

$$F_n(\theta_n) \triangleq \frac{1}{m_n} \sum_{k=1}^{m_n} l(\theta_n; d_{n,k}). \tag{2}$$

For simplicity, we abuse the notation of l by ignoring the other parameters in θ but not in θ_n. The core of the optimization problem is to select subset θ_n from the global model parameters θ based on model capacity C_n.

3.2 Partial Training and `FedRolex`

As discussed in Section 2, PT-based methods train heterogeneous models by extracting a subset of the global model and training the smaller model on private data. The primary step in PT-based methods is this sub-model extraction. Consider $\theta_n^{(j)}$ selected from global model weights, θ for client n at the jth communication round. Assume that the proportion of nodes extracted from each layer in θ for client n be β_n, which belongs to the set of unique model capacities $\boldsymbol{\beta}$. Let the total number of nodes in the ith layer be K_i and the selected nodes are $\mathcal{S}_i^{(j)}$. Then the parameters $\theta_{n,[i,\cdot]}^{(j)}$ for the ith layer in extracted model parameters $\theta_n^{(j)}$ consist of $\{\theta_{n,[i,\mathcal{S}_{i,0}^{(j)}]}^{(j)}, \theta_{n,[i,\mathcal{S}_{i,1}^{(j)}]}^{(j)}, ..., \theta_{n,[i,\mathcal{S}_{i,K_i}^{(j)}]}^{(j)}\}$.

Let, $\mathcal{M} \subset \mathcal{N}$ be the set of selected clients from the client pool from which the server pulls model parameters at round j. Let $\theta_{[i,k]}$ be the kth trainable weight of the ith layer of global model and $\theta_{m,[i,k]}; m \in \mathcal{M}$ be the kth trainable weight of the ith layer of the client m. The model parameters are aggregated as follows:

$$\theta_{[i,k]} = \frac{1}{\sum_{m \in \mathcal{M}} p_m} \sum_{m \in \mathcal{M}} p_m \theta_{m,[i,k]} \tag{3}$$

Here, p_m is an empirical weight given to a client. This can be based on a variety of factors like the client model capability, the number of data points a client has, etc. However, there is no theoretical foundation in the literature to quantify the relative importance of a client update nor are there empirical studies that accurately study this. We consider this tangent to our goals in this study and so, throughout the paper, unless otherwise stated, the weight of all clients is assumed to be the same, i.e, $p_m = 1$. For different PT-based methods, the method in which the subset of selected parameters, $\mathcal{S}$ is chosen varies according to their design. Finally, the optimal sub-model is extracted from the global model for each client according to client capability.

3.3 Comparing Sub-model Extraction Techniques

`FedRolex` proposes a new way to extract the sub-models. In general, sub-models are created by pulling a subset of weights from the global model. This subset can be dynamic (Federated Dropout, `FedRolex`) in each round or remain static for all rounds (HeteroFL). In this section, we compare `FedRolex` with two existing methods: static sub-model extraction (HeteroFL) and random sub-model extraction (Federated Dropout). For simplicity, we focus on the ith layer at the nth client during the jth communication round only. Let K_i denote the number of nodes in the ith layer of the global model and β_n denote the capacity of the client model.

Static Sub-Model Extraction Scheme In this scheme (used by HeteroFL), the global model remains static and the extracted sub-models are always the same. This is shown in the left column of Figure 1. Unused nodes do not see the full data distribution. The subset of node indices is given by:

$$\mathcal{S} = \{1, 2, \ldots, \lfloor \beta_n K_i \rfloor\}. \tag{4}$$

Random Sub-Model Extraction Scheme. In this scheme (used by Federated Dropout), the sub-models are extracted by sampling nodes uniformly and randomly. This is shown in the middle column of Figure 1. The subset is given by:

$$\mathcal{S} = \{k_c \mid \text{integer } k_c \in [1, K_i] \text{ and } 1 \leq c \leq \lfloor \beta_n K_i \rfloor\} \tag{5}$$

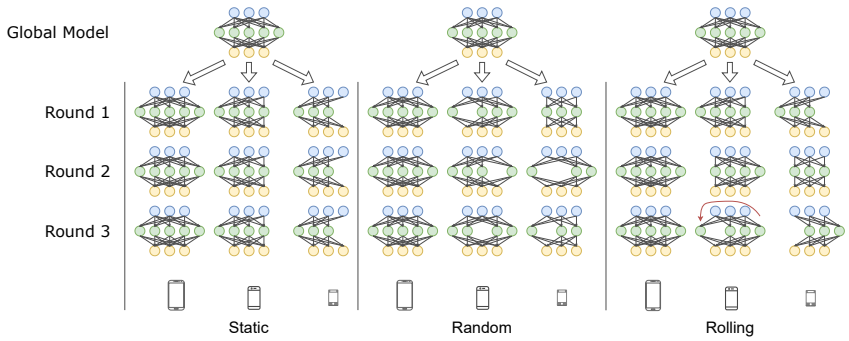


Figure 1: Illustration of how sub-models are extracted by (left) static extraction (HeteroFL); (middle) random extraction (Federated Dropout); and (right) rolling extraction (`FedRolex`).

where $\lfloor \beta_n K_i \rfloor$ nodes are randomly chosen from the global model. Unlike static sub-model extraction, since nodes are shuffled, all nodes end up seeing the entire data given enough communication rounds. However, this may require a large number of communication rounds.

Rolling Sub-Model Extraction. In this scheme (used by `FedRolex`), the global model is rolled before the subset is extracted. This is shown in the right column of Figure 1. Let $\hat{j} = j \bmod K_i$, then the subset is given by:

$$\mathcal{S} = \begin{cases} \{\hat{j}, \hat{j}+1, \ldots, \hat{j}+\lfloor \beta_n K_i \rfloor - 1\}, & \text{if } \hat{j}+\lfloor \beta_n K_i \rfloor \leq K_i + 1, \\ \{\hat{j}, \hat{j}+1, \ldots, K_i\} \cup \{1, 2, \ldots, \hat{j}+\lfloor \beta_n K_i \rfloor - 1 - K_i\} & \text{else.} \end{cases} \tag{6}$$

That is, the selected nodes start at $\hat{j}$ and increase until $\lfloor \beta_n K_i \rfloor$ nodes are chosen. When the node index reaches K_i, it goes back to one again.

Why Rolling Sub-model Extraction is better? PT-based algorithms HeteroFL and FjORD extract the model parameters θ_n from specific locations in the global model. Here the client model size is based on the client capacity, and the nodes are selected starting from the first index. The extracted sub-models are then trained on selected devices from the client pool, and the updates for each parameter are aggregated and averaged across participating clients for that parameter. The nodes in the beginning indices of each layer will therefore see updates that were averaged across more clients than end indices. In comparison, the nodes at the end indices will see updates averaged across much fewer clients, and therefore the updates to these nodes will have more uncertainty. Dropout-based algorithms like Federated Dropout Caldas et al. (2018) propose random dropout to create smaller sub-models. Though the expected value of the frequency for updating each index is the same for all the indices in Federated Dropout, their exact frequencies are not the same in each experiment because of the randomness. This could potentially cause client drift. In fact, given I indices, if one index is chosen at each round, the probability to update all indices once in I rounds is $\prod_{i=1}^{I} \frac{i}{I}$. Furthermore, the expected number of rounds to go through all indices at least once is close to $I \ln(I)$, see Appendix A.1. It shows that random dropout can not balance the update frequencies of the indices, and it takes a large number of rounds to update all the indices.

4 Experiments and Analysis

4.1 Experimental Setup

Datasets and Platform. We validate the performance of `FedRolex` on three datasets – CIFAR-10, CIFAR-100, and StackOverflow. We apply bounding box crop (Zoph et al., 2020) to augment the images. To simulate the non-iid nature of the data, we use balanced non-iid data distribution following HeteroFL. We skew the data distribution by restricting the clients to having a fixed number of unique labels L. We define $L = 2$ as high data heterogeneity (i.e., high non-iid) and $L = 5$ as low data heterogeneity (i.e., low non-iid) for CIFAR-10. Similarly, we use $L = 20$ as high non-iid and $L = 50$ as low non-iid for CIFAR-100. These roughly correspond to Dirichlet distribution $Dir_K(\alpha)$ with α set to 0.1 and 0.5, respectively. The data is partitioned into 100 non-iid partitions and assigned to the client pool. In each communication round, 10% of the clients are randomly selected from the client pool of 100. For StackOverflow, the data is partitioned over user ids. The total number of clients, in this case, was $342,477$ and in each round, 200 clients were sampled. We use a vocabulary size

Algorithm 1: FedRolex

Initialization ; $\theta^{(0)}, \mathcal{N}$
Input : D_n β_n $\forall n \in \mathcal{N}$,
Output : θ^J
Server Executes
for $j \leftarrow 0$ **to** J **do**
  Broadcast $\theta^{(j)}_{n,[i,\mathcal{S}^{(j)}_{i,k}]}$
  $\forall i, \mathcal{S}$ from Equation 6, $1 \leq k \leq K_i$ and $n \in \mathcal{N}$
  Sample subset $\mathcal{M}$ from $\mathcal{N}$
  for *each* *client* $m \in \mathcal{M}$ **do**
    `clientStep`($\theta^{(j)}_n$, D_n)
  end
  Aggregate $\theta^{(j+1)}_{[i,k]} = \frac{1}{\sum_{m \in \mathcal{M}} 1} \sum_{m \in \mathcal{M}} \theta_{m,[i,k]}$
end

Subroutine `clientStep`($\theta^{(j)}_n$, D_n)
  $m_n \longleftarrow len(D_n)$
  for $k \leftarrow 0$ **to** m_n **do**
    $\theta_n \longleftarrow \theta_n - \eta \nabla l(\theta_n; d_{n,k})$
  end
  return θ_n

of 10,000 words. We run our experiments on 8 Nvidia A6000 GPUs using the PyTorch library for machine learning and the Ray library for simulating the distributed system.

Models. We use pre-activated ResNet18 (PreResNet18) models (He et al., 2016) for CIFAR-10 and CIFAR-100. We replace the batch Normalization in PreResNet18 with static batch normalization (Diao et al., 2020; Andreux et al., 2020) and add a scalar module after each convolution layer (Diao et al., 2020). For StackOverflow experiments, we trained a modified 3-layer Transformer model, where the dimension of the token embeddings was 128, and the hidden dimension of the feed-forward network(FFN) block is 2048. We use 8 heads for the multi-head attention, where each head is based on 12-dimensional (query, key, value) vectors. We use ReLU activation and set the dropout rate to 0.1. To simplify our experiments and ablation studies, the model sizes are restricted to a set of 5, i.e, β_n would take any value in $\{1, 1/2, 1/4, 1/8, 1/16\}$. Note that we only vary the number of kernels in convolution layers and keep the nodes in the output layers the same in both our method and the PT-based baselines to keep the comparison fair. Similarly, for the Transformer model, we only vary the number of nodes in the hidden layer of the attention heads. We provide pseudocode for both HeteroFL and our Federated Dropout variant in Appendix A.7. Unless otherwise stated, the client capacity distribution is assumed to be uniform over the entire client pool for all methods. We list the exact hyperparameters and architecture of the models in Appendix A.6

Baselines. We compare `FedRolex` against state-of-the-art PT-based methods HeteroFL and Federated Dropout and state-of-the-art KD-based methods FedDF, DS-FL, and Fed-ET obtained from Lin et al. (2020). For a fair comparison, all the baselines from the PT-based methods are trained on the same learning rate, the same number of communication rounds, and the same multi-step learning rate decay schedule. The exact schedule for each dataset and experiment is given in Appendix A.6.

Evaluation Metrics. We use local and global model accuracy as our evaluation metrics. Specifically, global model accuracy is defined as the mean accuracy on the test set totally held out during training; local model accuracy is defined as the mean accuracy of the server model on each of the client datasets. For vision datasets CIFAR-10 and CIFAR-100, we measure classification accuracy and for the StackOverflow dataset, we measure next word prediction accuracy. Since there is some randomness due to the non-iid partitioning, we run our experiments on 5 different seeds and list the mean and standard deviation of our metrics. Note that the results of KD-based methods were obtained from Cho et al. (2022) using 3 different seeds.

4.2 Performance Comparison with State-of-the-Art Model-Heterogeneous FL Methods

Table 2 shows the global model accuracy comparison between `FedRolex` and the baselines under both high and low data heterogeneity settings. We also include the global model accuracy of two model-homogeneous cases where all clients have the highest capacity ($\beta = 1.0$) and lowest capacity

Table 2: The global model accuracy comparison on CIFAR-10 and CIFAR-100 under both high and low data heterogeneity. Comparison on StackOverflow dataset is shown as well. Note that the StackOverflow is partitioned by users and so has natural heterogeneity and so the average local accuracy is shown. Additionally, KD-based methods cannot be used directly for language modeling and so their results are marked N/A for StackOverflow

	Method	High Data Heterogeneity		Low Data Heterogeneity		StackOverflow
		CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	
KD based	FedDF	73.81 (± 0.42)	31.87 (± 0.46)	76.55 (± 0.32)	37.87 (± 0.31)	N/A
	DS-FL	65.27 (± 0.53)	29.12 (± 0.51)	68.44 (± 0.47)	33.56 (± 0.55)	N/A
	Fed-ET	**78.66 (± 0.31)**	**35.78 (± 0.45)**	**81.13 (± 0.28)**	**41.58 (± 0.36)**	N/A
PT based	HeteroFL	63.90 (± 2.74)	52.38 (± 0.80)	73.19 (± 1.71)	57.44 (± 0.42)	27.21 (± 0.22)
	Federated Dropout	46.64 (± 3.05)	45.07 (± 0.07)	76.20 (± 2.53)	46.40 (± 0.21)	23.46 (± 0.12)
	`FedRolex`	**69.44 (± 1.50)**	**56.57 (± 0.15)**	**84.45 (± 0.36)**	**58.73 (± 0.33)**	**29.22 (± 0.24)**
	Homogeneous (Smallest)	38.82 (± 0.88)	12.69 (± 0.50)	46.86 (± 0.54)	19.70 (± 0.34)	27.32 (± 0.12)
	Homogeneous (Largest)	**75.74 (± 0.42)**	**60.89 (± 0.60)**	**84.48 (± 0.58)**	**62.51 (± 0.20)**	**29.79 (± 0.32)**

($\beta = 0.0625$) representing the upper and lower-bound performance, respectively. We have three key observations. (1) In comparison with state-of-the-art PT-based methods, `FedRolex` consistently outperforms HeteroFL and Federated Dropout in both high and low data heterogeneity scenarios. In particular, under high data heterogeneity, Federated Dropout that uses random dropout has drastically worse performance than `FedRolex` and HeteroFL which both extract sub-models in a deterministic manner. (2) In comparison with state-of-the-art KD-based methods, `FedRolex` only performs worse than Fed-ET and FedDF on CIFAR-10 under high data heterogeneity, but outperforms all the state-of-the-art in more challenging CIFAR-100 dataset that has many more classes than CIFAR-10 under both high and low data heterogeneity scenarios. It is important to emphasize that KD-based methods leverage public data to boost their model accuracy while `FedRolex` does not. (3) In comparison with model-homogeneous cases, compared to other PT-based methods, `FedRolex` considerably reduces the gap between model-heterogeneous and model-homogeneous settings, particularly on CIFAR-10 under low data heterogeneity. Note that Fed-ET achieves a higher global model accuracy than the model-homogeneous upper bound on CIFAR-10 under high data heterogeneity, which showcases the advantage of having access to public data.

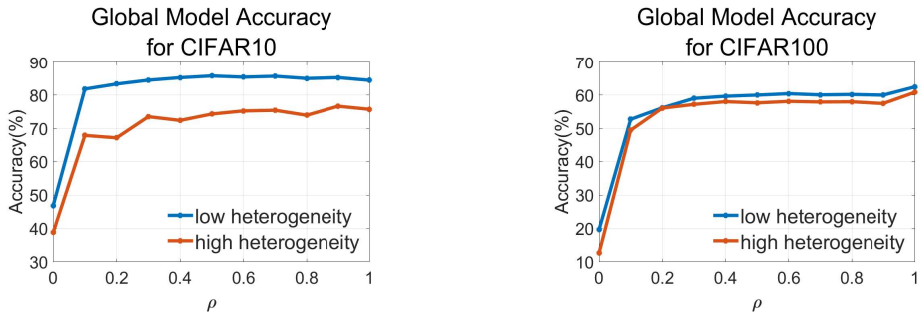
4.3 Impact of Model Capacity Distribution

In our previous experiment, in line with existing model-heterogeneous FL studies, we have performed our experiment where the distributions of model capacities are uniform for a fair comparison. In practical scenarios, however, the distributions of model capacities will likely be skewed. Therefore, we conduct an experiment to understand how global model accuracy changes when the proportion of higher capacity clients varies. To do so, we set the proportion of large to small models to ρ and the set of unique client capacities is $\beta = \{1, 1/16\}$ where $\rho = 1$ represents the edge case in which the models of all clients have the architecture of the full ResNet18 model, and $\rho = 0$ is the other edge case in which all clients have the smallest model.

Figure 2 shows how global model accuracy changes when ρ varies from 0 to 1 for CIFAR-10 and CIFAR-100 under both high and low data heterogeneity. We have three key observations. (1) For CIFAR-10, there is a large gap in global model accuracy between high and low data heterogeneity for a wide range of ρ (from 0.1 to 1). This is because CIFAR-10 is a relatively simple task and hence the global model accuracy is bottlenecked by the level of data heterogeneity instead of model capacity. This result indicates that having more high-capacity models in the cohort has limited contribution to global model accuracy. (2) For the more challenging CIFAR-100, the gap in global model accuracy is much lower between high and low data heterogeneity. In contrast to CIFAR-10, the global model accuracy is bottlenecked by the highest capacity of the models rather than the level of data heterogeneity. (3) For both CIFAR-10 and CIFAR-100, we observe that having a small fraction of high-capacity models is enough to significantly boost the global model accuracy.

4.4 Scalability

One limitation of HeteroFL and FjORD is that the server model size is restricted to the highest-capacity client model. In contrast, `FedRolex` is flexible to train a global model that can be much

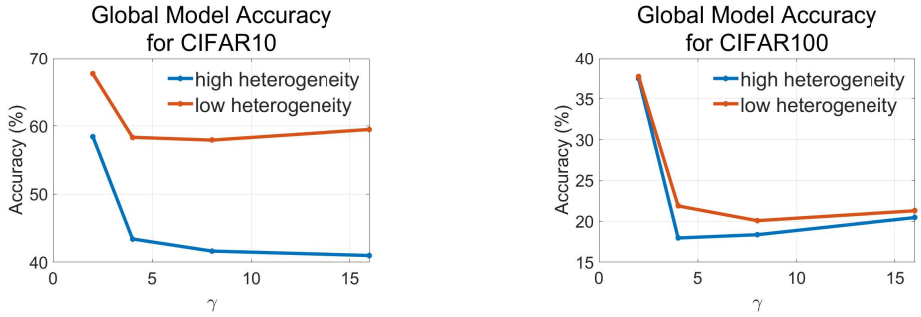


(i) (ii)

Figure 2: Impact of model capacity distribution on global model accuracy under low and high data heterogeneity for (i) CIFAR-10 and (ii) CIFAR-100.

larger than the largest client model. This is more akin to real-world settings where the server has a much higher capacity.

We consider the case where the global model size is γ times the size of the highest capacity client model. Figure 3 shows the global model accuracy of $\boldsymbol{\gamma} = \{2, 4, 8, 16\}$ for CIFAR-10 and CIFAR-100 under high and low data heterogeneity. As shown, for both CIFAR-10 and CIFAR-100, the global model accuracy drastically drops when γ increases from 2 to 4 but stay relatively stable when γ increases from 4 to 8 and from 8 to 16. This result indicates that there is a minimum client model capacity below which the server model cannot be trained effectively.



(i) (ii)

Figure 3: The global model accuracy when the server model is γ times the size of the largest client model under low and high data heterogeneity for (i) CIFAR-10 and (ii) CIFAR-100.

4.5 Enhance Inclusiveness of FL in Real-world Distribution

A primary goal of our method is to enhance the inclusiveness of FL. If the highest model capacity is used for deployment, a lot of data from the low-end devices will be unused. The exclusion of low-end devices limits the usefulness of the global model for these low-end devices. For example, low-end devices may have more grainy, low-resolution photos which will not be used in training, and so, the server model cannot make accurate predictions on such images and hence limiting its usefulness for these devices. If we consider homogeneous FL where we want to include all the client data in training, the smallest model would need to be used. The accuracy of the model is then limited by the lowest capacity client.

In this experiment, we show that `FedRolex` is a valid solution that enhances the inclusiveness of federated learning. To illustrate our point, we conducted an experiment that uses real-world household income distribution to emulate real-world device distribution. Specifically, we retrieve household income distribution information from Bureau (2021). We map $\beta_n = 1/16$ with the income group with earning less than $\$75000$ and assign proportions of remaining groups in $\$25000$ increments with

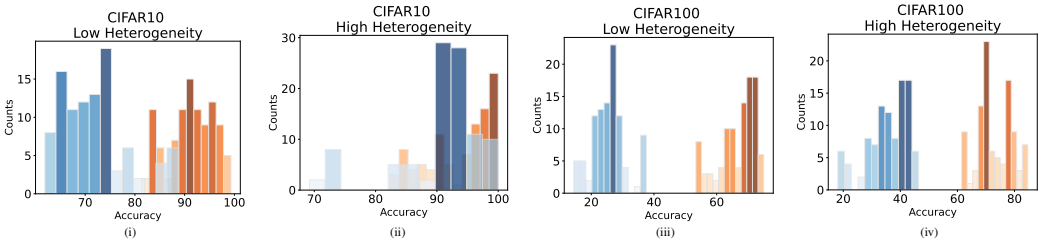


Figure 4: The local model accuracy distribution of `FedRolex` (in orange color) and the smallest homogeneous case (in blue color) for CIFAR-10 and CIFAR-100 under low and high data heterogeneity.

increasing values of β_n in the statistic. Detailed mapping of this distribution to the corresponding income distribution is given in Appendix A.6.

Table 3: Performance of FedRolex under Real-World Distribution.

Dataset	Method	High Heterogeneity		Low Heterogeneity	
		Local Accuracy	Global Accuracy	Local Accuracy	Global Accuracy
CIFAR-10	Homogeneous (smallest)	85.90 (± 0.46)	38.82 (± 0.88)	66.02 (± 0.52)	46.86 (± 0.54)
	Homogeneous (largest)	95.54 (± 0.26)	75.74 (± 0.41)	93.54 (± 0.44)	84.48 (± 0.58)
	`FedRolex`	94.05 (± 1.01)	63.17 (± 1.45)	91.03 (± 0.36)	80.14 ± 0.52)
CIFAR-100	Homogeneous(smallest)	34.51 (± 0.56)	12.69 (± 0.50)	33.22 (± 0.10)	19.70 (± 0.34)
	Homogeneous(largest)	81.99 (± 0.78)	60.89 (± 0.60)	76.43 (± 0.54)	62.51 (± 0.20)
	`FedRolex`	73.33 (± 0.96)	45.78 (± 1.71)	66.31 (± 0.34)	48.44 (± 0.51)
StackOverflow	Homogeneous(smallest)	27.32 (± 0.12)			
	Homogeneous(largest)	29.79 (± 0.32)			
	`FedRolex`	29.55 (± 0.41)			

Table 3 shows the global and local model accuracy of `FedRolex` on this emulated setting compared to two model-homogeneous cases where all clients have the smallest and largest model capacities, representing lower and upper bound on accuracy respectively. We make two key observations. (1) `FedRolex` consistently and significantly outperforms the lower bound model-homogeneous case in terms of local model accuracy for CIFAR-10 and CIFAR-100 under both high and low data heterogeneity. This result indicates that `FedRolex` can effectively boost the local model accuracy of low-end devices, which would otherwise not benefit from FL. (2) `FedRolex` achieves a local model accuracy that is close to the upper bound model-homogeneous case, especially for CIFAR-10 under both high and low data heterogeneity. This result indicates that `FedRolex` can significantly reduce the gap with the best-performing scenario without being constrained to only using high-end devices.

Finally, Figure 4 provides a detailed illustration of how the local model accuracy shifts for CIFAR-10 and CIFAR-100 under low and high data heterogeneity when `FedRolex` is used compared to the smallest homogeneous case with the same client outreach. As shown, the local model accuracy of individual clients is significantly boosted by `FedRolex`.

5 Conclusion and Future Directions

In this work, we present a partial training (PT)-based model-heterogeneous method named `FedRolex` that tackles the fundamental issue of existing PT-based methods. At the core of `FedRolex` is a simple yet effective rolling sub-model extraction scheme that enables parameters of the global model to be evenly trained to minimize client drift. Our experimental results show that `FedRolex` consistently outperforms state-of-the-art PT-based methods in both high and low data heterogeneity scenarios. We demonstrate its effectiveness in real-world scenarios by showing its performance on a more practical model capacity distribution and show `FedRolex` contributes to making FL more inclusive. In the meantime, we note that although we provide a statistical analysis of `FedRolex`, we do not derive a rigorous convergence analysis. This is not a trivial task and therefore left for future work. In addition, our goal in this paper is to train a large global model using a federation of clients with heterogeneous on-device resources. Determining what models to deploy onto each heterogeneous client after the global model has finished training during the federated learning process is a separate task. This can be achieved by a few techniques such as compressing/quantizing/KD the global model into smaller models to fit the resources of the target client. However, this is not the focus of this work and is a separate study in itself. We will pursue it in our future work as well.

References

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Malhar S Jere, Tyler Farnan, and Farinaz Koushanfar. A taxonomy of attacks on federated learning. *IEEE Security & Privacy*, 19(2):20–28, 2020.

Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE, 2019.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33: 14068–14080, 2020.

Sijie Cheng, Jingwen Wu, Yanghua Xiao, and Yang Liu. Fedgems: Federated learning of larger server models via selective knowledge fusion. *arXiv preprint arXiv:2110.11027*, 2021.

Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. *arXiv preprint arXiv:2204.12703*, 2022.

Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34, 2021.

Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

tff.simulation.datasets.stackoverflow.load_data | TensorFlow Federated. URL https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data.

Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *arXiv preprint arXiv:2008.06180*, 2020.

Yae Jee Cho, Jianyu Wang, Tarun Chiruvolu, and Gauri Joshi. Personalized federated learning for heterogeneous clients with clustered knowledge transfer. *arXiv preprint arXiv:2109.08119*, 2021.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5191–5198, 2020.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*, 2019.

Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Dhruv Guliani, Lillian Zhou, Changwan Ryu, Tien-Ju Yang, Harry Zhang, Yonghui Xiao, Françoise Beaufays, and Giovanni Motta. Enabling on-device training of speech recognition models with federated dropout. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8757–8761. IEEE, 2022.

Jae Hun Ro, Theresa Breiner, Lara McConnaughey, Mingqing Chen, Ananda Theertha Suresh, Shankar Kumar, and Rajiv Mathews. Scaling language model size in cross-device federated learning. *arXiv preprint arXiv:2204.09715*, 2022.

Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. *CoRR*, abs/2011.04050, 2020. URL https://arxiv.org/abs/2011.04050.

Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *European conference on computer vision*, pages 566–583. Springer, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Mathieu Andreux, Jean Ogier du Terrail, Constance Beguier, and Eric W Tramel. Siloed federated learning for multi-centric histopathology datasets. In *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, pages 129–139. Springer, 2020.

U.S. Census Bureau. Percentage distribution of household income in the u.s. in 2020. In Statista, September 2021. Retrieved May 18, 2022, from https://www.statista.com/statistics/203183/percentage-distribution-of-household-income-in-the-us.

Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, San Diego, CA, USA, 11th edition, 2014.

Donald J Newman. The double dixie cup problem. *The American Mathematical Monthly*, 67(1): 58–61, 1960.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Our work' claims are verified through experiment in Section 4.2 and statistical study in Appendix A.1
   (b) Did you describe the limitations of your work? [Yes] The limitations are mentioned in Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We do not show any rigorous theoretical results nor claim as such
   (b) Did you include complete proofs of all theoretical results? [N/A] We do not show any rigorous theoretical results nor claim as such
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is made available to reviewers. Data is open source. Instructions to run are available in code
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All major training details are mentioned in Section 4 and fine details provided in Appendix A.6
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All experiments were performed with 5 different seeds and the mean and standard deviations were reported when applicable.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]