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

Samiul Alam[1,2], **Luyang Liu**[3], **Ming Yan**[4,1], **Mi Zhang**[2,1]
[1]Michigan State University, [2]The Ohio State University,
[3]Google Research, [4]The Chinese University of Hong Kong, Shenzhen
alamsami@msu.edu, luyangliu@google.com, yanming@cuhk.edu.cn, mizhang.1@osu.edu

Abstract

Most cross-device federated learning (FL) studies focus on the model-homogeneous setting where the global server model and local client models are identical. However, such constraint not only excludes low-end clients who would otherwise make unique contributions to model training but also restrains clients from training large models due to on-device resource bottlenecks. In this work, we propose `FedRolex`, a partial training (PT)-based approach that enables model-heterogeneous FL and is able to train a global server model larger than the largest client model. At its core, `FedRolex` employs a rolling sub-model extraction scheme that allows different parts of the global server model to be evenly trained, which mitigates the client drift induced by the inconsistency between individual client models and server model architectures. Empirically, we show that `FedRolex` outperforms state-of-the-art PT-based model-heterogeneous FL methods (e.g. Federated Dropout) and reduces the gap between model-heterogeneous and model-homogeneous FL, especially under the large-model large-dataset regime. In addition, we provide theoretical statistical analysis on its advantage over Federated Dropout. Lastly, we evaluate `FedRolex` on an emulated real-world device distribution to show that `FedRolex` can enhance the inclusiveness of FL and boost the performance of low-end devices that would otherwise not benefit from FL. Our code is available at: https://github.com/MSU-MLSys-Lab/FedRolex.

1 Introduction

Federated learning (FL) is a machine learning paradigm that trains models from distributed clients with private data under the coordination of a central server [1, 2]. In this work, we focus on cross-device FL where clients are usually resource-constrained edge devices. The majority of existing cross-device FL studies focus on the *model-homogeneous* setting [3–6], in which the server model and the client models across all the participating client devices are *identical*. However, model-homogeneous FL are confronted with two fundamental constraints: (1) **device heterogeneity** is a more realistic consideration when deploying FL systems in real-world applications: different client devices could have very diverse on-device resources and are only capable of training models with capacities that match their on-device resources. Having the same model on all the devices would, unfortunately, exclude clients with low-end devices who would otherwise make unique contributions to the model training from their local data; (2) state-of-the-art machine learning has moved towards **large models** [7] such as Transformer [8]. Restricting server and client models to be the same inevitably causes model-homogeneous FL to fail to train such large models due to the resource constraint of client devices.

To relax the fundamental constraints of model-homogeneous FL, *model-heterogeneous* FL was proposed where heterogeneous models with different capacities across the server and the clients are trained during the federated training process. One primary challenge in model-heterogeneous FL is

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

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

	Model Heterogeneity	Aggregation Scheme	Sub-model Extraction Scheme	Need of Public Data	Server Model Size	Compatibility with Secure Aggregation
FedAvg [3]	No	-	-	No	= Client Model	Yes
FedProx [4]				No	= Client Model	Yes
SCAFFOLD [5]				No	= Client Model	Yes
FedBE [6]				Unlabeled	= Client Model	No
FedGKT [9]	No	Knowledge Distillation	-	No	$\geq$ Largest Client Model	No
FedDF [10]				Unlabeled	= Largest Client Model	No
DS-FL [11]				Unlabeled	= Largest Client Model	No
Fed-ET [12]				Unlabeled	$\geq$ Largest Client Model	No
Federated Dropout [13]	Yes	Partial Training	Random	No	$\geq$ Largest Client Model	Yes
HeteroFL [14]			Static	No	= Largest Client Model	Yes
FjORD [15]			Static	No	= Largest Client Model	Yes
FedRolex (Our Approach)			**Rolling**	**No**	$\geq$ **Largest Client Model**	**Yes**

the aggregation of heterogeneous client models. To address this challenge, knowledge distillation (KD)-based approaches have been proposed [9–12], in which the client models serve as teachers and the server ensembles the knowledge distilled from the individual client models. However, KD-based approaches in general require public data on the server to achieve competitive model accuracy, whereas the desired public data may not be always available in practice. Moreover, since KD-based approaches need the individual client models (whole models or prediction layers) or their outputs to be sent to the server, they are incompatible with secure aggregation protocols [16], which limits their privacy guarantee. To remove the dependency on public data and ensure compatibility with secure aggregation, partial training (PT)-based approaches such as random sub-model extraction (Federated Dropout [13]) and static sub-model extraction (HeteroFL [14], FjORD [15]) were proposed. In these approaches, each client trains a smaller sub-model extracted from the larger global server model, and the server model is updated by aggregating those trained sub-models. However, the fundamental issue of existing PT-based methods is that the sub-models are extracted in ways (either random or static) such that the parameters of the global server model are *not evenly trained*. This makes the server model vulnerable to client drift[1] induced by the *inconsistency between individual client model and server model architectures* – a unique challenge of model-heterogeneous FL.

To tackle the fundamental issue of existing PT-based methods, in this work, we propose a simple yet effective PT-based model-heterogeneous FL approach named `FedRolex`. The key difference between `FedRolex` and existing PT-based methods is how the sub-models are extracted for each client over communication rounds in the federated training process. Specifically, instead of extracting sub-models in either random or static manner, `FedRolex` proposes a **rolling sub-model extraction** scheme, where the sub-model is extracted from the global server model using a rolling window that advances in each communication round. Since the window is rolling, sub-models from different parts of the global model are extracted in sequence in different rounds. As a result, all the parameters of the global server model are *evenly trained* over the local data of client devices.

The proposed rolling sub-model extraction scheme, though simple, has equipped `FedRolex` with multifold merits compared to prior arts (Table 1): (1) `FedRolex` enables different parts of the global server model to be evenly trained, which mitigates the client drift induced by model heterogeneity. (2) Contrary to static sub-model extraction approaches (HeteroFL, FjORD), `FedRolex` is able to train a global server model that is *larger* than the largest client model, enabling FL to benefit from the superior performance brought by large models. It echos some concurrent efforts in developing FL primitives to support training large server models in cross-device setting [17]. (3) Compared to random sub-model extraction (Federated Dropout), as we show in our theoretical statistical analysis in Section 3 and Appendix A.1, the global server model is trained more evenly by `FedRolex` as the expected number of rounds for `FedRolex` going through all the parameters of the global model for at least certain times is smaller than that of Federated Dropout. (4) `FedRolex` only needs to transmit the sub-model that is needed by a given client instead of the full server model to the client. This allows clients to contribute to federated training under resource constraints and reduces communication overheads. (5) Lastly, `FedRolex` is fully compatible with existing secure aggregation protocols that enhance the privacy properties of FL systems.

We evaluate the performance of `FedRolex` under two regimes: i) small-model small-dataset regime (most existing cross-device FL studies use this combination), and ii) large-model large-dataset regime

[1] In model-homogeneous FL, client drift is primarily induced by *data heterogeneity* across clients. In model-heterogeneous FL, *model heterogeneity* across clients is *another* critical source that induces client drift.

(this combination echos recent efforts on pushing the frontier of cross-device FL towards training large server models on large-scale datasets [18–21]). We highlight five of our findings: (1) `FedRolex` consistently outperforms state-of-the-art PT-based model-heterogeneous FL methods under both small-model small-dataset and large-model large-dataset regimes (§4.1). (2) `FedRolex` reduces the gap between model-heterogeneous and model-homogeneous FL, especially under large-model large-dataset regime (§4.2). (3) With `FedRolex`, under both regimes, having a small fraction of large-capacity models could significantly boost the global model accuracy (§4.3). (4) `FedRolex` is able to train a global server model that is larger than the largest client model and outperforms Federated Dropout in terms of global model accuracy (§4.4). (5) Using an emulated real-world device distribution, we show that `FedRolex` enhances the **inclusiveness** of FL and boosts the performance of low-end devices that would otherwise not benefit from FL (§4.5).

2 Related Work

Knowledge Distillation (KD)-based Model-Heterogeneous FL. One primary approach for model-heterogeneous FL in cross-device settings is based on knowledge distillation (KD) [22]. In particular, FedDF [10] distills knowledge from a set of classifiers trained with private data from a federation of client devices. The logit outputs of each classifier against an unlabeled public dataset are then used to train a student model at the server with KD. Similarly, DS-FL [11] utilized an unlabeled public dataset at the server, and proposed a distillation-based semi-supervised FL approach to enhance performance by pseudo-labeling the public data. FedGKT [9] proposed group knowledge transfer in which knowledge is transferred from a large model in the server to clients without public data. Fed-ET [12] proposed a weighted consensus distillation scheme with diversity regularization that enables the training of a large server model with smaller client models. KD-based approaches, however, have several limitations: these often require public data to achieve competitive model accuracy. This is because model accuracy is dependent on the size of public data as well as the domain similarity of public data with client data [10, 12, 23]. Furthermore, as KD-based approaches use client model weights partially or entirely as teachers to transfer knowledge to the server, they are incompatible with secure aggregation protocols, making them vulnerable to backdoor attacks [20].

Partial Training (PT)-based Model-Heterogeneous FL. To address the limitations of KD-based approaches, partial training (PT) has emerged as another solution for model-heterogeneous FL. Depending on how the sub-models are extracted from the global server model, existing PT-based methods can be in general categorized into two groups: *static* sub-model extraction and *random* sub-model extraction. HeteroFL [14] and FjORD [15] proposed static sub-model extraction schemes where sub-models are *always* extracted from a *designated* part of the global server model. However, such a static extraction strategy has two key drawbacks. First, the global server model is restricted to the *same* size as the largest client model. As such, the size and capability of the global model are implicitly restricted by the resources of client devices, making it not able to train large models due to resource bottlenecks at client devices. Second and more importantly, under static extraction, depending on their resource demands, different sub-models can *only* be trained on clients whose on-device resources are matched. As a consequence, part of the global server model cannot be trained on data at low-end client devices, causing different parts of the global model to be trained on data with different distributions. This would significantly degrade the performance of the global model, especially under high data heterogeneity. In contrast, inspired by the dropout technique commonly used in centralized training [24], Federated Dropout [13] proposed a random sub-model extraction method. Although avoiding the drawbacks of the static sub-model extraction strategy, as reported in [25], it becomes less effective when the data heterogeneity is high and the client cohort is small due to its randomness in selecting sub-models. In this work, we propose a rolling sub-model extraction scheme that tackles the issues of both static and random sub-model extraction methods.

3 Methodology

3.1 Formulation of Model-Heterogeneous FL

Let $\mathcal{N}$ denote N client devices with non-IID (non-identically and independently distributed) local data $D = \{D_1, D_2, ..., D_N\}$. Model-homogeneous FL trains a global model of parameter θ by solving the following optimization problem:

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

with

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

where $D_n \triangleq \{d_{n,1}, d_{n,2}, d_{n,3} ... d_{n,m_n}\}$ is the set of the nth client's local data samples and p_n is its corresponding weight such that $p_n \geq 0$ and $\sum_{n=1}^{N} p_n = 1$.

In comparison, in model-heterogeneous FL, clients train local models with heterogeneous capacities $\{C_1, C_2, ..., C_N\}$, and the local objective function at the nth client becomes

$$F'_n(\theta_n) \triangleq \frac{1}{m_n} \sum_{k=1}^{m_n} l(\theta_n; d_{n,k}). \tag{3}$$

The size of θ_n depends on the client's model capacity C_n, and the parameter θ_n is extracted by selecting a sub-model from the global model θ, which can change from one iteration to another. If θ_n changes, the objective function also changes. For simplicity, we use the same notation l for the loss function at all clients and iterations, though they differ between clients and iterations. The key of model-heterogeneous FL is selecting θ_n from the global model θ given model capacity C_n.

3.2 FedRolex: Model-Heterogeneous FL with Rolling Sub-Model Extraction

As a partial training (PT)-based approach, `FedRolex` trains only a sub-model extracted from the global server model at one client and sends back the corresponding sub-model updates to the server for aggregation. Figure 1 illustrates three rounds of federated training of `FedRolex` on two heterogeneous clients. During each iteration, the server selects a subset of the trainable variables from the global model $\theta^{(j)}$ and broadcasts the selected sub-model to each participating client. The clients train the received sub-models on their private data and post their updates $\theta_m^{(j)}$ to the server. The server then aggregates the variables from the updates and formulates the new global model $\theta^{(j+1)}$ for the next round.

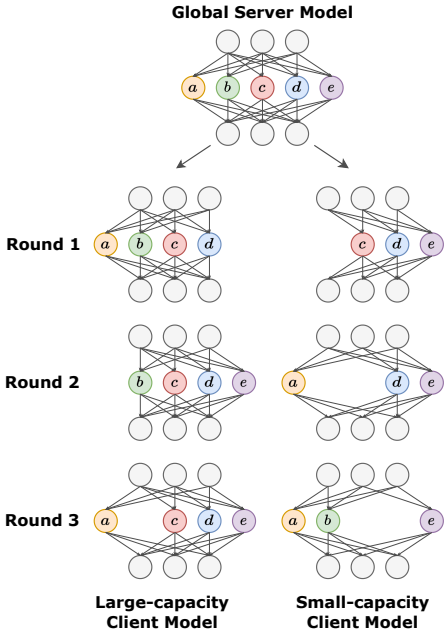


Figure 1: Overview of FedRolex.

There are two primary design decisions involved in building this partial training algorithm. (1) How to extract the sub-models from the global model? (2) How to aggregate client updates from heterogeneous devices to update the global model?

How to extract sub-models? Equation (1) shows us that we must find θ that minimizes the objective concerning all private datasets. In centralized settings, this is achieved by simply iterating over the entire data. In FL, however, we do not have this option. Instead, the parts of the global model are dynamically trained by iterating them over the entire client federation. Consider $\theta_n^{(j)}$ selected from global model weights 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 $\{\theta_{n,[i,\mathcal{S}_{i,0}^{(j)}]}^{(j)}, \theta_{n,[i,\mathcal{S}_{i,1}^{(j)}]}^{(j)}, ..., \theta_{n,[i,\mathcal{S}_{i,K_i-1}^{(j)}]}^{(j)}\}$. Intuitively, we can think of this as iterating over the model instead of the data.

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

$$\mathcal{S} = \begin{cases} \{\hat{j}, \hat{j}+1, \ldots, \hat{j} + \lfloor \beta_n K_i \rfloor - 1\} & \text{if } \hat{j} + \lfloor \beta_n K_i \rfloor \leq K_i, \\ \{\hat{j}, \hat{j}+1, \ldots, K_i - 1\} \cup \{0, 1, \ldots, \hat{j} + \lfloor \beta_n K_i \rfloor - 1 - K_i\} & \text{else.} \end{cases} \tag{4}$$

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 returns to index 0 again.

How to aggregate updates? In model homogeneous FL, θ_n equals θ as all devices have the same capacity. The updates are therefore straightforward to aggregate as it is a simple average of all the updates. However, in model-heterogeneous FL, this scheme will not work. Instead, we must compute the average separately for each variable based on how many participating clients in the federation updated that variable based on their data. 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 the global model and $\theta_{m,[i,k]}$ be the kth trainable weight of the ith layer of the client m. Denote $\mathcal{M}_k \subset \mathcal{M}$ as the set of clients updating the kth trainable weight, and the model parameters are aggregated as follows:

$$\theta_{[i,k]} = \frac{1}{\sum_{m \in \mathcal{M}_k} p_m} \sum_{m \in \mathcal{M}_k} p_m \theta_{m,[i,k]}, \tag{5}$$

which is supported by Federated Select [17]. The empirical weight p_m is assigned based on factors like the client model capacity, 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, $p_m = 1/N$. For different PT-based methods, how $\mathcal{S}$ is chosen varies according to their design. In our work, we propose a rolling process for computing $\mathcal{S}$ and thereby extract the sub-models. The pseudocode of `FedRolex` is summarized in Algorithm 1.

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** $J-1$ **do**
  Sample subset $\mathcal{M}$ from $\mathcal{N}$
  Broadcast $\theta^{(j)}_{m,[i,\mathcal{S}^{(j)}_{i,k}]}$ to client $m \in \mathcal{M}$
    $\forall i, \mathcal{S}$ from equation (4)
  for *each* *client* $m \in \mathcal{M}$ **do**
    `clientStep`($\theta^{(j)}_m$, D_m)
  end
  Aggregate $\theta^{(j+1)}_{[i,k]}$ according to Equation (5)
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

3.3 Comparison with Other Sub-model Extraction Schemes

In contrast to our approach, there are other proposed sub-model extraction schemes in the literature, as discussed in §2. They can primarily be grouped as static (HeteroFL, Fjord) and random (Federated Dropout). We describe these extraction approaches' primary differences with `FedRolex`. 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 client's capacity. The pseudocodes of both HeteroFL and Federated Dropout are included in Appendix A.7.

3.3.1 Comparison with 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 2. Unused nodes do not see the full data distribution. The subset of node indices is given by:

$$\mathcal{S} = \{0, 1, 2, \ldots, \lfloor \beta_n K_i \rfloor - 1\}. \tag{6}$$

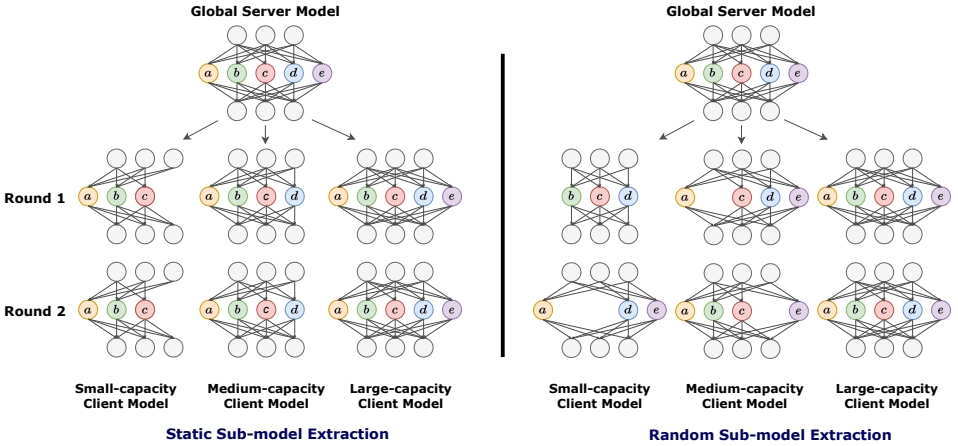


Figure 2: Illustration of how sub-models are extracted by (left) static extraction (HeteroFL) and (right) random extraction (Federated Dropout).

Limitations of Static Sub-model Extraction: 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.

3.3.2 Comparison with 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 right column of Figure 2. The subset is given by:

$$\mathcal{S} = \{k_c \mid \text{integer } k_c \in [0, K_i - 1] \text{ for } 1 \leq c \leq \lfloor \beta_n K_i \rfloor\}. \tag{7}$$

Here $\lfloor \beta_n K_i \rfloor$ nodes are randomly chosen from the global model. Unlike the static sub-model extraction, all nodes see the entire data given a sufficiently large number of communication rounds.

Limitations of Random Sub-model Extraction: Dropout-based algorithms like Federated Dropout [13] use 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, their exact frequencies are not the same in each experiment because of the randomness. This could potentially cause client drift. Given I components, if one component is chosen at each round, the expected number of rounds to go through all components at least once is $\prod_{i=1}^{I} \frac{i}{I}$, which is close to $I \ln(I)$; see Appendix A.1 for more details and the results on updating all components at least more than one time. It shows that random dropout can not balance the update frequencies of the components, and it takes too many rounds to update all the components.

4 Experiments

Datasets and Models. We evaluate the performance of `FedRolex` under two regimes. Under small-model small-dataset regime, we train pre-activated ResNet18 (PreResNet18) models [26] on CIFAR-10 and CIFAR-100 [27]. We replace the batch Normalization in PreResNet18 with static batch normalization [14, 28] and add a scalar module after each convolution layer [14]. Under large-model large-dataset regime, we use Stack Overflow [29] and followed [2] to train a modified 3-layer Transformer [8] with a vocabulary of $10,000$ words, where the dimension of token embeddings is 128, and the hidden dimension of the feed-forward network (FFN) block is 2048. We use ReLU activation and use 8 heads for the multi-head attention where each head is based on 12-dimensional (query, key, value) vectors. The statistics of the datasets are listed in Table 2.

Table 2: Dataset statistics.

Dataset	Train Clients	Train Examples	Validation Clients	Validation Examples	Test Clients	Test Examples
CIFAR-10	100	50,000	N/A	N/A	N/A	10,000
CIFAR-100	100	50,000	N/A	N/A	N/A	10,000
Stack Overflow	342,477	135,818,730	38,758	16,491,230	204,088	16,586,035

Data Heterogeneity. For CIFAR-10 and CIFAR-100, we followed HeteroFL [14] to model non-IID distributions by restricting each client to have L labels. In our evaluation, we consider two levels of data heterogeneity. For CIFAR-10, we define $L = 2$ as high data heterogeneity and $L = 5$ as low data heterogeneity. For CIFAR-100, we use $L = 20$ as high data heterogeneity and $L = 50$ as low data heterogeneity. These two levels roughly correspond to Dirichlet distribution $Dir_K(\alpha)$ with α equal to 0.1 and 0.5, respectively. For Stack Overflow, the dataset is partitioned over user IDs, making the dataset naturally non-IID distributed.

Model Heterogeneity. Without loss of generality, in our evaluation, we consider five different client model capacities $\boldsymbol{\beta} = \{1, 1/2, 1/4, 1/8, 1/16\}$ where for instance, $1/2$ means the client model capacity is half of the largest client model capacity (full model). To generate these client models, for ResNet18, we vary the number of kernels in convolution layers and keep the nodes in the output layers the same. For Transformer, we vary the number of nodes in the hidden layer of the attention heads.

Baselines. We compare `FedRolex` against both state-of-the-art PT-based model-heterogeneous FL methods including Federated Dropout [13] and HeteroFL [14][2] as well as state-of-the-art KD-based model-heterogeneous FL methods including FedDF [10], DS-FL [11] and Fed-ET [12] [3]. To ensure fair comparison, all the PT-based baselines are trained using the same learning rate, number of communication rounds, and multi-step learning rate decay schedule. The details of the schedule for each dataset and experiment are described in Appendix A.6.

Configurations and Platform. For CIFAR-10 and CIFAR-100, we apply bounding box crop [30] to augment the images. In each communication round, 10% of the clients are randomly selected from a pool of 100 clients. For Stack Overflow, we followed [2] to use a 10% dropout rate to prevent overfitting, and 200 clients are randomly selected from a pool of $342,477$ clients in each communication round. The details of the hyper-parameters for model training are included in Appendix A.6. We implemented `FedRolex` and PT-based baselines using PyTorch [31] and Ray [32], and conducted our experiments on 8 Nvidia A6000 GPUs.

Evaluation Metrics. We use global and local model accuracy as our evaluation metrics. Specifically, global model accuracy is defined as the server model accuracy on the test set; and local model accuracy is defined as the accuracy of the server model on each of the client local datasets. For CIFAR-10 and CIFAR-100, we report the classification accuracy. For Stack Overflow, we report the next word prediction accuracy which includes both out-of-vocabulary (oov) and end-of-sentence (eos) tokens. We run our experiments using five different seeds for CIFAR-10 and CIFAR-100 and using three different seeds for Stack Overflow.

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

First, we compare the performance of `FedRolex` with state-of-the-art PT and KD-based model-heterogeneous FL methods. For fair comparison, we followed the experimental settings used in prior arts where the distributions of client model capacities are uniform and the global server model is the same as the largest client model.

Evaluation Results: Table 3 summarizes our results. We have two observations. (1) In comparison with state-of-the-art PT-based methods, under small-model small-dataset regime, `FedRolex` consistently outperforms HeteroFL and Federated Dropout under both low and more challenging high data heterogeneity scenarios. In particular, under high data heterogeneity, Federated Dropout which extracts sub-model randomly has worse performance than `FedRolex` and HeteroFL which both extract sub-models in a deterministic manner. Under large-model large-dataset regime, `FedRolex` also outperforms both HeteroFL and Federated Dropout. These results together demonstrate the

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

[3]We did not compare with FedGKT [9] as it is only compatible with CNN models.

Table 3: Global model accuracy comparison between `FedRolex`, PT and KD-based model-heterogeneous FL methods, and model-homogeneous FL methods. Note that the results of KD-based methods were obtained from [12]. For Stack Overflow, since KD-based methods cannot be directly used for language modeling task, their results are marked as N/A.

	Method	**High Data Heterogeneity**		**Low Data Heterogeneity**		**Stack Overflow**
		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)**

superiority of `FedRolex` under both regimes. (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 KD-based methods on the more challenging CIFAR-100 which has a larger number classes than CIFAR-10 under both low and high data heterogeneity scenarios. It is important to note that KD-based methods leverage public data to boost their model accuracy while `FedRolex` does not.

4.2 Performance Comparison with Model-Homogeneous FL Methods

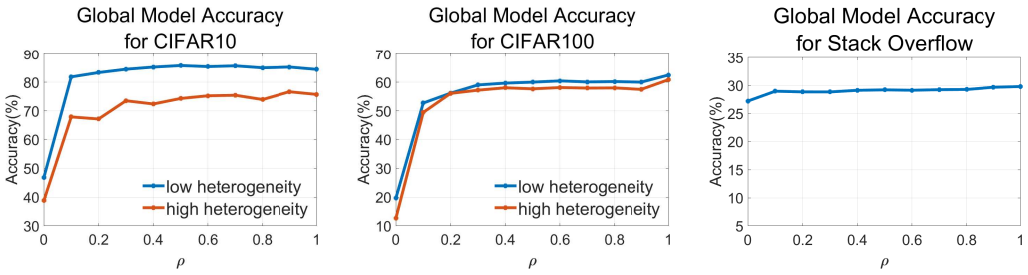
We also compare the global model accuracy of `FedRolex` with two model-homogeneous cases where all the clients have the largest capacity model ($\beta = \{1\}$) and the smallest capacity model ($\beta = \{1/16\}$), representing the upper and lower-bound performance, respectively.

Evaluation Results: As listed in Table 3, compared with other PT-based methods, `FedRolex` reduces the gap in global model accuracy between model-heterogeneous and upper-bound model-homogeneous settings. In particular, `FedRolex` is on par with the upper-bound model-homogeneous case for Stack Overflow, whereas both HeteroFL and Federated Dropout perform even worse than the model homogeneous case using the smallest model. This result indicates that with `FedRolex`, we will not be constrained to only using high-end devices to achieve competitive global model accuracy. 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 using public data.

4.3 Impact of Client Model Heterogeneity Distribution

In our previous experiments, the distributions of model capacities across client devices are set to be uniform. In this experiment, we aim to understand the impact of the client model heterogeneity distribution. To do so, without loss of generality, we use two client model capacities $\beta = \{1, 1/16\}$ and vary the distribution ratio between the two (denoted as ρ) where $\rho = 1$ represents the case in which all the clients have the largest capacity model ($\beta = \{1\}$) and $\rho = 0$ represents the case in which all the clients have the smallest capacity model ($\beta = \{1/16\}$).

Evaluation Results: Figure 3 shows how the global model accuracy changes when ρ varies from 0 to 1 for CIFAR-10, CIFAR-100 and Stack Overflow. We have three observations. (1) For CIFAR-10 (Figure 3(i)), 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 (Figure 3(ii)), 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 regimes (Figure 3(i)(ii) vs. Figure 3(iii)), we observe that having a small fraction of large-capacity models significantly boosts the global model accuracy, but keeping increasing the ratio of large-capacity models has limited contribution to global model accuracy.



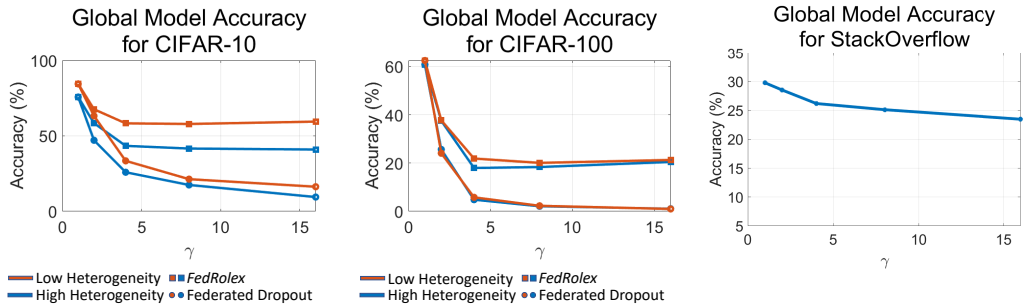
(i) (ii) (iii)

Figure 3: Impact of client model heterogeneity distribution on global model accuracy for (i) CIFAR-10, (ii) CIFAR-100, and (iii) Stack Overflow.

4.4 Performance on Training Larger Server Model

Similar to Federated Dropout, one advantage of `FedRolex` over static sub-model extraction methods (HeteroFL and FjORD) is that `FedRolex` is able to train a global server model that is larger than the largest client model. In this experiment, we aim to evaluate the performance of `FedRolex` on training larger server models. To do so, we consider the case where the size of the global server model is $\gamma = \{2, 4, 8, 16\}$ times the size of client models. For simplicity, all client models have the same size.

Evaluation Results: Figure 4(i) and Figure 4(ii) compare `FedRolex` with Federated Dropout in terms of global model accuracy when γ for CIFAR-10 and CIFAR-100, respectively. As shown, although the global model accuracy drops for both `FedRolex` and Federated Dropout when γ increases, especially from 1 to 4, `FedRolex` consistently achieves higher global model accuracy than Federated Dropout across $\gamma = \{2, 4, 8, 16\}$ under both low and high data heterogeneity. For Stack Overflow (Figure 4(iii)), the global model accuracy has a much smaller drop when γ increases. This demonstrates the superiority of using large models on large-scale datasets for training larger server models.



(i) (ii) (iii)

Figure 4: Performance on training larger server model when the server model is γ times the size of the client model for (i) CIFAR-10, (ii) CIFAR-100, and (iii) Stack Overflow.

4.5 Enhance Inclusiveness of FL in Real-world Distribution

A primary vision of `FedRolex` is to enhance the inclusiveness of FL. To demonstrate this, in this experiment, we use real-world household income distribution to emulate real-world device distribution. Specifically, we retrieve household income distribution information from Bureau [33]. We map $\beta_n = 1/16$ with the income group with earning less than $\$75,000$ and assign proportions of remaining groups in $\$25,000$ increments with increasing values of β_n. Detailed mapping of this distribution to the corresponding income distribution is provided in Figure 7 in Appendix A.6.

Evaluation Results: Table 4 shows both the global and local model accuracies of `FedRolex` for CIFAR-10 and CIFAR-100 as well as the global model accuracy on Stack Overflow under the emulated real-world device distribution. Again, we compare with two model-homogeneous cases where all clients have the smallest and largest model capacities, representing lower and upper-bound

Table 4: Performance of FedRolex under emulated real-world device distribution.

Dataset	Method	High Data Heterogeneity		Low Data 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)
Stack Overflow	Homogeneous (smallest)	27.32 (± 0.12)			
	Homogeneous (largest)	29.79 (± 0.32)			
	FedRolex	29.55 (± 0.41)			

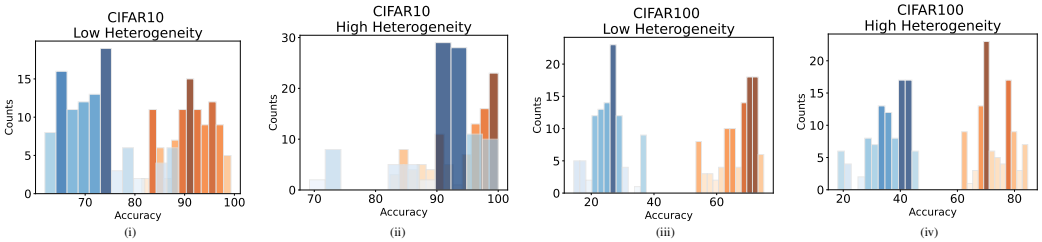


Figure 5: Local model accuracy distribution of FedRolex (orange color) vs. the smallest model-homogeneous case (blue color) for CIFAR-10 and CIFAR-100 under low and high data heterogeneity.

accuracy, respectively. We make two observations. (1) Looking at the global model accuracy, FedRolex consistently outperforms the lower-bound model-homogeneous case across CIFAR-10, CIFAR-100, and Stack Overflow. This result indicates that FedRolex enhances the inclusiveness of FL and improves the accuracy of the global model, which would otherwise not be able to achieve. (2) Looking at the local model accuracy, FedRolex significantly outperforms the lower-bound model-homogeneous case on CIFAR-10 and CIFAR-100 under both low and high data heterogeneity. This result indicates that FedRolex effectively boosts the performance of low-end devices, which would otherwise not benefit from FL. A detailed illustration on how local model accuracy distribution of individual clients shifts when FedRolex is used compared to the smallest model-homogeneous case with the same client outreach is shown in Figure 5.

5 Conclusion

We presented FedRolex, a partial training (PT)-based model-heterogeneous FL approach that is able to train a global server model larger than the largest client model. FedRolex proposed a rolling sub-model extraction scheme that enables parameters of the global server model to be evenly trained to mitigate client drift induced by model heterogeneity. We provided a theoretical statistical analysis on its advantage over Federated Dropout. Our experimental results show that FedRolex consistently outperforms state-of-the-art PT-based methods across models and datasets at both small and large scales. Moreover, we demonstrated its performance on an emulated real-world device distribution and show FedRolex contributes to making FL more inclusive.

Limitations and Future works. In this work, we provided a statistical analysis of FedRolex. A full convergence analysis of FedRolex is not trivial and left for future work. In addition, the goal of this work is to train a global server model using a federation of heterogeneous client models. Determining what models to deploy onto each client after the global server model is trained is a separate task, especially when the global server model is large. We will pursue it as our future work.

6 Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. We also want to thank Jiahui Yu, Zheng Xu, and Chaoyang He for their constructive discussions. This work was partially supported by NSF Awards PFI:BIC-1632051, CNS-1814551, DMS-2012439, and a Google Computing Platform (GCP) credit grant.

References

[1] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[2] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Blaise Aguera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, Suhas Diggavi, Hubert Eichner, Advait Gadhikar, Zachary Garrett, Antonious M. Girgis, Filip Hanzely, Andrew Hard, Chaoyang He, Samuel Horvath, Zhouyuan Huo, Alex Ingerman, Martin Jaggi, Tara Javidi, Peter Kairouz, Satyen Kale, Sai Praneeth Karimireddy, Jakub Konecny, Sanmi Koyejo, Tian Li, Luyang Liu, Mehryar Mohri, Hang Qi, Sashank J. Reddi, Peter Richtarik, Karan Singhal, Virginia Smith, Mahdi Soltanolkotabi, Weikang Song, Ananda Theertha Suresh, Sebastian U. Stich, Ameet Talwalkar, Hongyi Wang, Blake Woodworth, Shanshan Wu, Felix X. Yu, Honglin Yuan, Manzil Zaheer, Mi Zhang, Tong Zhang, Chunxiang Zheng, Chen Zhu, and Wennan Zhu. A field guide to federated optimization, 2021.

[3] 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.

[4] 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.

[5] 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.

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

[7] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[9] 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.

[10] 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.

[11] 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.

[12] Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[13] 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.

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

[15] 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.

[16] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[17] Zachary Charles, Kallista Bonawitz, Stanislav Chiknavaryan, Brendan McMahan, et al. Federated select: A primitive for communication-and memory-efficient federated learning. *arXiv preprint arXiv:2208.09432*, 2022.

[18] 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.

[19] Zachary Charles, Kallista Bonawitz, Stanislav Chiknavaryan, Brendan McMahan, and Blaise Agüera y Arcas. Federated select: A primitive for communication- and memory-efficient federated learning, 2022. URL `https://arxiv.org/abs/2208.09432`.

[20] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16070–16084. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/b8ffa41d4e492f0fad2f13e29e1762eb-Paper.pdf`.

[21] Tien-Ju Yang, Dhruv Guliani, Françoise Beaufays, and Giovanni Motta. Partial variable training for efficient on-device federated learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4348–4352. IEEE, 2022.

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

[23] Samuel Don Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew Gordon Wilson. Does knowledge distillation really work? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=7J-fKoXiReA`.

[24] 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.

[25] Gary Cheng, Zachary Charles, Zachary Garrett, and Keith Rush. Does federated dropout actually work? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3387–3395, 2022.

[26] 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.

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

[28] 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.

[29] TFF. Tensorflow federated stack overflow dataset. *Online: https://www. tensorflow. org/federated/api_docs/python/tff/simulation/datasets/stackoverflow*, 2019.

[30] 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.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[32] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

[33] 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`.

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

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

[36] 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]
   (b) Did you describe the limitations of your work? [Yes]
   (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? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
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]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (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]

A Appendix

A.1 Statistical Analysis

Lemma 1. *Given I indices, and one index is chosen at each round equally randomly. The expected number of rounds of choosing all indices at least once is*

$$I\left(\frac{1}{I}+\frac{1}{I-1}+\cdots+\frac{1}{1}\right),$$

which is the same as

$$I\int_0^\infty \left(1-(1-e^{-t})^I\right)dt.$$

Proof. We denote the expected number of rounds to choose exactly i indices at least once as $E(i)$. Then we have $E(1)=1$, because, after the first round, one index is chosen. After the first round, the expected number of rounds to choose a new index is $\frac{I}{I-1}$, because one of the remaining $I-1$ out of the total I indices needs to be chosen. That is, $E(2)=E(1)+\frac{I}{I-1}$. Similarly, we have

$$E(i)=E(i-1)+\frac{I}{I+1-i}, \qquad \forall i=2,\ldots,I.$$

Thus, we have

$$E(I)=E(I-1)+I=E(I-2)+\frac{I}{2}+\frac{I}{1}=\cdots=I\left(\frac{1}{I}+\frac{1}{I-1}+\cdots+\frac{1}{1}\right).$$

The lemma is proved. □

It shows that the expected number of rounds to choose all indices at least once is $I\log(I)$ when $I\to\infty$. This proof can not be generalized to the case for choosing all indices at least m times for $m\geq 2$. Therefore, we provide alternative proof for it [34, Example 5.17].

Alternative proof of Lemma 1. This proof considers picking the indices as Poisson processes. Assume that the Poisson process to choose one index has a rate $\lambda=1$. Since the index is chosen equally randomly, choosing the jth index also follows a Poisson process with a rate $1/I$ for any j [34, Proposition 5.2]. We let X_j be the time to choose the first index j, and

$$X=\max_{1\leq j\leq I} X_j \tag{8}$$

is the time all indices are chosen at least once. Since all X_j are independent with rate $1/I$, we have

$$\begin{aligned}P\{X<t\}=&P\{\max_{1\leq j\leq I} X_j<t\}=P\{X_j<t,\text{ for } j=1,\ldots,I\}\\=&(1-e^{-t/I})^I.\end{aligned}$$

Therefore, we have

$$\mathbb{E}[X]=\int_0^\infty P\{x>t\}dt=\int_0^\infty\left(1-(1-e^{-t/I})^I\right)dt$$

We let N be the number of rounds to choose all indices at least one, and T_i be the ith interarrival time of the Poisson process for choosing one index. Then we have

$$X=\sum_{i=1}^N T_i,$$

and T_i are independent. Thus we have

$$\mathbb{E}[X|N]=N\mathbb{E}[T_i]=N,$$

and which gives

$$\mathbb{E}[X] = \mathbb{E}\{\mathbb{E}[X|N]\} = \mathbb{E}[N].$$

Thus we have

$$\mathbb{E}[N] = \int_0^\infty \left(1 - (1 - e^{-t/I})^I\right) dt = I \int_0^\infty \left(1 - (1 - e^{-t})^I\right) dt.$$

The lemma is proved. □

Next, we will present the lemma for choosing each index at least m times.

Lemma 2. *Given I indices, and one index is chosen at each round equally randomly. The expected number of rounds of choosing all indices at least m times is*

$$I \int_0^\infty \left(1 - (1 - S_m(t)e^{-t})^I\right) dt,$$

where

$$S_m(y) := 1 + y + \frac{y^2}{2!} + \cdots + \frac{y^{m-1}}{(m-1)!} = \sum_{l=0}^{m-1} \frac{y^l}{l!}. \tag{9}$$

Proof. We consider picking the indices as Poisson processes again. Assume that the Poisson process to choose one index has a rate $\lambda = 1$. Since the index is chosen equally randomly, choosing the jth index also follows a Poisson process with a rate of $1/I$ for any j. We let X_j be the time to choose index j for the mth time, and

$$X = \max_{1 \le j \le I} X_j \tag{10}$$

is the time all indices are chosen at least m times. Since all X_j are independent with rate $1/I$, we have

$$\begin{aligned} P\{X < t\} =& P\{\max_{1 \le j \le I} X_j < t\} = P\{X_j < t, \text{ for } j = 1, \ldots, I\} \\ =& (1 - S_m(t/I)e^{-t/I})^I. \end{aligned}$$

Therefore, we have

$$\mathbb{E}[X] = \int_0^\infty P\{x > t\} dt.$$

We let N be the number of rounds to choose all indices at least m times, and T_i be the ith interarrival time of the Poisson process for choosing one index. Then we have

$$X = \sum_{i=1}^N T_i,$$

and T_i are independent. Thus we have

$$\mathbb{E}[X|N] = N\mathbb{E}[T_i] = N,$$

and which gives

$$\mathbb{E}[X] = \mathbb{E}\{\mathbb{E}[X|N]\} = \mathbb{E}[N].$$

Thus we have

$$\mathbb{E}[N] = \int_0^\infty \left(1 - (1 - S_m(t/I)e^{-t/I})^I\right) dt = I \int_0^\infty \left(1 - (1 - S_m(t)e^{-t})^I\right) dt.$$

The lemma is proved. □

It shows that the expected number of rounds to choose all indices at least once is $I \log(I) + I(m - 1) \log \log I$ when $I \to \infty$ [35].

A.2 Ablation Study: Impact of Different Weighing Schemes

[12] reported that weighting clients is important to improving model accuracy. Therefore, we did an ablation study and evaluated three client weighting schemes: (1) **model size-based weighting scheme**: client weight is proportional to the number of kernels in the model; (2) **model update-based weighting scheme**: client weight is proportional to the number of updates; and (3) **hybrid weighting scheme**: client weight is proportional to both (1) model size and (2) model update.

Table 5 lists the results. As shown, the performance of the three weighting schemes is not significantly better than the non-weighting scheme. Therefore, we used the non-weighting scheme in `FedRolex`.

Table 5: Impact of weighting schemes on model accuracy under high data heterogeneity.

	Weighting Scheme	Local Model Accuracy	Global Model Accuracy
CIFAR-10	Non-Weighting	95.95 (±0.81)	**69.44 (±1.50)**
	Model Size-based	95.98 (±0.67)	69.09 (±1.42)
	Model Update-based	96.01 (±0.71)	68.83 (±0.89)
	Hybrid	**96.05 (±0.96)**	68.78 (±0.89)
CIFAR-100	Non-Weighting	**81.58 (±0.59)**	56.57 (±0.15)
	Model Size-based	81.23 (±1.56)	**56.99 (±0.27)**
	Model Update-based	81.23 (±1.07)	56.63 (±0.36)
	Hybrid	81.49 (±1.07)	56.71 (±0.20)

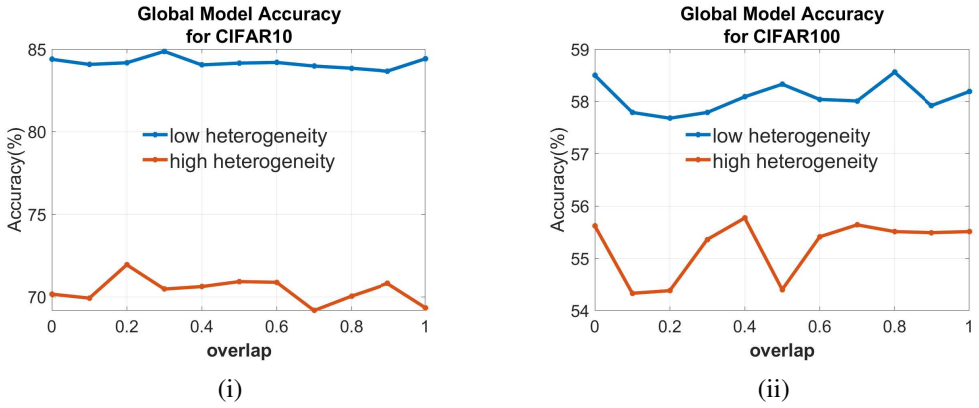
A.3 Ablation Study: Impact of Overlapping Kernels

We also studied the impact of overlapping kernels between rounds. Specifically, we extracted sub-models using a rolling window that advances and loops over all the kernels of each convolution layer in the global model in strides. Let the degree of overlap between each stride of the rolling window be $r \in [0, 1]$. In each iteration, each convolution layer in the global model is advanced by $1 + \lfloor \beta_n (1 - r) K_i \rfloor$ where $\lfloor \cdot \rfloor$ is the floor function. In `FedRolex`, $r = 1$, i.e., the kernels are advanced by 1 from one iteration to the next iteration.

Figure 6 shows the impact of different r on global model accuracy. As shown, the value of r does have some influence on the global model accuracy, but the impact is non-linear and inconsistent.

A.4 Ablation Study: Impact of Client Participation Rate

In our main paper, we followed prior arts [14, 24, 15, 36, 9] and used a 10% client participation rate. To examine the effect of client participation rate, we conducted experiments with both lower (5%) and higher (20%) client participation rates using CIFAR-10 as an example for `FedRolex`, HeteroFL and Federated Dropout.



(i) (ii)

Figure 6: Impact of inter-round kernel overlap on global model accuracy under low and high data heterogeneity for (i) CIFAR-10 and (ii) CIFAR-100.

The results are summarized in Table 6. As shown, `FedRolex` consistently outperforms both Federated Dropout and HeteroFL across 5%, 10% and 20% client participation rates.

Table 6: Performance of FedRolex, HeteroFL, and Federated Dropout under different client participation rates.

		Client Participation Rate		
		5%	10%	20%
CIFAR-10	HeteroFL	48.43 (+/- 1.78)	63.90 (+/-2.74)	65.07 (+/- 2.17)
	Federated Dropout	42.06 (+/- 1.29)	46.64 (+/-3.05)	55.20 (+/- 4.64)
	`FedRolex`	**57.90 (+/- 2.72)**	**69.44 (+/-1.50)**	**71.85 (+/- 1.22)**

A.5 Communication and Computation Costs of FedRolex

To calculate the communication cost, we use the average size of the models sent by all the participating clients per round as the metric. To calculate the computation overhead, we calculate the FLOPs and numbers of parameters in the models of all the participating clients per round and take the average as the metric. To put these metrics in the context, we also calculate the upper and lower bounds of the communication cost and computation overhead (i.e., all the clients were using the same largest model and smallest model, respectively).

Table 7 lists the results. As shown, compared to the upper bound, `FedRolex` significantly reduces the communication cost and computation overhead while being able to achieve comparable model accuracy. Compared to the lower bound, although `FedRolex` has higher communication cost and computation overhead, the model accuracy achieved is much higher than the lower bound. These results indicate that `FedRolex` is able to achieve comparable high model accuracy as the upper bound with much less communication cost and computation overhead.

Table 7: Computation and communication costs of FedRolex compared to upper and lower bounds represented by homogeneous settings with largest and smallest models respectively.

	Homogeneous (largest)	**FedRolex**	**Homogeneous (smallest)**
Average Number of Parameters per Client (Million)	11.1722	2.9781232	0.04451
Average FLOPs per Client (Million)	557.656	149.048384	2.41318
Average Model Size per Client (MB)	42.62	11.36	0.17

A.6 Experimental Setup Details

Experimental Setup Details for Table 3. The experimental setup for PT-based methods is listed in Table 8. The experimental setup for model-homogeneous baselines was slightly different from the PT-based methods, and hence is listed separately in Table 9.

Table 8: Experimental setup details of PT-based methods in Table 3 on CIFAR-10, CIFAR-100 and Stack Overflow.

		CIFAR-10	**CIFAR-100**	**Stack Overflow**
Local Epoch		1	1	1
Cohort SIze		10	10	200
Batch Size		10	24	24
Initial Learning Rate		2.00E-04	1.00E-04	2.00E-04
Decay Schedule	High Data Heterogeneity	800, 1500	1000, 1500	600, 800
	Low Data Heterogeneity	800, 1250	1000, 1500	
Decay Factor		0.1	0.1	0.1
Communication Rounds	High Data Heterogeneity	2500	3500	1200
	Low Data Heterogeneity	2000	3500	
Optimizer		SGD	SGD	SGD
Momentum		0.9	0.9	0.9
Weight Decay		5.00E-04	5.00E-04	5.00E-04

Experimental Setup Details for Figure 3. The experimental setup details are tabulated in Tables 10 and 11.

Experimental Setup Details for Figure 4. The experimental setup details are tabulated in Table 12.

Table 9: Experimental setup details of model-homogeneous baselines in Table 3 on CIFAR-10 and CIFAR-100 and Stack Overflow.

		CIFAR-10	**CIFAR-100**	**Stack Overflow**
Local Epoch		1	1	1
Cohort Size		10	10	200
Batch Size		10	24	24
Initial Learning Rate		2.00E-04	1.00E-04	2.00E-04
Decay Schedule	High Data Heterogeneity	500, 1000	1000, 1500	300
	Low Data Heterogeneity	500, 1000	1000, 1500	
Decay Factor		0.1	0.1	0.1
Communication Rounds$\sim$	High Data Heterogeneity	1250	3500	1000
	Low Data Heterogeneity	1500	3500	
Optimizer		SGD	SGD	SGD
Momentum		0.9	0.9	0.9
Weight Decay		5.00E-04	5.00E-04	5.00E-04

Table 10: Experimental setup for results shown in Figure 3. ρ between 0.0 and 0.5 in 0.1 increments.

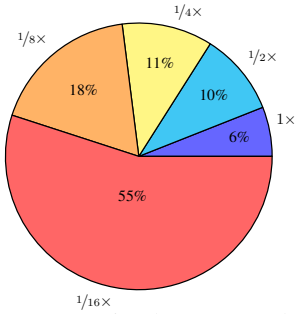
Dataset		ρ	0.0	0.1	0.2	0.3	0.4
CIFAR-10	High Heterogeneity	Decay Schedule	500, 1000	500, 1000	500, 1000	700, 1200	700, 1200
		Communication Rounds	1250	1250	1250	1500	1500
	Low Heterogeneity	Decay Schedule	500, 1000	500, 1000	500, 1000	700, 1200	700, 1200
		Communication Rounds	1250	1250	1250	1500	1500
CIFAR-100	High Heterogeneity	Decay Schedule	1000, 1500	1000, 1500	1000, 1500	1000, 1500	1000, 1500
		Communication Rounds	2000	2000	2000	2000	2000
	Low Heterogeneity	Decay Schedule	1000, 1500	1000, 1500	1000, 1500	1000, 1500	1000, 1500
		Communication Rounds	2000	2000	2000	2000	2000
Stack Overflow	High Heterogeneity	Decay Schedule	800	800	800	800	800
		Communication Rounds	1500	1500	1500	1500	1500
	Low Heterogeneity	Decay Schedule	800	800	800	800	800
		Communication Rounds	1500	1500	1500	1500	1500

Table 11: Experimental setup for results shown in Figure 3. ρ between 0.5 and 1.0 in 0.1 increments.

Dataset		ρ	0.6	0.7	0.8	0.9	1.0
CIFAR-10	High Heterogeneity	Decay Schedule	700, 1200	700, 1200	500, 1000	500, 1000	500, 1000
		Communication Rounds	1500	1500	1250	1250	1250
	Low Heterogeneity	Decay Schedule	700, 1200	700, 1200	500, 1000	500, 1000	500, 1000
		Communication Rounds	1500	1500	1250	1250	1250
CIFAR-100	High Heterogeneity	Decay Schedule	1000, 1500	1000, 1500	1000, 1500	1000, 1500	1000, 1500
		Communication Rounds	2000	2000	2000	2000	2000
	Low Heterogeneity	Decay Schedule	1000, 1500	1000, 1500	1000, 1500	1000, 1500	1000, 1500
		Communication Rounds	2000	2000	2000	2000	2000
Stack Overflow	High Heterogeneity	Decay Schedule	800	800	800	800	800
		Communication Rounds	1500	1500	1500	1500	1500
	Low Heterogeneity	Decay Schedule	800	800	800	800	800
		Communication Rounds	1500	1500	1500	1500	1500

Table 12: Experimental setup for results shown in Figure 4

Dataset		γ	2	4	8	16
CIFAR-10	High Heterogeneity	Decay Schedule	800, 1200	800, 1200	800, 1200	800, 1200
		Communication Rounds	1500	1500	1500	1500
	Low Heterogeneity	Decay Schedule	800, 1200	800, 1200	800, 1200	800, 1200
		Communication Rounds	1500	1500	1500	1500
CIFAR-100	High Heterogeneity	Decay Schedule	800, 1200	800, 1200	800, 1200	800, 1200
		Communication Rounds	1500	1500	1500	1500
	Low Heterogeneity	Decay Schedule	800, 1200	800, 1200	800, 1200	800, 1200
		Communication Rounds	1500	1500	1500	1500
Stack Overflow	High Heterogeneity	Decay Schedule	800	800	800	800
		Communication Rounds	1500	1500	1500	1500
	Low Heterogeneity	Decay Schedule	800	800	800	800
		Communication Rounds	1500	1500	1500	1500



Model Capacity	**Annual Household Income**
$1/16\times$	$< \$75,000$
$1/8\times$	$\$75,000 - \$100,000$
$1/4\times$	$\$100,000 - \$150,000$
$1/2\times$	$\$150,000 - \$200,000$
$1\times$	$> \$200,000$

Figure 7: Mapping between real-world annual household income and model capacity.

Table 13: Experimental setup for Table 4 for CIFAR-10, CIFAR-100 and Stack Overflow.

		CIFAR-10	CIFAR-100	Stack Overflow
Local Epoch		1	1	1
Cohort SIze		10	10	200
Batch Size		10	24	24
Initial Learning Rate		2.00E-04	1.00E-04	2.00E-04
Decay Schedule	High Heterogeneity	800, 1500	1000, 1500	600, 800
	Low Heterogeneity	800, 1250	1000, 1500	
Decay Factor		0.1	0.1	0.1
Communication Rounds	High Heterogeneity	2500	3500	1200
	Low Heterogeneity	2000	3500	
Optimizer		SGD	SGD	SGD
Momentum		0.9	0.9	0.9
Weight Decay		5.00E-04	5.00E-04	5.00E-04

A.7 Algorithm Pseudocodes

The pseudocodes for HeteroFL and Federated Dropout are given in Algorithms 2 and 3 respectively. Their differences from `FedRolex` are marked using blue color.

Algorithm 2: HeteroFL

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** $J-1$ **do**
  Sample subset $\mathcal{M}$ from $\mathcal{N}$
  Broadcast $\theta^{(j)}_{m,[i\ ;\ 0,1,\ \dots\ \lfloor \beta_n K_i \rfloor - 1]} \forall i$ and $m \in \mathcal{M}$
  for *each* *client* $m \in \mathcal{M}$ **do**
    `clientStep(`$\theta^{(j)}_m$, D_m`)`
  end
  Aggregate $\theta^{(j+1)}_{[i,k]}$ according to Equation (5)
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

Algorithm 3: Federated Dropout

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** $J-1$ **do**
  Sample subset $\mathcal{M}$ from $\mathcal{N}$
  Broadcast $\theta^{(j)}_{m,[i\ ;\ k_1,\dots,k_{\lfloor \beta_n K_i \rfloor}]} \forall i$ and $m \in \mathcal{M}$
  for *each* *client* $m \in \mathcal{M}$ **do**
    `clientStep(`$\theta^{(j)}_m$, D_m`)`
  end
  Aggregate $\theta^{(j+1)}_{[i,k]}$ according to Equation (5)
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