Federated Fingerprint Learning with Heterogeneous Architectures

Tianshi Che[†], Zijie Zhang[†], Yang Zhou[†*], Xin Zhao[†], Ji Liu[‡],
Zhe Jiang[§], Da Yan[¶], Ruoming Jin[♯], Dejing Dou[‡]
[†]Auburn University [‡]Baidu Research [§]University of Florida [¶]University of Alabama at Birmingham [♯]Kent State University
{tzc0029, zzz0092, yangzhou}@auburn.edu, cmkk684735@gmail.com, liuji04@baidu.com,
zhe.jiang@ufl.edu, yanda@uab.edu, rjin1@kent.edu, doudejing@baidu.com

2022 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-5099-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDM54844.2022.00013

***Abstract*—Recent studies on federated learning (FL) have sought to solve the system heterogeneity issue by designing customized local models for different clients. However, public dataset introduction, sensitive information exchange, non-trivial computational cost, or particular architecture requirement limit the applicability of most of them in real scenarios. This paper presents a novel federated fingerprint learning model for making full use of the computing power of each client with the customized local models for improving the FL convergence, while keeping the data and sensitive information safe and local. First, we decompose the parameters of each local model into two types of parameters: rigid ones that have fixed model architecture for ensuring the convergence of global model training and elastic ones that contain customized model structure and size for allowing to make full use of the computing power of each client based on individual data scale. Second, we adopt the standard FL scheme to update and aggregate the local rigid parameters. We introduce a Gaussian distribution as auxiliary input and output K local fingerprints respectively for the elastic parameters of all K local models. The server aggregates K local fingerprints into a global one and sends it back to the clients. A fingerprint-based aggregation strategy makes the local models indirectly receive the aggregated elastic parameters through the aggregation of K local fingerprints while fixing data locally. Last but not least, we design a parameter masking method to mask the rigid parameters irrelevant to the local classification task in the local models. We develop a parameter separation method to guarantee that the combination of unmasked rigid parameters in all local models are able to cover all the rigid parameters as many as possible, for further raising the utilization rate of each rigid parameter.**

***Index Terms*—federated learning, system heterogeneity, fingerprint, mask**

I. Introduction

Recent advances in federated learning (FL) present a promising machine learning (ML) paradigm that enables collaborative training of shared ML models over multiple distributed clients without the need for data sharing, while protecting the data privacy [1]–[5]. One of critical challenges in the FL paradigm is system heterogeneity due to different computational resources and capabilities across clients [6]–[8]. In the standard FL paradigm, the server learns a single global model for prediction by aggregating the parameters of local models at clients. However, exchanging knowledge among the server and clients is only viable if all of them share the same model structure and size. This implies that the computational capability of FL models is limited by the clients with the poorest hardware configuration [9]. In this case, the clients with strong configuration fail to make full use of their computing power and thus the convergence of both local and global models can be significantly delayed. In addition, the underutilized computing power may seriously slow down the convergence of FL models.

[*]Corresponding author

Only recently, researchers attempt to solve the system heterogeneity issue by designing customized local models for each client with different hardware capabilities. Knowledge distillation techniques transfer knowledge from the local models with different model architectures to a centralized server model [10]–[16]. However, these methods either rely on access to common public datasets which may be impractical [10]–[12], [15], or transfer sensitive information (e.g., predictions or embeddings) among the clients and server, raising potential privacy concerns [13], [16]. Hypernetwork-based methods feed local clients' representations to a global hypernetwork, which can output personalized heterogeneous models [17], [18]. The hypernetwork has a large memory footprint already for small clients' models, which may lead to non-trivial efficiency and scalability issues. Embedding-based algorithms communicate the learnt representations among the clients and server [19], [20]. The embedding transferring on the Internet can increase privacy exposure risks. HeteroFL proposes to upload a different subset of global model to the server for aggregation with the objective to produce a single global inference model [21]. However, HeteroFL can train CNN-style models only in federated settings. In addition, it requires that all clients share the same architecture type (e.g. CNN or ResNet-18) and differ only in the number of channels per layer. FedBABU decomposes the entire network into the body with the same model structure and size for the global model training and the head for the personalized learning [22]. In FedBABU, the head is never updated during federated training. Therefore, the global model training still follows the standard FL strategy by aggregating the local updates.

This motivates the central questions investigated in this work: Can we make full use of the computing power of each client with the customized local model for improving the

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.

convergence of both local and global model training, while keeping the data or sensitive information safe and local? How to tailor the local models based on unique data distribution at each client for enhancing the convergence of FL training?

A key technical challenge in solving this problem is how to aggregate the parameters of local models (e.g., neural networks) with different structure, width, and depth into a global model without privacy leaking. We decompose the parameters W_k $(1 \leq k \leq K)$ of each local model at client C_k into two types of parameters: rigid ones U_k that have fixed model architecture for ensuring the convergence of global model training and elastic ones V_k that contain customized model structure and size for allowing to make full use of the computing power of each client based on individual data scale. Efficient neural architecture search (NAS) methods [23]–[28] can be leveraged to identify the importance of the neural network layers in all K local models, partition them into the rigid and elastic layers based on their importance, and divide the parameters into the rigid and elastic ones.

The rigid parameters in both global model (U) and all K local models $(U_1, \cdots, U_K)$ share the common structure, width, and depth. We follow the standard FL scheme to update and aggregate the local rigid parameters $U_1, \cdots, U_K$ to generate a global one U. The elastic parameter V_k of each local model have the customized structure, width, and depth and output a fingerprint F_k. As for the update of the local elastic parameters $V_1, \cdots, V_K$, instead of introducing external public datasets or exchanging sensitive information (e.g., predictions or embeddings) among the server and clients, we introduce a Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$ as auxiliary input for the elastic parameters of all K local models under the restriction of privacy preservation in the federated settings. The original data and the auxiliary input share the same elastic parameters at each client C_k. The output for the auxiliary Gaussian distribution at each client is still a distribution and serves as a fingerprint F_k to implicitly capture the behavior and characteristic of the elastic parameters of the local model at client C_k. Each client uploads only its local fingerprint F_k to the central server for the aggregation. We only require that the first and last elastic layers of each local model have the same size. Each client freely chooses the width and depth of other elastic layers. The server aggregates the local fingerprints $F_1, \cdots, F_K$ into a global one F and then sends F back to the clients. At the beginning of next FL round, each client C_k first refines its local elastic parameter V_k by forcing that the local fingerprint F_k is close to the global one F. This strategy makes the local models indirectly receive the aggregated elastic parameters $V_1, \cdots, V_K$ through the aggregation of F while fixing data locally.

Deep learning techniques utilize the architecture of large model size and over-parameterization to achieve state-of-the-art performance [29], [30]. A recent study reports that a lossless neural networks pruning can be derived for any deep learning models, as long as the networks meet a certain condition about their spectrum and Lipschitz constant [31]. We argue that the local models in federated settings are particular over-parameterization when local data distributions are quite different from each other. A local model that is inherited from the global model and trained on a part of the entire dataset may make contributions to the update of only some parameters of the global model. For example, two clients contain the samples from classes 1 and 2, and classes 3 and 4 respectively. The local model training at client 1 is related to only the classification regarding classes 1 and 2. The local model inherited from the global model that addresses the classification about all four classes is over-parameterization for client 1. It is unnecessary to update the parameters irrelevant to the classification regarding classes 1 and 2 during the local model training for slowing down the FL convergence. By leveraging the lossless neural network pruning techniques [31], a parameter masking method is designed to mask the rigid parameters in the local models in terms of local data distribution. In order to further raise the utilization rate of each rigid parameters, a parameter separation method is developed to guarantee that the combination of unmasked rigid parameters in all local models are able to cover all the rigid parameters as many as possible.

Our proposed federated fingerprint learning model brings three tremendous benefits: (1) the elastic parameters allow to make full use of the computing power of each client with customized local model for improving the convergence, while there are no external public datasets or sensitive information exchanging; (2) the shared rigid parameters among the clients ensure that the local models can converge to a single global model; and (3) the parameter masking method is able to improve the convergence of local and global model training.

Empirical evaluation on real federated datasets demonstrates the superior performance of our federated fingerprint learning model against several state-of-the-art methods for tackling the challenge of statistical and system heterogeneity.

II. Problen Statement

Given K clients $C_1, \cdots, C_K$, each C_k $(1 \leq k \leq K)$ has a local dataset D_k with N_k samples $\{(x_k^1, y_k^1), \cdots, (x_k^{N_k}, y_k^{N_k})\}$, where each x_k^i and y_k^i denote the input feature and class label of a sample respectively. Each client C_k trains a local model $f(x; W_k)$ with a local parameter W_k over D_k. Federated learning aims to solve the following optimization problem.

$$\begin{aligned}\min_{W \in \mathbb{R}^d} \mathcal{L}(f(x;W), y) = \sum_{k=1}^{K} \frac{N_k}{N} \mathcal{L}_k(f(x;W), y) \\ \text{where } \mathcal{L}_k(f(x;W), y) = \mathbb{E}_{(x,y)\sim D_k} L_k(f(x;W), y)\end{aligned} \tag{1}$$

where N is the total number of training samples from all K clients, i.e., $N = N_1 + \cdots + N_K$. W is the parameter of global model. f is the shared model by the clients and server. $L_k(f(x;W), y)$ denotes the loss (e.g., cross-entropy loss) of the prediction on a sample (x, y) made with W. $\mathcal{L}_k(f(x;W), y)$ denotes the local loss function at client C_k. $\mathcal{L}(f(x;W), y)$ represents the global loss for all K clients.

On the other hand, the optimization goal of local model training at client C_k is defined as follows.

$$\min_{W_k \in \mathbb{R}^d} \mathcal{L}_k(f(x;W_k), y) = \mathbb{E}_{(x,y)\sim D_k} L_k(f(x;W_k), y) \tag{2}$$

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.

client的模型由两部分组成，即U和V。
server只有rigid这层模型，即W=U

In the standard FL paradigm, the global and local models share the same model architecture f, i.e., their parameters W and $W_1, \cdots, W_K$ have the same structure, width, and depth. It aggregates the local model parameters $W_1, \cdots, W_K$ with the same structure and size to generate the global one W. However, in real world, the data at the clients collected by different owners are often non-IID (Independent and Identically Distributed), i.e., the data distributions at the clients are different from each other, and thus the statistical heterogeneity issue is raised. In addition, the end clients such as computers, smartphones, and Internet of Things devices are usually heterogeneous, in terms of different computational resources and capabilities across the clients, and thus the system heterogeneity problem is rised. The above standard FL paradigm with the same model architecture for both the global and local models is not the optimal solution to tackle such heterogeneous clients and data.

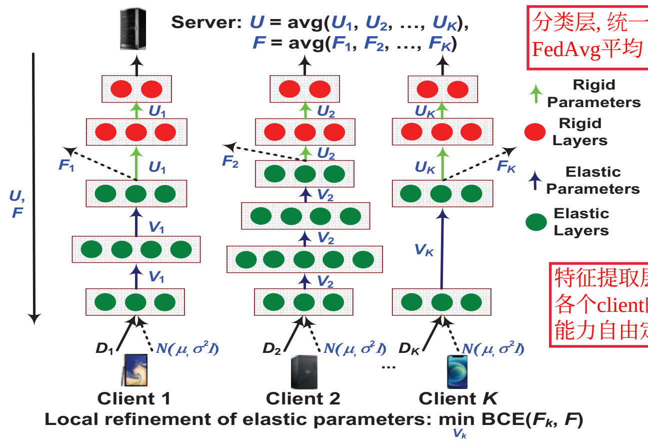
This work aims to develop a novel flexible FL model for making full use of the computing power of each client with the customized local models for improving the convergence of local training, while indirectly connecting local models together for ensuring the convergence of global model training. The FL model should keep the data and any sensitive information safe and local, except model parameter exchanging.

In our proposed federated fingerprint learning model, the neural network layers of each local model is decomposed into two types of layers: (1) Rigid layers with the fixed size are shared with all K clients for ensuring the convergence of global model training and (2) Elastic layers that are allowed to customize for each client, in order to make full use of its computing power. Two types of layers can locate anywhere on the neural networks, such as input, intermediate, or output layers. Therefore, our FL model has two corresponding kinds of parameters: (1) Rigid parameters that occur in both the global model and all K local models and (2) Elastic parameters that occur in the K local models.

Rigid layers. A rigid layer for a client C_k takes the original data on C_k, other rigid layer, or elastic layer as input, participates in the local model training for the FL task, and outputs the predicted class labels or connects to other rigid layer or elastic layer. The rigid layers in both global model and all local ones share common structure, width, and depth.

Elastic layers. An elastic layer for a client C_k is involved in two learning tasks simultaneously: (1) takes the original data on C_k, rigid layer, or other elastic layer as input, participates in the local model training for the FL task, and outputs the predicted class labels or connects to rigid layer or other elastic layer and (2) takes a Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$, rigid layer, or other elastic layer as input and outputs a fingerprint F_k or connects to rigid layer or other elastic layer. The owner of C_k can feel free to customize the number of elastic layers and the dimension of each elastic layer in the local model on C_k. The only exception is that the first and last elastic layers of each local model have the same size.

Rigid parameters. A rigid parameter U_k for a client C_k (or



分类层, 统一固定, FedAvg平均

特征提取层, 根据各个client的计算能力自由定制

Fig. 1: Federated Fingerprint Learning

U for the server) is the flattened combination of the weights between a rigid layer and a neighbor rigid or elastic layer. $U_1, \cdots, U_K$ and U share the common size and can be used to exchange the knowledge among the clients and server for conducting the FL task.

Elastic parameters. An elastic parameter V_k for a client C_k is the flattened combination of the weights between pairwise consecutive elastic layers. Notice that the global model does not have the elastic layers, since the corresponding elastic parameters on different clients often have diverse size and cannot be used to aggregate into a global model.

Therefore, the local model parameter W_k at client C_k is the flattened combination of the above two kinds of parameters, i.e., $W_k = [U_k \; V_k]$, while the global model parameter W on the server contains only the elastic parameter, i.e., $W = U$.

III. Federated Fingerprint Learning

Figure 1 presents the architecture of our federated fingerprint learning model that utilizes fingerprints as intermediaries to exchange the knowledge among the clients and server, while keeping the data or sensitive information safe and local. The local models update the rigid parameters $U_1, \cdots, U_K$, elastic parameters $V_1, \cdots, V_K$, and fingerprints $F_1, \cdots, F_K$ from the elastic parameters. The central server receives $U_1, \cdots, U_K$ and fingerprints $F_1, \cdots, F_K$ from all K local clients and aggregate them to produce the global model parameter U and global fingerprint F. In our federated fingerprint learning model, we do not need to introduce external public datasets or exchange sensitive information (e.g., predictions or embeddings) among the clients and server. This implies that our model is able to follow the same requirement of privacy preservation in the standard FL settings. In addition, although a client C_k with strong configuration may have a large number of elastic layers and elastic parameters V_k, we upload only the fingerprint F_k to the server, which can dramatically reduce the communication cost between the clients and server. Thus, our federated fingerprint learning model provides a flexible and lightweight heterogeneous FL architecture.

Partitioning of rigid and elastic parameters. Recently, the deep learning community has seen a shift from manually designing architectures of neural networks to developing neural architecture search (NAS) algorithms that automate

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.

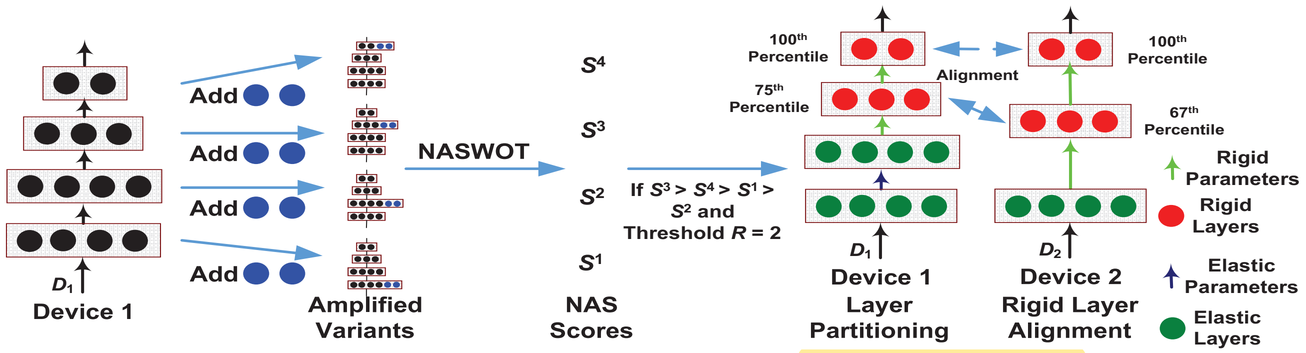


Fig. 2: Partitioning of Rigid and Elastic Parameters with Neural Architecture Search

the discovery of effective architectures [23]–[28]. NASWOT proposes a score measure that is indicative of the final trained accuracy of a neural network [27]. Based on the score measure, it develops a fast NAS algorithm that searches for neural networks without training in a matter of seconds on a single GPU. For example, NASWOT is able to search for a neural network that achieves 92.81% accuracy in 30 seconds within the NAS-Bench-201 search space on CIFAR-10 [32].

The NASWOT method [27] can be leveraged to identify the importance of the neural network layers in all K local models and partition them into the rigid and elastic layers in terms of their importance. First, a score measure is defined as an indicator of the final trained accuracy of a neural network. Consider a mini-batch of data $x_k = \{x_k^i\}_{i=1}^{N_k}$ mapped through a neural network as $f_k(x_k^i)$. The indicators from the rectified linear units in f_k at x_k^i form a binary code c_k^i that defines the linear region. Namely, the linear maps of the network are uniquely identified by the binary code corresponding to the activation pattern of the rectified linear units.

A kernel matrix $\mathbf{K}_k$ at client C_k measures the correlations between binary codes on the whole mini-batch data.

$$\mathbf{K}_k = \begin{pmatrix} n_k - d\left(\mathbf{c}_k^1, \mathbf{c}_k^1\right) & \cdots & n_k - d\left(\mathbf{c}_k^1, \mathbf{c}_k^{N_k}\right) \\ \vdots & \ddots & \vdots \\ n_k - d\left(\mathbf{c}_k^{N_k}, \mathbf{c}_k^1\right) & \cdots & n_k - d\left(\mathbf{c}_k^{N_k}, \mathbf{c}_k^{N_k}\right) \end{pmatrix} \quad (3)$$

where n_k is the number of rectified linear units of the local model at client C_k. d denotes the Hamming distance between two binary codes induced by the untrained network at two inputs. It measures how dissimilar the two inputs are.

Thus, the score of a neural network f_k is generated below.

$$S_k = \log |\mathbf{K}_k| \quad (4)$$

Between two kernel matrices with the same trace, S_k is larger if the kernel matrix $\mathbf{K}_k$ is closer to a diagonal matrix. A higher S_k at initialization implies the improved final accuracy after training.

Second, the server chooses the largest local model f_k with the most layers to partition its layers into rigid and elastic layers. The same number of neurons is added to each layer respectively to generate L_k variants of f_k, each with one layer amplified, where L_k is the number of layers in f_k. A score is calculated for each variant and top-R variants with the highest scores are selected, given a threshold R regarding the number of rigid layers. A variant with a higher score implies that the corresponding layer after amplification can improve final accuracy more after training and thus has greater impact on the prediction quality. Therefore, the layers corresponding to the top-R variants are identified as rigid layers and other layers are recognized as elastic layers.

Finally, the server identifies the rigid and elastic layers for other local models with different size, in terms of the corresponding positions of the rigid and elastic layers in the largest local model f_k. For example, f_1 is the largest local model with four layers and layer 3 is identified as a rigid layer. The position of layer 3 is equal to $3/4 = 75$th percentile. In another local model f_2 with three layers, layers 2 and 3 have positions of $2/3 = 67$th percentile and $2/3 = 100$th percentile respectively. Layer 2 in f_2 is selected as a rigid layer since 67th percentile is closer to 75th percentile.

Training objective. Our federated fingerprint learning model aims to solve a joint optimization problem in the federated settings: minimize the local loss function at all K clients regarding the FL task, while matching the output fingerprints $F_1, \cdots, F_K$ with each other.

$$\min_{U_1,\cdots,U_K,V_1,\cdots,V_K} \sum_{k=1}^{K} \frac{N_k}{N} \mathcal{L}_k(f(x; U_k, V_k), y) + \lambda \cdot \mathbf{BCE}(F_k, F)$$
$$\text{where } \mathcal{L}_k(f(x; U_k, V_k), y) = \mathbb{E}_{(x,y)\sim D_k} L_k(f(x; U_k, V_k), y) \quad (5)$$

where K, N_K, N, x, y, and $L_k(f(x; U_k, V_k), y)$ have the same definitions as the ones in Eqs.(1)-(2). f denotes a local model at client C_k that is parameterized with the corresponding rigid parameter U_k and elastic one V_k. F_k represents the output fingerprint of f. $\mathcal{L}_k(f(x; U_k, V_k), y)$ denotes the local loss function at client C_k. **BCE** specifies the binary cross-entropy to measure how the local fingerprint distribution F_k is different from the global one F.

The above training objective can be divided two optimization subproblems. We alternately solve these two subproblems: (1) fix $V_1, \cdots, V_K$ to optimize $U_1, \cdots, U_K$ and (2) fix $U_1, \cdots, U_K$ to optimize $V_1, \cdots, V_K$. In the federated setting, subproblem (1) uses the local data at client C_k to solve the original FL task (e.g., classification), while subproblem (2) tries to match the fingerprint distributions in both the local and global models. The binary cross-entropy tends to achieve the minimum when two fingerprint distributions are identical.

Global model and fingerprint aggregation. We follow the standard FL scheme to update and aggregate the local rigid parameters $U_1, \cdots, U_K$ to generate a global one U. The central server aggregates the local fingerprints $F_1, \cdots, F_K$

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.

produced by the local elastic parameters $V_1, \cdots, V_K$ into a global one F and then sends F back to the clients.

$$U = \sum_{k=1}^{K} \frac{N_k}{N} U_k, \quad F = \sum_{k=1}^{K} \frac{N_k}{N} F_k \tag{6}$$

In our federated fingerprint learning model, we utilize and aggregate the local rigid parameters with the common structure and size shared by all K clients for ensuring the convergence of global model training. To tackle the system heterogeneity issue in the federated setting, the local elastic parameters have the customized structure and size for making full use of the computing power of the corresponding clients.

Local model update. In the local models, we update the local parameters W_k for optimizing the local loss function $\mathcal{L}_k(f(x; W_k), y)$ at client C_k. Notice that W_k at client C_k is the flattened combination of the local rigid and elastic parameters, i.e., $W_k = [U_k \; V_k]$. Instead of introducing external public datasets or exchanging sensitive information (e.g., predictions or embeddings) among the server and clients, we introduce an auxiliary input following a Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$ for the elastic layers of each local model to output a fingerprint F_k through a mapping function m parameterized with V_k to implicitly capture the behavior and characteristic of the elastic layers. In order to produce a consensus fingerprint across the clients, a regularization term is added to the local loss function, enabling the local fingerprints $F_1, \cdots, F_K$ to approach the global one F, while minimizing the loss of the FL task.

类似proxy data的用法

$$\begin{aligned} &\min_{U_k, V_k} \mathcal{L}_k(f(x; U_k, V_k), y) + \lambda \cdot \mathbf{BCE}(F_k, F) \\ &W_k \leftarrow W_k - \eta_1 \nabla_{W_k} \mathcal{L}_k(f(x; W_k), y) \\ &V_k \leftarrow V_k - \eta_2 \nabla_{V_k} \lambda \cdot \mathbf{BCE}(F_k, F) \text{ where } F_k = m(z; V_k) \end{aligned} \tag{7}$$

但是BCE是基于二分类的，这里应该要对hidden求softmax再计算吧？

where $z \sim \mathcal{N}(0, \sigma^2 I)$ denotes an auxiliary input sampled from a Gaussian distribution. η_1 and η_2 are the learning rates for optimizing W_k and refining V_k respectively.

The original data and the auxiliary input share the same elastic layers at each client k. At current FL round, we update the local rigid and elastic parameters U_k and V_k. Each client uploads its local rigid parameter U_k and fingerprint F_k to the server for aggregation. At the beginning of next FL round, after the server sends U and F back to the clients, each client C_k refines its local elastic parameter V_k by forcing the local fingerprint F_k close to the global one F, i.e., $\min_{V_k} \mathbf{BCE}(F_k, F)$. This makes the local models indirectly receive the aggregated elastic parameters $V_1, \cdots, V_K$ through the aggregation of F while fixing data locally. By assembling all the pieces in Section III, the pseudo code of our federated fingerprint learning model is provide in Algorithm 1.

IV. Mask Generation before Training

Although deep neural networks have become ubiquitous in fields such as computer vision and natural language processing, large model size and extreme over-parameterization in deep learning are typically required to achieve state-of-the-art results, incurring higher training costs and hindering applications limited by memory or inference time [29], [30]. Recent advances in neural network pruning and lottery ticket hypothesis suggest the existence of an extremely sparse subnetwork, within an over-parameterized dense neural network,

Algorithm 1 Federated Fingerprint Learning

Input: K clients $C_1, \cdots, C_K$, local datasets $D_1, \cdots, D_K$ at K clients, standard deviation σ in Gaussian distribution, learning rate η_1, learning rate η_2, number of FL rounds T, and number of local epochs t_1.

```
1:  Initialize local parameters $W_1, \cdots, W_K$ for all $K$ clients;
2:  for each round $1, \ldots, T$ do
3:  On Client:                                    Local Training: U, V have updated
4:      for each client $k$ in parallel do
5:        Update local parameter $W_k \leftarrow$ LocalUpdate($k$);
6:        Sample $z$ from Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$;
7:        Calculate fingerprint $F_k = m(z; V_k)$;   基于随机noise计算得到的hidden
8:        Upload $U_k$ and $F_k$ to server.
9:  On Server:
10:     Update global rigid parameter $U$ in Eq.(6);   分类层：FedAvg fingerprint: hidden avg
11:     Update global fingerprint $F$ in Eq.(6).
12: On Client:
13:     for each client $k$ in parallel do
14:       Download $U$ and $F$ from Server;
15:       Refine local rigid parameter $U_k \leftarrow U$;
16:       Refine local elastic parameter $V_k$ through training in Eq.(7).
                                      Binary CE做KD, 基于noise的hidden和其他client的hidden avg
17: LocalUpdate($k$)
18:     for each local epoch $1, \ldots, t_1$ do
19:       for batch $(x_k, y_k) \in D_k$ do
20:           Update local parameter $W_k$ through local model training in Eq.(7);
                                      client update with private data
21:     Return $W_k$.
```

that can reach similar performance as the dense network when trained with the same parameter initialization [33]–[37]. IMC is a recently proposed lossless neural network pruning algorithm that reformulates the neural network optimization problem as a gradient dynamical system and reduces this high-dimensional system onto inertial manifolds to obtain a low-dimensional system regarding pruned subnetworks [31]. The extensive evaluation has shown that, in some settings, pruned subnetworks can be successfully retrained and yield better performance than their original dense networks.

We argue that the local models in federated settings are particular over-parameterization due to the statistical heterogeneity issue, i.e., local data distributions are quite different from each other. A local model that is inherited from the global model and trained on a part of the entire dataset may make contributions to the update of some neurons of the global model. Especially, when class distributions over local clients are quite imbalanced, the over-parameterization issue of local models becomes worse. For example, two clients contain the samples from classes 1 and 2, and classes 3 and 4 respectively. The local model training at client 1 is related to only the classification regarding classes 1 and 2. The local model inherited from the global model that addresses the classification about all four classes is over-parameterization for client 1. It is unnecessary to update the parameters irrelevant to the classification regarding classes 1 and 2 during the local model training for slowing down the convergence. Thus, using the parameter of a global model that covers the entire data

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.

on all classes to train a local model over the local data on partial classes may lead to the overfitting issue, slow down the convergence of global model training, and thus delay the convergence of global model training by aggregating the overfitted local models.

The lossless neural network pruning techniques [31] can be leveraged to mask the rigid parameters in the local models in terms of local data distribution. Each local model is first trained few iterations (10 epochs in our experiment) to generate an approximate local parameter W_k^* and then the IMC algorithm [31] can be utilized to produce a binary pruning mask $M_k \in \{0,1\}^d$ of W_k^* for pruning the unnecessary rigid parameters in the local models in terms of local data distribution. As shown in the experiments in the IMC paper, the training with 10 epochs is enough to guarantee the generation of a good neural network pruning.

In order to further raise the utilization rate of the rigid parameters for alleviating the over-parameterization issue, a parameter separation method is developed to guarantee that the combination of unmasked rigid parameters in all local models are able to cover the shared rigid parameters as many as possible. In addition, the parameter separation method is able to dramatically reduce the communication cost among the clients and server when using sparse storage of masked parameters in the FL, such as sparse tensors in TensorFlow or PyTorch. The objective of rigid parameter separation can be formulated as follows.

$$\max_{g_k,h_k} \sum_{k=1}^{K} \mathbf{cos}\Big(g_k(h_k(M_k)), g_k(M_k)\Big) + \alpha \cdot \mathbf{nnz}\Big(h_1(M_1) \odot \cdots \odot h_k(M_k)\Big) \tag{8}$$

where $\mathbf{cos}$ represents the cosine similarity between a transformed mask $h_k(M_k)$ with the parameter separation and a original mask M_k. In order to further reduce the cost of similarity computation, we project $h_k(M_k)$ and M_k into embedding spaces with an encoder g_k. $\odot$ denotes the Hadamard product between two flattened vectors. $\mathbf{nnz}$ specifies the number of non-zero elements in a vector. The second term is to encourage to utilize every possible rigid parameter among all K clients, to alleviate the over-parameterization issue.

The above mask generation operation is a one-time operation before the training process of FL. We exchange the iteratively expensive communication cost and the convergence accelerating of global model training in the FL with the one-time mask generation cost. The pseudo code of mask generation before FL training is presented in Algorithm 2.

Algorithm 2 Mask Generation before FL Training

Input: K clients $C_1, \cdots, C_K$, local datasets $D_1, \cdots, D_K$ at K clients, and number of few local epochs t_2.

```
1:  On Client:
2:    for each client $k$ in parallel do
3:      for each local epoch $1, \ldots, t_2$ do
4:        for batch $(x_k, y_k) \in D_k$ do
5:          Generate local parameter $W_k$ through local model training in Eq.(7);
6:      Produce a parameter mask $M_k$ for local rigid one $U_k$;
7:      Upload $M_k$ to server.
8:  On Server:
9:    Train an encoder $g_k$ and a transformation function $h_k$ based on objective function in Eq.(8);
10:   Produce a transformed mask $\tilde{M}_k \leftarrow h_k(M_k)$.
11:   Send $\tilde{M}_k$ to client $C_k$.
12: On Client:
13:   Download $\tilde{M}_k$ from Server;
14:   Initialize local parameters $W_k \leftarrow (U_k \odot \tilde{M}_k) \cup V_k$ and enter the FL training process in Algorithm 1.
```

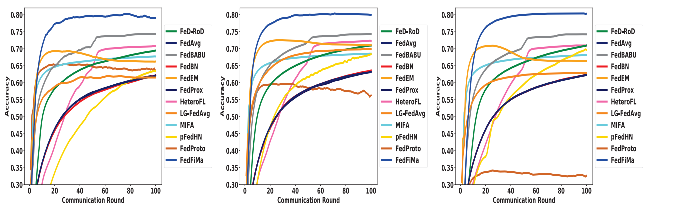
V. Experiments

In this section, we have evaluated the performance of our FedFiMa model and other comparison methods in serval representative federated datasets and learning tasks to date. We show that our FedFiMa method can achieve faster convergence and higher test accuracy in federated settings against several state-of-the-art approaches.

Datasets. We focus on three popular computer vision tasks over three representative benchmark datasets respectively: (1) We train a multilayer perceptron (MLP) on EMNIST [38] for character recognition; (2) We train LeNet over CIFAR-10 [32] for image classification; and (3) We train a convolutional neural network (CNN) on SVHN for street view house number identification [39]. The same experimental design in [4], [40] can be followed to produce a highly non-IID setting by assigning examples from at most $s \in \{2,3,4,5\}$ classes to each client. The value of s simulates client variance: $s = 2$ represents highest client variance. A smaller value of s indicates a stronger degree of non-IID. For the datasets with 10 classes, $s = 10$ represents a complete IID split of labels to clients. We run 100 global communication rounds and 5 local epochs in each round with varying 3, 10, or 50 clients and an active-user ratio 100%.

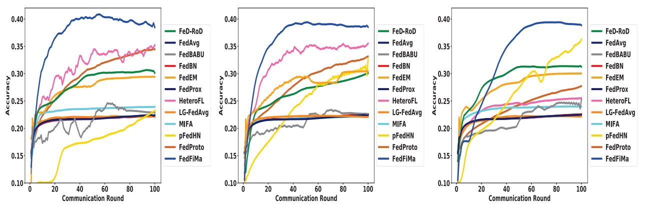
Baselines. We compare the FedFiMa model with eleven state-of-the-art federated learning algorithms, including one regular FL method, four approaches for alleviating the statistical heterogeneity problem, six algorithms for tackling the system heterogeneity issue. **FedAvg** is a classical as well as practical method for the federated learning of deep networks based on iterative model averaging [4]. **LG-FedAvg** a FL algorithm that jointly learns compact local representations on each device and a global model across all devices [40]. **FedBN** addresses non-iid training data which keeps the client BN layers updated locally, without communicating, and aggregating them at the server [41]. **FedEM** allows clients to jointly learn shared component models and personalized mixture weights in client-server and fully decentralized settings [22]. **Fed-RoD** a two-loss, two-predictor FL framework that learns personalized predictor to minimize each client's empirical risk on top of the generic predictor [22]. **FedProx** is a generalization and re-parametrization of FedAvg that addresses the challenges of heterogeneity both theoretically and empirically [7]. **HeteroFL** trains heterogeneous local models and aggregates them stably and effectively into a single global inference model [21]. **pFedHN** feeds local clients' representations to a global (across

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.



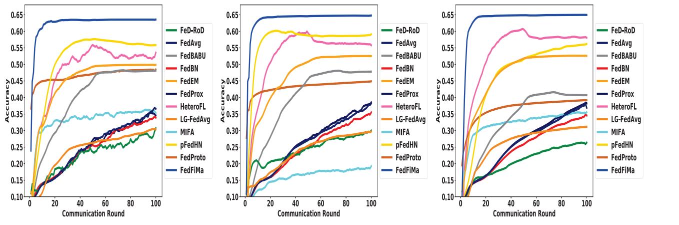
(a) #Clients = 3 (b) #Clients = 10 (c) #Clients = 50

Fig. 3: Convergence on EMNIST



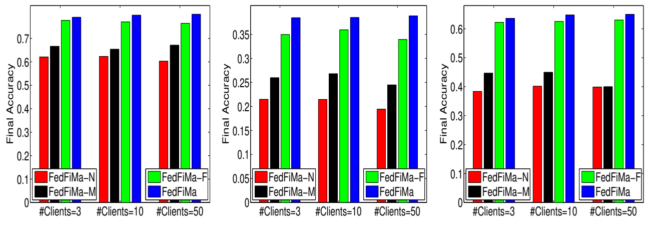
(a) #Clients = 3 (b) #Clients = 10 (c) #Clients = 50

Fig. 4: Convergence on CIFAR-10



(a) #Clients = 3 (b) #Clients = 10 (c) #Clients = 50

Fig. 5: Convergence on SVHN



(a) EMNIST (b) CIFAR-10 (c) SVHN

Fig. 6: Final Accuracy of FedFiMa variants

clients) hypernetwork, which can output personalized heterogeneous models [17]. **MIFA** is a memory-augmented impatient federated averaging algorithm under arbitrary device unavailability [42]. **FedProto** aggregates the local prototypes collected from different clients, and then sends the global prototypes back to all clients to regularize the training of local models [20]. **FedBABU** decomposes the entire network into the body with the same model structure and size for the global model training and the head for the personalized learning [22].

Variants of FedFiMa model. We evaluate four versions of FedFiMa to show the strengths of different techniques. FedFiMa utilizes our proposed fingerprint techniques with mask generation method to make full use of the computing power of each client with the customized local model for tackling the system heterogeneity issue as well as for accelerating the convergence of global model training and reducing the communication cost. FedFiMa-F makes use of the fingerprint techniques only, while FedFiMa-M employs the mask generation method only. FedFiMa-N without the support of both techniques has similar performance to FedAvg [4].

Evaluation metrics. We use one popular measures in federated learning and plot the measure curves with increasing training rounds to verify the convergence of different methods: **Accuracy** [43]–[48]. A larger Accuracy indicates a better federated learning result. We run 100 rounds of training over three datasets. In addition, we use final Accuracy to evaluate the quality of the federated learning algorithms.

Convergence on performance. Figures 3-5 exhibit the $Accuracy$ curves of twelve federated learning models for image classification through MLP on EMNIST, LeNet on CIFAR-10, and CNN on SVHN with the varying number of clients respectively. It is obvious that the performance curves by federated learning algorithms initially keep increasing with training rounds and remains relatively stable when the curves are beyond convergence points, i.e., turning points from a sharp $Accuracy$ increase to a flat curve. This phenomenon indicates that most federated learning algorithms are able to converge to the invariant solutions after enough training rounds. However, among one regular FL method, four approaches for alleviating the statistical heterogeneity problem, seven algorithms for tackling the system heterogeneity issue, our FedFiMa federated optimization method can significantly speedup the convergence on three datasets in most experiments, showing the superior performance of FedFiMa in federated settings. Compared to the learning results by other federated learning models, based on training rounds at convergence points, FedFiMa, on average, achieves 18.2%, 27.8%, and 31.4% convergence improvement on three datasets.

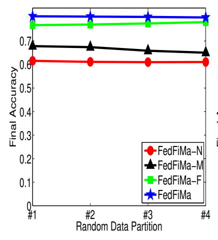
Final $Accuracy$. Table I show the quality of thirteen federated learning algorithms over EMNIST, CIFAR-10, and SVHN respectively. We have observed that our FedFiMa federated learning solution outperforms the competitor methods in most experiments. FedFiMa achieves the highest $Accuracy$ values (> 0.790 over EMNIST, > 0.385 on CIFAR-10, and > 0.636 on SVHN respectively), which are better than other eleven baseline methods in all tests. A reasonable explanation is that the combination of fingerprint and mask generation techniques is able to achieve faster convergence as well as higher test accuracy in the federated settings. In addition, the promising performance of FedFiMa over all three datasets implies that FedFiMa has great potential as a general FL solution with heterogeneous architectures of local models to learning tasks over federated datasets, which is desirable in practice.

Ablation study. Figures 6 (a)-(c) present the performance of image classification on three datasets with four variants of the FedFiMa model. We have observed that the complete FedFiMa achieves the highest accuracy, which are obviously better than other versions. FedFiMa on average, achieves 16.7%, 42.2%, and 33.1% accuracy boost on three datasets respectively. Notice that FedFiMa-F performs quite well in most experiments, compared with FedFiMa-M. A reasonable explanation is that the fingerprint techniques allow to make full use of the computing power of each client with the customized local model for improving the convergence. In

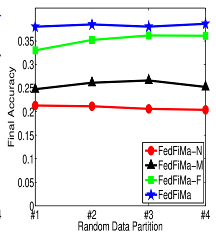
Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.

TABLE I: Final Accuracy on Three Datasets

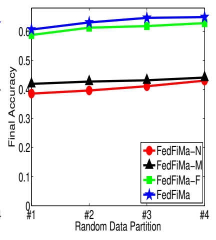
	EMNIST			CIFAR-10			SVHN		
#Clients	**3**	**10**	**50**	**3**	**10**	**50**	**3**	**10**	**50**
FedAvg	0.620	0.631	0.623	0.224	0.225	0.225	0.366	0.382	0.369
LG-FedAvg	0.615	0.699	0.629	0.221	0.221	0.222	0.304	0.297	0.312
FedBN	0.621	0.636	0.623	0.221	0.226	0.226	0.339	0.357	0.346
FedEM	0.662	0.710	0.665	0.294	0.319	0.300	0.498	0.525	0.526
Fed-RoD	0.695	0.709	0.710	0.301	0.308	0.313	0.307	0.300	0.264
FedProx	0.623	0.633	0.623	0.223	0.226	0.225	0.346	0.385	0.382
HeteroFL	0.708	0.724	0.709	0.320	0.337	0.255	0.518	0.559	0.579
pFedHN	0.635	0.685	0.697	0.232	0.327	0.363	0.558	0.591	0.562
MIFA	0.677	0.686	0.682	0.240	0.222	0.240	0.353	0.367	0.355
FedProto	0.635	0.563	0.326	0.344	0.331	0.277	0.485	0.449	0.392
FedBABU	0.743	0.743	0.743	0.229	0.259	0.252	0.481	0.478	0.407
FedFiMa	**0.790**	**0.799**	**0.803**	**0.385**	**0.385**	**0.389**	**0.636**	**0.648**	**0.650**



(a) EMNIST

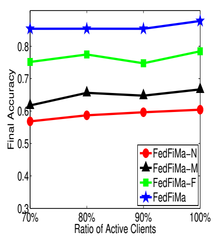


(b) CIFAR-10

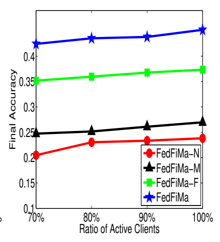


(c) SVHN

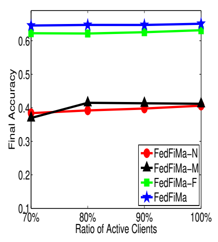
Fig. 7: Final Accuracy with Varying Data Partitions



(a) EMNIST



(b) CIFAR-10



(c) SVHN

Fig. 8: Final Accuracy with Varying Ratios of Active Clients

addition, FedFiMa-M achieves the better performance than FedFiMa-N. A rational guess is that the mask generation techniques is able to alleviate the over-parameterization issue and thus accelerate the convergence of global model training in the FL. These results illustrate both the fingerprint and mask generation techniques are important in tackling the system heterogeneity issue as well as improving the FL convergence.

Impact of data partitions. Figure 7 shows the impact of different data distributions in our FedFiMa model by randomly changing data partitions among clients with different client variance over the datasets of EMNIST, CIFAR-10, and SVHN respectively. It is observed that when changing the data partitions, the final $Accuracy$ of heterogeneous architectures of local models by our FedFiMa method keeps relatively stable, i.e., the accuracy fluctuates within the range of less than 4%. This shows the performance of our FedFiMa method is insensitive to the different data distributions. A reasonable explanation is that the customized local models always make full use of the computing power of each client for improving the FL convergence and achieving the superior accuracy.

Impact of ratios of active clients. Figures 8 (a)-(c) present the performance achieved by our FedFiMa method with varying the ratios of active clients from 70% to 100% over three datasets. It is obvious that the performance curves with each variant of our FedFiMa method keep increasing with the increased ratios of active clients. This phenomenon indicates that the accuracy in the federated settings are sensitive to the ratios of active clients. This is because the special data and system heterogeneity issues in the FL increase the difficulty in converging in a short time and the FL models need the participation of more active clients to obtain the desired learning results. However, as shown in the above experiments of convergence in Figures 3-5, our FedFiMa method presents superior convergence performance, compared with other FL algorithms, including all of one regular FL method, four approaches for alleviating the statistical heterogeneity problem, six algorithms for tackling the system heterogeneity issue.

VI. Related Work

Parallel, distributed, and federated learning have attracted active research in the last decade [31], [49]–[91]. Only recently, researchers attempt to solve the system heterogeneity issue in the federated learning by designing customized local models for each client with different hardware capabilities and configurations. The heterogeneous modeling techniques can be categorized into three types: (1) Knowledge distillation techniques use knowledge distillation to aggregate local soft predictions instead of local model parameters on the server, which allows to implement heterogeneous local models on the clients, due to the independence of model structure [10]–[16]. Most of the above knowledge distillation techniques either are based on the assumption of available public datasets in the federated setting [10]–[12], [15], or transfer the predictions among the clients and server [13], [16]. However, the access to some common public dataset which may be impractical and the use of the public datasets conflicts with the general FL setting. In addition, the prediction transferring over the Internet may result in serious privacy concerns. Although QuPeD uses personalized models with different quantization parameters and model dimensions/structures for solving the problem of personalized model compression, local models that have identical model structure and size fail to make full use of the capabilities of all clients for speeding up the convergence of global model training [14]; (2) Hypernetwork-based methods feed local clients' representations to a global hypernetwork, which can output personalized heterogeneous models [17], [18]. Unfortunately, the hypernetwork has a large memory footprint already for small clients' models. Moreover, they require each client to communicate multiple times for

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.

the server to learn meaningful representations. Hence, it is not clear if these methods can scale to more complex models in a reasonable time; and (3) Embedding-based algorithms where the clients and server communicate the learnt representations instead of the gradients, such that the FL training is independent with model structure [19], [20]. The transferring of the embeddings among the clients and server can easily raise severe privacy exposure risks. Besides the above three types of algorithms, HeteroFL proposes to upload a different subset of global model to the server for aggregation with the objective to produce a single global inference model [21]. However, HeteroFL can vary the width of hidden channels, which implies that it can train CNN-style models only in federated settings. FedBABU decomposes the entire network into the body with the same model structure and size for the global model training and the head for the personalized learning [22]. FedBABU only update the body of the model in the local update stage, i.e., the head is never updated during federated training. Therefore, the global model training still follows the standard FL strategy.

VII. Conclusions

This work proposes a novel federated fingerprint learning model for solving the system heterogeneity issue in the FL paradigm. First, the parameters of each local model are decomposed into rigid ones with the fixed model architecture for ensuring the convergence of global model training and elastic ones with the customized model structure and size for making full use of the computing power of each client. Second, a fingerprint-based aggregation strategy is developed to make the local models indirectly receive the aggregated elastic parameters through the aggregation of local fingerprints while fixing data locally. Last but not least, a parameter masking method is designed to mask the rigid parameters irrelevant to the local classification tasks in the local models.

References

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS*, 2016.

[2] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, vol. abs/1610.02527, 2016.

[3] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS 2017*, pp. 1273–1282.

[5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1-2, pp. 1–210, 2021.

[6] K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *MLSys*, 2019.

[7] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *MLSys*, 2020.

[8] G. Long, Y. Tan, J. Jiang, and C. Zhang, "Federated learning for open banking," *Lecture Notes in Computer Science*. 2020, vol. 12500, pp. 240–254.

[9] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *ECML PKDD 2018 Workshops*, pp. 21–28.

[10] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *CoRR*, vol. abs/1811.11479, 2018.

[11] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *CoRR*, vol. abs/1910.03581, 2019.

[12] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *NeurIPS*, 2020.

[13] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *ICML*, 2021, pp. 12 878–12 889.

[14] K. Ozkara, N. Singh, D. Data, and S. N. Diggavi, "Quped: Quantized personalization via distillation with applications to federated learning," in *NeurIPS*, 2021.

[15] G. Long, T. Shen, Y. Tan, L. Gerrard, A. Clarke, and J. Jiang, "Federated learning for privacy-preserving open innovation future on digital health," *CoRR*, vol. abs/2108.10761, 2021.

[16] A. Afonin and S. P. Karimireddy, "Towards model agnostic federated learning using knowledge distillation," in *ICLR*, 2022.

[17] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, "Personalized federated learning using hypernetworks," in *ICML*, 2021.

[18] O. Litany, H. Maron, D. Acuna, J. Kautz, G. Chechik, and S. Fidler, "Federated learning with heterogeneous architectures using graph hypernetworks," *CoRR*, vol. abs/2201.08459, 2022.

[19] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," in *NeurIPS*, 2020.

[20] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "FedProto: Federated Prototype Learning across Heterogeneous Clients," in *AAAI*, 2022.

[21] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *ICLR'21*.

[22] J. Oh, S. Kim, and S. Yun, "Fedbabu: Towards enhanced representation for federated image classification," in *ICLR*, 2022.

[23] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.

[24] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *ICML*, 2018.

[25] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, pp. 55:1–55:21, 2019.

[26] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *CoRR*, vol. abs/1905.01392, 2019.

[27] J. Mellor, J. Turner, A. J. Storkey, and E. J. Crowley, "Neural architecture search without training," in *ICML*, 2021, pp. 7588–7598.

[28] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Zen-nas: A zero-shot NAS for high-performance deep image recognition," in *ICCV*, 2021.

[29] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010, pp. 249–256.

[30] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016.

[31] Z. Zhang, J. Jin, Z. Zhang, Y. Zhou, X. Zhao, J. Ren, J. Liu, L. Wu, R. Jin, and D. Dou, "Validating the lottery ticket hypothesis with inertial manifold theory," in *NeurIPS*, 2021.

[32] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.

[33] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *NIPS*, 1990.

[34] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *CoRR*, vol. abs/1506.02626, 2015.

[35] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *ICLR*, 2017.

[36] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *ICLR*, 2019.

[37] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "The lottery ticket hypothesis at scale," *CoRR*, vol. abs/1903.01611, 2019.

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.

[38] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *ICML*, 2020.
[39] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS*, 2011.
[40] P. P. Liang, T. Liu, Z. Liu, R. Salakhutdinov, and L. Morency, "Think locally, act globally: Federated learning with local and global representations," *CoRR*, vol. abs/2001.01523, 2020.
[41] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," *ICLR'21*.
[42] X. Gu, K. Huang, J. Zhang, and L. Huang, "Fast federated learning in the presence of arbitrary device unavailability," in *NeurIPS*, 2021.
[43] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: stochastic controlled averaging for federated learning," in *ICML*, 2020, pp. 5132–5143.
[44] A. Mitra, R. Jaafar, G. Pappas, and H. Hassani, "Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients," in *NeurIPS*, 2021.
[45] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Trans. Parallel Distributed Syst.*, vol. 31, no. 8, pp. 1754–1766, 2020.
[46] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *ICLR'21*.
[47] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Mime: Mimicking centralized stochastic algorithms in federated learning," in *ICLR*, 2021.
[48] J. Wang, Z. Xu, Z. Garrett, Z. Charles, L. Liu, and G. Joshi, "Local adaptivity in federated learning: Convergence and consistency," in *FL-ICML*, 2021.
[49] Q. Zhang, L. Liu, Y. Ren, K. Lee, Y. Tang, X. Zhao, and Y. Zhou, "Residency aware inter-vm communication in virtualized cloud: Performance measurement and analysis," in *CLOUD*, 2013, pp. 204–211.
[50] Y. Zhou and L. Liu, "Social influence based clustering of heterogeneous information networks," in *KDD*, 2013, pp. 338–346.
[51] K. Lee, L. Liu, Y. Tang, Q. Zhang, and Y. Zhou, "Efficient and customizable data partitioning framework for distributed big rdf data processing in the cloud," in *CLOUD*, 2013, pp. 327–334.
[52] Z. Su, L. Liu, M. Li, X. Fan, and Y. Zhou, "Servicetrust: Trust management in service provision networks," in *SCC*, 2013.
[53] Y. Zhou and L. Liu, "Activity-edge centric multi-label classification for mining heterogeneous information networks," in *KDD*, 2014.
[54] B. Palanisamy, L. Liu, K. Lee, S. Meng, Y. Tang, and Y. Zhou, "Anonymizing continuous queries with delay-tolerant mix-zones over road networks," *DAPD*, vol. 32, no. 1, pp. 91–118, 2014.
[55] Q. Zhang, L. Liu, K. Lee, Y. Zhou, A. Singh, N. Mandagere, S. Gopisetty, and G. Alatorre, "Improving hadoop service provisioning in a geographically distributed cloud," in *CLOUD*, 2014, pp. 432–439.
[56] Z. Su, L. Liu, M. Li, X. Fan, and Y. Zhou, "Reliable and resilient trust management in distributed service provision networks," *ACM Transactions on the Web (TWEB)*, vol. 9, no. 3, pp. 1–37, 2015.
[57] Y. Zhou, L. Liu, K. Lee, C. Pu, and Q. Zhang, "Fast iterative graph computation with resource aware graph parallel abstractions," in *HPDC*, 2015, pp. 179–190.
[58] X. Bao, L. Liu, N. Xiao, Y. Zhou, and Q. Zhang, "Policy-driven autonomic configuration management for nosql," in *CLOUD*, 2015.
[59] Y. Zhou, L. Liu, K. Lee, and Q. Zhang, "Graphtwist: Fast iterative graph computation with two-tier optimizations," *PVLDB*, 8(11):1262–1273, 2015.
[60] K. Lee, L. Liu, K. Schwan, C. Pu, Q. Zhang, Y. Zhou, E. Yigitoglu, and P. Yuan, "Scaling iterative graph computations with graphmap," in *SC*, 2015, pp. 57:1–57:12.
[61] Y. Jiang, C.-S. Perng, A. Sailer, I. Silva-Lepe, Y. Zhou, and T. Li, "Csm: A cloud service marketplace for complex service acquisition," *ACM TIST*, vol. 8, no. 1, pp. 1–25, 2016.
[62] Y. Zhou, L. Liu, S. Seshadri, and L. Chiu, "Analyzing enterprise storage workloads with graph modeling and clustering," *IEEE JSAC*, vol. 34, no. 3, pp. 551–574, 2016.
[63] Y. Zhou, "Innovative mining, processing, and application of big graphs," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, GA, 2017.
[64] Y. Zhou, S. Wu, C. Jiang, Z. Zhang, D. Dou, R. Jin, and P. Wang, "Density-adaptive local edge representation learning with generative adversarial network multi-label edge classification," in *ICDM*, 2018.
[65] Y. Zhou, A. Amimeur, C. Jiang, D. Dou, R. Jin, and P. Wang, "Density-aware local siamese autoencoder network embedding with autoencoder graph clustering," in *BigData*, 2018, pp. 1162–1167.
[66] B. Palanisamy, L. Liu, Y. Zhou, and Q. Wang, "Privacy-preserving publishing of multilevel utility-controlled graph datasets," *ACM TOIT*, vol. 18, no. 2, pp. 24:1–24:21, 2018.
[67] K. Lee, L. Liu, R. L. Ganti, M. Srivatsa, Q. Zhang, Y. Zhou, and Q. Wang, "Lightwieight indexing and querying services for big spatial data," *IEEE TSC*, vol. 12, no. 3, pp. 343–355, 2019.
[68] J. Ren, Y. Zhou, R. Jin, Z. Zhang, D. Dou, and P. Wang, "Dual adversarial learning based network alignment," in *ICDM*, 2019.
[69] Y. Zhou and L. Liu, "Approximate deep network embedding for mining large-scale graphs," in *CogMI*, 2019, pp. 53–60.
[70] S. Goswami, A. Pokhrel, K. Lee, L. Liu, Q. Zhang, and Y. Zhou, "Graphmap: Scalable iterative graph processing using nosql," *The Journal of Supercomputing (TJSC)*, vol. 76, no. 9, pp. 6619–6647, 2020.
[71] Y. Zhou, J. Ren, D. Dou, R. Jin, J. Zheng, and K. Lee, "Robust meta network embedding against adversarial attacks," in *ICDM*, 2020.
[72] Z. Zhang, Z. Zhang, Y. Zhou, Y. Shen, R. Jin, and D. Dou, "Adversarial attacks on deep graph matching," in *NeurIPS*, 2020.
[73] S. Wu, Y. Li, D. Zhang, Y. Zhou, and Z. Wu, "Topicka: Generating commonsense knowledge-aware dialogue responses towards the recommended topic fact," in *IJCAI*, 2021, pp. 3766–3772.
[74] Y. Zhou, Z. Zhang, S. Wu, V. Sheng, X. Han, Z. Zhang, and R. Jin, "Robust network alignment via attack signal scaling and adversarial perturbation elimination," in *WWW*, 2021, pp. 3884–3895.
[75] X. Zhao, Z. Zhang, Z. Zhang, L. Wu, J. Jin, Y. Zhou, R. Jin, D. Dou, and D. Yan, "Expressive 1-lipschitz neural networks for robust multiple graph learning against adversarial attacks," in *ICML*, 2021.
[76] J. Ren, Z. Zhang, J. Jin, X. Zhao, S. Wu, Y. Zhou, Y. Shen, T. Che, R. Jin, and D. Dou, "Integrated defense for resilient graph matching," in *ICML*, 2021, pp. 8982–8997.
[77] Z. Zhang, Z. Zhang, Y. Zhou, L. Wu, S. Wu, X. Han, D. Dou, T. Che, and D. Yan, "Adversarial attack against cross-lingual knowledge graph alignment," in *EMNLP*, 2021, pp. 5320–5337.
[78] S. Wu, Y. Li, M. Wang, D. Zhang, Y. Zhou, and Z. Wu, "More is better: Enhancing open-domain dialogue generation via multi-source heterogeneous knowledge," in *EMNLP*, 2021, pp. 2286–2300.
[79] G. Zhang, Y. Zhou, S. Wu, Z. Zhang, and D. Dou, "Cross-lingual entity alignment with adversarial kernel embedding and adversarial knowledge translation," *CoRR*, vol. abs/2104.07837, 2021.
[80] S. Wu, M. Wang, D. Zhang, Y. Zhou, Y. Li, and Z. Wu, "Knowledge-aware dialogue generation via hierarchical infobox accessing and infobox-dialogue interaction graph network," in *IJCAI*, 2021.
[81] C. Zhou, J. Liu, J. Jia, J. Zhou, Y. Zhou, H. Dai, and D. Dou, "Efficient device scheduling with multi-job federated learning," in *AAAI*, 2022.
[82] D. Yan, Y. Zhou, and G. Guo, "Think-like-a-task programming model," in *Encyclopedia of Big Data Technologies*, J. T. Albert Zomaya and S. Sakr, Eds. Springer, 2022.
[83] D. Yan, W. Qu, G. Guo, X. Wang, and Y. Zhou, "Prefixfpm: A parallel framework for general-purpose mining of frequent and closed patterns," *The VLDB Journal (VLDBJ)*, vol. 31, no. 2, pp. 253–286, 2022.
[84] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From distributed machine learning to federated learning: A survey," *KAIS*, vol. 64, no. 4, pp. 885–917, 2022.
[85] L. Y. J. K. C. L. Z. J. Guimu Guo, Da Yan and Y. Zhou, "Maximal directed quasi-clique mining," in *ICDE*, 2022.
[86] H. Zhang, J. Liu, J. Jia, Y. Zhou, H. Dai, and D. Dou, "Fedduap: Federated learning with dynamic update and adaptive pruning using shared data on the server," in *IJCAI*, 2022.
[87] Y. Zhou, J. Ren, R. Jin, Z. Zhang, J. Zheng, Z. Jiang, D. Yan, and D. Dou, "Unsupervised adversarial network alignment with reinforcement learning," *ACM TKDD*, vol. 16, no. 3, pp. 50:1–50:29, 2022.
[88] Y. Zhou and L. Liu, "Clustering analysis in large graphs with rich attributes," in *Data Mining: Foundations and Intelligent Paradigms: Volume 1: Clustering, Association and Classification*, D. E. Holmes and L. C. Jain, Eds. Springer, 2012.
[89] J. Jin, Z. Zhang, Y. Zhou, and L. Wu, "Input-agnostic certified group fairness via gaussian parameter smoothing," in *ICML*, 2022.
[90] J. Jin, J. Ren, Y. Zhou, L. Lv, J. Liu, and D. Dou, "Accelerated federated learning with decoupled adaptive optimization," in *ICML*, 2022.
[91] Z. Zhang, Y. Zhou, X. Zhao, T. Che, and L. Lyu, "Prompt certified machine unlearning with randomized gradient smoothing and quantization," in *NeurIPS*, 2022.

Authorized licensed use limited to: National University of Singapore. Downloaded on May 19,2023 at 08:40:35 UTC from IEEE Xplore. Restrictions apply.