DistFL: Distribution-aware Federated Learning for Mobile Scenarios

BINGYAN LIU*, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University
YIFENG CAI*, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University
ZIQI ZHANG, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University
YUANCHUN LI, Microsoft Research
LEYE WANG, Dept of Computer Science, School of EECS, Peking University
DING LI, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University
YAO GUO†, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University
XIANGQUN CHEN, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University

Federated learning (FL) has emerged as an effective solution to decentralized and privacy-preserving machine learning for mobile clients. While traditional FL has demonstrated its superiority, it ignores the non-iid (independently identically distributed) situation, which widely exists in mobile scenarios. Failing to handle non-iid situations could cause problems such as performance decreasing and possible attacks. Previous studies focus on the "symptoms" directly, as they try to improve the accuracy or detect possible attacks by adding extra steps to conventional FL models. However, previous techniques overlook the root causes for the "symptoms": blindly aggregating models with the non-iid distributions. In this paper, we try to fundamentally address the issue by decomposing the overall non-iid situation into several iid clusters and conducting aggregation in each cluster. Specifically, we propose **DistFL**, a novel framework to achieve automated and accurate **Dist**ribution-aware **F**ederated **L**earning in a cost-efficient way. DistFL achieves clustering via extracting and comparing the *distribution knowledge* from the uploaded models. With this framework, we are able to generate multiple personalized models with distinctive distributions and assign them to the corresponding clients. Extensive experiments on mobile scenarios with popular model architectures have demonstrated the effectiveness of DistFL.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Computing methodologies** → **Neural networks**.

Additional Key Words and Phrases: federated learning, neural networks, distribution knowledge, privacy

---

*The first two authors contributed equally.
†Correspondence to: Yao Guo.

Authors' addresses: Bingyan Liu, lby_cs@pku.edu.cn, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University, Beijing, China, 100871; Yifeng Cai, caiyifeng@pku.edu.cn, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University, Beijing, China, 100871; Ziqi Zhang, ziqi_zhang@pku.edu.cn, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University, Beijing, China, 100871; Yuanchun Li, Yuanchun.Li@microsoft.com, Microsoft Research, Beijing, China, 100871; Leye Wang, leyewang@pku.edu.cn, Dept of Computer Science, School of EECS, Peking University, Beijing, China, 100871; Ding Li, ding_li@pku.edu.cn, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University, Beijing, China, 100871; Yao Guo, yaoguo@pku.edu.cn, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University, Beijing, China, 100871; Xiangqun Chen, cherry@pku.edu.cn, MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University, Beijing, China, 100871.
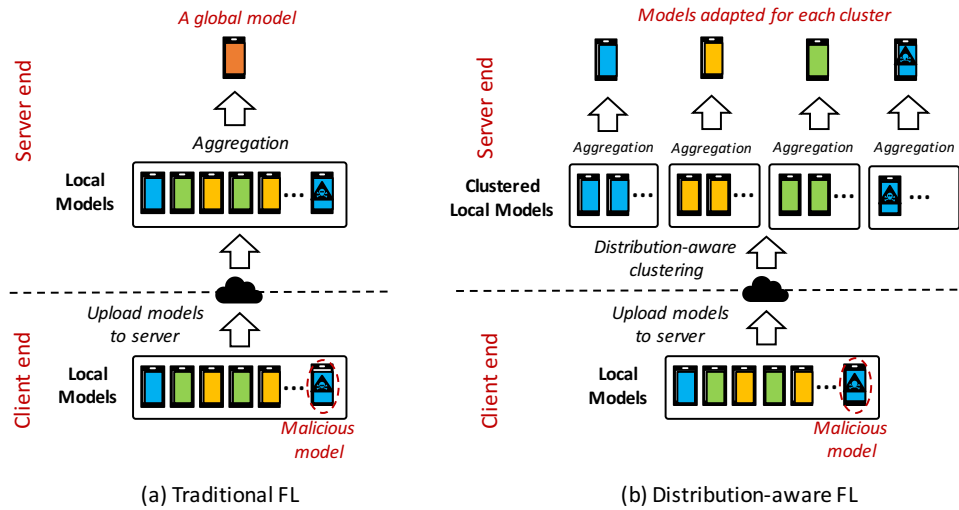

Fig. 1. Comparison between traditional FL and our distribution-aware FL. Here different colors represent different distributions.

1 INTRODUCTION

As General Data Protection Regulation (GDPR) [54] has been officially proposed to guarantee user privacy, it is no longer appropriate to transfer user data for cloud centralized training. Under this condition, *federated learning* (FL), which enables a number of devices to collaboratively generate a global model without exposing their client data, has become an important and emerging research topic in recent years [56, 57, 61]. A conventional FL solution consists of three steps: (1) the server end distributes a model to each client [1]; (2) each client trains the offloaded model locally and uploads it to the server; (3) the server collects and coordinates these models by some aggregation algorithms (e.g., FedAvg [40]) to generate a global model that owns the knowledge of each client. These three steps may repeat many times until the model converges.

Despite achieving tremendous success, conventional FL techniques may not be practical in certain scenarios that involve mobile devices. Existing FL techniques assume that the overall data distributions of all clients are iid (independently identically distributed). However, in mobile scenarios, this assumption may not always be valid. Data collected by a mobile device can be strongly biased on its context. For instance, pictures collected by a white-collar office worker may only contain indoor views while data from a traveller may be biased on outdoor landscapes. Simply averaging the models trained on these biased data sets may not achieve desired results.

Violating the iid assumption could be problematic for existing FL techniques in two ways. First, it may dramatically decrease the accuracy of trained machine learning models. In our experiment, conventional FL techniques, such as FedAvg [40], can only achieve an accuracy of 21% to 40% with strongly biased data tasks.

---

[1]In the rest of this paper, we use *client* to refer to *mobile device*.

Second, in an even worse scenario, a malicious client can deliberately create a poisonous data set that misleads the global model to wrong predictions [3, 51].

The research community has noticed the problem of violation of the iid assumption and many techniques have been proposed to fix the *symptoms*. For example, some researchers propose to achieve personalization by manually conducting a loss-based clustering [12] or a local adaptation after the FL process [29], which can improve the accuracy of traditional FL on non-iid data. People also propose to utilize model weights [28, 59] or borrow extra data [23] to assist the detection of malicious clients and remove them. Unfortunately, existing techniques suffer from limitations such as requiring massive manual tuning or lacking of sufficient accuracy.

We notice that the techniques mentioned in the above paragraph only focus on the "symptoms" rather than the root cause. Conventional FL techniques are inaccurate or vulnerable because data from clients are usually from different distributions while previous techniques still blindly aggregate models from these clients. They just specifically improve the accuracy or detect malicious clients by adding extra steps to the conventional FL process. However, these techniques fail to identify, distinguish and process clients with different data distributions separately, which is the fundamental cause of the symptoms of conventional FL techniques.

Unfortunately, to the best of our knowledge, there is no existing technique that directly addresses the root cause of non-iid data distribution issues in FL. To fill the gap, in this paper, we try to fundamentally address the issue caused by the non-iid situation by proposing **distribution-aware FL**, a technique that clusters uploaded local models in terms of their *distribution knowledge* and then aggregates them in each cluster. In this way, the overall non-iid situation can be approximately decomposed into several iid clusters and conventional FL techniques could be applied to each iid cluster. We believe this distribution-based clustering is possible because usually there are a large number of clients involved in the mobile FL system, some of which will inevitably come from the identical/similar data distribution and thus generate models with a similar distribution. For example, some mobile users may have a similar preference and thus collect data with a similar distribution.

Specifically, we propose **DistFL**, a framework that aims at achieving automated and accurate **Dist**ribution-aware **F**ederated **L**earning in a cost-efficient way. As shown in Figure 1, compared with traditional FL, our framework conducts a selective aggregation step based on the results of distribution-aware clustering. In this way, we are able to generate several personalized models with distinctive distributions and distribute them to corresponding clients with a similar distribution. Note that the malicious clients can also be clustered into a single group, eliminating their influence on normal clients and making FL more robust.

Achieving distribution-aware FL is non-trivial. The key challenge is how to identify the data distribution of each client. Note that in FL, data on each client are private. DistFL cannot directly access these data. In other words, DistFL needs to infer the data distribution of each client based on the uploaded models. However, directly extracting input data distribution from a DNN could be hard as the DNN is usually viewed as a black box whose knowledge is uninterpretable. To address this challenge, we borrow the idea of synthesis technique [39]: we use the uploaded models to generate a set of bogus data and use it to measure the similarity of models. We say two models are similar if they generate similar classification results on the same data set, which is the generated bogus data set in our case. Here the classification results refer to the probability distribution output at the last layer of the model, which is applicable to any DNN architectures. By measuring the similarity between models, we can infer the data distribution that these models are generated from so that we can cluster models that are trained with similar data distributions.

Clustering uploaded models with the inferred bogus data set requires overcoming three technical challenges. We list them as follows:

- *How to effectively generate bogus data that reflect the distribution of data from a model?* A straight forward idea to generate bogus data is to directly use existing synthesis techniques. However, this simple idea suffers three main limitations: (i) existing synthesis techniques pay more attention to the label information,

which can reflect the specific category features while may not benefit the clustering process on models with different distribution features; (ii) The synthesis process is computationally expensive especially when the number of labels is huge (e.g., The ImageNet dataset [7] has 1000 labels); (iii) Existing synthesis techniques focus on image tasks and do not try to generalize to other domains. In a word, it is not appropriate to use this method directly to generate representations of data distribution.

- *How to guarantee the efficiency during the generation process?* In real-world mobile scenarios the number of clients might be huge, which means that if we directly generate the distribution knowledge from each model, the computation costs may be unacceptable. So there is a high demand to design some optimization techniques in order to accomplish this process efficiently. This is a challenging problem considering the tremendous number of clients.

- *How to automatically and accurately figure out the cluster strategy?* Unlike the previous work [12] that accomplishes the clustering process by manually setting the number of clusters and using loss-based metric, we seek to an automated and precise method to decide the cluster information with the help of the extracted distribution knowledge.

DistFL addresses the first challenge mainly based on the following observation: *The statistical information, which can represent the distribution knowledge [33], are exactly encoded in the widely used BatchNorm (BN) layers (i.e., running_mean, running_variance) of a DNN.* This finding inspires us to generate some "data" that can well fit the statistics of the BN layers. With the help of the synthesis technique [39], we are able to finally generate the "data" with distribution knowledge. To address the second challenge, we design two optimization methods to make the generation faster, ensuring the efficiency of the whole extraction process. To address the third challenge, we introduce a distribution-based clustering technique, where we feed the extracted "data" into each uploaded model and obtain the corresponding probability distribution vector at the Softmax layer. By calculating the KL divergence between them, we are able to generate a similarity matrix, which provides guidance for clustering by setting an apparent threshold, getting rid of manual tuning. In terms of the clustering results, we finally conduct aggregation inside each cluster and generate several personalized models for corresponding clients.

To summarize, DistFL consists of the following three steps: (1) a distribution extraction step to generate some data with distribution knowledge based on the uploaded models (see §4.2); (2) a distribution-based clustering step to achieve automated and accurate clustering process by utilizing the generated distribution knowledge (see §4.3); (3) an aggregation step to selectively federate models with similar distributions (see §4.4). Besides, we would like to highlight that our framework does not need any raw data or feature information. Instead, we only operate the uploaded models in the server end, which indicates that our approach can be seamlessly integrated into the traditional FL pipeline to effectively cope with the existing non-iid situation.

We evaluate DistFL on four simulated mobile scenarios since it is difficult to conduct FL in real-world mobile applications. The simulated scenarios include *category-imbalance* scenario, *environment-difference* scenario, *privacy-protection* scenario and *attack-injection* scenario, where we use public computer vision (CV) datasets and human activity recognition (HAR) datasets for concrete simulation (see §5.1). Experimental results on these scenarios show that our proposed framework can reduce the relative error to other state-of-the-art personalization methods by up to 33.68%. In addition, we are able to accurately detect the malicious clients and remove their influence, especially when the scale of malicious clients is large. For example, when there are 80% attackers in the FL system, our approach can still achieve 8.05% accuracy improvement and 7.33% ASR (attack success rate) dropping compared to other defense methods, which validates the superiority of DistFL. Finally, we conduct an experiment on a real-world dataset and provide some in-depth analyses to further illustrate the effectiveness of the proposed framework.

The contributions of this paper are summarized as three folds.

- We propose *distribution-aware federated learning*, a new paradigm to improve both the personalization and robustness performance of FL for mobile scenarios. By clustering similar models into iid groups and dealing with them separately, we can significantly mitigate the data heterogeneity and malicious clients influence caused by the non-iid situation.
- We design and implement DistFL to accomplish our goal, which achieves clustering through extracting and utilizing the distribution knowledge from the federated models. The whole process is efficient, automated and accurate. To the best of our knowledge, this is the first framework to explore and study the paradigm of distribution-aware federated learning.
- We conduct extensive experiments on multiple scenarios with different model architectures. The results demonstrate that the proposed DistFL performs better than state-of-the-art approaches.

2 RELATED WORK

This section presents a brief summary of existing literature related to our study.

2.1 Mobile Federated Learning

Federated learning (FL) is a prevalent distributed machine learning paradigm with improved privacy, which has been applied in many privacy-sensitive areas. For example, Google has developed the first product-level federated learning application *Gboard* for next-word prediction [16]. NVIDIA has begun to cooperate with medical institutions aiming at building the first federated AI platform for medical diagnosis and drug researches [36]. In addition, both industry and academic community have released some basic FL frameworks or benchmarks to support and contribute this promising field [6, 17, 47, 55]. According to a recent survey [27], federated learning can be categorized into two typical types by the scale of federation: *cross-silo* and *cross-device*. Here we mainly focus on the cross-device setting, where the participants are a large number of mobile or wearable devices rather than big organizations or data centers. In other words, this paper pays more attention to **mobile federated learning**, which has also been widely studied in recent years. For instance, Feng *et al.* [8] borrowed the idea of federated learning to predict human mobility. Samarakoon *et al.* and Lu *et al.* [37, 49] designed a distributed federated learning system for connected vehicles. Guo *et al.* [15] proposed an edge-accelerated federated learning framework for POI recommendation. Next we describe some existing issues and investigate corresponding solutions in the context of mobile federated learning.

2.1.1 Data Heterogeneity Problem and Personalization on Mobile FL. A key problem in mobile FL is that the data on the users' personal devices are usually non-iid (independently identically distributed) [21] due to different user environments or user preference. Faced with the challenge of statistical differences, a series of studies have been proposed to personalize the model. In particular, Zhao *et al.* [61] maintained a shared dataset across clients to improve training on non-iid data, which is impractical since it is intractable to find such a dataset and may cause privacy leakage. Liang *et al.* [29] used traditional FL to generate a global model and further fine-tune it with data in each local client. However, this method needs massive computation costs and cannot completely address the data heterogeneity problem since it still requires a traditional FL process (i.e., aggregating models with different distributions) before fine-tuning. Ghosh *et al.* [12] proposed the K-Cluster method to conduct FL, where they first set K clusters manually before different clients are respectively clustered and optimized in terms of the output loss. However, loss-based clustering is not accurate since the loss cannot reflect valuable distribution knowledge in the model. Moreover, this clustering algorithm needs to set the number of clusters manually before learning, which leads to big uncertainty because nobody has the ability to give a precise judgment to the number setting in advance. Different from these methods, DistFL partitions clients through inferring the data distribution directly from federated models, making the clustering process more efficient, automated and accurate.

2.1.2 Attack on Mobile FL. DNN attack is a hot and important research topic even before the emergence of FL [14, 35, 48]. The key idea is to inject some triggers inside the model to achieve malicious behaviors. Unlike existing attacks that target only one model in the centralized scenario, in mobile FL, the attack can happen in a single or several clients and is introduced on their local models to mislead the global model. We believe attacks on mobile FL are harder to defend because it is difficult to detect and locate the source of malicious behaviors considering the tremendous number of clients.

A recent survey [38] has summarized two types of attacks under the FL scenario: poison attacks [3, 4, 10, 51] and inference attacks [20, 41, 44, 62]. The goal of poison attacks is to induce the FL model to output the target label specified by the adversary. For example, Tolpegin *et al.* [51] implemented data poison attack by flipping the labels of training data from one class to another class in the local training epoch to mislead the global model output. Bagdasaryan *et al.* [3] conducted model poison attack by replacing the local model with elaborate weights, which can achieve a significantly higher attack success rate than data poison attacks. For inference attacks, model updates are utilized to infer an amount of private information, such as class representatives [20], membership as well as properties associated with a subset of the training data [41].

2.1.3 Defense on Mobile FL. Many efforts have been made to design defensive strategies to resolve attacks on mobile FL. Towards poison attacks, Konstantinov & Lampert [23] proposed the idea to maintain a small dataset in the server to provide extra guidance to mitigate the attack. However, the server end may not own such data and it is difficult to determine which data should be collected. Besides, Li *et al.* [28] proposed to use conv layers' weights of uploaded models to build an auto-encoder to detect malicious clients and Yu & Wu [59] attempted to directly use the weights to distinguish malicious clients for robust aggregation. Fu *et al.* [9] applied a weighting scheme to give a low weight to uploaded attacked models, mitigating their negative influences. Different from these methods that devote to borrowing extra data or simply utilizing model weights, our approach isolates these malicious clients into a single cluster by detecting whether their *distribution knowledge* is abnormal, which is more accurate and efficient.

For inference attacks, researchers applied differential privacy (DP) [1] to federated learning by (1) clipping each client's update, and (2) adding random noise [11, 60]. As a result, the privacy property of user data can be formally protected. In our experiments, we also implement DP on the client local model to defend the attack and test whether our approach is still effective under this *privacy-protection* scenario.

2.2 Knowledge Extraction from DNNs

DistFL is motivated by the idea of synthesis while paying more attention to generate *distribution knowledge* in an efficient manner. Generally, DistFL's process of generating bogus data can be categorized as the problem of extracting knowledge from DNN models. There are two lines of such work. The first is to use knowledge distillation (KD) [19], where the model knowledge can be extracted/transferred to another model by mimicking the output or intermediate features. During the last few years, KD has been widely used in various fields, such as model compression [30, 31, 42], compact neural network architecture design [46] and semantic segmentation [34]. However, KD relies on the raw data to conduct training on the server end, which is unacceptable for FL due to privacy concerns. Besides, image synthesis [39] can also be used to conduct knowledge extraction, where the model knowledge is derived to image-like data. For example, DeepDream [43] and DeepInversion [58] tried to synthesize high fidelity and high resolution natural images from a trained DNN by gradient techniques. However, most of them focus on concrete label information and require massive computation costs.

3 PRELIMINARIES

This section starts with a problem analysis and then explains the *distribution knowledge* in detail. Finally, we formulate our objective.

3.1 Problem Analysis

The aim of this subsection is to analyze how the data heterogeneity problem and the client malicious behaviors affect the FL process. Here we do not consider the inference attacks since they do not inject any malicious behaviors to the local model and thus will not harm FL.

For the data heterogeneity problem, each client owns a specific data distribution due to its environment or preference, which naturally satisfies the non-iid situation. For malicious behaviors in FL (i.e., poison attacks), they try to mislead specific data to wrong results, which can be considered to inject abnormal data information into the model. In other words, these attacks change the original data distribution of clients and also form a type of non-iid situation (normal and abnormal). So we can find that both of the issues come from aggregating models with non-iid distributions, which motivates us to address them fundamentally from a new perspective: *Is it possible to decompose the non-iid situation into several iid situations by distribution-aware clustering?* If this clustering can be achieved, the data heterogeneity problem will disappear and models with malicious behaviors can be clustered into a single group, getting rid of their harmful influence on normal clients. In this paper, we design and implement distribution-aware FL to accomplish our goal.

3.2 Statistical Information and Distribution Knowledge

3.2.1 Statistical Information. Here the statistical information refers to a type of statistic that can well describe the overall distribution situation of a dataset. Usually the distribution parameters such as *mean* and *var* are used to represent the information [33]. Note that these parameters are exactly encoded in the widely used BatchNorm (BN) layers (i.e., running_mean, running_variance) of a modern DNN model, which can be easily obtained in our scenario because all of the local models have been uploaded to the central server.

3.2.2 Distribution Knowledge. In our work, we attempt to extract the *distribution knowledge* from a DNN model to implement the later clustering. Intuitively, the distribution knowledge characterizes the situation of data distribution. In real-world applications, the same object image collected from different environments, the same activity coming from different people, the number of certain data based on different users, or some malicious behaviors injected by different attackers can be considered as typical examples that can lead to various distributions. In other words, the *distribution knowledge* indicates that the specific feature information (e.g., the feature of the object image), which may be related to certain data, is identical while the overall data distribution situation (e.g., the environment that the object images are in) may be significantly different. Therefore, we believe the distribution knowledge has a close relation to environments, special user preferences, and targeted behaviors, which is commonly seen in mobile scenarios.

In order to extract the *distribution knowledge*, we first denote it as a set of bogus data generated from an uploaded model that complies to the original distribution of the model's input data. As aforementioned, the original data distribution can be characterized by the statistical information (i.e., distribution parameters), which motivates us to generate data whose statistical information is similar to the original ones. Specifically, we employ the synthesis technique to optimize the noises to fit the distribution parameters for generating the *distribution knowledge*. Details can be founded in Section 4.2.

3.3 Problem Formulation

We now introduce the symbols and annotations to formally define the optimization objective. Generally speaking, there are two parties involved in the FL process: *client end* and *server end*. The client end owns local datasets $D = \{D_1, D_2, ..., D_N\}$ and use them to conduct training, generating local models $M = \{M_1, M2, ..., M_N\}$. Here N denotes the number of clients. The server end collects these models and aggregates them to generate a global model $M_{global} = Agg(M_1, M_2, ..., M_N)$, where Agg represents the aggregation algorithm (e.g., FedAvg). Unlike the

traditional FL that outputs a global model, our objective is to generate several personalized models based on different distributions. In order to provide a formal definition of *distribution-aware federated learning*, we first give an assumption.

Assumption 1. *Suppose X represents the data distribution. In the mobile FL system, there exist Q types of data distributions $X_1, X_2, ..., X_Q$ and each local model $M_i \in M$ reflects one of them.*

This assumption is commonly seen in the practical mobile FL scenario, where different distributions may come from diverse environments, user preference or injected malicious behaviors. Note that any client can find at least one client with similar distributions considering the huge number of clients in the mobile FL system and thus $Q < N$. Based on the assumption, we define *distribution-aware federated learning* as follows.

Definition 1. ***Distribution-aware Federated Learning (DistFL)**: Suppose clients can be ideally partitioned into Q clusters $(G_1, G_2, ..., G_Q)$ in terms of their data distributions. At the aggregation stage, instead of aggregating all models as the traditional FL does, we conduct aggregation algorithms inside each cluster, generating Q personalized models $(M_p^1, M_p^2, ..., M_p^Q)$ for clients, getting rid of the influence of the non-iid situation.*

To accomplish DistFL, we first construct a mapping $F : M_{target} \rightarrow K_{target}$, in order to map the target model M_{target} to K_{target}, where K_{target} is the distribution knowledge embedded in M_{target}. In terms of K_{target}, we attempt to characterize the relation among the models (clients) and allocate them into different clusters $(G_1, G_2, ..., G_Q)$. In the following sections, we introduce a series of techniques to obtain K_{target}, the concrete cluster information $(G_1, G_2, ..., G_Q)$ and the final personalized models $(M_p^1, M_p^2, ..., M_p^Q)$.

4 THE DISTFL FRAMEWORK

This paper proposes an extraction-clustering based framework to achieve DistFL defined in Definition 1. In this section, we first give an overview of our framework and then describe each of the core components in detail.

4.1 Overview

We first discuss the overall process of DistFL.

Figure 2 illustrates the three key components in our framework, which we briefly summarize as follows:

- **Distribution extraction**: We first need to extract distribution knowledge for later clustering. Specifically, we randomly initialize a noise and feed it into an uploaded model(*step 1*). Then in *step 2* and *step 3*, we select the extraction targets that may contain distribution information of the models and adapt the noise to data with distribution knowledge using backpropagation (i.e., synthesis techniques [43, 58]). In order to make the process more efficient, we conduct a *pre-aggregation* scheme to generate a global model and only extract the knowledge from it. Besides, the extraction targets of the model are decided by an *importance-based target selection* method, which aims to remove some redundant information to save adaptation costs. As a result, we are able to extract the distribution knowledge of the target model (here $M_{target} = M_{global}$) and generate K_{target}.
- **Distribution-based clustering**: After obtaining the distribution knowledge K_{target}, we feed them into each uploaded model to observe their response (*step 4*). Concretely, we generate the probability distribution vector at the Softmax layer (*step 5*). By calculating the KL divergence [52] between these vectors, we are able to construct a similarity matrix, which provides guidance to cluster models without manual efforts (*step 6* and *step 7*). In this way, the cluster information $(G_1, G_2, ..., G_Q)$ is formed.
- **Aggregation**: Instead of blindly aggregating all of the uploaded models as traditional FL does, our approach conducts a selective aggregation process by only aggregating models inside the same cluster, generating corresponding personalized models $(M_p^1, M_p^2, ..., M_p^Q)$ for clients with different data distributions.
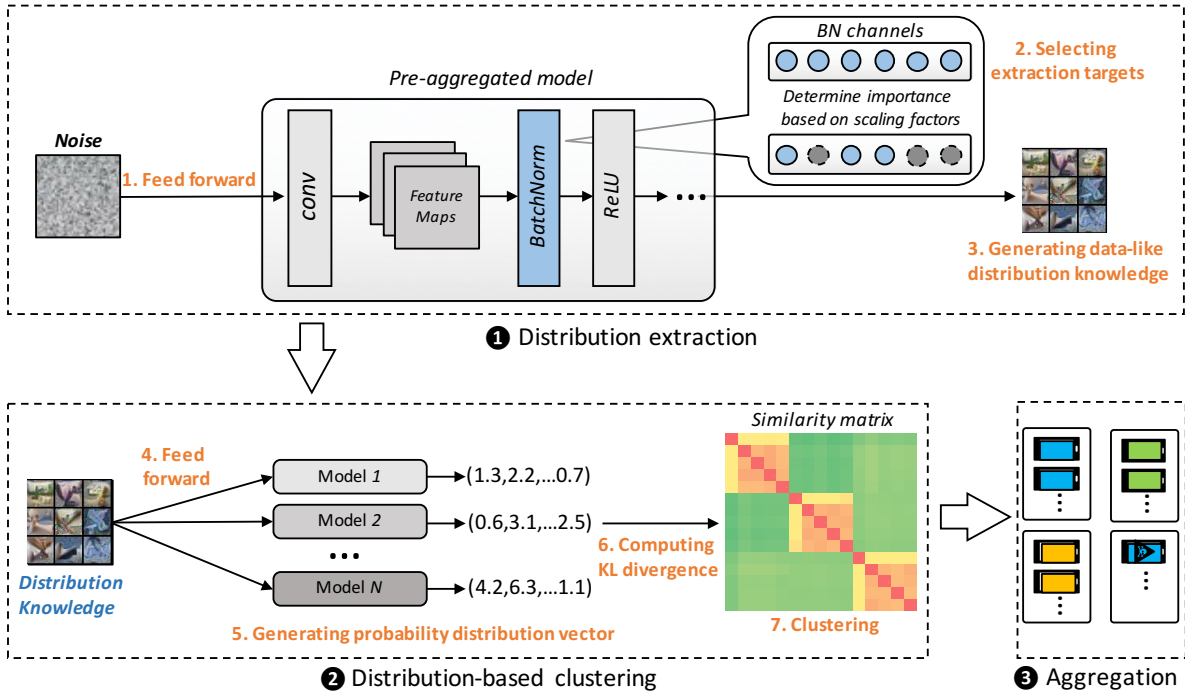
Fig. 2. Pipeline of our framework. The *distribution extraction* step extracts the distribution knowledge from the target model and the *distribution-based clustering* step leverages the extracted knowledge to cluster models with similar data distributions. Finally, the *aggregation* step aggregates models inside each cluster, generating specialized models for each cluster.

We dynamically implement these steps in each round of FL until convergence. Note that the whole process is operated in the server, without consuming extra resources on the resource-constrained clients.

4.2 Distribution Extraction

The first step is to extract the distribution knowledge from uploaded models. Given an uploaded model, DistFL generates the bogus input that represents the input data distribution of the model. To achieve this, we borrow the idea of the image synthesis technique [43], whose goal is to synthesize high fidelity and high resolution natural images given a single model, which can be regarded as the knowledge of the model. Here we do not use GAN [13] because it requires not only the model but also the user raw data to generate bogus data, which is unrealistic in our scenario.

Specifically, given a randomly initialized noise $\hat{x} \in \mathcal{R}^{C \times H \times W}$, and a target label y, the synthesis technique is able to generate label-related images (knowledge) by optimizing the following objective

$$\min_{x^r} \mathcal{L}(\hat{x}, y) + R(\hat{x}) \tag{1}$$

where C, H, and W are the number of channels, height and width of the noise. x^r is the reconstructed image. $\mathcal{L}(\cdot, \cdot)$ denotes the loss function and $R(\cdot)$ represents an image regularization term to improve the visual quality.

In our work, different from previous works that devote to generating high fidelity natural images with target labels, we focus on optimizing the noises to fit the statistics encoded in the BatchNorm (BN) layers (i.e.,

running_mean, running_variance) of the uploaded model as much as possible, no matter whether the generated items are natural images or not. Therefore, our approach can be applied to any tasks by changing the dimension of the initialized noise. For instance, to the task of human activity recognition (HAR), the noise is a 1-dimension vector, which can also achieve good performance as shown in our experiments (see 5.3). Here we still take the image task as an example for a better comparison to traditional synthesis techniques.

Concretely, given a series of noises $\hat{X} = \{\hat{x_1}, \hat{x_2}, ..., \hat{x_z}\}$ (z represents the number of noises) and the target model M_{target}, we attempt to address an optimization problem as follows

$$\min_{K_{target}} \sum_{i=1}^{H} \left(||\mu_i(\hat{X}) - BN_i(running\_mean)||_2 + ||\sigma_i^2(\hat{X}) - BN_i(running\_variance)||_2 \right) \qquad (2)$$

where $K_{target} = \{k_{target}^1, k_{target}^2, ..., k_{target}^z\}$ is the reconstructed distribution knowledge from the target model M_{target}. $\mu_i(\hat{X})$ and $\sigma_i^2(\hat{X})$ denote the mean and variance estimates of feature maps to the i_{th} layer, which are generated by feeding the noises into the model for an inference process. H represents the number of BN layers of the model. $BN_i(running\_mean)$ and $BN_i(running\_variance)$ are the parameters in the i_{th} BN layer, which can reflect the statistical information. $|| \cdot ||_2$ represents the L_2 norm. After solving this optimization problem by gradient-descent based back propagation, we can obtain the final distribution knowledge K_{target}.

4.2.1 Efficiency Optimization 1. The naive approach of knowledge extraction is to generate the distribution knowledge for each of the uploaded models respectively. However, in an FL system, we may have a huge number of clients, each of which holds a local model. Thus directly extracting the distribution knowledge from all uploaded models could be time-consuming and computationally expensive. To improve the efficiency of knowledge extraction, instead of extracting the distribution from all of the uploaded models, we apply a *pre-aggregation* scheme to generate a global model and use it to extract the distribution knowledge. The intuition is that no matter what changes the attackers make to their local models, the aggregation process must be enabled. Otherwise, FL cannot be continued, let alone injecting malicious behaviors. Besides, the aggregated model must be a usable model for later deployment, indicating that we are able to extract the distribution knowledge from it although the distribution may be abnormal.

Specifically, we first generate the global model M_{global} by averaging all of the uploaded models as the FedAvg algorithm does [40]:

$$M_{global} = \frac{1}{N} \sum_{i=1}^{N} M_i \qquad (3)$$

The generated M_{global} can be regarded as the target model M_{target}.

4.2.2 Efficiency Optimization 2. As DNNs become larger and larger for processing more complex tasks, the number of BN layers have also increased dramatically. Under this situation, selecting all of the BN layers to conduct aforementioned optimization may cause considerable computation costs. Therefore, to make the process more efficient, we present an *importance-based target selection* method to selectively extract the *important* channels of BN layers to implement the optimization. Here the importance is measured by the *scaling factor* attached in each BN channel and we pick out $E\%$ (i.e., the ratio of selected channels to the whole channels of the BN layer) channels with the larger factor value in each layer. Thus, the optimization problem is reformulated as

$$\min_{K_{target}} \sum_{i=1}^{H} \left(||\mu_i(\hat{X}) - BN_i^S(running\_mean)||_2 + ||\sigma_i^2(\hat{X}) - BN_i^S(running\_variance)||_2 \right), \qquad (4)$$

where S represents the selected BN channel set in the corresponding layer.

In terms of Equation 4, we are able to generate the distribution knowledge of the target model M_{target} and obtain the final K_{target}.

4.3 Distribution-based Clustering

This component determines the concrete cluster information $(G_1, G_2, ..., G_Q)$ based on the extracted K_{target}. Because the extracted knowledge is in a form of "data", we can directly feed them into each uploaded model to observe the responses, which can then be used to guide clustering.

Our intuition is that, given the same distribution knowledge, models with similar distributions will give similar responses. Therefore, clustering models has the same effect as clustering their corresponding responses, which simplifies our goal. Specifically, given an uploaded model M_c and the target distribution knowledge K_{target}, we feed each input k_{target}^i into the model and concatenate the probability distribution vector at the final Softmax layer

$$V_c = concat\{M_c(k_{target}^i)\}_{i=1,2,...,z} \tag{5}$$

In this way, we generate $V_1, V_2, ..., V_N$ from each uploaded model to represent its response to the certain distribution knowledge (i.e., K_{target}). Notice that the responses generated from each model are a series of probability distribution vectors, and thus we can calculate their similarity with the KL divergence [25], which has been proved effective in measuring the difference degree between two probability distributions. Next, we calculate the KL divergence between these vectors as follows

$$Div(p, q) = D_{KL}(V_p || V_q) \tag{6}$$

Based on Equation 6, we can generate a similarity matrix SIM, where SIM_{pq} represents the distribution similarity degree for the p_{th} model to the q_{th} model and the lower value implies higher similarity. In terms of SIM, we are able to easily determine the number of clusters and which model is in which cluster (i.e., $(G_1, G_2, ..., G_Q)$) by setting a threshold (the threshold is apparent as shown in the right part of Fig 6, where every 5 clients with similar distributions have a significantly higher similarity compared to others and thus can be clearly clustered), getting rid of the complex and inaccurate manual efforts.Specifically, given an uploaded model, we set a threshold to pick out the similar models based on SIM to form a rough iid cluster. For other unselected models, we follow the above step and form another iid cluster. In this way, we can finally partition these models into different iid clusters without manual setting of the concrete number of clusters.

Besides, we would like to point out that although we do not know the concrete information of the dataset or problem settings (e.g., how many clusters), the manual threshold is always effective because the computed matrix SIM has clearly described the difference between models generated from the iid data distribution and the non-iid data distribution, which benefits the threshold-based partition. In other words, the value of SIM is close in the iid group while varying greatly between two groups (i.e., iid and non-iid). Therefore, the threshold can be always and easily generated in terms of the distance among the SIM values and we finally form different model groups based on it. In summary, what makes the threshold works is that the uploaded models intrinsically come from different distributions according to our Assumption 1 and the matrix SIM is able to characterize their similarity, which make the threshold feasible in partitioning them.

4.4 Aggregation

Model aggregation is the final step to finish FL. A prevalent and effective aggregation is FedAvg [40], which has been proved effective especially when the data is iid. Here we adopt this aggregation algorithm because we have partitioned uploaded models into several iid clusters. Concretely, for the cluster G_i, we average the models inside

this cluster (i.e., $M_{G_i}^1, M_{G_i}^2, ..., M_{G_i}^{N_i}$) and generate the corresponding personalized model M_p^i

$$M_p^i = \frac{1}{N_i} \sum_{j=1}^{N_i} M_{G_i}^j, \tag{7}$$

where N_i is the number of models inside the cluster G_i. In this way, we are able to obtain the final personalized models $(M_p^1, M_p^2, ..., M_p^Q)$ and deploy them to corresponding clients.

4.5 Theoretical Analysis

In this subsection, we provide theoretical analysis to support our framework. A possible direction is to quantify the uncertainty of models built by different aggregation algorithms based on the theory proposed by Gal [22]. After exploration, we find that the key factor to affect model uncertainty is the amount of data. In other words, the more data we have, the smaller uncertainty the model holds. In our FL scenario, the client data is not changed when applying different aggregation algorithms and thus we cannot explain the improved performance from this perspective. Here we use the *weight divergence* to provide feasible theoretical analysis.

Specifically, we analyze the *weight divergence* among the candidate models that need to be aggregated in traditional FL and DistFL, which has been proven to be a key factor to the final FL performance (i.e., higher divergence leads to worse performance [61]). According to the work [61], the *weight divergence* satisfies the following inequality

$$\begin{aligned}
\left\| \boldsymbol{w}_{mT}^{(f)} - \boldsymbol{w}_{mT}^{(c)} \right\| &\leq \sum_{k=1}^{N} \frac{n^{(k)}}{\sum_{k=1}^{N} n^{(k)}} \left(a^{(k)} \right)^T \left\| \boldsymbol{w}_{(m-1)T}^{(f)} - \boldsymbol{w}_{(m-1)T}^{(c)} \right\| \\
&+ \eta \sum_{k=1}^{N} \frac{n^{(k)}}{\sum_{k=1}^{N} n^{(k)}} \sum_{i=1}^{C} \left\| pro^{(k)}(y=i) - pro(y=i) \right\| \sum_{j=1}^{T-1} \left(a^{(k)} \right)^j g_{\max} \left(\boldsymbol{w}_{mT-1-k}^{(c)} \right)
\end{aligned} \tag{8}$$

where N is the number of clients. $\boldsymbol{w}_{mT}$ denotes the model weight at the mT round. f and c represent the federated setting and the centralized setting. C is the number of category and y is the ground truth. g_{max} denotes the max gradient and a is a hyper-parameter. $pro_{(k)}$ represents the probability of the k_{th} client and pro is the overall distribution under the centralized setting.

Here we mainly focus on how to compare the candidate models' *weight divergence* between traditional FL and our DistFL. Based on the inequality, the distribution heterogeneity (i.e., $\sum_{i=1}^{C} \left\| pro^{(k)}(y=i) - pro(y=i) \right\|$ in the inequality) plays a key role in the divergence degree: higher heterogeneity leads to larger divergence. In traditional FL, the distribution heterogeneity is high since the data distribution under the candidate models is typically non-iid. Different from it, DistFL partitions uploaded models into several iid clusters to conduct aggregation, which significantly mitigates the heterogeneity and decreases the weight divergence degree. Therefore, compared with traditional FL, DistFL maintains a low weight divergence degree during aggregation, contributing to improved performance.

5 EVALUATION

This section presents the evaluation of DistFL over different mobile scenarios and compares it against various baseline methods. Specifically, we primarily look at the following two aspects:

- What is the performance of DistFL on our simulated mobile scenarios and the real-world scenario? How does it compared to existing methods? (§5.2, §5.3, §5.4, §5.5, §5.6, §5.7)
- Is the extracted distribution knowledge effective in distinguishing models with different distributions? How to select parameters for better efficiency during the extraction process?(§5.8)

Table 1. The simulation settings of different datasets. Note that the *HAR-ADL* dataset is collected from real mobile users, which means that it does not have the specific distribution type.

Dataset	Distribution type	Client index	#Training sample	#Testing sample
CIFAR-10	Type1	1,2,...,20	800	2000
	Type2	21,22,...,40	800	2000
	Type3	41,42,...,60	800	2000
	Type4	61,62,...,80	800	2000
	Type5	81,82,...,100	800	2000
Office-Home	Ar	1,2,...,5	1940	485
	Cl	6,7,...,10	3490	873
	Pr	11,12,...,15	3550	887
	Rw	16,17,...,20	3485	871
HAR-ADL	unknown	1,2,...,20	5408	1984

5.1 Experimental Settings

In the following, we will describe the simulation settings of our experiments since it is hard to conduct experiments on real-world mobile applications. The source code is available at https://github.com/YikFungTsai/DistFL.

5.1.1 Used Public Datasets. We mainly use public computer vision (CV) datasets and human activity recognition (HAR) datasets for simulation. Concretely, we select CIFAR-10 [24], Office-Home [53] and HAR-ADL [2] to simulate various situations. The detailed usage and processing of these datasets are explained in the next part.

5.1.2 Simulated Mobile Scenarios. We design four commonly seen mobile scenarios based on public datasets, whose detailed simulation settings are illustrated in Table 1. Here we briefly summarize each scenario as follows.

(1) *Category-imbalance scenario.* This scenario means that different clients may hold different categories due to the user preference. For example, one client may own many "dog" images while another prefers to collect "cat" images. Similar to the previous work [40], we simulate this situation by the public CIFAR-10 dataset, which contains 10 different image categories. Considering that the client end may own limited data, we uniformly select 4000 samples from the training set and allocate them into 100 clients, where every 20 clients belong to a type of distribution. Therefore, in total we have 5 types of distribution, each of which holds two disjoint categories of CIFAR-10. For testing, we use the official testing set and partition them into 5 parts based on the 5 types of distribution. Here we use *TypeX (X=1,2,...,5)* to denote each type of distribution.

(2) *Environment-difference scenario.* This scenario suggests that the main recognition object is identical while the background is different due to diverse environments. Under this circumstance, we use the Office-Home dataset for simulation, which contains 15,500 images with four distinctive domains: Artistic images (Ar), Clipart images (Cl), Product images (Pr) and Real-World images (Rw). Each domain has 65 identical object categories but different backgrounds. As a recent work does [32], we first partition each domain into a training set (80%) and a testing set (20%) since this dataset has no official train/test split. For training sets, we further divide each domain into 5 parts and each client owns one part. Therefore, in total we have 20 clients with 4 types of data distribution and each type is evaluated on the testing set of the corresponding domain.

(3) *Privacy-protection scenario.* Considering the inference attack may steal some sensitive user information from the uploaded models or gradient updates, we apply differential privacy (DP) for the local training to defend this type of attack as prior works do [11, 60]. Under this privacy-protection scenario, we want to

examine whether our approach can still work given the DP-based local models. Here we take Office-Home as an example to observe the performance on different privacy budgets.

(4) *Attack-injection scenario.* This scenario indicates that there exist some attackers who inject some malicious behaviors to their local models to mislead FL. Here we mainly focus on the poison attack in the context of FL because the inference attack can be effectively defended if we adopt DP techniques. Specifically, we implement the label flipping attack [51] and the model replacement attack [3] to some clients for simulation. Notice that many baseline methods do not target the non-iid problem, thus we only simulate the iid setting on CIFAR-10 (i.e., 50 clients, each of which has 10 categories with 1000 samples) to make a fair comparison.

Besides, we use the HAR-ADL [2] dataset to simulate both the category-imbalance and the environment-difference scenario, which is collected from accelerometers and gyroscopes of 30 smartphone users who perform 6 different activities (*i.e., walking, walking_upstairs, walking_downstairs, sitting, standing, laying*). Because HAR-ADL is a real world dataset, where different users may have specific user preference under diverse environments, we believe it can cover both of the first two scenarios. Specifically, we select 20 users as 20 clients and use a simple 2-layer CNN to conduct experiments.

5.1.3 Models. We use three types of model architectures: simple 2-layer CNN, VGGNet [50] and ResNet [18], to test the performance of our approach. Towards VGGNet, we replace the original fully connected layers with a global average pooling layer in order to alleviate overfitting [45].

5.1.4 Compared Baseline Methods. We compare the proposed approach to two lines of work. The first one is the personalization methods targeting at the non-iid problem in FL systems. Specifically, the following methods are compared:

- *Local*: This method means that we only train a model at each local client without federation.
- *LG-Net* [29]: LG-Net first learns a global model by traditional FL and then fine-tunes the model with the data in each client.
- *K-Cluster* [12]: Similar to our approach, K-Cluster also aims to cluster the uploaded models and conduct aggregation accordingly. However, this method is only based on simple loss values and requires setting the cluster manually, which is inaccurate and impractical since it is hard to know the cluster information before learning.

Another line of work is the defense methods, which aim to resist poison attacks for FL systems. Concretely, we implement three methods for comparison:

- *Credit* [28]: Credit attempts to use conv layers' weights of uploaded models to build an auto-encoder such that the abnormal clients can be selected out.
- *IRLS* [9]: IRLS applies a weighting scheme that gives a lower weight to uploaded attacked models, thus mitigating their negative influence.
- *GRA* [59]: The goal of GRA is to directly use the weights to distinguish malicious clients for robust aggregation.

Besides, the typical *FedAvg* [40] is also compared as a baseline for both of the two lines.

5.1.5 Implementation Details. We performed all experiments with the PyTorch framework, based on a server that has 4 GeForce GTX 2080Ti GPUs, 48 Intel Xeon CPUs, and 128GB memory. In the *category-imbalance scenario*, we tested the performance on VGG-11 and ResNet-18 with the CIFAR-10 dataset, where input images are 32*32 and normalized to zero mean for each channel. We assigned the learning rate to 1e-4 for VGG-11 and 1e-2 for ResNet-18 respectively. In the *background-difference scenario*, we used the pre-trained ResNet-50 model to test the performance on the Office-Home dataset, where we randomly cropped the input images to 224*224 and these images are also normalized to zero mean for each channel. The learning rate was set to 1e-2. Besides, we

Table 2. Results on the category-imbalance scenario with CIFAR-10. Each *"Type"* represents a unique data distribution.

Method	VGG-11						ResNet-18					
	Type1	*Type2*	*Type3*	*Type4*	*Type5*	*Avg*	*Type1*	*Type2*	*Type3*	*Type4*	*Type5*	*Avg*
FedAvg	38.65	40.35	21.25	32.55	37.05	33.97	28.60	35.10	25.30	32.00	30.45	30.29
Local	81.50	81.37	69.38	81.44	83.11	79.36	74.00	70.40	63.37	74.16	76.09	71.60
LG-Net	83.36	85.94	76.87	85.43	84.99	83.32	80.43	79.65	68.05	79.71	82.15	78.00
K-Cluster	88.60	92.55	79.45	91.20	91.35	88.63	86.40	86.55	74.80	55.95	58.25	72.39
Ours	**89.70**	**93.05**	**89.50**	**91.40**	**91.75**	**91.08**	**87.80**	**88.25**	**76.60**	**87.00**	**87.40**	**85.41**

Table 3. Results on the environment-difference scenario with Office-Home. Here *Art, Clipart, Product, Real-World* are the four data domains under different environments.

Method	*Art*	*Clipart*	*Product*	*Real-World*	*Avg*
FedAvg	78.35	71.82	89.52	83.36	80.76
Local	54.06	54.30	79.79	75.84	62.72
LG-Net	75.91	74.18	89.06	84.59	80.94
K-Cluster	74.67	77.09	89.63	84.39	80.46
Ours	**79.49**	**78.62**	**93.01**	**87.73**	**83.71**

conducted experiments in the HAR-ADL dataset with a simple 2-layer CNN, where we sampled data at every 50 Hz and set the learning rate to 1e-2.

Towards the *privacy-protection scenario* and the *attack-injection scenario*, we followed the processing settings in aforementioned scenarios and then applied different schemes (i.e., DP and attacks) into the FL pipeline. By default, all of the models were trained using SGD with a momentum of 0.9. We set the local training epoch to 5 and conducted 50 federated rounds to guarantee convergence. In addition, the hyper-parameter z and $E\%$, which represent the number of initialized noises and the channel percentage of BN layers, were respectively set to 200 and 50%. Finally, we repeated all the experiments 3 times and took the average of them as the reported results.

5.2 Results on the Category-imbalance Scenario

As we stated in the experimental settings, there are 5 types of distributions and we test the average performance on each type. Table 2 illustrates the results on VGG-11 and ResNet-18. We can clearly see that the proposed DistFL consistently achieves higher accuracy than other methods across all of the types. On average, the improvement is over 2% for VGG-11 and 7% for ResNet-18 compared to other methods. This validates that by distribution-aware FL, we can greatly boost the performance for the category-imbalance scenario. Besides, it is worth noting that: (1) The *FedAvg* baseline is worse than the *Local* training, which suggests that the typical FL algorithm fails to cope with this non-iid setting and it is even not necessary to conduct federated learning under this condition; (2) The overall performance on VGG-11 outperforms ResNet-18, demonstrating that the skip connections existed in ResNets may harm the FL process when each client owns images with different categories. Despite the interference, DistFL still exceeds other methods by a large margin, which further confirms the effectiveness of our approach.

5.3 Results on the Environment-difference Scenario

5.3.1 Performance on Office-Home. Here Office-Home is used to simulate the environment-difference scenario and each domain in the dataset is considered as a type of distribution. We use ResNet-50 as the backbone since it

Table 4. The accuracy (%) achieved by different methods on HAR-ADL. Note that this dataset is collected from real world users, which can cover both the category-imbalance and environment-difference scenarios.

Client Index	FedAvg	Local	LG-Net	K-Cluster (K=2)	K-Cluster (K=3)	K-Cluster (K=4)	Ours
1	96.55	83.91	97.70	83.91	100.00	98.85	100.00
2	100.00	38.16	100.00	88.16	100.00	100.00	100.00
3	58.14	91.86	75.58	80.23	100.00	96.51	96.51
4	78.95	30.26	94.74	94.74	97.37	97.37	97.37
5	100.00	96.34	100.00	70.73	98.78	98.78	98.78
6	89.61	76.62	98.70	92.21	98.70	98.70	98.70
7	67.61	81.69	53.52	46.48	57.75	54.93	67.61
8	68.06	68.06	34.72	68.06	52.78	38.89	72.22
9	100.00	87.34	97.47	98.73	97.47	100.00	100.00
10	100.00	71.25	97.50	80.00	80.00	97.50	97.50
11	98.78	93.90	75.61	80.49	98.78	85.37	95.12
12	65.22	45.65	86.96	79.35	100.00	90.22	97.83
13	83.33	87.78	97.78	98.89	98.89	97.78	98.89
14	56.86	98.04	58.82	66.67	100.00	59.80	100.00
15	91.36	86.42	85.19	98.77	96.30	83.95	98.96
16	95.83	91.67	100.00	97.92	88.54	100.00	98.96
17	96.94	94.90	96.94	96.94	96.94	94.90	98.98
18	90.63	90.63	98.96	98.96	100.00	100.00	100.00
19	100.00	21.67	100.00	100.00	100.00	100.00	100.00
20	98.96	89.58	96.88	98.96	96.88	100.00	98.96
Avg	86.84	76.29	87.35	86.01	92.96	89.68	**95.82**

is commonly evaluated on this dataset. We average the achieved accuracy of each domain/type and report the domain performance in Table 3. Obviously, we can see that DistFL shows superiority in all domains with up to 3.38% average improvement (i.e., on the *Product* domain), demonstrating its applicability when the environments of clients are diverse. In addition, there are two interesting observations: (1) Different from the category-imbalance scenario, in this scenario the *FedAvg* baseline can significantly outperform the *Local* training, which means that we can benefit from FL although the environment of client data is different; (2) The state-of-the-art method *K-Cluster* might be not very applicable to this setting considering that the achieved average accuracy is even lower than the *FedAvg* baseline. We believe this is because the environment difference is hard to be distinguished only using the inference loss calculated in this method, thus leading to wrong clustering results.

5.3.2 Performance on HAR-ADL. HAR-ADL is a human activity recognition (HAR) dataset whose data are collected from different real-world users. Therefore, the distribution is typically non-iid, including both the category-imbalance and environment-difference situations. Different from the Office-Home dataset that has specific distribution types, it is hard to estimate the detailed distribution information of HAR-ADL, which poses a higher demand to implement clustering automatically. Therefore, for the *K-cluster* method, we manually try several hyper-parameter settings (i.e., K=2,3,4) and record their corresponding results. Table 4 demonstrates the overall performance with different methods. We can observe that: (1) On average, our DistFL can outperform other methods by a large margin, which suggests that distribution-aware FL does benefit the real-world user
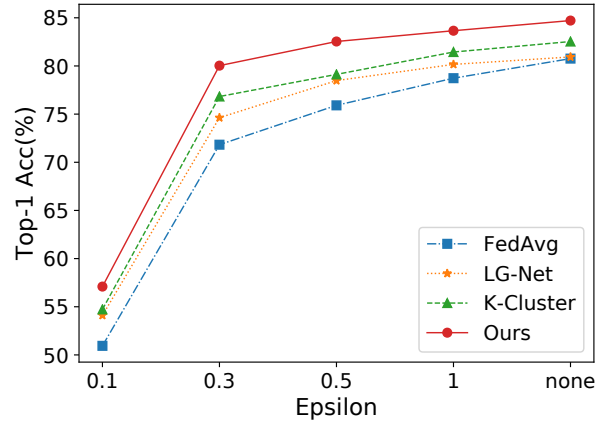
Fig. 3. Results on the privacy-protection scenario. Here we test the effectiveness of different methods by enforcing different privacy budgets (i.e., epsilon).
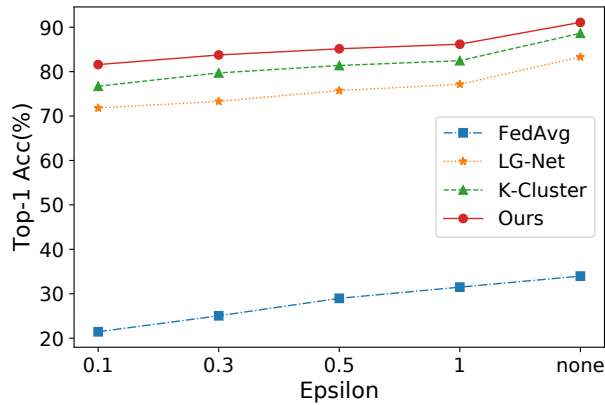

Fig. 4. Results on the mixed scenario (i.e., category-imbalance and privacy-protection). Here we test the effectiveness of different methods by enforcing different privacy budgets (i.e., epsilon).

scenario; (2) For *K-cluster*, the performance varies significantly given different hyper-parameter settings. This demonstrates that *K-cluster* is largely dependent on the accurate manual setting, making it hard to be used in real-world applications since we cannot give the suitable manual setting before FL; (3) Besides the cluster-based method, other personalization methods can only reach an accuracy of 87.35%, which is roughly 10% lower than our approach. In other words, these methods fail to achieve good performance when the category-imbalance and environment-difference situations happen simultaneously.

5.4 Results on the Privacy-protection Scenario

We simulate this scenario by adding differential privacy (DP) based noise to the local models for stronger privacy guarantees. The intensity of the noise is controlled by the privacy budget ϵ. The smaller ϵ is, the larger noises

Table 5. Results on the attack-injection scenario. Here is the defense performance to the label flipping attack on VGG-11. *ACC* is the testing accuracy on the final aggregated model and *ASR* represents the attack success rate.

Method	5		10		15		20		40	
	ACC(%)	ASR(%)	ACC(%)	ASR(%)	ACC(%)	ASR(%)	ACC(%)	ASR(%)	ACC(%)	ASR(%)
FedAvg	67.89	5.18	59.33	19.57	59.33	40.33	40.85	52.87	36.61	70.08
Credit	69.48	3.82	62.22	14.62	49.44	37.65	42.12	51.92	36.45	70.45
IRLS	70.24	3.03	69.93	3.77	68.14	4.19	68.06	4.28	37.09	65.87
GRA	67.40	4.23	63.26	4.37	60.65	7.13	58.14	8.69	57.25	10.71
Ours	**70.71**	**2.82**	**70.01**	**2.72**	**69.65**	**2.93**	**69.47**	**2.98**	**65.30**	**3.38**

Table 6. Results on the attack-injection scenario. Here is the defense performance to the model replacement attack on ResNet-18.

Metric	FedAvg	Credit	IRLS	GRA	Ours
ACC(%)	31.84	72.68	72.63	71.53	**72.69**
ASR(%)	56.15	2.26	2.35	2.48	**2.22**

are added, which indicates stronger defense to the inference attack. Under this privacy-protection scenario, we explore whether our approach is still effective in clustering models with different distributions.

Here we use the results on *Office-Home & ResNet-50* as an example and apply different ϵ values to local models to control the degree of privacy protection. As shown in Figure 3, the average accuracy of four domains with several methods is reported under the different ϵ. From the figure, we can draw the following conclusions: (1) a larger ϵ (weaker privacy guarantee) usually leads to higher accuracy, which is intuitive since it introduces smaller noises to the models; (2) as we strengthen the degree of privacy protection (decrease ϵ), the performance on all of the methods is degraded. In particular, when we decrease the ϵ from 0.3 to 0.1, almost all of the methods drop significantly and become nearly invalid; (3) our approach can achieve consistently higher accuracy regardless of the intensity of ϵ, which further indicates the effectiveness of DistFL.

5.4.1 Performance on A Mixed Scenario. To explore whether our framework is effective in the mixed scenario, we further conduct an experiment on both the category-imbalance and privacy-protection scenario. Concretely, we employ the simulation setting in the category-imbalance scenario and incorporate different privacy budgets (i.e., ϵ) to local models to enable privacy protection. Here we use the VGG-11 as the backbone and the average performance among these clients is recorded for comparison. Figure 4 demonstrates the results. We can observe that the proposed framework can achieve better accuracy performance than others for all of the privacy budgets (e.g., 4.88% improvement when $\epsilon = 0.1$), validating its usefulness in the mixed scenario. Besides, the baseline *FedAvg* performs significantly worse than other methods, which keeps the same observation as the category-imbalance scenario where the non-iid situation largely degrades the performance of traditional aggregation used in *FedAvg*.

5.5 Results on the Attack-injection Scenario

Although DP can be introduced to provide stronger privacy protection as stated in the last subsection, it is impossible to defend the poison attacks, which inject malicious behaviors on the internal models. In this scenario, we demonstrate how to inject poison attacks and the corresponding defense performance with different methods.

5.5.1 Threat Models. We construct two threat models by implementing two types of poison attacks. The first one is the label flipping attack [51], where the labels of honest training examples of one class are flipped to another class while the features of the data are kept unchanged. Another is the model replacement attack [3], which accomplishes the attack by replacing the local model with elaborate weights to mislead FL. Note that this replacement attack will generate an unusable local model that cannot be processed using our extraction techniques. Thus, we resort to a *pre-aggregation* scheme since the aggregated model is always a usable model.

5.5.2 Defense on These Attacks. We use two metrics to evaluate the defense performance: *testing accuracy on the final aggregated model* and *ASR*. The first metric is to examine whether the influence of malicious clients can be removed. *ASR* represents the attack success rate, which is higher if the misleading accuracy on the targeted input increases. In our scenario, *ASR* is defined as the success probability of perturbing a specific image class to a target class, which can be calculated by

$$ASR = \frac{\sum_{i=1}^{|D_s|} \mathbb{I}(M_{agg}(x_i) == y_{target})}{|D_s|} \tag{9}$$

Here $|D_s|$ is the number of the specific image class and M_{agg} is the aggregated model. y_{target} is the label of the target class and x_i is the i_{th} sample of the specific class. $\mathbb{I}$ is an indicator function. If the attack includes multiple classes, we average their ASR as the final result.

Table 5 shows the defense results to the label flipping attack on VGG-11. Here label 0-3 and label 4-7 are mutually flipped to conduct the misleading training. We respectively implement the attack on 5, 10, 15, 20 and 40 clients of the simulated 50 clients, to test the defense performance on different attack scales. Note that under the label flipping attack, DistFL will generate two aggregated models for the normal and abnormal cluster respectively, and we use the model in the normal cluster as the aggregated model because this model is the decent model to be deployed for mobile users (another one is just an attack model). From the table, we can observe that although other methods such as *IRLS* and *GRA* can resist the attack, they fail to accomplish the defense goal as the number of attacked clients grows, especially when the attack scale is large enough (e.g., 40 malicious clients in our simulation). Different from them, our approach has the capability to achieve good defense performance even though the attack scale is large (e.g., 40/50 users are attackers and they flip their data labels). This is because in DistFL, the malicious clients are solely and accurately clustered, getting rid of their negative influence on the final aggregated model of normal clients. Besides, our accurate clustering also benefits more to the testing accuracy compared with other methods since no malicious behavior can be introduced to disturb the FL process in the cluster of the normal clients.

For the model replacement attack, we also use the same label setting as the flipping attack and conduct the misleading training. After training, we replace the model weights using techniques stated in [3] to achieve a stronger attack with only one malicious client. As shown in Table 6, the attack can significantly harm traditional FL (FedAvg). However, all of the defense methods are able to defend it due to its over weird weights, which can be easily detected and removed. Compared to other methods, our approach performs slightly better in an efficient and automated way.

5.6 Results on the Real-world Scenario

Aforementioned scenarios are based on the simulation setting with public datasets. However, the data distribution in the real-world scenario is more complex and even has no specific pattern. To further test the generality of the proposed approach, we conduct an experiment on a human activity recognition (HAR) dataset collected from a mobile app to represent the real-world scenario [26], where the data distribution is not clear as our simulated scenarios. Specifically, an Android app is first developed for activity signal logging and 6 types of activities,

Table 7. The accuracy (%) achieved by different methods on the real-world dataset collected from a mobile app.

Client Index	FedAvg	Local	LG-Net	K-Cluster (K=2)	K-Cluster (K=3)	K-Cluster (K=4)	Ours
1	75.63	50.00	86.25	85.63	86.25	83.13	87.50
2	80.94	77.05	92.62	90.33	90.12	90.53	92.83
3	97.06	42.65	95.59	96.32	95.59	94.85	97.06
4	73.96	70.83	89.58	83.33	89.58	88.54	89.58
5	78.41	80.11	86.43	81.82	85.23	81.82	86.43
6	83.82	93.38	94.12	94.12	91.91	91.18	95.59
7	83.33	83.33	89.46	90.69	90.69	90.69	91.38
8	41.67	35.42	79.17	75.35	75.00	75.69	80.21
9	68.45	53.81	81.55	70.83	77.38	77.98	85.12
10	75.40	81.85	88.71	93.15	91.53	89.92	93.15
11	96.02	89.77	100.00	98.30	98.86	98.30	100.00
12	85.00	82.50	91.75	90.00	90.00	85.00	91.75
13	83.93	50.00	91.07	83.93	87.50	82.14	92.64
14	62.50	71.20	77.72	75.54	75.54	76.63	78.26
15	80.56	45.83	87.50	87.50	81.02	87.50	87.50
16	87.69	53.85	90.79	87.69	87.69	89.90	92.71
17	88.54	67.81	91.04	82.31	91.04	91.04	92.38
18	93.75	68.75	91.88	96.88	93.75	93.75	93.75
19	83.17	57.07	90.22	86.61	86.61	89.50	92.93
20	79.51	68.40	92.93	90.63	92.93	94.64	94.64
Avg	79.97	66.18	89.42	87.05	87.91	87.64	**90.77**

including *Walking, Biking, (walking) Upstairs, (walking) Downstairs, Running and Taking Bus/Taxi*, are collected to form the dataset. Similar to the setting used in HAR-ADL, we select 20 users as 20 clients (17,382 training samples and 4,381 testing samples) and employ a simple 2-layer CNN to conduct the experiment.

Table 7 summaries the results. From the table, we can obtain the following observations: (1) It is obvious that the proposed approach achieves the best performance on average, which suggests the effectiveness of DistFL under the real-world scenario; (2) Unlike the results of the HAR-ADL dataset where *LG-Net* performs worse than *K-Cluster*, in this context, *LG-Net* can exceed *K-Cluster* by up to 2.37% average accuracy. We believe this is because in this scenario we own more training samples than HAR-ADL, which will significantly mitigate the overfitting problem to the local fine-tuning process existed in *LG-Net*. However, *LG-Net* needs more computation cost than others since it includes a *FedAvg* process plus a fine-tuning process, which increases the burden on the resource-constrained mobile devices; (3) The hyper-parameter K in *K-Cluster* is not very sensitive in this scenario since the final accuracy of different K values is similar. However, all of the hyper-parameter settings for *K-Cluster* perform worse than DistFL, which indicates that our clustering result is more precise and thus benefits the final performance.

5.7 Convergence Performance

To better observe the training process of different methods, we record their corresponding testing accuracy in each FL round and plot the convergence lines. Here we conduct the experiment on the environment-difference
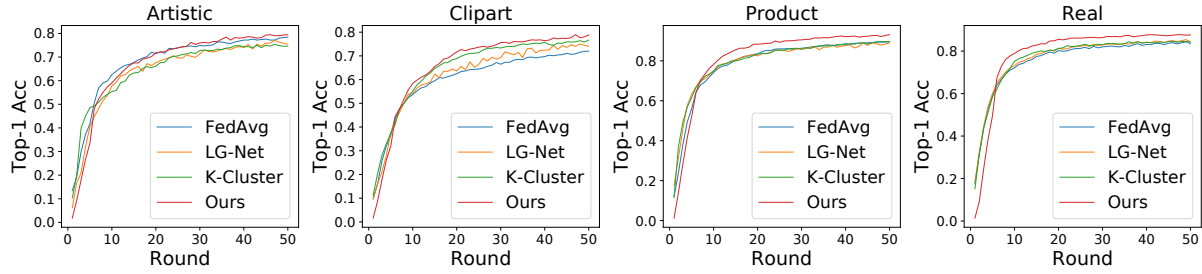
Fig. 5. Convergence performance of the environment-difference scenario with Office-Home.
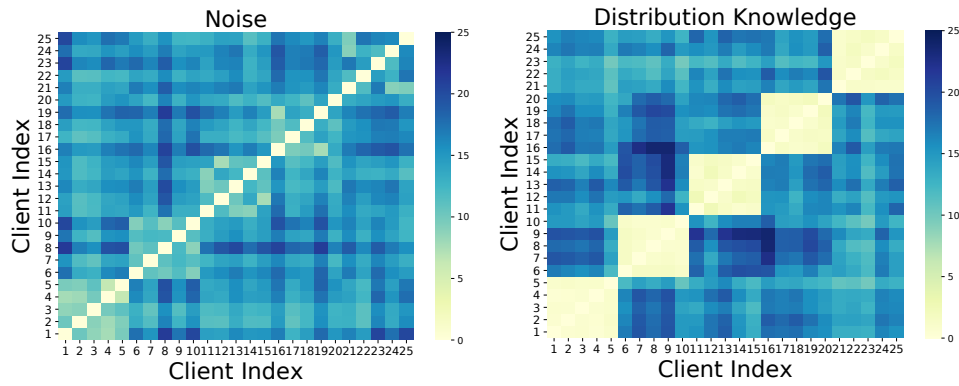

Fig. 6. Visualization of the similarity matrix generated by noises (left) and our distribution knowledge (right). Lighter color indicates higher similarity. We can clearly see that using the distribution-based matrix can provide better guidance to the clustering process.

scenario with Office-Home and for each domain, we randomly pick out one device as an example. As shown in Figure 5, all the methods can reach convergence within 50 FL rounds and DistFL achieves better performance compared to others. In addition, it is worth noting that: (1) For the *Product* and the *Real-World* domain, the convergence speed is significantly faster than the other two domains. Specifically, we can obtain desirable performance at roughly 30 FL rounds, which suggests that the two domain tasks are easier to train; (2) Other methods behave unstably on different domains. For example, *FedAvg* can achieve comparable convergence speed as ours on the *Artistic* domain while failing to perform well on the *Clipart* domain. Different from them, DistFL can keep consistent superiority under any domain situation.

5.8 Effectiveness and Efficiency Analysis

This subsection analyzes different modules and parameters used in DistFL to provide readers a better understanding of the effectiveness and efficiency of our framework. All of the analyses, unless otherwise specified, are based on the setting of the *category-imbalance scenario*.

5.8.1 Effectiveness of Extracted Knowledge. We conduct two control experiments to verify the effectiveness of the distribution knowledge, which are summarized as follows.

- *Noise vs. Distribution knowledge.* In this experiment, we respectively feed some noises and the generated distribution knowledge into models to obtain the similarity matrix. Here we randomly picked out 25 clients
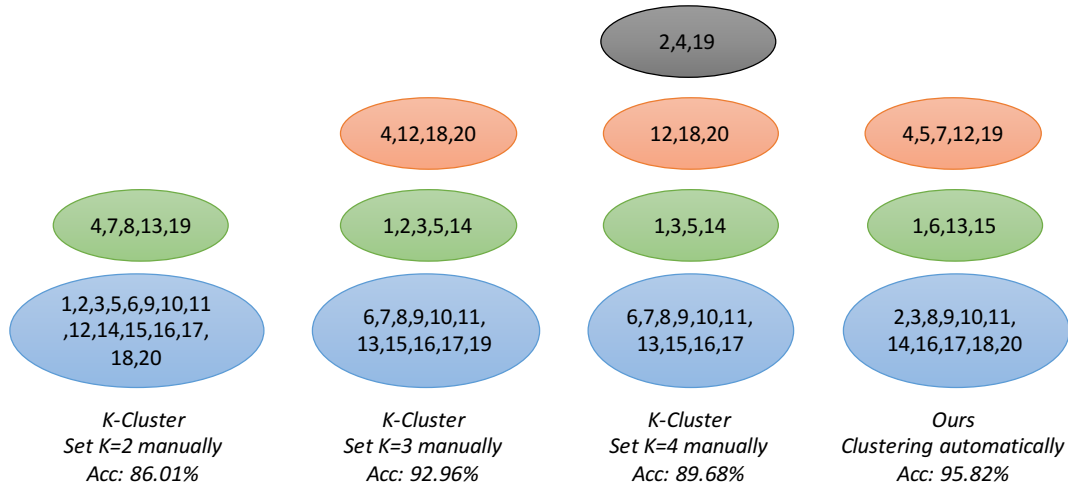
Fig. 7. Illustration of the concrete cluster information on HAR-ADL, which owns data collected from 20 different users. In the figure we use the client index to represent each user and allocate them into different clusters based on the K-Cluster method and our method. Compared to K-Cluster, our distribution-aware clustering can achieve a better accuracy without manual efforts.

and every 5 with continuous indexes come from the same distribution. Figure 6 depicts the results, where the lighter color denotes higher similarity. Obviously, it is hard to achieve clustering from the noise-based matrix since we cannot distinguish suitable clients from others given a target client. Different from the noises, our distribution knowledge is able to obtain a desirable similarity matrix, where every 5 clients of the same distribution are clearly grouped, confirming the usefulness of the generated knowledge.

- *Loss-based clustering vs. Distribution-based clustering.* In this experiment, we compare the concrete cluster results with the loss-based clustering (K-Cluster [12]) and our distribution-based clustering. Here we use the HAR-ADL dataset as an example. As shown in Figure 7, the 20 client models are distributed to different clusters denoted in different colors. We can conclude that the loss-based scheme obtains an inappropriate clustering result despite using various parameter settings since all of their final accuracy are significantly lower than ours. Besides, among these manual settings, partitioning models into 3 clusters can achieve the best performance, which is consistent to our approach. However, we would like to point out that our approach is automated and the concrete model index in each cluster is different from the loss-based scheme. That is why our approach can outperform others.

5.8.2 Parameter Selection for Better Efficiency. In our framework, the computation costs come from two aspects: (1) the amount of knowledge we extract; (2) the number of extraction target in a model. Here we implement some experiments with different parameters to explore the best choice for efficiency.

For the first aspect, as stated in Section 4.3, the distribution knowledge can be denoted as $K_{target} = \{k_{target}^1, k_{target}^2, ..., k_{target}^z\}$, where the amount of knowledge is reflected by the parameter z. Thus we explore the influence of z on the clustering performance. Specifically, the value of z is set to 1, 10, 20, 50, 100, 200 and 500 and we respectively use these values to generate the corresponding knowledge and the similarity matrix of 10 randomly sampled clients, where every two continuous clients come from the same distribution. We plot the similarity matrix of each parameter pair (i.e., $(z, E\%)$, we will analyze the influence of $E\%$ in the following part) and for each matrix,
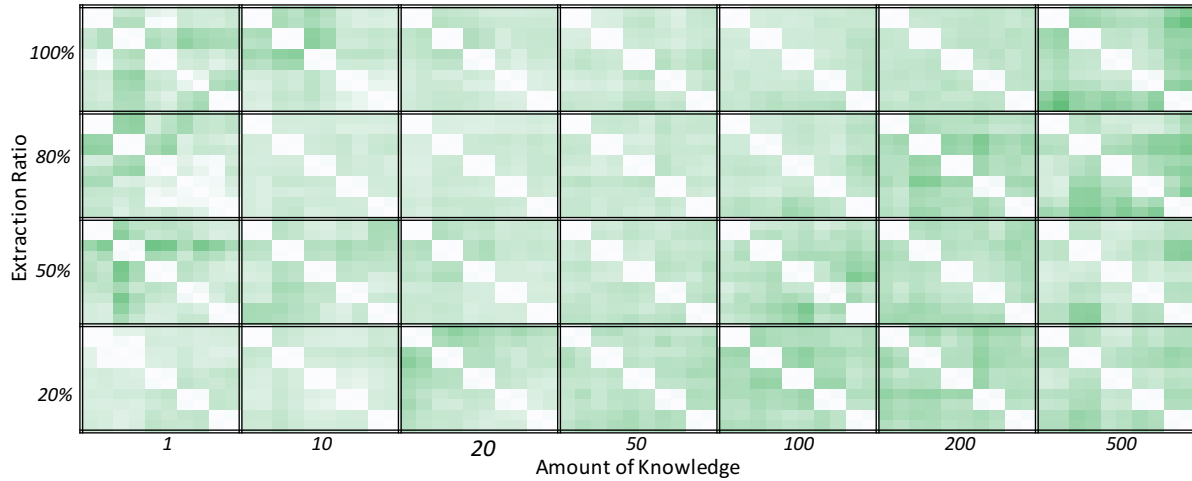
Fig. 8. Influence of the extraction ratio and the amount of knowledge on the similarity matrix. Here we randomly sample 10 clients and every 2 continuous clients come from a similar distribution. Thus, for each matrix, every four blocks on the diagonal indicate the models with similar distributions. Lighter color means higher similarity.

every four blocks on the diagonal indicate the models with similar distributions. Here lighter color means higher similarity. As shown in Figure 8, the clustering performance becomes more significant when we increase the amount of generated knowledge (i.e., the value of z). Generally, 100 "distribution knowledge" (i.e., $z = 100$) is enough to achieve good clustering regardless of the extraction ratio $E\%$.

Besides, deciding the number of extraction target is also necessary for efficiency since there are many BN layers in a modern DNN. In DistFL, we propose to only optimize the important BN channels during the extraction. To study the impact of different extraction ratios (i.e., $E\%$) of BN layers, we respectively select 20%, 50%, 80% and 100% BN channels of each BN layer to conduct our pipeline. The performance is also measured by the similarity matrix as illustrated in Figure 8. From the figure, we can clearly see that it is not necessary to optimize the whole BN channels, especially when the amount of the extracted knowledge is enough (e.g., 100). This demonstrates that we can accomplish the extraction process efficiently by employing fewer BN channels such as 50% or even 20% when we have enough quantity of the distribution knowledge.

6 DISCUSSIONS

One limitation of the paper is the lack of experiments on real-world applications. Although we have simulated different mobile scenarios to validate the effectiveness of our approach, the real-world study is also an important aspect worth exploring further. In our future work, we will look for ways to evaluate DistFL in real-world applications. Besides, this paper mainly focuses on computer vision tasks and human activity recognition tasks. It is interesting to apply our framework for other deep learning tasks by analyzing and extracting the distribution knowledge from their models. Finally, although we have simulated the privacy-protection scenario by enforcing differential privacy (DP) to the local models, researchers may want to use some secure aggregation schemes [5] to further protect the model information. However, secure aggregation requires massive computation power, which may be unacceptable to the resource-constrained clients under our mobile scenarios.

Besides, we provide some findings and lessons learned from this work. First, before generating bogus data, researchers should check if the BN layer has been used in the model although it is prevalent in the current

architectures. If not, the statistical information needs to be collected manually and then uploaded to the server for data generation. Second, the performance of our clustering process depends on the responses of each model, which are obtained by feeding the distribution knowledge (i.e., bogus data) into each local model. Therefore, it is necessary to generate enough bogus data to make the distribution knowledge more precise, which will then benefit the final clustering process.

7 CONCLUSION

This paper proposes *distribution-aware federated learning* to address the non-iid problem, which widely exists in mobile scenarios. We design and implement a framework named DistFL to accomplish our goal efficiently, automatically and accurately, where the distribution knowledge of federated models is extracted to distinguish and cluster different client models. Experimental results demonstrate that DistFL achieves significantly better performance on all simulated scenarios with different models compared to state-of-the-art approaches.

ACKNOWLEDGMENTS

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[4] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[5] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[6] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[8] Jie Feng, Can Rong, Funing Sun, Diansheng Guo, and Yong Li. 2020. PMF: A privacy-preserving human mobility prediction framework via federated learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1 (2020), 1–21.

[9] Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. 2019. Attack-resistant federated learning with residual-based reweighting. *arXiv preprint arXiv:1912.11464* (2019).

[10] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2018. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866* (2018).

[11] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

[12] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Neural Information Processing Systems(NeurIPS)* (2020).

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[14] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.

[15] Yeting Guo, Fang Liu, Zhiping Cai, Hui Zeng, Li Chen, Tongqing Zhou, and Nong Xiao. 2021. PREFER: Point-of-interest REcommendation with efficiency and privacy-preservation via Federated Edge leaRning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–25.

[16] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[17] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. 2020. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518* (2020).

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

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[20] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 603–618.

[21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[22] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *NIPS* (2017).

[23] Nikola Konstantinov and Christoph Lampert. 2019. Robust learning from untrusted sources. In *International Conference on Machine Learning*. PMLR, 3488–3498.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[25] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[26] Chenglin Li, Di Niu, Bei Jiang, Xiao Zuo, and Jianming Yang. 2021. Meta-HAR: Federated Representation Learning for Human Activity Recognition. In *Proceedings of the Web Conference 2021*. 912–922.

[27] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2019. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *arXiv preprint arXiv:1907.09693* (2019).

[28] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen. 2019. Abnormal client behavior detection in federated learning. *arXiv preprint arXiv:1910.09933* (2019).

[29] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523* (2020).

[30] Bingyan Liu, Yifeng Cai, Yao Guo, and Xiangqun Chen. 2021. TransTailor: Pruning the Pre-trained Model for Improved Transfer Learning. *arXiv preprint arXiv:2103.01542* (2021).

[31] Bingyan Liu, Yao Guo, and Xiangqun Chen. 2019. WealthAdapt: A general network adaptation framework for small data tasks. In *Proceedings of the 27th ACM International Conference on Multimedia*. 2179–2187.

[32] Bingyan Liu, Yao Guo, and Xiangqun Chen. 2021. PFA: Privacy-preserving Federated Adaptation for Effective Model Personalization. *The Web Conference (WWW)* (2021).

[33] Bingyan Liu, Yuanchun Li, Yunxin Liu, Yao Guo, and Xiangqun Chen. 2020. PMC: A Privacy-preserving Deep Learning Model Customization Framework for Edge Computing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 4 (2020), 1–25.

[34] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. 2019. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2604–2613.

[35] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2017. Trojaning attack on neural networks. (2017).

[36] King's College London. 2019. King's and NVIDIA join forces to build UK's rst AI platform for hospitals. https://www.kcl.ac.uk/news/kings-and-nvidia-to-build-uks-first-al-platform.

[37] Sidi Lu, Yongtao Yao, and Weisong Shi. 2019. Collaborative learning on the edges: A case study on connected vehicles. In *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*.

[38] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133* (2020).

[39] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5188–5196.

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

[41] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 691–706.

[42] Asit Mishra and Debbie Marr. 2017. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv preprint arXiv:1711.05852* (2017).

[43] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going deeper into neural networks. (2015).

[44] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 739–753.

[45] Siyuan Qiao, Zhe Lin, Jianming Zhang, and Alan L Yuille. 2019. Neural rejuvenation: Improving deep network training by enhancing computational resource utilization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 61–71.

[46] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* (2014).

[47] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017* (2018).

[48] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 11957–11965.

[49] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. 2019. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications* 68, 2 (2019), 1146–1159.

[50] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[51] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*. Springer, 480–501.

[52] Tim Van Erven and Peter Harremos. 2014. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory* 60, 7 (2014), 3797–3820.

[53] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5018–5027.

[54] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* (2017).

[55] WeBank. 2019. An Industrial Level Federated Learning Framework. https://github.com/FederatedAI/FATE.

[56] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[57] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).

[58] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8715–8724.

[59] Lei Yu and Lingfei Wu. 2020. Towards Byzantine-Resilient Federated Learning via Group-Wise Robust Aggregation. In *Federated Learning*. Springer, 81–92.

[60] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. 2020. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758* (2020).

[61] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).

[62] Ligeng Zhu and Song Han. 2020. Deep leakage from gradients. In *Federated Learning*. Springer, 17–31.