Federated Learning for IoT Devices with Domain Generalization

Liling Zhang, Xinyu Lei, *Member, IEEE*, Yichun Shi, Hongyu Huang, *Member, IEEE*, and Chao Chen, *Member, IEEE*

***Abstract*—Federated Learning (FL) is a distributed machine learning technique that allows numerous Internet of Things (IoT) devices to jointly train a machine learning model using a centralized server for help. Local data never leaves each IoT device in FL, so the local data of IoT devices are protected. In FL, distributed IoT devices usually collect their local data independently, so the dataset of each IoT device may naturally form a distinct source domain. In real-world applications, the model trained over multi-source domains may have poor generalization performance on unseen target domains. To address this issue, we propose FedADG to equip federated learning with domain generalization capability. FedADG employs the federated adversarial learning approach to measure and align the distributions among different source domains via matching each distribution to a reference distribution. The reference distribution is adaptively generated (by accommodating all source domains) to minimize the domain shift distance during alignment. Therefore, the learned feature representation tends to be universal, and thus, it has good generalization performance over the unseen target domains while protecting local data privacy. Intensive experiments on various datasets demonstrate that FedADG has comparable performance with the state-of-the-art.**

***Index Terms*—Federated learning, Internet of Things, domain generalization, adversarial learning.**

I. Introduction

Nowadays, the development of the Internet of Things (IoT) has brought great convenience to people's lives. The unprecedented data generated by IoT devices are used for prediction, classification, and detection in deep learning [1]. To train a deep learning model, a straightforward way is to let the IoT devices upload their local data to a centralized server for training [2]. However, some devices' local data (e.g., biometric health records, financial records, location information) may be highly privacy-sensitive and their owners are reluctant to share with any other entities. Fortunately, the proposal of Federated Learning (FL) [3] provides a privacy-preserving mechanism that enables a centralized server to train the model without requiring devices' to share their private data. In one iteration of FL, the server sends the global model to all devices. Then, each device trains the global model using the local data. Next, each device sends the model update to the server, which is used for model update aggregation and new global model generation. After multiple rounds of iteration, the model can be well-trained.

In recent years, FL has been intensively studied in both academia and industry [4]–[6]. In FL, since distributed IoT devices collect their local data independently, each device's dataset may naturally form a distinct domain (a domain is defined as a set of labeled training data that are sampled from a specific distribution [7], [8]). For example, when people use a camera-capable IoT device to collect bird pictures, each device often collects a different bird species (using different cameras and different shot angles). Therefore, the data collected by each IoT device forms an independent domain. Here, the domain formed by one IoT device's dataset is called a source domain, so there are multiple source domains in the FL. In deep learning, multi-source domains often collaboratively train a model for the object classification task. In this article, we aim to develop a solution to learn a classifier (on multiple source domains) that can be used for "unseen domain" with good performance.

Most previous FL studies assume that the test dataset is a subset of device dataset. There is a lack of studies for another common practical scenario in which the data of the target dataset (i.e., test dataset) is absent from the FL training process. It is required to build a model that has high performance when testing over the related but unseen target dataset (note that the target dataset forms the target domain). However, the FL-trained model may have poor performance on target domains due to the discrepancies between source domains and target domains.

The above issue can be addressed by Domain Generalization (DG) [7], [9], [10] technique, but the previous techniques of domain generalization cannot be directly applied to the FL setting. Domain generalization aims to train a Machine Learning (ML) model from one or several different source domains while ensuring the trained model can be generalized to target domains. Most conventional solutions achieve domain generalization in a centralized manner. That is, a centralized server (with access to all source domain data) is responsible for the domain generalization task. For example, **Ji**gsaw puzzle based **Gen**eralization (JiGen) [11] requires data decomposed from multi-source domains to be mixed to train a classifier.

This work was supported in part by the National Key Research and Development Project under Grant 2021YFB1714200, in part by the National Natural Science Foundation of China under Grant 62172066 and 62272073, and in part by the National Science Foundation under Grant CNS-2153393. *(Corresponding author: Hongyu Huang.)*

Liling Zhang, Hongyu Huang, and Chao Chen are with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: ZhangLiling@cqu.edu.cn; hyhuang@cqu.edu.cn; cschaochen@cqu.edu.cn).

Xinyu Lei is with the Department of Computer Science, Michigan Technological University, Houghton, MI 49931, USA (e-mail: xinyulei@mtu.edu)

Yichun Shi is with the Department of Computer Science, Michigan State University, East Lansing, MI 48824, United States (e-mail: shiyichu@msu.edu).

Besides, MixStyle [12] needs to mix features from different source instances to synthesize new domains. However, accessing sources domain by the centralized server is prohibitive in FL to meet the security requirements. Therefore, these conventional techniques cannot be easily applied to domain generalization in FL. There are two proposed schemes (i.e., COPA [13] and FedDG [14]) that study domain generalization problems in FL. COPA is the abbreviation of **C**ollaborative **OP**timization and **A**ggregation, while FedDG is the abbreviation of **Fed**erated **D**omain **G**eneralization. Both of them suffer from some limitations. For COPA, it requires each IoT device to share its local data size. Moreover, it leaks the global information (i.e., domain variation) to each device for batch normalization (BN) layer parameters tuning. In a nutshell, COPA sacrifices security for domain generalization. For FedDG, it allows each device's local data information (i.e., image amplitude spectrum) to be shared with other entities. However, the shared image amplitude spectrum contains class-relevant information, which can be used for training a classifier [15]. It leaks sensitive information about the device's local data. In summary, both COPA and FedDG sacrifice security for domain generalization. Different from the two schemes, our solution aims to achieve domain generalization without the above information leakage.

In this paper, we propose the **Fed**erated **A**dversarial **D**omain **G**eneralization (FedADG) scheme to address the domain generalization problem in FL for IoT devices. FedADG design has two key insights as described below. First, FedADG exploits the idea to learning the domain-invariant feature representation by aligning each distribution of source domain data to a reference distribution in a distributed manner. In the alignment, we employ an Adversarial Learning Network (ALN) to measure the distance between distributions in FL setting. Furthermore, we propose the Federated ALN (FedALN) technique to train ALN in FL setting. In this way, FedADG can learn the domain-invariant features while eliminating the requirement for a centralized server to access IoT devices' local data. Second, FedADG uses the idea of adaptively learning a dynamic distribution (by accommodating all source domains) as the reference distribution. This approach can minimize the domain shift distance during alignment.

Compared with using a pre-selected fix reference distribution, our approach reduces the distortion of extracted feature representation. Therefore, the key information of the original source domain data can be largely preserved, resulting in high generalization performance. Besides, FedADG takes the label information (encoding in a one-hot vector) as input during alignment. Hence, FedADG supports the class-wise alignment of the source domain data, which can further improve its performance on target domains. Furthermore, compared with using the fixed reference distribution, using the dynamically generated reference distribution approach can get more discriminative features after alignment. The discriminative features are helpful to improve the performance of FedADG.

The high performance of FedADG can be explained via visualization, so FedADG gains some explainability to some extent. The more explainability a FedADG scheme has, the deeper understanding that users achieve. An explainable machine learning model can help users in two folds. First, it can help users to tune model parameters efficiently, making it easier for further model optimization. Second, it is more trustworthy to be used in sensitive and critical areas, where its value can be enormous. Note that most previous domain generalization solutions lack explainability.

We summarize our contributions as follows.

1) We propose the FedALN technique to learn the domain-invariant features in FL while eliminating the requirement for a centralized server to access IoT devices' local data.
2) We propose FedADG which employs the adaptively generated reference distribution and class-wise alignment technique in FedADG to ensure its high performance.
3) The explainability of FedADG's high performance brings in two immediate benefits. First, it is easier for users to tune parameters and have further model optimization. Second, it is more trustworthy to be used in practice.

The remainder of the paper is organized as follows. Sec. II introduces some preliminary knowledge. Sec. III introduces some related works of this paper. Sec. IV presents the FedADG scheme and its training process in detail. Sec. V analyzes the principle of FedADG. Sec. VI demonstrates the experimental results, and the efficiency and effects of FedADG scheme are analyzed. Sec. VII concludes this paper.

II. Preliminaries

A. Federated Learning

Federated learning [3], [16] is a distributed machine learning method that learns a global model across multiple devices without revealing the device's local dataset. In FL, each IoT device is referred to as a client. Fig.1 illustrates the framework of FL, which includes K clients and a centralized server. Learning a global model on FL requires multiple iterations of training on both the server and the client. In one iteration of FL, the server sends the initialized global model to all clients. Then, each client trains the global model on their local dataset. Next, each client's model updates are sent to the centralized server and used for aggregation to generate a new global model. After multiple rounds of iterations, the global deep learning model can be well trained.

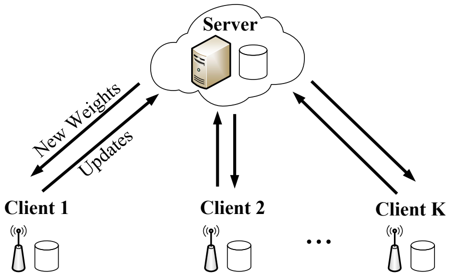


Fig. 1. The federated learning framework.

B. Generative Adversarial Network

Generative Adversarial Network (GAN) is first proposed in [17]. GAN endows the generative model with the ability to generate given distribution outputs via an adversarial

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3234977

procedure. GAN has two components: a generator (G) and a discriminator (D). For generator model, the generator takes random noise samples z from a given prior distribution as input. Then, the generator model is trained to output fake samples that are similar to the real training samples. For discriminator model, the discriminator takes the samples output by the generator model and the real training samples as input. Next, the discriminator learns to distinguish whether an input is fake (generated) or true (from real training samples). The generator and discriminator perform multiple rounds of adversarial training. The training objective can be expressed as

$$\arg\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_g(z)}[\log(1-D(G(z)))], \quad (1)$$

where $p_r(x)$ and $p_g(z)$ denote the distribution of the real training sample and the prior distribution used in generator, respectively. Compared with the classic GAN, FedADG introduces several new components to achieve our purposes in the FL setting.

III. Related Work

Federated Learning. Federated learning [3], [16] is a distributed approach that leaves training data distributed on multiple clients and learns a global model by aggregating the locally-uploaded parameters on a server. In FL, local data never leaves each client, so local data privacy is protected. To improve the performance of the FL-trained model, researchers have proposed many optimized schemes, such as **Fed**erated learning with the **Prox**imal term (FedProx) [18], **Fed**erated **No**rmalized a**v**eraging **a**lgorithm (FedNova) [19], and **MO**del-c**ON**trastive learning (MOON) [20]. Recently, FL has been used to enable numerous intelligent IoT applications [21]. For instance, IoT devices mitigate single points of failure and network scaling issues by integrating FL and consensus-based approaches [22]. Intrusion detection plays an important role in ensuring the security of data in the IoT. Researchers use some techniques to achieve success in intrusion detection, such as lightweight neural networks [23] and novel clustering methods [24]. An intrusion detection system based on federated transfer learning can secure patients' sensitive data on medical devices and applications [25]. Besides, FL combined with deep neural network architecture can detect zero-day botnet attacks on IoT devices [6]. In the real world, most previous FL studies assume that the test dataset is a subset of client dataset. Different from the previous papers, this paper mainly focuses on enabling FL to train a model that has good performance on unseen target domains.

Domain Generalization. The requirement of learning a model from multiple seen source domains for unseen domains motivates the research of domain generalization [7]. Most previous solutions [11], [26], [27] consider the domain generalization problem in a centralized setting. In these papers, a centralized server has access to data from all source domains and it is responsible for training an ML model that has domain generalization capability. However, these solutions expose the source domain data to the server. This is not allowed in FL, so these solutions cannot be directly used in FL setting. To sum up, we summarize the comparison between the previous solutions and FedADG, as shown in Table I.

Domain Adaptation. A similar concept is Unsupervised Domain Adaptation (UDA), which aims to learn an ML model from one or multi-source domain(s) that performs well on a different (but related) target domain [29]. UDA techniques assume the availability of unlabeled target domain data. Even if Peng *et al.* [30] propose a privacy-preserving approach, but its test dataset participates in the training process, which is prohibited in domain generalization. Therefore, UDA techniques cannot be directly used in this paper.

IV. Problem Statement and FedADG Scheme

In this section, we first have the problem statement. Then, we introduce the FedADG scheme. For ease of reading, we summarize the frequently used notations in Table II.

A. Problem Statement

In this paper, we aim to develop a solution to learning a ML model with non-shared data from multi-source domains. Suppose that there are K source domains $\mathcal{S} = \{\mathcal{S}_k\}_{k=1}^{K}$, and a sample-label pair from source domain k is denoted by $(\mathbf{x}_{k_i}, \mathbf{y}_{k_i})$, where $\mathbf{x}_{k_i} \in \mathbb{R}^{d\times 1}$ and $\mathbf{y}_{k_i} \in \mathbb{R}^{m\times 1}$. The ML model trained over the K source domains should have high performance on the unseen target domains. Besides, the proposed solution should follow the same security principle as the traditional FL: only model parameters (e.g., updated gradients) can be sent to the server, and no information about local data can be shared directly. In this paper, each IoT device can also be called a client.

B. FedADG Components

Fig. 2 shows the FedADG scheme. It can be seen from the figure that each client's local model mainly consists of four components, which are described as follows.

Feature Extractor. Feature extractor can extract latent features from the raw data of each client. Besides, the extracted features can be applied to the classification task.

Discriminator. Given features extracted from raw data (from a source domain) and features generated by distribution generator, the discriminator is used to distinguish the extracted features and the generated features. During training, the discriminator gains the ability to distinguish the above two types of features. Besides, a Random Projection (RP) layer is prepended to the discriminator. The RP layer is used to stabilize the training of ALN.

Distribution Generator. On input random noise samples and one-hot vector (used for label encoding), distribution generator generates features, which follow a certain distribution (i.e., the reference distribution). Note that the above three components constitute the Adversarial Learning Network (ALN).

Classifier. Given features as the input, the classifier outputs the predicted label.

TABLE I
THE COMPARISON BETWEEN THE PREVIOUS SOLUTIONS AND FEDADG.

Papers **Properties**	DANN [28] JiGen [11] Epi-FCR [26] RSC [27] MixStyle [12]	FedAvg [3]	FedDG [14]	COPA [13]	**FedADG**
Data storage mode	Centralized	Distributed			
Support strong privacy protection?	×	✓	×	×	✓
Support domain generalization?	✓	×	✓	✓	✓
Support distributed domain generalization?	×	×	✓	✓	✓

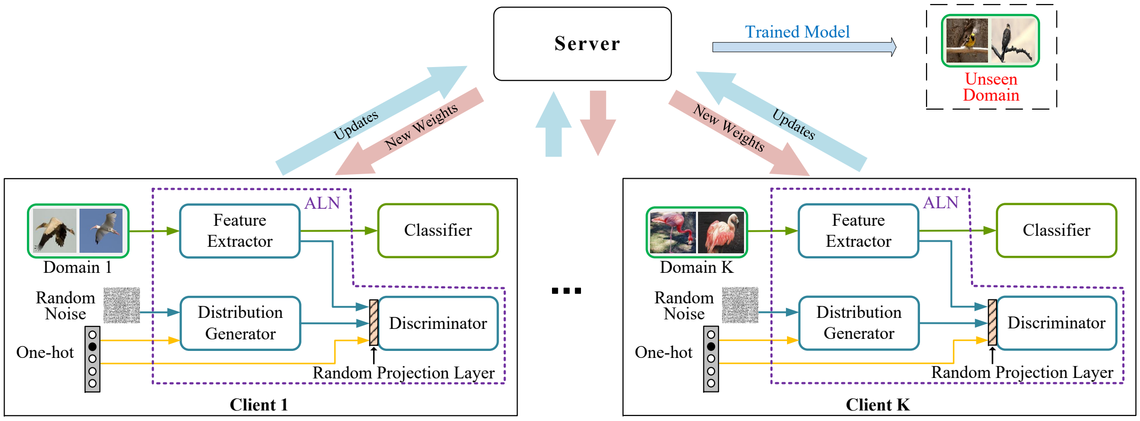


Fig. 2. Illustration of the proposed FedADG scheme. FedADG first aligns each distribution of source domain data to the generated reference distribution through ALN on each client. Meanwhile, via minimizing the loss function of distribution generator, the generated reference distribution is moving close to the "center" of all source domain distributions. Note that the above alignment process is performed in a class-wise manner by using a one-hot vector (encoding the data label). Besides, FedADG uses the loss function of the classifier to assist the learning of the feature extractor. After training, in FedALN, the reference distribution and all the distributions of source domains features are aligned to learn a domain-invariant representation for domain generalization.

TABLE II
NOTATIONS.

Notation	Description
K	number of clients
$\mathcal{S}$	source domain
n	number of source domain data
(x, y)	data and its label
$\mathbf{z}$	random noise
$\mathbf{h}$	feature of source domain data
$p(\mathbf{h})$	feature distribution
$F(\cdot)$	feature extractor
$G(\cdot)$	distribution generator
$D(\cdot)$	discriminator
$C(\cdot)$	classifier

C. FedADG Loss Function

FedADG loss function consists of adversarial loss function and classification loss function.

Adversarial Loss Function. The adversarial loss function includes three loss functions: $\mathcal{L}_{adv\_d}$, $\mathcal{L}_{adv\_f}$, and $\mathcal{L}_{adv\_g}$. They are elaborated on below.

$\mathcal{L}_{adv\_d}$. The loss function $\mathcal{L}_{adv\_d}$ is used to update the parameters in discriminator. During adversarial learning, features extracted by feature extractor $F(\cdot)$ are regarded as negative samples, while features generated by distribution generator $G(\cdot)$ are regarded as positive samples. Given the two types of features with the same one-hot vector (encoding a label $\mathbf{y}$), the discriminator $D(\cdot)$ outputs the probability that they are positive samples. Besides, the output of the $D(\cdot)$ is used to calculate $\mathcal{L}_{adv\_d}$ to measure the difference between the two types of samples. $\mathcal{L}_{adv\_d}$ is defined as

$$\begin{aligned}\mathcal{L}_{adv\_d} = -\mathbb{E}_{\mathbf{h}\sim p(\mathbf{h})}[(1 - D(\mathbf{h}|\mathbf{y}))^2] - \\ \mathbb{E}_{\mathbf{h}'\sim p(\mathbf{h}')}[D(\mathbf{h}'|\mathbf{y})^2],\end{aligned} \tag{2}$$

where $\mathbf{h} = F(\mathbf{x})$ and $p(\mathbf{h})$ is the $F(\cdot)$ generated distribution over input data $\mathbf{x}$. Likewise, $\mathbf{h}' = G(\mathbf{z})$ and $p(\mathbf{h}')$ is the $G(\cdot)$ generated distribution over input data $\mathbf{z}$. The random noise $\mathbf{z}$ is drawn from $[0, 1)$ uniformly.

A random projection layer is pre-pended to the discriminator (as shown in Fig. 2). The random projection function is used to linearly transform data from d_1 dimensions to d_2 dimensions [31], where $d_1 > d_2$. It can be represented as $d_1 \times d_2$ matrix R. Let an $n \times d_1$ matrix $\mathbf{h}$ represent a d_1-dimensional data set. Each row in $\mathbf{h}$ represents d_1-dimensional data and n is the number of data. Let $\bar{\mathbf{h}}$ denote the projected data set and we have $\bar{\mathbf{h}} = \mathbf{h} \times R$. In this work, the random projection layer helps stabilize the ALN training as well as reduce computation.

$\mathcal{L}_{adv\_f}$. For $\mathcal{L}_{adv\_f}$, it is used by the discriminator to evaluate the possibility that $\mathbf{h}$ is the positive sample. In adversarial learning, given a fixed $D(\cdot)$, $\mathcal{L}_{adv\_f}$ is used to update the parameters in the feature extractor. During the training of the feature extractor, the negative samples $\mathbf{h}$ extracted by $F(\cdot)$ are used to deceive the discriminator (in a successful deception, discriminator treats $\mathbf{h}$ as positive samples). Thus, $\mathcal{L}_{adv\_f}$ is given by

$$\mathcal{L}_{adv\_f} = \mathbb{E}_{\mathbf{h}\sim p(\mathbf{h})}[(1 - D(\mathbf{h}|\mathbf{y}))^2]. \tag{3}$$

$\mathcal{L}_{adv\_g}$. For $\mathcal{L}_{adv\_g}$, it is used by discriminator to evaluate the possibility that $\mathbf{h}'$ is the positive sample. In adversarial learning, given a fixed $D(\cdot)$, $\mathcal{L}_{adv\_g}$ is used to update the parameters in the distribution generator. Specifically, $\mathcal{L}_{adv\_g}$ is given by

$$\mathcal{L}_{adv\_g} = \mathbb{E}_{\mathbf{h}'\sim p(\mathbf{h}')}[(1 - D(\mathbf{h}'|\mathbf{y}))^2]. \tag{4}$$

In the definitions of $\mathcal{L}_{adv\_d}$, $\mathcal{L}_{adv\_f}$, and $\mathcal{L}_{adv\_g}$, we borrow the idea from [32] to use the least-squared term instead of the log-likelihood term. This approach helps to address the non-convergence problem during training.

Classification Loss Function. Let $\mathcal{L}_{err}$ be the loss on the classifier's predictions. It is used to measure the error between the label $C(\mathbf{h})$ ($\mathbf{h} = F(\mathbf{x})$) predicted by the classifier $C(\cdot)$ and the real label y of the data. $\mathcal{L}_{err}$ is the standard cross-entropy loss [33] in FedADG. During training, $\mathcal{L}_{err}$ controls the update of both feature extractor and classifier. To prevent overfitting, label smoothing regularization [34] is adopted in computing $\mathcal{L}_{err}$ to reduce the weight of the positive samples in $\mathcal{L}_{err}$.

Complete Loss Function. The complete loss function of FedADG is

$$\mathcal{L}_{\text{FedADG}} = \mathcal{L}_{adv\_d} + \mathcal{L}_{adv\_g} + \lambda_0\mathcal{L}_{adv\_f} + \lambda_1\mathcal{L}_{err}, \tag{5}$$

where $\mathcal{L}_{adv\_d}$, $\mathcal{L}_{adv\_g}$, and $\mathcal{L}_{adv\_f}$ are given in Eq. (2)-(4), respectively. Both λ_0 and λ_1 are adjustable weight hyper-parameters, where $0 < \lambda_0 < 1$, $0 < \lambda_1 < 1$, and $\lambda_0 + \lambda_1 = 1$. During training, the objective of FedADG is to minimize $\mathcal{L}_{\text{FedADG}}$.

Algorithm 1 FedADG Training Algorithm

```
Input: source domains $\mathcal{S} = \{\mathcal{S}_k|k = 1, ..., K\}$, one-hot vector $\mathbf{y}$, model parameters of $F(\cdot)$, $C(\cdot)$, and $G(\cdot)$ $\mathbf{w} = \{w_f, w_c, w_g\}$, parameter of $D(\cdot)$ $w_d$, etc.
Output: Feature extractor $F(\cdot)$ and Classifier $C(\cdot)$

Server executes:
Step s1: Initialize $\mathbf{w}_1$
for round $t = 1, 2, \ldots, T$ do
    for each client $k = 1, 2, \ldots, K$ in parallel do
        Step s2: $\mathbf{w}^k_{t+1} \leftarrow$ ClientUpdate$(k, \mathbf{w}_t)$
    end for
    Step s3: $\mathbf{w}_{t+1} \leftarrow \frac{1}{K}\sum_{k=1}^{K}\mathbf{w}^k_{t+1}$
end for

ClientUpdate$(k, \mathbf{w})$: // Execute on client k
Receive $\mathbf{w} = \{w_f, w_c, w_g\}$ from server
for epoch $i = 1, 2, \ldots, E_0$ do
    Step c2: Sample one mini-batch $S_x$ from $S_k$
    Step c3: Update $w_f$ and $w_c$ on $S_x$ to minimize $\mathcal{L}_{err}$
end for
for epoch $j = 1, 2, \ldots, E_1$ do
    Step c4: Sample one mini-batch $S_x$ from $S_k$
    Step c5: Update $w_f$ and $w_c$ on $S_x$ to minimize $\lambda_0\mathcal{L}_{adv\_f} + \lambda_1\mathcal{L}_{err}$
    Step c6: Use random number generator to generate one mini-batch random numbers $S_z$
    Step c7: Update $w_d$ on $S_x$ and $S_z$ to minimize $\mathcal{L}_{adv\_d}$
    Step c8: Update $w_g$ on $S_z$ and $\mathbf{y}$ to minimize $\mathcal{L}_{adv\_g}$
end for
Step c9: Upload the trained $\mathbf{w}$ to server
```

D. FedADG Training Algorithm

The detailed FedADG training is presented in Algorithm 1. The FedADG training process includes two phases: server execution and client update.

Server Execution Phase. The server is used to aggregate the model parameters uploaded by the clients. To begin the training, in Step s1, the server initializes the parameters $\mathbf{w} = \{w_f, w_c, w_g\}$ of three network components (i.e., feature extractor $F(\cdot)$, classifier $C(\cdot)$, and distribution generator $G(\cdot)$) and distributes them to all clients. During the training process, in Step s2 and Step s3, the server receives and aggregates model parameters from all clients to obtain new parameters. Then, the server sends the aggregated parameters to the clients. After multiple rounds of server-client interaction, the model can be well-trained. Note that the ML model (that is constructed as the series connection of feature extractor and classifier) is applied to target domains.

Client Update Phase. In the training process, the client uses the local discriminator and receives the parameters $\mathbf{w}$ of other components from the server to train on the local data. Specifically, as shown in Step c2 and Step c3, $\mathcal{L}_{err}$ is used to control the training of classifier $C(\cdot)$ and feature extractor $F(\cdot)$. Then, the parameters of $F(\cdot)$ and $C(\cdot)$ are updated to minimize the loss $\lambda_0\mathcal{L}_{adv\_f} + \lambda_1\mathcal{L}_{err}$. The parameters of the discriminator $D(\cdot)$ are updated to minimize the loss $\mathcal{L}_{adv\_d}$. In Step c8, the output of $D(\cdot)$ for the given positive samples with $\mathbf{y}$ is used to update the parameters of $G(\cdot)$ to minimize the loss $\mathcal{L}_{adv\_g}$. After the local training is completed, the client uploads the parameters $\mathbf{w}$ of $F(\cdot), C(\cdot)$, and $G(\cdot)$ to the server.

V. FedADG Analysis

In this section, we first analyze how to learn domain-invariant features in FedADG. Then, we explain how FedADG achieves high performance on target domains.

A. How to Learn Domain-Invariant Features

Under the FL settings, FedADG aligns the distributions of all source domain data to learn the domain-invariant features. In the previous domain generalization techniques, the centralized server can access each client's local data. Thus, it can learn a domain-invariant feature via directly minimizing the discrepancy between the source domains (e.g., using the Maximum Mean Discrepancy (MMD) distance metric [35]). However, in FL, the server can not access each client's local data, making it hard to learn the domain-invariant features. In our proposed Adversarial Learning Network (ALN), the distribution generator is shared among clients, indicating that the reference distribution is identical for all clients. Thus, once

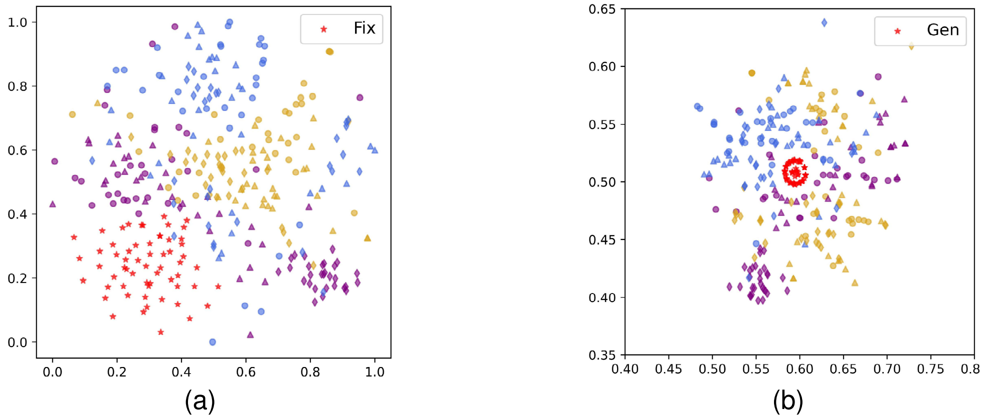


Fig. 3. T-SNE visualization of FedADG features before training (fixed distribution v.s. adaptively generated distribution). The *red asterisk* marker represents the features of reference distribution. Each marker (except the asterisk) represents a distinct source domain. Each color (except red) represents a distinct class label. Each feature is projected into two-dimensional space. (a) Source domain features and fixed distribution features. (b) Source domain features and generated distribution features.

the discriminator is hard to distinguish between the feature extracted from feature extractor and the feature generated from distribution generator, the generated features are considered to be invariant across multi-source domains. Note that ALN can be trained in a federated manner (i.e., FedALN), which eliminates the requirement for centralized training.

B. How to Achieve High Performance

There are two candidate approaches to obtain the reference distribution in FedADG: pre-selected fixed distribution and adaptively generated distribution. Using the adaptively generated distribution can increase the performance of FedADG due to the following three reasons.

Less Distortion During Alignment. As shown in Fig. 3, we employ t-SNE [36] to visualize the source domain features and the reference distribution features before training the model. Gaussian distribution is used as the fixed distribution. It can be observed that the adaptively generated distribution would locate close to the "center" of the distributions from all the source domain features. Hence, the distances between the adaptively generated distribution and the distributions (of source domain data) are smaller than the distances between the fixed reference distribution and the source domain distributions. Thus, using the adaptively generated distribution can reduce the distortion of extracted feature representation during alignment. Less distortion means that the key information of the original source domain data can be largely preserved, resulting in the high generalization performance of FedADG.

Class-Wise Alignment. FedADG uses the label information (encoded in a one-hot vector) in the adversarial training. Thus, the distribution generator generates features for each class in training. It means that the distributions of source domains data are aligned in a class-wise manner. This fine-grained class-wise alignment approach can further improve the performance of FedADG.

More Discriminative Features. Fig. 4 shows the source domain features and the reference distribution features after training the model. The distances between different class clusters in Fig. 4b are more evident than that in Fig. 4a. It indicates that FedADG is capable of learning more discriminative features among different classes for different source domains. Therefore, FedADG has good domain generalization performance.

VI. Experiments

In this section, we conduct experiments to evaluate the performance of FedADG under FL setting for IoT devices with domain generalization. We first compare FedADG with some recent centralized domain generalization solutions on three different datasets. Then, we have an ablation study of the FedADG scheme. Afterward, we investigate the in-domain performance of FedADG. Last, we study the impact of the different reference distributions.

A. Experimental Settings

Implementation. We conduct our experiments using Pytorch 1.7.1 deep learning framework and Python 3.6.5 on Ubuntu 16.04. We use four Linux terminals to simulate the deployment of FedADG in IoT devices. Our server uses Geforce RTX 2080ti GPU with 24G RAM for computing. Following most of the previous studies on FL [1], [6], we simulate the computation of IoT devices (i.e., clients) on the Linux server and then measure FedADG performance. Since the learning process is exactly the same, the performance metrics measured are accurate in our experiments.

Datasets. All experiments are based on three widely used datasets in DG, i.e., VLCS (Pascal [37], LabelMe [38], Caltech-101 [39], and SUN [40]), PACS [41] (Photo, Art painting, Cartoon, and Sketch), and Office-Home [42] (Real-World, Clipart, Product, and Art). All of them have four sub-datasets, which form distinct domains. For each dataset, we utilize the leave-one-domain-out validation strategy. That is, we let one domain serve as the target domain and use the rest domains as source domains. Like [10], each domain is divided into a training set (70%) and a validation set (30%) randomly. The well-trained model is tested on the target domain data.

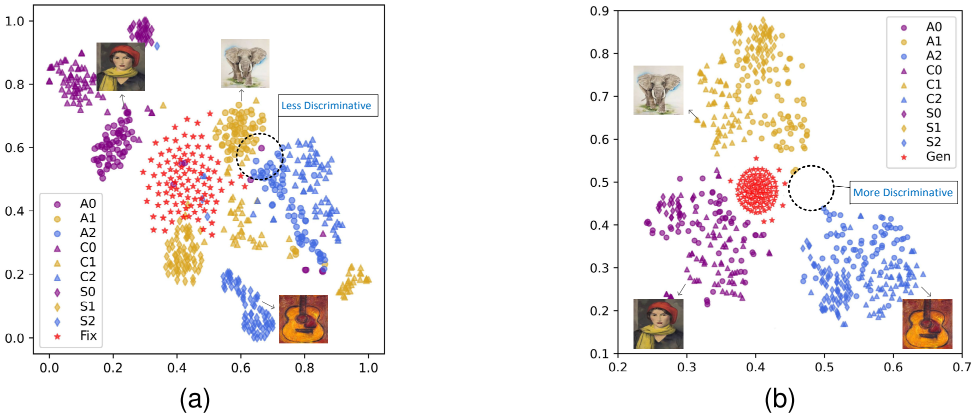


Fig. 4. T-SNE visualization of FedADG features after training (fixed distribution v.s. adaptively generated distribution). There are three source domains (*A, C, S*) data of the PACS dataset. In the legend, Ai represents the class label i of source domain A. The meanings of different colors and markers are the same as Fig. 3. (a) Source domain features and fixed distribution features. (b) Source domain features and generated distribution features.

TABLE III
PARAMETERS SETTING IN EXPERIMENTS ON DIFFERENT DATASETS.

datasets (→)	VLCS	PACS	Office-Home
λ_0	0.85		
λ_1	0.15		
lr_f	0.01	0.001	0.05
lr_g	0.007	0.0007	0.007
lr_d	0.007	0.0007	0.001

Besides, we follow the protocol of [41] to perform experiments on PACS. For Office-Home, we use the same protocol as [43]. Since Office-Home and PACS have related domain types, conducting experiments on these two datasets can check the scalability of FedADG when the number of categories varies from 7 to 65. The three used datasets are standard ones used for studying domain generation. Thus, we can compare our experimental results with prior solutions.

Network Architecture. We test the performance of FedADG by using three pre-trained network architectures as the feature extractor. The three network architectures are the main structure of AlexNet [44], ResNet18 [45], or ResNet50 [45] without including their last layers. Besides, the classifier consists of the last layer of these pre-trained network structures and an additional output layer. For distribution generator and discriminator, both of them have two fully connected layers. The two layers of distribution generator and the first layer of discriminator have the same size as the hidden representation. The size of the second layer in discriminator is set to one.

Model Training. When updating the components' parameters in FedADG, each client uses Stochastic Gradient Descent (SGD) to calculate the model gradient. The Rectified Linear Unit (ReLU) is used as an activation function. We use the data augmentation protocol from JiGen [11] to improve model performance. The source domain local epoch E_0 (for classification) is 3, E_1 (for feature alignment) is 7, and batch size is 16. The global model with the highest accuracy across all source domains is used to test accuracy on the unseen target domain.

Parameter Settings. In all experiments, the parameters of feature extractor are initialized with pre-trained weights using ImageNet [46]. The hyper-parameters of the feature extractor and the initial learning rates of different components on different datasets are detailed in Table III. Notice that the hyper-parameters λ_0 $(0 < \lambda_0 < 1)$ and λ_0 $(0 < \lambda_1 < 1)$ train the feature extractor together, and $\lambda_0 + \lambda_1 = 1$. In particular, lr_f is the learning rate of the feature extractor and classifier, the learning rate of distribution generator and discriminator are lr_g and lr_d, respectively. In our experiments, unless otherwise stated, the hyper-parameters and learning rates are set as the above default configuration.

B. Performance Evaluation

In this section, we compare FedADG with several recent domain generalization solutions on VLCS, PACS, and Office-Home datasets. These solutions are briefly introduced as follows.

1) DANN [28], a neural network that can both accurately classify source data and have features that are invariant across multiple source domains. DANN is the abbreviation of **D**omain-**A**dversarial **N**eural **N**etwork.
2) JiGen [11], a supervised framework for learning to generalize across visual domains by solving jigsaw puzzles.
3) Epi-FCR [26], a scheme to learn domain shift using episodic training. Epi-FCR is the abbreviation of **Epi**sodic-**F**eature and **C**lassifier **R**egularisation.
4) MTSSL [47], a method for enabling models to learn transferable features through a self-supervised task of Gabor filter bank response prediction. MTSSL is the abbreviation of **M**ulti-**T**ask **S**elf-**S**upervised **L**earning.
5) EISNet [48], a network that uses self-supervised learning and metric learning to improve classifier performance on target domains. EISNet is the abbreviation of **E**xtrinsic and **I**ntrinsic **S**upervision **Net**work.
6) L2A-OT [49], a method to learn domain-invariant features by augmenting the source domain with synthetic data. L2A-OT is the abbreviation of **L**earning **to** **A**ugment by **O**ptimal **T**ransport.

TABLE IV
THE AVERAGE CLASSIFICATION ACCURACY USING LEAVE-ONE-DOMAIN-OUT VALIDATION ON VLCS DATASET.

Paradigm	Backbone	Method	Sun	Pascal	Labelme	Caltech	Avg.
Centralized w/o privacy concern	AlexNet	MTSSL [47]	58.88	62.59	64.99	89.15	67.67
	AlexNet	DANN [28]	63.60	66.40	64.00	92.60	72.40
	AlexNet	Epi-FCR [26]	65.90	67.10	64.30	94.10	72.85
	AlexNet	JiGen [11]	64.30	70.62	60.90	96.93	73.19
	AlexNet	RSC [27]	**68.32**	**73.93**	**61.86**	**97.61**	**75.43**
Distributed	AlexNet	FedAvg [3]	46.65	48.77	52.32	71.43	54.79
	AlexNet	FedADG (ours)	**71.81**	**73.40**	**61.07**	**93.44**	**75.09**
Centralized w/o privacy concern	ResNet18	JiGen [11]	71.40	70.93	62.06	96.17	75.14
	ResNet18	RSC [27]	**72.10**	**73.81**	**62.51**	**96.21**	**76.16**
Distributed	ResNet18	FedAvg [3]	62.78	65.12	57.48	90.63	69.00
	ResNet18	FedADG (ours)	**74.95**	**73.20**	**61.20**	**95.78**	**76.28**

7) DSON [50], a scheme that combines batch normalization and instance normalization to enhance generalization performance on target domains. DSON is the abbreviation of **D**omain **S**pecific **O**ptimized **N**ormalization.
8) Mixstyle [12], a method for mixing features across source domains to synthesize new source domains to optimize model generalization.
9) RSC [27], a method to discard dominant features of training data to optimize the generalization ability of a model. RSC is the abbreviation of **R**epresentation **S**elf-**C**hallenging.

All prior solutions require centralized data access, whereas FedADG is used for domain generalization in a distributed way. We also compare it with a recent state-of-the-art DG method: FedDG [14], which does not centralize the dataset and is also trained in the FL setting. Besides, FedAvg [3] is used as a baseline. We do not compare COPA [13] because of the following reasons. First, it sacrifices security (refer to Section I for more details). Second, the project codes are not publicly available. We also do not compare FL optimization methods (e.g., FedProx, FedNova, and MOON) since these papers do not focus on the domain generation problem. These optimization methods differ from FedADG in the following aspects. First, the source domain data in FedADG are from different IoT devices with domain discrepancy, instead of different subsets from the same dataset [13]. Second, the discrepancy between the test datasets and training datasets in FedADG also makes it more complex than those in federated optimization methods. Moreover, FedADG requires building a model that has high performance when testing over the related but unseen target dataset rather than seen dataset. Note that all solutions used in the comparison are constructed using the same pre-trained network as FedADG. For each test, we run 5 trails and report the average results which are shown in Table IV, Table V, and Table VI. In the three tables, each column containing experimental results (except Avg. column) shows the results when one domain is chosen as the target domain. We highlight the best results in bold font.

VLCS. Table IV shows the domain generalization accuracy on VLCS. We use two pre-trained networks, AlexNet and ResNet18, as the backbone to compare FedADG with some recent domain generalization solutions. Table IV shows that FedADG outperforms most of the compared centralized solutions. The performance is comparable to the recent RSC solution. Besides, FedADG has good performance in both small and large backbone networks.

PACS. Table V shows the domain generalization accuracy on PACS. We use the same backbone network as in VLCS. In Table V, we find that the performance of FedADG is better than most of the compared centralized solutions. Furthermore, the performance of FedADG is obviously improved compared to FedDG, which is also a distributed DG method. Besides, we observe that FedDG does not improve performance like FedADG (as the backbone size increases from AlexNet to ResNet18).

The accuracy of FedADG is slightly worse than the recent solution L2A-OT. Specifically, FedADG significantly improves the performance in the Sketch domain.

Office-Home. We also evaluate FedADG on the Office-Home dataset and the results are shown in Table VI. ResNet18 and ResNet50 are applied as the backbone. In Table VI, we observe that FedADG is better than other solutions.

Compared with Traditional Centralized Method. Table IV, Table V, and Table VI present the domain generalization accuracy of FedADG and prior traditional centralized machine learning solutions without FL. These traditional ML approaches are described in Section VI-B. In these tables, the paradigm of these traditional ML solutions is represented as "centralized w/o privacy concerns". That is, these solutions require the centralized server to access source domain data and expose sensitive local information. In Table IV and Table V, we find that the performance of FedADG is comparable to the traditional ML methods. In particular, the generalization accuracy of the AlexNet-based FedADG on PACS is over 1% higher than the centralized approaches (e.g., MTSSL, Epi-FCR, JiGen). Besides, Table VI shows that FedADG performs significantly better than other traditional ML methods on Office-Home dataset. In summary, our proposed FedADG

TABLE V
THE AVERAGE CLASSIFICATION ACCURACY USING LEAVE-ONE-DOMAIN-OUT VALIDATION ON PACS DATASET.

Paradigm	Backbone	Method	Sketch	Artpaint	Cartoon	Photo	Avg.
Centralized w/o privacy concern	AlexNet	DANN [28]	57.00	63.20	67.50	88.10	68.95
	AlexNet	MTSSL [47]	63.91	61.67	67.41	84.31	69.32
	AlexNet	Epi-FCR [26]	65.00	64.70	72.30	86.10	72.03
	AlexNet	JiGen [11]	**65.18**	**67.63**	**71.71**	**89.00**	**73.38**
Distributed	AlexNet	FedAvg [3]	60.52	65.97	62.93	86.95	69.09
	AlexNet	FedDG [14]	67.63	66.50	63.51	**89.26**	71.73
	AlexNet	FedADG (ours)	**69.15**	**71.68**	**70.14**	87.01	**74.50**
Centralized w/o privacy concern	ResNet18	Epi-FCR [26]	73.00	82.10	77.00	93.90	81.50
	ResNet18	JiGen [11]	71.35	79.42	75.25	96.03	80.51
	ResNet18	EISNet [48]	**74.33**	81.89	76.44	95.93	82.15
	ResNet18	L2A-OT [49]	73.60	**83.30**	**78.20**	**96.20**	**82.81**
Distributed	ResNet18	FedAvg [3]	70.51	77.18	73.97	89.86	77.88
	ResNet18	FedDG [14]	61.53	64.08	72.70	89.26	71.89
	ResNet18	FedADG (ours)	**78.56**	**81.39**	**75.39**	**93.64**	**82.25**

TABLE VI
THE AVERAGE CLASSIFICATION ACCURACY USING LEAVE-ONE-DOMAIN-OUT VALIDATION ON OFFICE-HOME DATASET.

Paradigm	Backbone	Method	Real	Clipart	Product	Art	Avg.
Centralized w/o privacy concern	ResNet18	JiGen [11]	72.79	47.51	71.47	53.04	61.20
	ResNet18	DSON [50]	**74.68**	45.70	**71.84**	**59.37**	62.90
	ResNet18	RSC [27]	74.54	**47.90**	71.63	58.42	**63.12**
Distributed	ResNet18	FedAvg [3]	71.31	52.11	67.60	48.00	59.76
	ResNet18	FedADG (ours)	**74.98**	**53.98**	**70.83**	**58.13**	**64.48**
Centralized w/o privacy concern	ResNet50	Mixstyle [12]	69.20	**53.20**	68.20	51.10	60.43
	ResNet50	RSC [27]	**75.10**	51.40	**74.80**	**60.70**	**65.50**
Distributed	ResNet50	FedAvg [3]	71.83	54.06	69.14	53.07	62.03
	ResNet50	FedADG (ours)	**76.48**	**56.09**	**74.87**	**60.27**	**66.93**

TABLE VII
THE AVERAGE IN-DOMAIN CLASSIFICATION ACCURACY USING LEAVE-ONE-DOMAIN-OUT VALIDATION ON PACS DATASET.

Paradigm	PACS	Backbone	Sketch	Artpaint	Cartoon	Photo	Avg.
Distributed	FedAvg (in)	AlexNet	**99.98**	**99.98**	**99.97**	**99.98**	**99.98**
	FedADG (in)	AlexNet	98.13	98.06	97.96	98.03	98.05
	FedAvg (in)	ResNet18	99.12	98.08	98.38	95.83	97.85
	FedADG (in)	ResNet18	**99.83**	**98.47**	**94.08**	**99.73**	**98.03**

can achieve good domain generalization capability while still protecting data privacy.

C. Ablation Study

An ablation study investigates the performance of FedADG by removing a certain component to understand the contribution of the component to the overall FedADG scheme. We perform ablation experiments on VLCS and PACS datasets using AlexNet. Specifically, we focus on the distribution generator and discriminator, along with the data label (encoding as a one-hot vector) in these two components. When we remove the one-hot vectors from both distribution generator and discriminator, the remained FedADG is denoted as “FedADG w/o one-hot”. We use “FedADG w/o RP” to denote FedADG without the Random Projection (RP) layer. “FedADG w/o G&D” represents FedADG without distribution generator and discriminator. Fig. 5 shows the ablation study results. The results analysis is performed as follows.

FedADG w/o one-hot. As shown in Fig. 5, FedADG has higher accuracy in each target domain than FedADG w/o one-

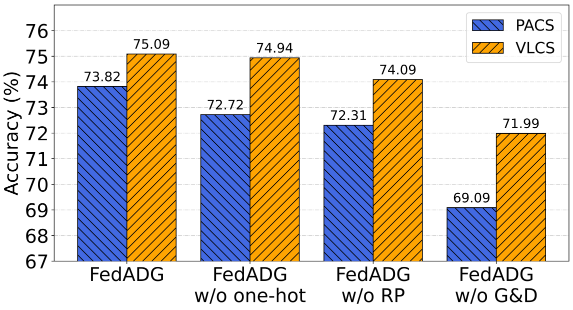


Fig. 5. Ablation study on VLCS and PACS datasets. "*FedADG w/o one-hot*" means to remove the one-hot vector from FedADG. "*FedADG w/o RP*" represents the model by removing the random projection layer from FedADG. Besides, "*FedADG w/o G&D*" represents the model by removing distribution generator and discriminator from FedADG.

hot. These results demonstrate that the class-wise alignment can increase the generalization performance of FedADG.

FedADG w/o RP. Fig. 5 shows that the accuracy of FedADG w/o RP is less than FedADG. The function of random projection is to decrease the dimension of features. The low-dimension features stabilize the training of ALN and help to do feature alignment. Thus, the random projection layer can improve the performance of FedADG on target domains.

FedADG w/o G&D. Fig. 5 shows that the accuracy of FedADG w/o G&D is less than FedADG. In terms of average accuracy, FedADG is over 3% higher than FedADG w/o G&D because the latter lacks of domain generalization design.

D. In-domain Performance Evaluation

The previous experimental results are tested on out-of-domain data. We also measure the performance of FedADG on in-domain data. We consider the commonly used experimental setting: both the training and testing data come from the same domain. AlexNet and ResNet18 are applied as the backbone and PACS dataset is used in experiments. Table VII compares the in-domain performance of FedADG and FedAvg. In Table VII, "FedAvg (in)" and "FedADG (in)" represent the in-domain performances of FedAvg and FedADG, respectively.

The results show that the in-domain performance of FedADG is comparable to that of FedAvg. It indicates that FedADG can be used on both source domains and the unseen target domain. In practice, the clients can train both FedAvg (used for in-domain data) and FedADG (used for out-domain data).

E. Impact of Different Reference Distributions

As we have discussed in Section IV-B, the reference distribution helps to align the feature distributions of all source domain data, which in turn improves the accuracy of classification on target domains. In general, most existing works adopt fixed reference distribution without considering the distortion it may cause to the source domain distribution. In this part, we investigate how the adaptively generated distribution can outperform the fixed settings such as Gaussian distribution ($\mathcal{N}$), Uniform distribution ($\mathcal{U}$), and Laplace distribution [35]. The experimental results of different reference distributions using AlexNet on PACS are shown in Table VIII.

TABLE VIII
EXPERIMENTAL RESULTS WHEN USING DIFFERENT REFERENCE DISTRIBUTIONS IN FEDADG ON PACS DATASET.

Unseen domain (→)	Sketch	Artpaint	Cartoon	Photo	Avg.
fixed reference distribution					
$\mathcal{N} \sim (0, \mathbf{I})$	49.45	53.32	53.54	75.69	58.00
$\mathcal{U} \sim [-1, 1]$	38.23	57.71	55.33	81.26	58.13
Laplace$(1/\sqrt{2})$	44.29	54.69	55.84	84.31	59.78
adaptively generated distribution					
FedADG (ours)	**69.15**	**68.99**	**70.14**	**87.01**	**73.82**

In Table VIII, the parameter of the Laplace distribution we compared in the experiment is $1/\sqrt{2}$, which is proved by Li *et al.* [35] to have the best effect on target domains. Moreover, we find that the accuracy of the Laplace distribution with the parameter of $1/\sqrt{2}$ is higher than the other two fixed distributions in the table. By observing all experimental results in Table VIII, we notice that the average accuracy of the adaptively generated reference distribution can be 10% higher than the accuracy of the fixed reference distributions. Especially in the target domains of Cartoon and Sketch, the accuracy of using the adaptively generated distribution in FedADG is about 20% higher than the accuracy of the fixed reference distribution. The remarkable result of FedADG supports the effectiveness of using adaptively generated distribution. It proves that the generated adaptive reference distribution can promote the performance of the model for target domains.

VII. CONCLUSION

In this paper, we propose the FedADG scheme under the federated learning setting for IoT devices with domain generalization. The main idea of FedADG is to learn the domain-invariant feature representation in FL while eliminating the requirement for a centralized server to access IoT devices' local data. First, we propose the federated adversarial learning approach to measure and align the distributions among different source domains via matching each distribution to the reference distribution. Specifically, we use the federated adversarial learning technique to adaptively learn a dynamic distribution (by accommodating all source domains) as the reference distribution. Therefore, the learned feature representation tends to be universal. Then, our proposed FedADG uses the adaptively generated reference distributions and class-wise alignment technique. It ensures that FedADG has good generalization performance over the unseen target domains while protecting local data privacy. Furthermore, we analyze the explainability of FedADG, which helps researchers to optimize the model and make the model more trustworthy. Finally, the effectiveness of FedADG has been demonstrated by intensive simulations. Thus, FedADG significantly boosts FL performance for IoT devices. There are two directions to launch further research. First, we aim to handle the scenario in which the unseen target domain contains more classes than the seen source domain. Second, we plan to find an optimization method to automatically balance the classification training epoch and the alignment training epoch to obtain a better federated generalization performance.

REFERENCES

[1] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal (IoT-J)*, no. 7, pp. 6360–6368, 2020.

[2] C. Chen, Q. Liu, X. Wang, C. Liao, and D. Zhang, "semi-traj2graph: Identifying fine-grained driving style with gps trajectory data via multi-task learning," *IEEE Transactions on Big Data*, pp. 1–1, 2021.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[4] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet of Things Journal (IoT-J)*, no. 3, pp. 1817–1829, 2021.

[5] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal (IoT-J)*, no. 1, pp. 1–24, 2022.

[6] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in iot-edge devices," *IEEE Internet of Things Journal (IoT-J)*, pp. 3930–3944, 2022.

[7] G. Blanchard, G. Lee, and C. Scott, "Generalizing from several related classification tasks to a new unlabeled sample," in *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[8] J. Wang, C. Lan, C. Liu, Y. Ouyang, and T. Qin, "Generalizing to unseen domains: A survey on domain generalization," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, 2021, pp. 4627–4635.

[9] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013, pp. 10–18.

[10] M. Ghifary, W. Kleijn, M. Zhang, and D. Balduzzi, "Domain generalization for object recognition with multi-task autoencoders," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2551–2559.

[11] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2229–2238.

[12] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang, "Domain generalization with mixstyle," in *9th International Conference on Learning Representations(ICLR)*, 2021.

[13] G. Wu and S. Gong, "Collaborative optimization and aggregation for decentralized domain generalization and adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 6484–6493.

[14] Q. Liu, C. Chen, J. Qin, Q. Dou, and P.-A. Heng, "Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 1013–1023.

[15] B. Stasiak and M. Yatsymirskyy, "Fast orthogonal neural network for adaptive fourier amplitude spectrum computation in classification problems," in *ICMMI*, vol. 59, 2009, pp. 327–334.

[16] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," *IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, 2017.

[17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[18] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems, (MLSys)*, 2020.

[19] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[20] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10 713–10 722.

[21] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[22] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, 2020.

[23] R. Zhao, G. Gui, Z. Xue, J. Yin, T. Ohtsuki, B. Adebisi, and H. Gaanin, "A novel intrusion detection method based on lightweight neural network for internet of things," *IEEE Internet of Things Journal (IoT-J)*, pp. 9960–9972, 2022.

[24] H. K. Mittal, A. K. Tripathi, A. C. Pandey, M. D. Alshehri, M. Saraswat, and R. Pal, "A new intrusion detection method for cyber-physical system in emerging industrial iot," *Comput. Commun.*, vol. 190, pp. 24–35, 2022.

[25] Y. Otoum, Y. Wan, and A. Nayak, "Federated transfer learning-based IDS for the internet of medical things (iomt)," in *IEEE Globecom 2021 Workshops, Madrid, Spain, December 7-11, 2021.* IEEE, 2021, pp. 1–6.

[26] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales, "Episodic training for domain generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1446–1455.

[27] Z. Huang, H. Wang, E. P. Xing, and D. Huang, "Self-challenging improves cross-domain generalization," in *European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, vol. 12347, 2020, pp. 124–140.

[28] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research (JMLR)*, pp. 2096–2030, 2016.

[29] S. Ben-David, J. Blitzer, K. Crammer, and F. C. Pereira, "Analysis of representations for domain adaptation," in *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[30] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," in *8th International Conference on Learning Representations, (ICLR)*, 2020.

[31] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach," in *Machine Learning, Proceedings of the Twentieth International Conference (ICML)*, 2003, pp. 186–193.

[32] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2813–2821.

[33] R. Y. Rubinstein, "Cross-entropy and rare events for maximal cut and partition problems," *ACM Trans. Model. Comput. Simul.*, pp. 27–53, 2002.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[35] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5400–5409.

[36] L. van der Maaten and G. E. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[37] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, pp. 303–338, 2009.

[38] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, pp. 157–173, 2007.

[39] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Conference on Computer Vision and Pattern Recognition Workshop*, pp. 178–178, 2004.

[40] M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky, "Exploiting hierarchical context on a large database of object categories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 129–136.

[41] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5543–5551.

[42] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5385–5394.

[43] A. D’Innocente and B. Caputo, “Domain generalization with domain-specific aggregation modules,” in *Pattern Recognition - 40th German Conference (GCPR)*, vol. 11269, 2018, pp. 187–198.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Conference on Neural Information Processing Systems (NIPS)*, pp. 84 – 90, 2012.

[45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2016, pp. 770–778.

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[47] I. Albuquerque, N. Naik, J. Li, N. Keskar, and R. Socher, “Improving out-of-distribution generalization via multi-task self-supervised pretraining,” 2020.

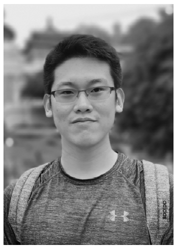
[48] S. Wang, L. Yu, C. Li, C. Fu, and P. Heng, “Learning from extrinsic and intrinsic supervisions for domain generalization,” in *European Conference on Computer Vision (ECCV)*, vol. 12354, 2020, pp. 159–176.

[49] K. Zhou, Y. Yang, T. M. Hospedales, and T. Xiang, “Learning to generate novel domains for domain generalization,” in *European Conference on Computer Vision (ECCV)*, vol. 12361, 2020, pp. 561–578.

[50] S. Seo, Y. Suh, D. Kim, G. Kim, J. Han, and B. Han, “Learning to optimize domain specific normalization for domain generalization,” in *European Conference on Computer Vision (ECCV)*, vol. 12367, 2020, pp. 68–83.

Liling Zhang received the bachelor’s degree in computer science and technology from Harbin Engineering University, Harbin, China, in 2020. She is currently pursuing the master’s degree in computer technology with the Chongqing University, Chongqing, China. Her research interest includes federated learning and domain generalization.

Xinyu Lei (Member, IEEE) received the Ph.D. degree with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA, in 2021. He is currently an Assistant Professor with the Department of Computer Science, Michigan Technological University, Houghton, MI, USA. His current research focuses on machine learning and cybersecurity. He worked as a Research Assistant with the Texas A&M University at Qatar, Doha, Qatar, in 2013. In 2017, he worked as a Research Intern with Ford Motor Company, Dearborn, MI, USA.



Yichun Shi received his PhD degree from Michigan State University, East Lansing, MI, USA, in 2021. His research focuses on machine learning and computer vision. He is currently a research scientist at ByteDance, CA, USA. He worked as a research intern at NEC Lab of America, San Jose, CA, USA. He worked as a research intern at Visa, Foster City, CA, USA.

Hongyu Huang (Member, IEEE) is an Associate Professor at the College of Computer Science, Chongqing University, Chongqing, China. He obtained his Ph.D. degree from Shanghai Jiao Tong University, China in 2009. He received the B.Sc. and M.Sc. degrees in computer science from Chongqing Normal University and Chongqing University in 2002 and 2005, respectively. Dr. Huang got published over 50 papers. His research interests include privacy protection, mobile computing, federated learning, and vehicular ad hoc networks.

Chao Chen (Senior Member, IEEE) is a Full Professor at College of Computer Science, Chongqing University, Chongqing, China. He obtained his Ph.D. degree from Pierre and Marie Curie University and Institut Mines-TELECOM/TELECOM Sud Paris, France in 2014. He received the B.Sc. and M.Sc. degrees in control science and control engineering from Northwestern Polytechnical University, Xi’an, China, in 2007 and 2010, respectively. His research interests include pervasive computing, mobile computing, urban logistics, data mining from large-scale GPS trajectory data, and big data analytics for smart cities. He got published over 80 papers including 20 ACM/IEEE Transactions. His work on taxi trajectory data mining was featured by IEEE SPECTRUM in 2011, 2016, and 2020 respectively. He was also the recipient of the Best Paper Runner-Up Award at MobiQuitous 2011.