Scientific Research Publishing

Open Journal of Applied Sciences, 2025, 15(4), 987-1001
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Federated Domain Generalization by Intra-Client Stable Feature Learning and Inter-Client Domain Adversarial Alignment

Xin Zhao[1], Wu Ai[1,2*]

[1]School of Mathematics and Statistics, Guilin University of Technology, Guilin, China
[2]Guangxi Colleges and Universities Key Laboratory of Applied Statistics, Guilin, China
Email: *aiwu818@gmail.com

How to cite this paper: Zhao, X. and Ai, W. (2025) Federated Domain Generalization by Intra-Client Stable Feature Learning and Inter-Client Domain Adversarial Alignment. *Open Journal of Applied Sciences*, **15**, 987-1001.
https://doi.org/10.4236/ojapps.2025.154067

Received: March 5, 2025
Accepted: April 12, 2025
Published: April 15, 2025

Abstract

Federated Learning (FL), which serves as a distributed learning model focused on privacy protection, has received widespread attention due to its ability to collaboratively train models without exchanging clients' raw data. However, since each client independently collects its local dataset, a single client may contain data from multiple domains, which leads to the existence of an agnostic distribution bias both intra-client and inter-client biases, and it is still a challenge to implement cross-domain generalization of the model in such an FL setting. In this paper, we propose a new stable federated adversarial domain generalization (Stable-FedADG) algorithm, which considers both intra-client and inter-client multi-domain distributions and through intra-client stable feature learning and inter-client domain adversarial alignment to learn domain-invariant features in federated scenarios. Experiments verify that the Stable-FedADG algorithm enhances the overall model performance across various federated environments.

Keywords

Federated Domain Generalization, Stable Feature Learning, Domain Adversarial Alignment, Agnostic Distribution Bias

1. Introduction

In recent years, with the popularization of smart devices, the amount of global data has experienced explosive growth, and AI technology, as the core of intelligent Internet technology, needs to mine useful information from massive datasets to help the Internet system to provide more accurate and broader services and

decisions [1]. However, due to the huge storage resources, computational overhead and privacy protection, the traditional centralized training method is no longer feasible, so federated learning has received widespread attention as a new distributed learning paradigm with privacy protection [2] [3]. In the federated architecture, data is stored in each client in a decentralized manner, and each client can collect data independently and has a certain computational capability. In every round of Federated Learning (FL), the client trains its local model using its own local data, leveraging the received global model as a foundation. Afterward, the client transmits the model updates back to the server. The server then aggregates these updates and uses them to refine the global model. Once the global model is updated, it is sent back to each client for the subsequent round of model refinement. Throughout this process, clients are not required to share their raw datasets.

FL is able to utilize datasets from various clients for collaborative model training, while safeguarding data privacy. FedAvg proposed in [4] is a classical FL paradigm, which mainly utilizes the weighted average of client updates to learn a global model, and is widely used for its ability to conduct several local updates on each client, thereby significantly minimizing communication costs. However, due to the different environments, data collection methods and user preferences of each client, the distribution of data stored in different clients may be significantly different and violate the traditional IID assumption, which is one of the main challenges of FL: the statistical heterogeneity problem [3] [5]. Although traditional FedAvg exhibits efficient performance in homogeneous scenarios, its convergence and training speed are usually challenged when facing heterogeneous data [6]. There have been many works that study the statistical heterogeneity problem. FedProx in [7] adds a proximal term for FedAvg to limit the local update bias for reducing the dispersion of the local model; SCAFFOLD [8] solves the clients drift problem by designing control variables to correct the local update direction. To solve the data heterogeneity problem, another solution is to construct personalized models based on the local distribution [9] [10]. The studies mentioned above primarily aim to enhance the model's performance within the participating clients, typically assuming that the test data is a subset of the client's dataset. However, they often overlook the issue of generalization to federated domains that have not been encountered during training. In this paper, we aim to learn a global model that can be generalized to other unseen clients.

Federated Domain Generalization (FDG) is an extension of FL that combines FL and Domain Generalization (DG) techniques to collaboratively train a model using multiple source domains (multiple clients). FDG, the same as DG, is mainly studied for the case that the distribution of the test set and training set is non-IID, with the aim of enhancing the model's predictive performance in the unseen domains [11]. Many works have been developed in FDG. References [12] [13] studied the problems related to unsupervised domain adaptation in the FL paradigm, these methods usually need the data from target domain to adapt the model, but

the data from the target domain are usually agnostic in real scenarios. COPA [14] solves FDG by aggregating the weights of domain-invariant feature extractors while maintaining an ensemble of domain-specific classifiers, where there may be a privacy leakage problem. FedDG [15] adapts local learning to domain distribution changes by exchanging data distribution information among different clients and utilizing a boundary-oriented episodic learning paradigm based on the obtained multi-source distributions. FedADG [16] uses a federated adversarial learning approach to learn domain-invariant features by aligning the distributions of different clients to a dynamic reference distribution. However, these methods typically treat each client as a source domain and align data distributions client by client, which only focuses on the inter-client domain distribution differences and ignores the multi-domain information that a single client might contain. When intra-client domain bias or unbalance also exists, merely considering inter-client domain distribution differences is insufficient for enhancing the model's generalization performance.

Stable Learning (SL) has been extensively studied in the field of out-of-distribution generalization. Reference [17] presents a causal regularization learning framework to mitigate agnostic data selection bias and enhance the robustness of predictive models. Additionally, it introduces the Causal Regularization Logistic Regression (CRLR) algorithm, which integrates causal techniques into traditional logistic regression models. Kuang *et al.* [18] propose a generalized Decorrelated Weighting Regression (DWR) algorithm designed to handle both multicategorical and continuous features. Cui *et al.* [19] [20] provide a theoretical explanation for the effectiveness of Stable Learning (SL) in addressing covariate bias generalization and its links to machine learning and causal inference, offering rigorous theoretical support. Zhang *et al.* [21] extend SL to deep learning models to better handle complex data types, introducing Deep Stable Learning (StableNet). Although SL has been rapidly developed in the centralized context, it has not been studied in the federated context where data are scattered and stored in different clients and restricted by privacy protection.

Based on the above approach, the main contribution of this paper is to propose a new federated domain generalization algorithm, which simultaneously focuses on the multi-domain distributions of intra-clients and inter-clients to further enhance the generalization capability of the global model. In this paper, the SL idea is introduced into the FL architecture to eliminate the agnostic distributional bias that exists intra-client by extracting stable features using a sample reweighting scheme. In addition, this paper combines the federated adversarial learning network (FedALN) [16] to dynamically align the feature distributions of each client to the same reference distribution, which indirectly realizes the alignment of feature distributions between clients, and thus eliminates the differences of feature distributions between clients. This paper introduces the stable federated adversarial domain generalization (Stable-FedADG) method only needs to pass model parameters during training which protects data privacy.

2. Preliminaries

2.1. Problem Statement

The aim of this paper is to develop a generalized federated domain generalization mechanism that takes into account both intra-client and inter-client domain distributions to collaboratively train a model with strong generalization performance using unshared data from different clients. Assume that there are K clients, each of which collects data independently to form its own local dataset, where the local dataset of the k-th client is denoted as $(\mathbf{X}_k, \mathbf{Y}_k)$, $\mathbf{X}_k \subset \mathbb{R}^{N_k \times M_X}$, $\mathbf{Y}_k \subset \mathbb{R}^{N_k \times M_Y}$, and $X_{i,k}$ and $Y_{i,k}$ denote the i-th sample data of the k-th client, each client can train the model independently and can transmit the information with the server. Let $\mathcal{X} \subset \mathbb{R}^{M_X}$, $\mathcal{Y} \subset \mathbb{R}^{M_Y}$ and $\mathcal{Z} \subset \mathbb{R}^{M_Z}$ denote respectively the input space, the output space and the feature space. $F: \mathcal{X} \to \mathcal{Z}$ denotes the feature extractor, $C: \mathcal{Z} \to \mathcal{Y}$ denotes the classifier, $\mathbf{Z}_k \subset \mathbb{R}^{N_k \times M_Z}$ denotes the sample features of the k-th client extracted through the feature extractor $F(\cdot)$, $\mathbf{Z}_{k,:,i}$ denotes the i-th feature variable in the feature space of the k-th client and N_k denotes the sample size of the k-th client. Since each client collects data independently, the problem of agnostic distribution bias may exist both intra-client and inter-client, so we would like to give a generalized federated domain generalization mechanism that improves the generalization performance of model on unseen clients while observing federated learning privacy preservation.

2.2. Federated Adversarial Learning Network (FedALN)

FedALN is a network structure for aligning inter-client feature distributions in the federated context proposed in [16], which is mainly based on Adversarial Learning Networks (ALNs), by generating a dynamic reference distribution and aligning the feature distributions of each client to this reference distribution to reduce the inter-clients' domain distribution differences and improve the generalization performance of the global model. Its effectiveness has been verified by a large number of experiments. Our proposed method also adopts FedALN to eliminate the impact of inter-client domain bias on model performance and below we provide a brief review of FedALN.

FedALN consists of 3 main components: the feature extractor $F(\cdot)$, the distribution generator $G(\cdot)$ and the discriminator $D(\cdot)$. Among them, the feature extractor $F(\cdot)$ is used to extract real data features $\mathbf{Z}$ from the local raw data and the distribution generator $G(\cdot)$ is used to generate the reference distribution and output the generated features $\mathbf{Z}'$, the discriminator $D(\cdot)$ is employed to differentiate between the real data features $\mathbf{Z}$ and the generated features $\mathbf{Z}'$.

Specifically, each client has a local ALN and the data distribution differences between different clients are converted into feature distribution differences. Firstly, each client aligns the local feature distribution with the generated reference distribution by training the local ALNs, and then the model parameters of the local ALNs of all clients are passed to the server for model parameter aggregation to form a new FedALN to be passed to each client again. Through multiple interac-

tions of the local clients and the server, the feature distributions of different clients will be aligned to the same reference distribution, thus indirectly realizing the alignment of the feature distributions between clients.

Then we will describe specifically the three components of FedALN and the corresponding adversarial loss functions.

Discriminator $D(\cdot)$: The discriminator $D(\cdot)$ is primarily responsible for distinguishing between the true data features $\mathbf{Z}$, which are obtained from the feature extractor $F(\cdot)$, and the false features $\mathbf{Z}'$, generated by the distribution generator $G(\cdot)$. To facilitate this task, one-hot encoded labels are used to ensure that the discriminator $D(\cdot)$ can discriminate between true and false features according to their respective categories.

In the training process, the true data feature $\mathbf{Z}$ is regarded as a negative sample and the false feature $\mathbf{Z}'$ is regarded as a positive sample, input data features (true data feature $\mathbf{Z}$, false feature $\mathbf{Z}'$) and one-hot encoded labels $\mathbf{y}$, and output the probability that the feature is judged as a positive sample, *i.e.*, the probability that the feature comes from the distribution generator $G(\cdot)$. The corresponding adversarial loss function is expressed as follows:

$$\mathcal{L}_{adv\_d} = -\left(\mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z})}\left[\left(1-D(\mathbf{Z}\,|\,\mathbf{y})\right)^2\right]+\mathbb{E}_{\mathbf{Z}'\sim p(\mathbf{Z}')}\left[D(\mathbf{Z}'\,|\,\mathbf{y})^2\right]\right), \tag{1}$$

where $\mathbf{Z}=F(\mathbf{X})$ denotes the real data features extracted from the feature extractor, $\mathbf{Z}'=G(\boldsymbol{\varepsilon})$ denotes the false features generated from the distribution generator and $\boldsymbol{\varepsilon}$ is a random noise sample from a uniform distribution $[0,1)$, $p(\mathbf{Z})$ denotes the real feature distribution generated by the raw data $\mathbf{X}$ through the feature extractor $F(\cdot)$, and $p(\mathbf{Z}')$ denotes the reference distribution generated by random noise $\boldsymbol{\varepsilon}$ through the distribution generator $G(\cdot)$. This adversarial loss function is mainly used to measure the ability of the discriminator to correctly discriminate.

Feature extractor $F(\cdot)$: The feature extractor $F(\cdot)$ is mainly used to extract real data features $\mathbf{Z}$ from local raw data, and when it inputs local raw data $\mathbf{X}$, it outputs extracted feature information $\mathbf{Z}$. During the ALNs training, the feature distribution $p(\mathbf{Z})$ extracted by the feature extractor $F(\cdot)$ needs to be progressively aligned to the reference distribution $p(\mathbf{Z}')$, *i.e.*, given a trained discriminator $D(\cdot)$ with fixed parameters, when the feature distribution $p(\mathbf{Z})$ is closer to the reference distribution $p(\mathbf{Z}')$, the more likely the discriminator $D(\cdot)$ fails to correctly discriminate the real data features $\mathbf{Z}$ from the feature distribution $p(\mathbf{Z})$. At this point the discriminator $D(\cdot)$ is used to measure the gap between the feature distribution $p(\mathbf{Z})$ and the reference distribution $p(\mathbf{Z}')$. The corresponding adversarial loss function of the feature extractor $\mathcal{L}_{adv\_f}$ is expressed as follows:

$$\mathcal{L}_{adv\_f} = \mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z})}\left[\left(1-D(\mathbf{Z}\,|\,\mathbf{y})\right)^2\right], \tag{2}$$

where the discriminator $D(\cdot)$ outputs the probability that the features come from the generated distribution $p(\mathbf{Z}')$. This adversarial loss function indicates

that the more likely the discriminator fails to correctly discriminate the true features $\mathbf{Z}$ from the feature distribution $p(\mathbf{Z})$, the closer the feature distribution $p(\mathbf{Z})$ is to the reference distribution $p(\mathbf{Z}')$.

Distribution generator $G(\cdot)$: The distribution generator $G(\cdot)$ is mainly used to generate the dynamic reference distribution $p(\mathbf{Z}')$ and the corresponding generated features $\mathbf{Z}'$ (also known as spurious features). It inputs random noise $\boldsymbol{\varepsilon}$ and one-hot encoded true label $\mathbf{y}$, and it outputs false feature $\mathbf{Z}'$ with the same dimension as the true feature $\mathbf{Z}$. In the training process, given a trained discriminator $D(\cdot)$ with fixed parameters, the $\mathcal{L}_{adv\_g}$ is primarily employed to adjust the parameters of the distribution generator $G(\cdot)$, and the specific distribution generator adversarial loss function $\mathcal{L}_{adv\_g}$ is expressed as follows:

$$\mathcal{L}_{adv\_g} = -\mathbb{E}_{\mathbf{Z}'\sim p(\mathbf{Z}')}\left[\left(1-D(\mathbf{Z}'\,|\,\mathbf{y})\right)^2\right]. \tag{3}$$

The above three adversarial loss functions constitute the loss function of FedALN, which indirectly reduces the differences in feature distributions between clients by introducing reference distributions. Training federated prediction models on this basis will be beneficial to avoid the impact of inter-client domain distribution differences on model performance, the FedADG mechanism [16] is proposed based on FedALN. This mechanism sets the classifier $C(\cdot)$ for constructing a complete prediction model. It inputs real data features $\mathbf{Z}$, it outputs prediction labels $C(\mathbf{Z})$, and uses the traditional cross-entropy loss as the loss function of the classifier for model training. The specific FedADG objective function is expressed as follows:

$$\mathcal{L}_{\text{FedADG}} = \mathcal{L}_{adv\_d} + \mathcal{L}_{adv\_g} + \lambda_0\mathcal{L}_{adv\_f} + \lambda_1\mathcal{L}_{err}, \tag{4}$$

where $\mathcal{L}_{err} = \frac{1}{N}\sum_{i=1}^{N} L\left(C\left(F\left(X_i\right)\right), Y_i\right)$ denotes the classifier loss function, $L(\cdot,\cdot)$ denotes the cross-entropy loss, $0<\lambda_0,\lambda_1<1$ and $\lambda_0+\lambda_1=1$.

3. Stable Federated Adversarial Domain Generalization (Stable-FedADG)

In this section, we will further improve on FedADG and propose a Stable-FedADG method, which combines FedALN and stable prediction loss function to further improve the generalization performance of the model through intra-client stable feature learning and inter-client domain adversarial alignment. This section describes the model and algorithm of Stable-FedADG in detail.

3.1. Stable-FedADG Model

Now consider that there are K clients, each of which collects its local data $(\mathbf{X}_k, \mathbf{Y}_k)$ independently, then the objective function of the above FedADG can be reformulated into the following form:

$$\mathcal{L}_{\text{FedADG}} = \sum_{k=1}^{K}\frac{N_k}{N}\left(\mathcal{L}_{adv\_d_k} + \mathcal{L}_{adv\_g_k} + \lambda_0\mathcal{L}_{adv\_f_k} + \lambda_1\mathcal{L}_{err_k}\right), \tag{5}$$

where $\mathcal{L}_{adv\_d_k}$, $\mathcal{L}_{adv\_g_k}$, $\mathcal{L}_{adv\_f_k}$ denote the corresponding local ALNs adversarial losses, which can be obtained by using the client's local data, and $\mathcal{L}_{err_k} = \frac{1}{N_k}\sum_{i=1}^{N_k} L\left(C\left(F\left(X_{i,k}\right)\right), Y_{i,k}\right)$ denotes the local classifier loss function.

FedADG aligns the inter-client feature distributions based on FedALN, eliminating the impact of inter-client feature distribution differences on model performance, but this distribution alignment operation is performed on a client-by-client basis and does not take into account the domain bias and imbalance problems that may also exist within a single client. Since a single client may contain multiple source domain data and there usually is no significant domain labels for intra-client domain distribution alignment, we introduce the idea of SL [21] within a single client, which is used to extract stable features by removing spurious correlations between feature variables.

Specifically, in the k-th client, we use the stable predictive loss function $\mathcal{L}_{S_k}$ instead of the traditional cross-entropy loss $\mathcal{L}_{err_k}$ as the loss function of the local classifier $C(\cdot)$, and the specific expression of the stable predictive loss function $\mathcal{L}_{S_k}$ is as follows

$$\begin{aligned} \mathcal{L}_{S_k} &= \frac{1}{N_k}\sum_{i=1}^{N_k} w_{i,k}^{*} L\left(C\left(F\left(\mathbf{X}_{i,k}\right)\right), y_{i,k}\right), \\ \mathbf{w}_k^{*} &= \arg\min_{\mathbf{w}_k \in \Delta_{N_k}} \sum_{1\le m<h\le M_Z} \left\| \hat{\Sigma}_{\mathbf{Z}_{k,:,m}\mathbf{Z}_{k,:,h};\mathbf{w}_k} \right\|_F^2, \end{aligned} \tag{6}$$

where $\Delta_{N_k} = \left\{\mathbf{w}_k \in \mathbb{R}_+^{N_k} \mid \sum_{i=1}^{N_k} w_{i,k} = N_k\right\}$, $w_{i,k} \in \mathbf{w}_k$ denotes the i-th sample weight of the k-th client, $F(\cdot)$ is the feature extractor, $C(\cdot)$ is the classifier, $L(\cdot,\cdot)$ is the cross-entropy loss, $\mathbf{Z}_{k,:,m}$ denotes the m-th feature variable in the feature space of the k-th client, and similarly, $\mathbf{Z}_{k,:,h}$ denotes the h-th feature variable in the feature space of the k-th client, and $\hat{\Sigma}_{\mathbf{Z}_{k,:,m}\mathbf{Z}_{k,:,h};\mathbf{w}_k}$ denotes the weighted partial cross-covariance matrix of the feature variables $\mathbf{Z}_{k,:,m}$ and $\mathbf{Z}_{k,:,h}$, which is used to measure the independence between $\mathbf{Z}_{k,:,m}$ and $\mathbf{Z}_{k,:,h}$. The specific expression of $\hat{\Sigma}_{\mathbf{Z}_{k,:,m}\mathbf{Z}_{k,:,h};\mathbf{w}_k}$ is as follows:

$$\begin{aligned} \hat{\Sigma}_{\mathbf{Z}_{k,:,m}\mathbf{Z}_{k,:,h};\mathbf{w}_k} = \frac{1}{N_k}\sum_{i=1}^{N_k} \Bigg[&\left(w_{i,k}\mathbf{u}\left(Z_{k,i,m}\right) - \frac{1}{N_k}\sum_{j=1}^{N_k} w_{j,k}\mathbf{u}\left(Z_{k,j,m}\right)\right)^{\mathrm{T}} \\ &\times \left(w_{i,k}\mathbf{v}\left(Z_{k,j,h}\right) - \frac{1}{N_k}\sum_{j=1}^{N_k} w_{j,k}\mathbf{v}\left(Z_{k,j,h}\right)\right)\Bigg], \end{aligned} \tag{7}$$

where

$$\begin{aligned} \mathbf{u}(\cdot) &= \left(u_1(\cdot), u_2(\cdot), \cdots, u_{n_A}(\cdot)\right), u_j(\cdot) \in \mathcal{H}_{\mathrm{RFF}}, \forall j, \\ \mathbf{v}(\cdot) &= \left(v_1(\cdot), v_2(\cdot), \cdots, v_{n_B}(\cdot)\right), v_j(\cdot) \in \mathcal{H}_{\mathrm{RFF}}, \forall j. \end{aligned}$$

here, $Z_{k,i,m}$ denotes the i-th sample of the m-th feature variable from the k-th client. The vectors $\mathbf{u}$ and $\mathbf{v}$ are the Random Fourier Features mapping functions from the Random Fourier Feature function space $\mathcal{H}_{\mathrm{RFF}}$, which is employed to capture the nonlinear dependence among the complex features, and the

Random Fourier Feature function space $\mathcal{H}_{\text{RFF}}$ is expressed as follows:

$$\mathcal{H}_{\text{RFF}} = \left\{ h : x \to \sqrt{2}\cos\left(\omega x + \phi\right) \mid \omega \sim N\left(0,1\right), \phi \sim \text{Uniform}\left(0, 2\pi\right) \right\}. \tag{8}$$

The core of SL is to remove the dependency between features by sample re-weighting, *i.e.*, learning the sample weights $\mathbf{w}_k$ to make the weighted partial cross-covariance matrix $\hat{\Sigma}_{\mathbf{Z}_{k,:,m}\mathbf{Z}_{k,:,h};\mathbf{w}_k}$ converge to 0. And the Frobenius norm is used to construct the independence test statistic, *i.e.*, when $I_{\mathbf{Z}_{k,:,m}\mathbf{Z}_{k,:,h};\mathbf{w}_k} = \left\| \hat{\Sigma}_{\mathbf{Z}_{k,:,m}\mathbf{Z}_{k,:,h};\mathbf{w}_k} \right\|_F^2$ tends to 0, the feature variables $\mathbf{Z}_{k,:,m}$ and $\mathbf{Z}_{k,:,h}$ tend to be independent. Within a single client, for all feature variables, we learn the optimal $\mathbf{w}_k^*$ by equation (0) to minimize the linear and nonlinear dependencies of all features within the client, and learn the stable features to eliminate the impact of feature distribution bias of intra-client on the model performance.

In summary, our proposed Stable-FedADG approach consists of two main aspects. First, within a single client, we use the stable prediction loss function $\mathcal{L}_{S_k}$ as the loss function of the local classifier $C(\cdot)$ to learn stable features, and second, we base our method on FedALN to adversarially align inter-client domain distributions. The method simultaneously eliminates the effects caused by intra-client and inter-client domain distribution bias to learn domain-invariant features in the federated context and further improves the generalization performance of the model. The specific Stable-FedADG objective function is expressed as follows:

$$\mathcal{L}_{\text{Stable-FedADG}} = \sum_{k=1}^{K} \frac{N_k}{N} \left(\mathcal{L}_{adv\_d_k} + \mathcal{L}_{adv\_g_k} + \lambda_0 \mathcal{L}_{adv\_f_k} + \lambda_1 \mathcal{L}_{S_k} \right). \tag{9}$$

3.2. Stable-FedADG Algorithm

The detailed training process of the Stable-FedADG method is shown in **Algorithm 1**. The training process of Stable-FedADG consists of two main parts: server aggregation and client update.

Algorithm 1 Stable-FedADG

Input: Local dataset of each clients $\mathbf{X}_k$ with corresponding labels $\mathbf{Y}_k$, model parameters $\boldsymbol{\beta} = \{\boldsymbol{\beta}_f, \boldsymbol{\beta}_c, \boldsymbol{\beta}_g\}$ of $F(\cdot)$, $C(\cdot)$, and $G(\cdot)$, as well as the parameter $\boldsymbol{\beta}_d$ of the discriminator $D(\cdot)$.
Output: Feature extractor $F(\cdot)$ and Classifier $C(\cdot)$.

Server aggregation:
Initialize model parameters $\boldsymbol{\beta}$;
for round $t = 1, 2, \ldots, t_{max}$ **do**
  for each client $k = 1, 2, \ldots, K$ **in parallel do**
    $\boldsymbol{\beta}_k^{(t+1)} \leftarrow \text{ClientUpdate}(k, \boldsymbol{\beta}^{(t)})$
  end for
  $\boldsymbol{\beta}^{(t+1)} \leftarrow \sum_{k=1}^{K} \frac{N_k}{N} \boldsymbol{\beta}_k^{(t+1)}$
end for

ClientUpdate$(k, \boldsymbol{\beta})$: // *Execute on client k*
Receive model parameters $\boldsymbol{\beta} = \{\boldsymbol{\beta}_f, \boldsymbol{\beta}_c, \boldsymbol{\beta}_g\}$ from the server.
for epoch $i = 1, 2, \ldots, E_0$ **do**
  Sample a mini-batch $\mathbf{X}_b$ from $\mathbf{X}_k$;
  Compute sample weights $\mathbf{w}_k^*$ using (6) via a saving and reloading method;
  Update $\boldsymbol{\beta}_f$ and $\boldsymbol{\beta}_c$ on $\mathbf{X}_b$ to minimize $\mathcal{L}_{S_k}$.
end for
for epoch $j = 1, 2, \ldots, E_1$ **do**
  Sample a mini-batch $\mathbf{X}_b$ from $\mathbf{X}_k$;
  Compute sample weights $\mathbf{w}_k^*$ using (6) with a saving and reloading method;
  Update $\boldsymbol{\beta}_f$ and $\boldsymbol{\beta}_c$ on $\mathbf{X}_b$ to minimize $\lambda_0 \mathcal{L}_{adv\_f_k} + \lambda_1 \mathcal{L}_{S_k}$;
  Generate a mini-batch random numbers $\mathbf{X}_\varepsilon$ using a random number generator;
  Update $\boldsymbol{\beta}_d$ using $\mathbf{X}_b$ and $\mathbf{X}_\varepsilon$ to minimize $\mathcal{L}_{adv\_d}$;
  Update $\boldsymbol{\beta}_g$ using $\mathbf{X}_\varepsilon$ and $\mathbf{Y}_b$ to minimize $\mathcal{L}_{adv\_g}$.
end for
Upload the updated local model parameters $\boldsymbol{\beta}$ to the server.

Server aggregation: The server is mainly used to aggregate the model parameters uploaded by each client and update the global model. Once the global model is updated, it is sent back to each client for the next round of model training. Each client is required to upload only three network components: the feature extractor $F(\cdot)$, the classifier $C(\cdot)$ and the distribution generator $G(\cdot)$, along with their respective model parameters $\boldsymbol{\beta} = \{\boldsymbol{\beta}_f, \boldsymbol{\beta}_c, \boldsymbol{\beta}_g\}$. Specifically, before training, the server first initializes the model parameters $\boldsymbol{\beta}$ and passes them to each client. During the training process, the server receives the model parameters from each client and updates the global model with the received client parameters, and then passes the updated global model to each client, and after several server-client iterations, the trained global model can be obtained, in which the feature extractor $F(\cdot)$ and classifier $C(\cdot)$ constitute the prediction model that can be used for prediction of unknown clients.

Client update: The client is mainly responsible for training local models using local data. In the training process, the client first receives the global model from the server, and then utilizes the local data and the local discriminator $D(\cdot)$ to train the local model $\boldsymbol{\beta}_k$. Specifically, the classifier $C(\cdot)$ and the feature extractor $F(\cdot)$ are trained using the stable prediction loss function $\mathcal{L}_{S_k}$, and the saving and reloading method proposed in [21] is used to learn the sample weights during the training process. Then aligning the feature distribution to the reference distribution, the loss $\lambda_0 \mathcal{L}_{adv\_f_k} + \lambda_1 \mathcal{L}_{S_k}$ is minimized by updating both the feature extractor $F(\cdot)$ and the classifier $C(\cdot)$. Meanwhile, the losses $\mathcal{L}_{adv\_d_k}$ and $\mathcal{L}_{adv\_g_k}$ are minimized by updating the discriminator $D(\cdot)$ and the distribution generator $G(\cdot)$, respectively. At the end of local training, the client sends the local model parameters $\boldsymbol{\beta}_k = \{\boldsymbol{\beta}_{f_k}, \boldsymbol{\beta}_{c_k}, \boldsymbol{\beta}_{g_k}\}$ to the server for model aggregation.

4. Experiments

4.1. Experimental Setup

In this section, we conducted experiments on PACS and OfficeHome datasets, which are both widely used benchmark datasets in the field of DG. The PACS dataset contains 9991 image data from four different domains, the four domains are art paintings, cartoons, photographs and sketches, and each domain contains 7 categories. And the OfficeHome dataset also contains images from four domains, Art, Clipart, Product and Real, each containing 65 categories and a total of 15,500 images.

In addition, to further validate the model performance of Stable-FedADG in various federated scenarios, we set up three types of experiments, namely, the leave-one-domain-out setting (LODO), the unbalanced setting and the mixed setting. In all settings, one domain from the dataset is chosen as the target domain for model performance validation, while the remaining domains serve as source domains for model training. The detailed configurations are outlined as follows.

LODO: LODO setting is a widely used validation strategy in FDG research, which treats each domain as a client, *i.e.*, each client contains only data from the

same domain. This setting simulates the scenario of small intra-client data distribution differences and large inter-client data distribution differences.

Unbalanced: Unbalanced setting proposed by [21] is employed to simulate the federated scenario with large intra-client distribution differences and small inter-client distribution differences. Here, each client holds data from all source domains, with different source domains having distinct proportions, while the same source domain in different clients have consistent proportion shares.

Mixed: Mixed setting is employed to simulate a more complex federated scenario with domain bias both within and between clients, we configured each client to hold only 2 source domains data, with one domain being dominant. Moreover, different clients contain distinct source domains.

Table 1. Initial learning rate for different datasets.

PACS			
	LODO	unbalanced	mixed
lr_f	0.001	0.001	0.001
lr_g	0.05	0.005	0.05
lr_d	0.05	0.005	0.05
lr_w	0.1	0.1	0.1
OfficeHome			
	LODO	unbalanced	mixed
lr_f	0.001	0.001	0.001
lrg	0.005	0.005	0.05
lr_d	0.005	0.005	0.05
lr_w	0.1	0.1	0.1

Each of the three settings above simulates a different federated scenario. Following [16], 70% of each client's data is randomly chosen as the training set and 30% as the validation set. The network architecture in this paper is the same as that in FedADG [16]. For the PACS dataset, the feature extractor is constructed using ImageNet pretrained AlexNet and the OfficeHome dataset used ResNet50, and both without its last layer. Additionally, the classifier, distribution generator, and discriminator are all composed of two fully-connected layers. The output layer of the discriminator is configured to have a size of 1. In all experiments, the number of local iterations is set as $E_0 = 3$ and $E_1 = 5$ with a batch size of 64. The communication occurs 30 times. The local model is updated via the SGD method, with $\lambda_0 = 0.85$ and $\lambda_1 = 0.15$. The specific initial learning rates are shown in **Table 1**, where lr_f represents the learning rate of the feature extractor and the classifier, lr_g and lr_d are for the distribution generator and discriminator respectively, and lr_w is for learning sample weights within the client.

4.2. Experimental Results

In this paper, we present the experimental results of different algorithms in three settings, including the classic FL algorithms FedAvg [4], FedProx [7] and SCAFFOLD [8], and the existing FDG algorithms FedIIR [22], FedSR [23] and FedADG [16]. Among them, FedADG is a method that improves the generalization performance of the model only by aligning the feature distributions among clients.

Table 2 and Table 3 display the test accuracies of different models on the PACS dataset and OfficeHome dataset, respectively. Each result is the average of three repeats. Analysing Table 2 and Table 3, it can be seen that the test accuracies of Stable-FedADG and FedADG outperform FedAvg significantly in both the LODO setting and the mixed setting, indicating that when inter-client feature distributions vary greatly, adversarial aligning inter-client feature distributions are conducive to the learning of domain-invariant features and boost model generalization. Meanwhile, Stable-FedADG outperforms FedAvg and FedADG in all three settings, indicating that intra-client stable feature learning can further improve the FL global model's generalization. Besides, Stable-FedADG also outperforms other FL and FDG algorithms in all three settings, indicating that considering simultaneously intra-client and inter-client domain distribution differences facilitates further improving the generalization performance of the model in various federated scenarios.

Table 2. Experimental results of different algorithms on PACS dataset.

		Art	Cartoon	Photo	Sketch	Avg
*LODO	FedAvg	62.52	59.09	91.98	62.85	69.11
	FedProx	63.04	59.43	91.26	62.91	69.16
	SCAFFOLD	63.15	59.45	92.09	62.97	69.42
	FedIIR	63.64	59.52	92.85	64.62	70.16
	FedSR	61.58	58.96	90.83	62.52	68.47
	FedADG	64.84	59.81	93.17	68.28	71.53
	Stable-FedADG	**67.63**	**61.05**	**93.65**	**70.81**	**73.29**
*Unbalanced	FedAvg	68.95	63.44	88.26	61.95	70.65
	FedProx	68.71	62.81	88.23	61.42	70.29
	SCAFFOLD	69.38	62.67	88.38	62.64	70.77
	FedIIR	70.09	63.53	89.13	**67.46**	72.55
	FedSR	71.39	61.82	86.22	63.73	70.79
	FedADG	70.02	63.78	88.82	64.57	71.8
	Stable-FedADG	**73.73**	**64.29**	**91.87**	65.49	**73.85**
*Mixed	FedAvg	65.58	61.73	91.42	64.88	70.9
	FedProx	67.86	60.58	92.56	65.49	71.62
	SCAFFOLD	67.97	60.19	92.75	66.15	71.77

Continued

	FedIIR	68.89	61.69	92.38	**67.12**	72.52
	FedSR	65.43	60.29	90.49	62.95	69.79
	FedADG	68.05	63.78	92.92	63.04	71.95
	Stable-FedADG	**70.19**	**63.94**	**92.97**	66.58	**73.42**

Table 3. Experimental results of different algorithms on OfficeHome dataset.

		Art	Clipart	Product	Real	Avg
	FedAvg	65.42	45.36	75.74	78.93	66.36
	FedProx	65.59	46.67	76.32	78.74	66.83
	SCAFFOLD	65.34	47.09	75.89	78.97	66.82
*LODO	FedIIR	66.87	46.71	75.94	79.38	67.23
	FedSR	65.05	45.98	74.75	79.06	66.21
	FedADG	67.16	46.6	76.55	79.21	67.38
	Stable-FedADG	**68.23**	**49.42**	**76.59**	**79.52**	**68.44**
	FedAvg	64.57	45.52	76.62	77.64	66.09
	FedProx	63.89	45.56	76.79	77.61	65.96
	SCAFFOLD	64.52	45.69	77.38	77.53	66.28
*Unbalanced	FedIIR	64.68	45.72	77.79	77.68	66.47
	FedSR	63.92	44.78	76.95	77.23	65.72
	FedADG	64.32	45.04	77.61	77.29	66.07
	Stable-FedADG	**65.51**	**46.37**	**77.87**	**77.79**	**66.89**
	FedAvg	64.94	47.93	75.63	78.91	66.85
	FedProx	65.29	47.03	75.95	79.02	66.82
	SCAFFOLD	65.98	47.83	75.71	79.32	67.21
*Mixed	FedIIR	66.74	47.51	76.35	79.52	67.53
	FedSR	66.17	46.99	75.59	78.97	66.93
	FedADG	66.96	47.19	**77.09**	79.64	67.72
	Stable-FedADG	**67.86**	**48.41**	76.89	**80.1**	**68.32**

5. Conclusion

In this paper, we give a generalized federated domain generalization algorithm for agnostic distribution bias called Stable-FedADG. This algorithm focuses on both intra-client and inter-client multidomain distributions, and eliminates the intra-client feature bias by intra-client stable feature learning, meanwhile, the algorithm incorporates FedALN to adversarially align the inter-client feature distribution. The experimental results show that the Stable-FedADG algorithm outperforms both the traditional FL algorithm and the FedADG algorithm in different federated scenarios, and can further improve the generalization performance of the

model. In addition, the proposed algorithm still has some limitations, firstly, the proposed algorithm is mainly for feature drift, *i.e.*, each client is required to contain the same label categories, and future work can be developed on the basis of label drift. Second, the proposed algorithm is an FDG algorithm that requires a centralized server, and future work considers investigating decentralized FDG algorithms.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 62166013), the Natural Science Foundation of Guangxi (No. 2022GXNSFAA035499) and the Foundation of Guilin University of Technology (No. GLUTQD2007029).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Xu, H., Seng, K.P., Ang, L.M. and Smith, J. (2024) Decentralized and Distributed Learning for AIoT: A Comprehensive Review, Emerging Challenges, and Opportunities. *IEEE Access*, **12**, 101016-101052. https://doi.org/10.1109/access.2024.3422211

[2] Li, Y., Wang, X., Zeng, R., Yang, M., Li, K., Huang, M., *et al.* (2023) VARF: An Incentive Mechanism of Cross-Silo Federated Learning in Mec. *IEEE Internet of Things Journal*, **10**, 15115-15132. https://doi.org/10.1109/jiot.2023.3264611

[3] Liu, B., Lv, N., Guo, Y. and Li, Y. (2024) Recent Advances on Federated Learning: A Systematic Survey. *Neurocomputing*, **597**, Article 128019. https://doi.org/10.1016/j.neucom.2024.128019

[4] McMahan, B., Moore, E., Ramage, D., Hampson, S. and y Arcas, B.A. (2017) Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings of the* 20*th International Conference on Artificial Intelligence and Statistics* (*AISTATS*), Fort Lauderdale, 20-22 April 2017, 1273-1282. https://api.semanticscholar.org/CorpusID:14955348

[5] Ye, M., Fang, X., Du, B., Yuen, P.C. and Tao, D. (2023) Heterogeneous Federated Learning: State-of-the-Art and Research Challenges. *ACM Computing Surveys*, **56**, 1-44. https://doi.org/10.1145/3625558

[6] Lu, Z., Pan, H., Dai, Y., Si, X. and Zhang, Y. (2024) Federated Learning with Non-IID Data: A Survey. *IEEE Internet of Things Journal*, **11**, 19188-19209. https://doi.org/10.1109/jiot.2024.3376548

[7] Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A. and Smith, V. (2020) Federated Optimization in Heterogeneous Networks. *Proceedings of the* 3*rd MLSys Conference*, Austin, 2-4 March 2020, 429-450. https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf

[8] Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S. and Suresh, A.T. (2020) Scaffold: Stochastic Controlled Averaging for Federated Learning. *Proceedings of the* 37*th International Conference on Machine Learning*, Virtual Event, 13-18 July 2020, 5132-5143. http://proceedings.mlr.press/v119/karimireddy20a.html

[9] Tan, A.Z., Yu, H., Cui, L. and Yang, Q. (2023) Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, **34**, 9587-9603. https://doi.org/10.1109/tnnls.2022.3160699

[10] Lin, S., Yang, G. and Zhang, J. (2020) A Collaborative Learning Framework via Federated Meta-Learning. 2020 *IEEE* 40*th International Conference on Distributed Computing Systems* (*ICDCS*), Singapore, 29 November-1 December 2020, 289-299. https://doi.org/10.1109/icdcs47774.2020.00032

[11] Li, Y., Wang, X., Zeng, R., Donta, P.K., Murturi, I., Huang, M. and Dustdar, S. (2024) Federated Domain Generalization: A Survey. arXiv: 2306.01334. https://arxiv.org/abs/2306.01334

[12] Li, X., Gu, Y., Dvornek, N., Staib, L.H., Ventola, P. and Duncan, J.S. (2020) Multi-Site fMRI Analysis Using Privacy-Preserving Federated Learning and Domain Adaptation: ABIDE Results. *Medical Image Analysis*, **65**, Article 101765. https://doi.org/10.1016/j.media.2020.101765

[13] Peng, X., Huang, Z., Zhu, Y. and Saenko, K. (2020) Federated Adversarial Domain Adaptation. https://openreview.net/forum?id=HJezF3VYPB

[14] Wu, G. and Gong, S. (2021) Collaborative Optimization and Aggregation for Decentralized Domain Generalization and Adaptation. 2021 *IEEE/CVF International Conference on Computer Vision* (*ICCV*), Montreal, 10-17 October 2021, 6464-6473. https://doi.org/10.1109/iccv48922.2021.00642

[15] Liu, Q., Chen, C., Qin, J., Dou, Q. and Heng, P. (2021) FedDG: Federated Domain Generalization on Medical Image Segmentation via Episodic Learning in Continuous Frequency Space. 2021 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Nashville, 20-25 June 2021, 1013-1023. https://doi.org/10.1109/cvpr46437.2021.00107

[16] Zhang, L., Lei, X., Shi, Y., Huang, H. and Chen, C. (2023) Federated Learning for IoT Devices with Domain Generalization. *IEEE Internet of Things Journal*, **10**, 9622-9633. https://doi.org/10.1109/jiot.2023.3234977

[17] Shen, Z., Cui, P., Kuang, K., Li, B. and Chen, P. (2018) Causally Regularized Learning with Agnostic Data Selection Bias. *Proceedings of the* 26*th ACM international conference on Multimedia*, Seoul, 22-26 October 2018, 411-419. https://doi.org/10.1145/3240508.3240577

[18] Kuang, K., Xiong, R., Cui, P., Athey, S. and Li, B. (2020) Stable Prediction with Model Misspecification and Agnostic Distribution Shift. *Proceedings of the AAAI Conference on Artificial Intelligence*, **34**, 4485-4492. https://doi.org/10.1609/aaai.v34i04.5876

[19] Cui, P. and Athey, S. (2022) Stable Learning Establishes Some Common Ground between Causal Inference and Machine Learning. *Nature Machine Intelligence*, **4**, 110-115. https://doi.org/10.1038/s42256-022-00445-z

[20] Xu, R., Cui, P., Shen, Z., Zhang, X. and Zhang, T. (2021) Why Stable Learning Works? A Theory of Covariate Shift Generalization. arXiv: 2111.02355. https://arxiv.org/abs/2111.02355v1

[21] Zhang, X., Cui, P., Xu, R., Zhou, L., He, Y. and Shen, Z. (2021) Deep Stable Learning for Out-of-Distribution Generalization. 2021 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Nashville, 20-25 June 2021, 5368-5378. https://doi.org/10.1109/cvpr46437.2021.00533

[22] Guo, Y., Guo, K., Cao, X., Wu, T. and Chang, Y. (2023) Out-of-Distribution Generalization of Federated Learning via Implicit Invariant Relationships. *Proceedings of*

the 40*th International Conference on Machine Learning*, Honolulu, 23-29 July 2023, 11905-11933. https://dl.acm.org/doi/10.5555/3618408.3618886

[23] Nguyen, A.T., Torr, P. and Lim, S.N. (2022) FedSR: A Simple and Effective Domain Generalization Method for Federated Learning. *Proceedings of the* 36*th International Conference on Neural Information Processing System*, New Orleans, 28 November-9 December 2022, 38831-38843. https://dl.acm.org/doi/10.5555/3600270.3603084