IEEE ComSoc IEEE Communications Society | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 17 October 2023; revised 4 February 2024 and 6 April 2024; accepted 22 April 2024.
Date of publication 25 April 2024; date of current version 1 May 2024.

The associate editor coordinating the review of this article and approving it for publication was M. Chen.

Digital Object Identifier 10.1109/TMLCN.2024.3393892

Federated Analytics With Data Augmentation in Domain Generalization Toward Future Networks

XUNZHENG ZHANG (Graduate Student Member, IEEE), JUAN MARCELO PARRA-ULLAURI (Member, IEEE), SHADI MOAZZENI (Member, IEEE), XENOFON VASILAKOS (Member, IEEE), REZA NEJABATI (Senior Member, IEEE), AND DIMITRA SIMEONIDOU (Fellow, IEEE)

High Performance Networks Group, Smart Internet Laboratory, School of Electrical, Electronic and Mechanical Engineering, Faculty of Engineering, University of Bristol, BS8 1QU Bristol, U.K.

CORRESPONDING AUTHOR: X. ZHANG (xunzheng.zhang@bristol.ac.uk)

This work was supported by the Future Open Network Research Challenge (FONRC) call, the project Realising Enabling Architectures and Solutions for Open Networks (REASON), U.K.

ABSTRACT Federated Domain Generalization (FDG) aims to train a global model that generalizes well to new clients in a privacy-conscious manner, even when domain shifts are encountered. The increasing concerns of knowledge generalization and data privacy also challenge the traditional gather-and-analyze paradigm in networks. Recent investigations mainly focus on aggregation optimization and domain-invariant representations. However, without directly considering the data augmentation and leveraging the knowledge among existing domains, the domain-only data cannot guarantee the generalization ability of the FDG model when testing on the unseen domain. To overcome the problem, this paper proposes a distributed data augmentation method which combines Generative Adversarial Networks (GANs) and Federated Analytics (FA) to enhance the generalization ability of the trained FDG model, called FA-FDG. First, FA-FDG integrates GAN data generators from each Federated Learning (FL) client. Second, an evaluation index called generalization ability of domain (GAD) is proposed in the FA server. Then, the targeted data augmentation is implemented in each FL client with the GAD index and the integrated data generators. Extensive experiments on several data sets have shown the effectiveness of FA-FDG. Specifically, the accuracy of the FDG model improves up to 5.12% in classification problems, and the R-squared index of the FDG model advances up to 0.22 in the regression problem.

INDEX TERMS Federated domain generalization, data augmentation, federated analytics, adversarial learning, future networks.

I. INTRODUCTION

FEDERATED Learning (FL) has lately arisen as a privacy-preserving paradigm for distributed learning while the data is deployed on different clients [1], [2]. Supported by edge computing and raw data protection techniques, the FL steps forward as the federated paradigm to deal with data-oriented missions collaboratively without sharing the raw data. In this case, federated analytics (FA) is proposed to reuse the FL infrastructure but without the learning part [3]. The novel FA paradigm no longer needs raw data in the centre aggregation server. Instead, the selected clients do the following steps before FL: 1) receive the data statistical computing model from the server; 2) calculate insight-oriented tasks based on its local data; 3) upload the abstracted results back to the FA server; and 4) interact with the FL server to finally improve the quality of FL models, as shown in green steps of Figure 1. The FA server aggregates the results and gives data discoveries to the FL server. The FL server will deploy the FA results and the original learning model for FL to each client. By doing this, we expect that the FL model eventually achieves a comparable performance just like the traditional data-centralized model after multiple iterations

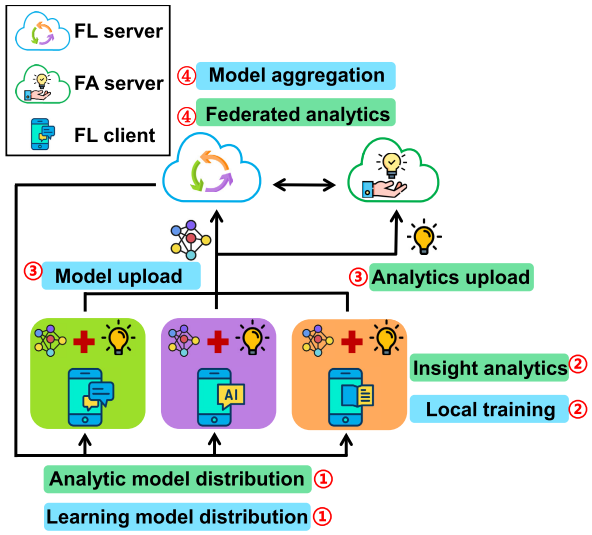


FIGURE 1. Workflow of FA assisted FL.

(blue steps in Figure 1). FL focuses on training neural networks, while FA targets operating data analytics tasks (like most often recognized song [3], statistics computing [4], local video analytics [5], etc) to improve the performance of FL.

In most seminal FL studies like [2], [6], [7], and [8], the test data set is a subset of the original data set. However, a more practical problem is how to train a model on clients with different data distributions so that this model performs well on unseen data. Especially for future network orchestrators, how to improve the model's generalization ability for unknown network resource configuration is significant. The above-mentioned problem can be addressed by the Domain Generalization (DG) technique [9], [10], [11], also with FL as Federated Domain Generalization (FDG) [12], [13]. Most DG approaches are designed in a centralized training manner, which means the server can access all domain data to train a Machine Learning (ML) model while trying to make the trained model generalize well on unseen domains. However, considering data privacy regulations like [14], the DG technique could combine with the FL as FDG.

Due to the distributed attribute, each domain is treated as a FL client in FDG. After model aggregation in FL, the global model will be tested on the unseen domain. Here 'domains' represent different contexts or sets of data with labels [9], [11], while "data distribution" describes how the individual data points are spread within a dataset. Domains encompass the broader context of data application, and data distribution is the characteristic of data within a specific domain. For instance, from the network perspective, each virtual network function (VNF) client may have different purposes with different resource configurations [15]. Each type of VNF collects individual network data as a domain. The domain formed by one FL client data set is called a source domain, there are multiple source domains in FDG. FDG can be seen as a new branch of federated transfer learning (FTL) since there exists knowledge migration among FL participants. The challenge of FDG lies in the domain shift both among the training FL clients and from training to testing clients [12].

To solve the domain shift problem among training clients, the best solution is to make each client have the data from the rest of the clients in FL. The generalization performance of the FL model also relies on the quantity and diversity of the training data [9] in each FL client. However, raw data cannot be directly exchanged by clients due to the privacy problem. Inspired by FL which moves the model rather than the raw data [16], in this paper, the motivation is training a GAN in each FL client, then the server collects well-trained generators and deploys this set of generators to every client in FDG. GANs have been proven to play an important role in the fields of privacy data augmentation [16], [17], [18]. Using GANs, on the one hand, FL clients can generate their synthetic data to increase the data quantity. On the other hand, by gathering well-trained generators from FL clients to the server which then deploys this set of generators to each FL participant, every FL user can generate synthetic data from other domains to improve its data diversity. These synthetic data with latent data distribution also include insights from generative AI models which may further enhance the generalization ability. In this case, the distribution difference and limited data among FL clients can be mitigated. Then the trained FL model could have better performance when facing the unseen test domain.

However, before generating data, generators cannot perform data augmentation without purpose at each FL client. Too much data will bring privacy issues, excessive training time and memory usage. On the contrary, it is necessary to get insight into which domains in FDG may contribute more to the generalization performance of the final FL model. Then do the targeted data augmentation in FL clients. FA can get this insight from the generators. Intuitively, the data set with higher variance [19] can have better generalization ability because the model can learn more spread based on this data set, which may contribute more when the test domains are unseen. In this case, after gathering the generators in the server, we give a FA algorithm based on a small amount of generated data from different domains to get an indicator called the generalization ability of domain (GAD). GAD denotes the average KL divergence [20] by using one domain to represent other domains. This insight ranks domains about which domain may act well on the unseen test domain, under the federated manner. The GAD then be deployed with the set of generators to each FL client and guide the data augmentation. From the results of FA, before FL, the generator from domains that may have better generalization ability will produce more data to give better FDG results.

In this paper, first, a novel distributed data augmentation method is proposed, which integrates GANs data generators from each FL client, and then deploys this set of generators to each FL user to achieve data augmentation for the FDG. Secondly, before the data augmentation in each FL client, an evaluation index called GAD is proposed with FA. This index involves targeted data augmentation from

data generators. GAD also avoids the decline of FDG model accuracy caused by uniformly augmenting the data. We call the proposed method as FA-FDG. Finally, the proposed FA-FDG is tested through extensive datasets and experiments. The aggregated FL model generalizes better with our methods than the traditional FL method without FA and data augmentation. The classification accuracy improves, and the regression error decreases when the aggregated FL model is tested on unseen domains.

A. CONTRIBUTION

The primary contributions of this article are given as follows.

1) MODELING

To achieve targeted data augmentation in FDG, this work first gives the FA-FDG architecture which includes 1) generators ensemble from GANs and GAD estimation in the FA server; 2) data augmentation in each FL client with the GAD result; 3) FL and testing the global model on the unseen domain.

2) ALGORITHMS

1) A Generator Ensemble Algorithm. This algorithm collects the well-trained GAN generators and deploys them to each FL client. In this way, the skewness and heterogeneity of FL clients can be reduced through the synthetic data from other domains.

2) A GAD Estimation Algorithm. In the FA server, the GAD insight ranks domains about which domain may act well on the unseen test domain. The GAD will guide the GANs data augmentation and make the generator from the domain with higher generalization ability produce more data in FL clients, thus improving the generalization performance of the FL global model.

3) A Data Augmentation Algorithm for FDG. After FA, the FL server will package together the original FL training model, the GAD result and the ensemble of generators (which is received from the FA server), then send the above three items to each client for FL. Each local user will enhance the domain data by the received generators under the guide of the GAD list. This distributed data augmentation method with generative AI not only improves the data quantity but also increases the data diversity in FL clients. This allows for privacy-preserving data sharing without directly exchanging raw data, leading to more robust and balanced FL models. When the test domain is unseen, the FL model trained with GANs augmented data has better performance.

3) SIMULATION EVALUATION

The proposed FA-FDG is evaluated on classification and regression problems considering both image data and tabular network data. FA-FDG realizes obtaining insights from different domains and performing targeted data augmentation. The generated data enriches the domain data, and improves the generalization accuracy of the trained FL model by up to 5.12%, compared with benchmarks from centralized [25], [26] to federated [2], [6], [27].

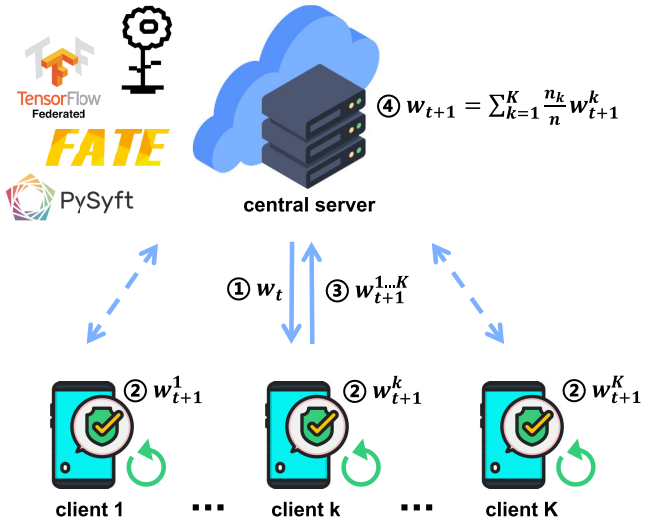


FIGURE 2. The FL framework with state-of-art platforms like Flower [21], FATE [22], TensorFlow Federated [23] and PyTorch PySyft [24].

4) TEST BED DEPLOYMENT

The proposed algorithms are deployed on the Kubernetes (K8s) [28]. We use the Link Layer Secure connectivity for Micro-service platforms (L2S-M) operator for privacy-preserving in FL. The network isolation creates secure resource sharing. The authentication and data packet encryption at the network layer level protects the in-transit data.

B. PAPER STRUCTURE

The rest of the paper is organized as follows. Related work is presented in Section II. The FA-FDG system is introduced in Section III. The problem formulation and algorithms are detailed in Section IV. Then, we give our test bed details in Section V, followed by experiments and analysis in Section VI. Finally, Section VII gives the conclusion and future directions of this work.

II. RELATED WORK

A. FEDERATED LEARNING AND ANALYTICS

FL and FA refer to a distributed approach in which multiple clients collaborate to solve data-oriented issues with only the parameters sharing. In FL, each iteration has four steps (Figure 2). To improve the performance of FL, some model aggregation methods [29] are given like federated average (Fed-Avg) [2], q-FedAvg [7] and FedProx [6]. Recently, FL has been considered an essential part of the next generation (6G) of networks [30], [31]. In FA [3], [5], [32], non-training data science is exploited. As a privacy-preserving framework, FA calculates or derives data analytics to extract insights from isolated entities. Wang et al. [33] use Hoeffding's inequality to estimate the federated clients' skewness. An interactive heavy hitters discovery algorithm is

proposed in [34] with central differential privacy. Zhang et al. [35] analyzed useful global features before FL, thus reducing the model training cost. Wang et al. [36] quantified the class distribution heterogeneity and formulate the client selection problem. The work [37] use FA to conduct statistical data analysis on various edge nodes towards 6G networks.

In future networks, the test domain of a FL model may not always be visible, which requires the model to have generalization performance on unseen domains. However, works like [2] do not guarantee convergence when the data in FL clients is highly heterogeneous. FA methods such as heavy hitters discovery [34] and skewness analytics [36], [37] only estimate frequent items and data heterogeneity, without considering how to reduce the clients' skewness and sensitive information through FA and data augmentation. If the user skewness is large and each FL client has a small number of samples, the trained FL model may have poor generalization ability to the unseen test domain. Under this condition, after finding some FL clients have skewness data [36], [37] by FA, GAN is particularly useful in this scenario because GANs can generate targeted synthetic data with guidance from FA in some FL clients, these new data can be combined with real data to augment datasets and reduce heterogeneity as well as skewness. GANs can be also employed to generate synthetic representations of data that preserve statistical properties but do not reveal sensitive information about individual data points like most frequent items in [34]. This allows for privacy-preserving data sharing without directly exchanging raw data, leading to more robust and balanced FL models. When the test domain is unseen, the FL model trained with GANs augmented data could have better performance. Besides, instead of training a GAN model in a federated way with high cost like [38] and [39], we directly send the trained GAN generator models from FL clients to the FA server.

B. FEDERATED DOMAIN GENERALIZATION

DG tries to learn a model from one or several different but related domains, and this model can generalize well on unseen test domains [9], [10], [11]. Considering data privacy and distributed ML methods, open issues in [9] and [10] both mentioned FDG. FDG [40], [41], [42], [43] realize the combination of federated paradigm and DG, the global model aggregates the parameters from different clients. Specifically, Chen et al. [40] propose a cross-client style transfer algorithm and a shared image style bank is introduced. Bai et al. [41] investigate the current DG algorithms adapted to FL, the authors give a standardized definition of client domain heterogeneity. In the article [42], a boundary-oriented episodic learning paradigm is designed to expose local learning to domain distribution shifts. In [43], GAN is used and the distribution generator is shared among clients. The work in [44] uses GANs to enrich attack data. A summary of FDG works is presented in [12].

However, less work considers using GANs and FA to realize distributed data augmentation for FDG. If each client uploads too many model parameters (e.g., the feature extractor, classifier, and distribution generator in [43]) for aggregation, it will lead the communication overhead in FL. If only one GAN is trained among domains, the high computation complexity may lead to model performance degradation [38], [39]. In this case, which domains will have more generalization performance among other domains should be considered by FA. Then, GAN is used to enhance the data of domains with more generalization ability. To the best of our knowledge, this work represents the first attempt using FA results to guide distributed data augmentation by GANs from generative AI, exploring the FDG in both regression and classification problems.

TABLE 1. Notations and definitions.

Notation	Definition
S^j, S^i	source domain j, source domain i
E_0	local GAN training epoch
E_1	local model training epoch
E_2	rounds of a complete FL
G_j, G_i	generator of domain j , generator of domain i
D_j	discriminator of domain j
$\mathbf{X}_{S^i}$	data generated by G_i of source domain S^i
$\mathbf{P}_{S^i}$	a set of basis of the subspace ($\mathbf{X}_{S^i}$ after PCA)
d	dimensionality of the subspace
d^*	optimal dimensionality of the subspaces
β_d	principal angles between two d-dimensional subspaces
$\mathbf{s}_d^i$	principal vectors from $\mathbf{P}_{S^i}$
$\boldsymbol{z}$	white noise
$\mathbf{X}^{(i)}$	data of domain i for Federated Learning
v_j	data augmentation coefficient
N_{S^i}	the number of samples in source domain S^i

III. FA-FDG ARCHITECTURE DESIGN

A. GENERATORS ENSEMBLE IN THE FA SERVER

Here, we first introduce the FA-FDG architecture in Figure 3. For ease of reading, the frequently used notations are summarized in Table 1. In this paper, each client in FL represents one domain. Suppose that there are S source domains $S = \{S^j\}_{j=1}^{S}$. First, we do the FA steps from Step 1 to Step 3 in Figure 3, to find the GAD guidance before distributed data augmentation. Instead of directly sharing the raw data, in Step 1, each domain will train a GAN model based on the local data. E_0 denotes the local GAN epoch. E_0 meets the requirement that the generators from GANs can be well-trained and generate good enough data samples. In each domain, the data generator can be obtained from the local GAN. The produced data from the generator looks similar to the original raw data, effectively reflecting some attributes of the local domain.

Then in Step 2, the FA server collects the well-trained generators from each domain. In the FA server, each generator $(G_1, G_2, \ldots, G_S)$ produces some domain-specific samples $(\mathbf{X}_{S^1}, \mathbf{X}_{S^2}, \ldots, \mathbf{X}_{S^S})$ for the GAD insight analysis, as shown

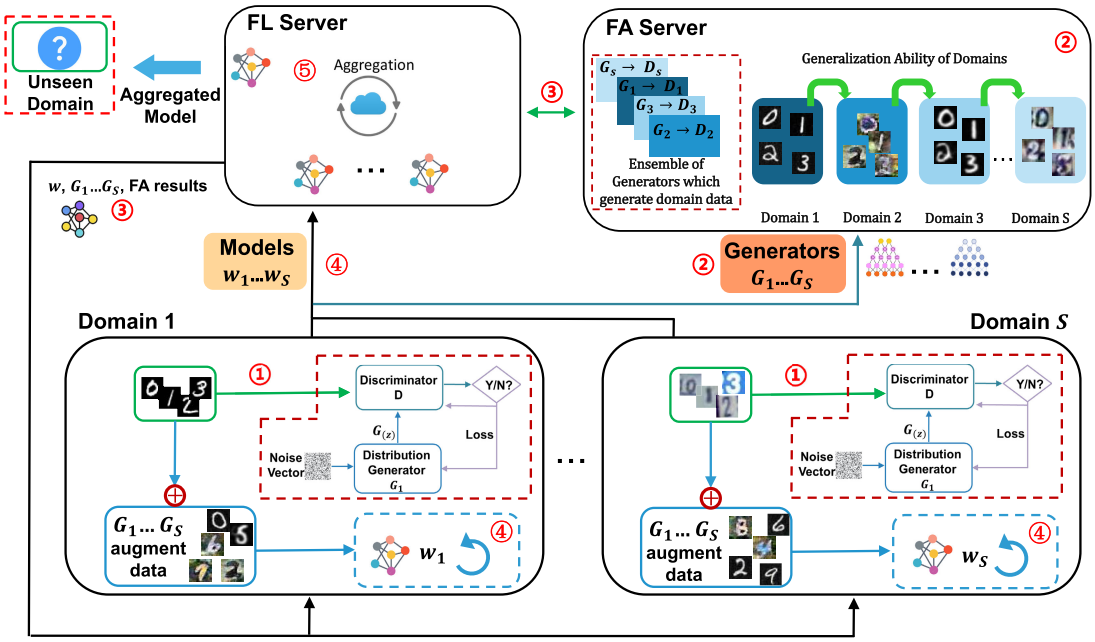


FIGURE 3. The architecture of FA-FDG. Step 1, training the local GANs. Step 2, uploading the generators and doing the FA. Step 3, deployment of the FA results, generators and the original FL model. Step 4 and Step 5, FL model training and aggregation.

in the FA server in Figure 3. In this paper, we calculate the improved KL divergence between the generated samples to evaluate the GAD. Due to the configuration of data sets and experiments, here we only consider the condition of Independent and Identically Distributed (IID) data generation which means the domains have the same label, more complicated non-IID generalization is left for future work. After obtaining the FA results (the GAD list), the FA server will send the ensemble of generators and the FA result to the FL server in Step 3. The FL server will pack the FA results, the ensemble of generators, and the FL model together, then deploy them to each domain that participates in the FL, as depicted in the left of Figure 3. The FA process concludes here. Each domain will receive three parts: the analytical outcomes of the FA, the ensemble of data-augmented generators and the initial training model for FL. Steps 1 to 3 are executed only once. Subsequently, the analytical outputs of the FA will support the FL process like generating more data on domains with higher GAD.

B. DISTRIBUTED DATA AUGMENTATION IN FDG

After receiving the GAD list, the FL model, and the generator ensemble, each client repeats Step 4 to Step 5 in Figure 3 until the model converges. Before training the FL model, each FL client initiates data augmentation using the GAD list. Expect the local raw data, the GAD list guides generators to produce data from other domains, augmenting local training data to enhance the model generalization performance. According to the GAD list from FA, for the domains which exhibit strong generalization ability, we encourage their corresponding generators to generate more samples. These additional data will then augment the local domain data. Conversely, for the domain demonstrating weaker generalization ability, we identify this domain as the 'target domain'. The data augmentation is unnecessary, the original data samples of this domain should not be changed to preserve the domain attribute.

Following the distributed data augmentation, each client uses the enhanced data for model training in Step 4. Once the local training epoch is completed, clients upload their local weights to the FL server for model updates. Subsequently, the FL server aggregates the weights and redistributes them to each FL user, repeating this process in Step 4 and Step 5 until the model converges. The final trained global model is used for testing on unseen domains. In summary, the FA is from Step 1 to Step 3. After getting the insights from FA as well as distributed data augmentation, the FL server repeats Step 4 and Step 5 to implement the FL.

IV. PROBLEM STATEMENT AND FA-FDG SCHEME

A. ENSEMBLE GENERATORS FROM GANS

First, we explain how to collect the generators from different clients and then use the generated data to give the FA results. In GANs training, two neural networks compete in a two-player mini-max game to simultaneously train a generator G and a discriminator D. The goal of the generator $G(\boldsymbol{z};\theta_g)$ is to learn a distribution $p_g(\boldsymbol{z})$ over data $\boldsymbol{x}$, by mapping input noise $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$ to real samples $\boldsymbol{x}$. $p_{\boldsymbol{z}}(\boldsymbol{z})$ is usually an easy-to-sample distribution like the uniform distribution with $(-1,1)$ or Gaussian distribution with $(0,1)$. Meanwhile, the discriminator $D(\boldsymbol{x},\theta_d)$ is trained to discriminate between the real samples $\boldsymbol{x}$ and generated samples $G(\boldsymbol{z})$. The value function

can be described as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (1)$$

In general, D strives to maximize the probability of assigning the correct label to both training examples and samples from G, simultaneously training G to make D cannot discriminate between the raw data and the generated data. In this way, D and G are alternatively optimized. The Jensen-Shannon (JS) divergence is used to measure the difference between $p_{data}(\boldsymbol{x})$ and $p_g(\boldsymbol{x})$ [45]. Firstly we maximize the $V(D, G)$ to get the optimal discriminator by $\frac{\partial V(D,G)}{\partial D} = 0$ and have $D^*_G = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}$. Then minimize the $V(D^*_G, G)$, which can be written as:

$$\min_G V(D^*_G, G) = \min_G [\text{KL}(p_{data}||\frac{p_{data} + p_g}{2}) + \text{KL}(p_g||\frac{p_{data} + p_g}{2}) - \log(4)] \quad (2)$$

The condition for formula (2) to have an optimal solution is $p_{data} = p_g = \frac{p_{data}+p_g}{2}$ with $D^*_G = \frac{1}{2}$. This means that the generated data successfully cheats the discriminator.

Inspired by EFFGAN [46] and synthetic data from GANs [47], we propose a novel method in the FA server that ensembles generators from local GANs, as shown in Algorithm 1. Here m indicates sampling mini-batch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data distribution $p_{data}(\boldsymbol{x})$, while mini-batch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ are selected from noise prior $p^j_g(\boldsymbol{z})$. Each FL client can set proper hyper-parameters (like learning rate and E_0) for the local GAN to avoid overfitting.

Under the principle of moving the model rather than the raw data, each local GAN first trains a generator which confuses the discriminator. Then the FA server collects generators from different clients and ensembles them as ΣG. To realize FA, the ΣG separately generates the virtual data of different domains at the FA server. In this way, the raw data are reserved for each client, and the virtual data in the FA server is similar to the raw data. Meanwhile, the FA server can analyze the virtual data from each collected GAN to have some useful insight into different domains in FA-FDG.

Here the GANs training can choose some popular GANs models like [48] or [49], based on the data type of FL clients. Each FL client first trains a local GAN in parallel with the same batch size and training epoch. The noise size and GAN architectures are given in **Experiment settings** and **Parameter settings** in Section VI. As long as the final well-trained generators have been saved as .pkl files or the parameters, the new trans-convolutional layers can be set to generate new samples. In the experiments, the GAN generators for classification problems are trained by each class or label, so every generator produces a specific class of image data. For example in the MNIST domain of Digit-Five data set [9], after GANs training, 10 .pkl files are gathered as the generator from MNIST domain, the data set PACS [50] is similarly where 7 generators are combined as one domain like Sketch [51]. For the regression problem from the tabular network profiling data, we choose one column from the original data set as the label to predict, then use the raw data (9 columns) to train the CTGAN [49]. The generated data (9 columns) from CTGAN is used for data augmentation in FL clients. The trained FL model will be tested on the raw data of the test domain which has no access during training.

Algorithm 1 Ensemble GANs in FA Server

Procedure 1: Ensemble generators from local GANs. S GAN models and each client holds the same hyper-parameters with the same function, i.e., Adam optimizer with learning.
$\Sigma G \leftarrow \{\}$
for *each client* $j \in [1, S]$ **do**
  for *each local GAN epoch* $E_0 = 1, 2, \ldots$ **do**
    Update the discriminator's (D_j) parameters by ascending its stochastic gradient: $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \{\log D_j(\boldsymbol{x}^{(i)}) + \log(1 - D_j(G_j(\boldsymbol{z}^{(i)})))\}$.
    Update the local generator's (G_j) parameters by descending its stochastic gradient: $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D_j(G_j(\boldsymbol{z}^{(i)})))$.
  end for
  $\Sigma G \leftarrow \{G_j\}$
  end for
Send ΣG to the FA server.
Procedure 2: Generate data from ΣG.
$\mathbf{X} \leftarrow \{\}$
for *each generator* $G_j, j \in [1, S]$ *in* ΣG **do**
  Generate virtual data $\mathbf{X}_{S^j}$ of domain j $\mathbf{X} \leftarrow \{\mathbf{X}_{S^j}\}$
  end for
return: Data sets $\mathbf{X} = \{\mathbf{X}_{S^1}, \mathbf{X}_{S^2}, \ldots, \mathbf{X}_{S^S}\}$

Unlike training GANs in a federated manner like [38] and [39] with high communication overhead and expensive one generator multiple discriminators mode [52], [53], the proposed Ensemble GANs algorithm has some significant advantages: 1) The FA server directly collects the well-trained generators from clients. Without exchanging the discriminators or training the GAN model in a federated aggregation way, this method relieves the high communication overhead and offline problems during the distributed model training. 2) The FA process is considered before FL in the FA server. After gathering the generators, virtual data from different domains can be obtained. Then the FA server can analyze data attributes and make some inferences from different domains, like domain relationships. In the next section, we will give a data augmentation reference (GAD) after virtual data analysis by FA. When the ΣG and GAD are downloaded by the FL clients, they can know which domain in ΣG has more generalization ability among the FDG domains and should augment with more data before FL.

Algorithm 2 GAD With FA

Input: Data sets generated by an ensemble of client generators $\mathbf{X}_{S^i} \in \{\mathbf{X}_{S^1}, \mathbf{X}_{S^2}, \ldots, \mathbf{X}_{S^S}\}$. The S sets of basis of the subspaces for S different source domains $\mathbf{P}_{S^i} \in \{\mathbf{P}_{S^1}, \mathbf{P}_{S^2}, \ldots, \mathbf{P}_{S^S}\}$.

Output: A GAD indicator list with an descending order.

function Domain generalization ability estimation.
  for $j = 1, 2, \ldots, S$ **do**
    GAD $\leftarrow \{\}$
    for $i = 1, 2, \ldots, S$ **do**
      The singular value decomposition from (3) to caculate the principal vectors in (4).
      Find the optimal dimensionality d^* for the PCA subspaces in (9).
      Cacluate the KL divergence in (5).
      Give the GAD estimation for domain S^j in (7), GAD $\leftarrow$ GAD(S^j).
    return: GAD
  for $i = 1, 2, \ldots, S-1$ **do**
    for $j = 1, 2, \ldots, S-i$ **do**
      if $GAD^{(j)} < GAD^{(j+1)}$ **then**
        Swap $\{\text{GAD}^{(j)}, \text{GAD}^{(j+1)}\}$
    end for
  return: List GAD

B. GENERALIZATION ABILITY OF DOMAIN (GAD)

1) FEDERATED ANALYTICS OF GAD

Since the data of the test domain is unseen, the knowledge of the existing data sets that participate in FDG should be fully utilized, and the potential relationship between the data sets should also be fully mined. Among all the domains participating in FDG, it is significant to discover the domain with better generalization performance than other domains. The above-mentioned method requires analysing the relationship between these domains. That is also the reason that we introduce FA with GANs in FDG to break the data silo. Before FL, if we can get insights from FA about the generalization ability among domains, the data augmentation in each FL client will become more focused. By analytics, the domain which can provide more generalization ability should be enhanced by more samples and act as the generalization domain. The domain with less generalization ability should be considered as the adaptation domain and preserve the original data characteristics. Inspired by the Rank of Domain (RoD) matrix in [20] for DA, here we introduce a Generalization ability of Domain (GAD) indicator for FDG by FA. After gathering the well-trained generators in the FA server, the generated data from different domains can be obtained. Let $\mathbf{X}_{S^i} \in \mathbb{R}^{N_{S^i} \times D}$ and $\mathbf{X}_{S^j} \in \mathbb{R}^{N_{S^j} \times D}$ denote the generated data from the source domain S^i and S^j $(i, j \in \{1, 2, \ldots, S\})$.

FA considers data analytics from two aspects, statistically and geometrically. High-dimensional data sets from generators may cause computation complexity. In statistical modeling, FA constructs low dimensional representations of $\mathbf{X}_{S^i}$ and $\mathbf{X}_{S^j}$. Here we use the principal component analysis (PCA) which liner maps the original data set into the subspace. Then statistically, KL divergence is used between data distributions when they are projected into the subspace.

The main ideas are described as follows. The detailed process of GAD is summarized in Algorithm 2, where $\mathbf{P}_{S^i}, \mathbf{P}_{S^j} \in \mathbb{R}^{D \times d}$ denote the two sets of basis of the subspaces for two different source domains, D is the dimensionality of the data and d is the dimensionality of the subspace. Given the singular value decomposition:

$$\mathbf{P}_{S^i}^T \mathbf{P}_{S^j} = \mathbf{U}_1 \Gamma \mathbf{V}^T \tag{3}$$

Here we briefly construct geodesic kernels considering manifold learning, Euclidean metrics are applied to Riemannian manifold spaces [20], [54], [55], [56]. $\mathbf{U}_1$ represents the principal component mapping vector matrix from the space $\mathbf{P}_{S^i}$ to $\mathbf{P}_{S^j}$ under this geodesic kernel. $\mathbf{V}$ represents the original orthogonal basis of mapping $\mathbf{P}_{S^i}^T \mathbf{P}_{S^j}$. The d-th principal angles and vectors from $\mathbf{P}_{S^i}$ and $\mathbf{P}_{S^j}$ can be computed as

$$\begin{aligned} \beta_d &= \arccos \gamma_d \\ \mathbf{s}_d^i &= (\mathbf{P}_{S^i} \mathbf{U}_1)_{.,d} \\ \mathbf{s}_d^j &= (\mathbf{P}_{S^j} \mathbf{V})_{.,d} \end{aligned} \tag{4}$$

where γ_d is the d-th diagonal element of the diagonal matrix Γ. The β_d are principal angles between two sub-spaces, and β_d measures the degree of "overlap" between two subspaces $\mathbf{P}_{S^i}$ and $\mathbf{P}_{S^j}$. The $(\mathbf{M})_{.,d}$ returns the d-th column of the matrix $\mathbf{M}$.

Given a pair of source domains, computing GAD includes 3 steps: 1) determine the optimal dimensionality d^* for the PCA subspaces (will describe in the next part); 2) at each dimension $d \leq d^*$, approximate the data distributions of the two domains with two one-dimensional Gaussians and then compute the KL divergences between them in (5):

$$\mathrm{D}_{\mathrm{KL}}(S^i || S^j) = \frac{1}{d^*} \sum_{d=1}^{d^*} \mathrm{KL}(P_d^{S^i} || Q_d^{S^j}) \tag{5}$$

where $P_d^{S^i}$ and $Q_d^{S^j}$ are two above-mentioned Gaussian distributions which are projected by their d-th principle vectors from PCA as $\mathbf{X}_{S^i}^T \mathbf{s}_d^i$ and $\mathbf{X}_{S^j}^T \mathbf{s}_d^j$ respectively. To facilitate the notation, the $P_d^{S^i}$ and $Q_d^{S^j}$ are simplified as P_d and Q_d. We project data onto the principal vectors and compare the similarity of the data from generators. Due to the excellent properties of the Gaussian distribution, we directly give the closed-form solution of the above (5) when $P_d \sim \mathcal{N}(\mu_{P_d}, \sigma_{P_d}^2)$ and $Q_d \sim \mathcal{N}(\mu_{Q_d}, \sigma_{Q_d}^2)$:

$$\begin{aligned} &\mathrm{D}_{\mathrm{KL}}(S^i || S^j) \\ &= \frac{1}{d^*} \sum_{d=1}^{d^*} \frac{1}{2} [\log \frac{\sigma_{Q_d}^2}{\sigma_{P_d}^2} + \frac{\sigma_{P_d}^2 + (\mu_{P_d} - \mu_{Q_d})^2}{\sigma_{Q_d}^2} - 1] \end{aligned} \tag{6}$$

3) The average KL divergence between one source domain S^j to other source domains is defined as GAD of S^j.

$$\mathrm{GAD}(S^j) = \frac{1}{S}\sum_{i=1}^{S} \mathrm{D_{KL}}(S^i||S^j) \tag{7}$$

The innovation of GAD is summarized as follows: 1. GAD is evaluated by data from GANs to break the data silo in FL clients. By using a small number of synthetic data, the relationship (GAD) among FL clients can be analyzed. That is also the reason we call this process FA. 2. GAD represents the average KL divergence by using one domain to represent other domains, this insight shows which domains may represent other domains well, and also may act well on the unseen test domain. Considering the previous work in domain adaptation like [19] and out-of-domain documents [57], domains with high KL can have better transfer learning performance. 3. The GAD list will guide the GANs data augmentation before FL. Improving generalization performance is not simply augmenting samples indiscriminately, because the test domain is unseen. Instead, it should involve targeted augmentation of samples from domains with strong generalization abilities.

The rationality of using GAD can both be understood from formula (6) and GAD visualization in Figure 12. In (6), the larger $\sigma^2_{Q_d}$ can lead high GAD value of S^j, which means the data distribution from domain S^j has a large variance and wider distribution than other domains, like dark green mn domain in Figure 12 (a). When facing the unseen DG problem, this kind of domain can cover more unknown data distributions whereas the red domain svhn in Figure 12 (a) cannot. Learning from domains with higher GAD can get more data distribution knowledge. The domain with lower GAD is steady, not easy to generalize and more suitable to act as the 'target domain'. The domain with higher GAD can generalize more easily and has higher variance, these domains are better to be the 'source domain'. The results in [20], [43], [57], [58] and Table 6 in experiments also proved this inference.

Algorithm 3 Data Augmentation for FDG

1. Aggregation FL Server executes:
prepare: ΣG, initialize: $w \leftarrow w_0$
for *each global round* $t = 1, 2, \ldots, E_2$ **do**
  for *each client* $i = 1, 2, \ldots, S$ *in parallel* **do**
    $w_i^t \leftarrow w^t$
    $w_i^{t+1} \leftarrow$ **ClientTraining**$(w_i^t, \mathbf{X}^{(i)})$
  $w^{t+1} \leftarrow \frac{1}{\sum_i^S n_i}\sum_i^S n_i w_i^{t+1}$ // Model Aggregatoin
return: w^{E_2}
2. Client Data Augmentation: // Run on each client i which receives ΣG from the server
for *each generator* G_j *in* ΣG **do**
  Generate vritural domain data $\mathbf{X}$ as (11)
  $\mathbf{X} = \{\mathbf{X}_{S^1}, \mathbf{X}_{S^2}, \ldots, \mathbf{X}_{S^S}\}$
  if *the GAD*(S^i) *is the last item in List GAD* **then**
    $\mathbf{X}^{(i)} = \{\mathbf{X} - \mathbf{X}_{S^i}\} \cup \mathbf{X}^{(i)}$
  else
    $\mathbf{X}^{(i)} = \mathbf{X} \cup \mathbf{X}^{(i)}$
  Caculate the MMD distance of $\mathbf{X}_{S^i}$ and $\mathbf{X}_{S^j}$ in $\mathbf{X}^{(i)}$ as (13)
  end for
return: $\mathbf{X}^{(i)}$ and MMD distance
3. Client Training: // Run on each client i
Receive w_i^t from server
if $t = 1$ **then**
  $\mathbf{X}^{(i)}$, MMD $\leftarrow$ **Client Data Augmentation**
else
  Break
for *each epoch* $e = 1, 2, \ldots, E_1$ **do**
  Update w_i^t to minimize $\mathcal{L} = \mathcal{L}_C + \lambda\mathcal{R}_{mmd}$ $(w_i^{t+1} \leftarrow w_i^t - \eta\nabla\mathcal{L})$
  end for
Upload w_i^{t+1} to the FL server

2) OPTIMAL DIMENSIONALITY DETERMINATION

After data sets are generated in FA, we should determine the optimal dimensionality of the subspaces. The optimal dimensionality d of the subspaces is automatically determined without using any labels. This task is accomplished by a subspace disagreement measure (SDM).

First, PCA subspaces of data sets are calculated, i.e., PCA_{S^j}. Then, the data sets are combined into one data set and computed the subspace $\mathrm{PCA}_{S^1+S^2+\ldots+S^S}$. Owing to different but related domains joining the DG, all these subspaces should not be too far away from each other. Based on this sight, the SDM in DG is defined based on principal angles:

$$\mathrm{SDM}(d) = \frac{1}{S}\sum_{j=1}^{S} \sin\alpha_d^{(j)} \tag{8}$$

where $\alpha_d^{(j)}$ denotes the d-th principal angle between the PCA_{S^j} and $\mathrm{PCA}_{S^1+S^2+\ldots+S^S}, j \in \{1, 2, \ldots, S\}$. Note that $\mathrm{SDM}(d)$ is at most 1. A small value indicates that $\alpha_d^{(j)}, j \in \{1, 2, \ldots, S\}$ are small, and thus $\mathrm{PCA}_{S^j}, j \in \{1, 2, \ldots, S\}$ are aligned at the d-th dimension. When $\mathrm{SDM(d)} = 1$, $\alpha_d^{(j)} = \frac{\pi}{2}$, $j \in \{1, 2, \ldots, S\}$, the directions of these S subspaces are orthogonal. Under this circumstance, domain relationship evaluation will become difficult since variances captured in one subspace would not be able to transfer to the other subspace. To determine the optimal dimensionality d, a greedy strategy is adopted,

$$d^* = \min\{d|\mathrm{SDM}(d) = 1\} \tag{9}$$

Intuitively, d^* should be as high as possible to preserve variances in the source domains in order to explicit the analytics results, meanwhile, not be so high that the directions

of the S subspaces start to be orthogonal. The FA of GAD is summarized in Algorithm 2.

After getting the GAD list, the FA server sends this list to the FL server. In the FL server, ΣG, the GAD list, and the initial model for FL are packed together and sent back to each client. Then the FL participants can know the rank of GAD between the domains and realize data augmentation based on this FA results. For domains ranked higher, ΣG generates more samples to enhance the model generalization ability. For domains ranked lower, we maintain the samples unchanged to preserve adaptability.

C. Domain Generalization With Data Augmentation

After FA, the GAD insight will be issued and used to guide the extent of data augmentation at each FL client. The performance of ML models heavily relies on the quality and quantity of training data, especially in FDG where the models are expected to generalize well to unseen domains. Under the guide of the GAD list, more samples should be generated for the domains in front of the GAD list.

In each FL client i with local raw data $\mathbf{X}^{(i)}$, the data augmentation can be expressed mathematically as:

$$\mathbf{X}^{(i)} = \mathbf{X}^{(i)} \cup \mathbf{X} \tag{10}$$

$$\mathbf{X} = \sum_{j=1}^{S} v_j G_j(z) = \sum_{j=1}^{S} \mathbf{X}_{S^j} \tag{11}$$

where $G_j(z)$ is the jth generator in ΣG, v_j is the data augmentation coefficient guided by GAD list. When GAD(S^i) is the last item of the GAD list (the domain i represented by client i has less generalization ability than other domains), we set $v_i = 0$. This means that the generalization ability of this local domain data is already weak, thus, there is no need to generate more data from this domain.

The intuition is that if we can learn a representation which minimizes the distance between the source distributions, then we can train a classifier on the source data and apply it to the unseen domain. The goal is to learn a feature space underlying all the seen source domains, by minimizing the distribution variance among them. The Maximum Mean Discrepancy (MMD) distance is chosen as the solution here. MMD is a statistical metric used to measure the discrepancy between probability distributions of different domains. The goal of using MMD in FDG is to minimize the distribution shift between domains and promote domain alignment. By minimizing the MMD, FDG aims to learn domain-invariant representations that capture shared knowledge across domains while preserving domain-specific characteristics, calculated as follows:

$$\begin{aligned}
&\mathrm{MMD}(\mathbf{X}_{S^i}, \mathbf{X}_{S^j})^2 \\
&= \left\| \frac{1}{N_{S^i}} \sum_{l=1}^{N_{S^i}} \phi(\mathrm{x}_{S^{i,l}}) - \frac{1}{N_{S^j}} \sum_{m=1}^{N_{S^j}} \phi(\mathrm{x}_{S^{j,m}}) \right\|_{\mathcal{H}}^2 \\
&= \frac{1}{N_{S^i}^2} \sum_{l=1}^{N_{S^i}} \sum_{l'=1}^{N_{S^i}} \phi(\mathrm{x}_{S^{i,l}}) \cdot \phi^T(\mathrm{x}_{S^{i,l'}}) \\
&\quad + \frac{1}{N_{S^j}^2} \sum_{m=1}^{N_{S^j}} \sum_{m'=1}^{N_{S^j}} \phi(\mathrm{x}_{S^{j,m}}) \cdot \phi^T(\mathrm{x}_{S^{j,m'}}) \\
&\quad - \frac{2}{N_{S^i} N_{S^j}} \sum_{l=1}^{N_{S^i}} \sum_{m=1}^{N_{S^j}} \phi(\mathrm{x}_{S^{i,l}}) \cdot \phi(\mathrm{x}_{S^{j,m}}),
\end{aligned} \tag{12}$$

where $\phi : \mathbb{R} \to \mathcal{H}$ is a feature map, $\mathrm{x}_{S^{i,l}} \in \mathbf{X}_{S^i}, (l = 1, 2, \ldots, N_{S^i})$, and $\mathrm{x}_{S^{j,m}} \in \mathbf{X}_{S^j}, (m = 1, 2, \ldots, N_{S^j})$. N_{S^i} and N_{S^j} are the number of samples in each domain. When we write as the kernel format as $K(\mathrm{x}_{S^{i,l}}, \mathrm{x}_{S^{j,m}}) = \phi(\mathrm{x}_{S^{i,l}}) \cdot \phi(\mathrm{x}_{S^{j,m}})$, the MMD distance can be written as:

$$\begin{aligned}
\mathrm{MMD}(\mathbf{X}_{S^i}, \mathbf{X}_{S^j})^2 &= \frac{1}{N_{S^i}^2} \sum_{l=1}^{N_{S^i}} \sum_{l'=1}^{N_{S^i}} K(\mathrm{x}_{S^{i,l}}, \mathrm{x}_{S^{i,l'}}) \\
&\quad + \frac{1}{N_{S^j}^2} \sum_{m=1}^{N_{S^j}} \sum_{m'=1}^{N_{S^j}} K(\mathrm{x}_{S^{j,m}}, \mathrm{x}_{S^{j,m'}}) \\
&\quad - \frac{2}{N_{S^i} N_{S^j}} \sum_{l=1}^{N_{S^i}} \sum_{m=1}^{N_{S^j}} K(\mathrm{x}_{S^{i,l}}, \mathrm{x}_{S^{j,m}})
\end{aligned} \tag{13}$$

We define the regularization term $\mathcal{R}_{mmd}$ on augmented data as:

$$\mathcal{R}_{mmd}(\mathbf{X}_{S^1}, \ldots, \mathbf{X}_{S^S}) = \sum_{1 \le i,j \le S} \mathrm{MMD}(\mathbf{X}_{S^i}, \mathbf{X}_{S^j})^2 \tag{14}$$

The targeted data augmentation of FDG is summarized in Algorithm 3. An example of the $\mathcal{R}_{mmd}$ is shown in Figure 4. Combing Algorithm 1 and Algorithm 3, the objective of data augmentation for each FL client can be expressed as:

$$\mathcal{L} = \begin{cases} \mathcal{L}_{GAN} & E_0 \\ \mathcal{L}_C(\mathbf{X}^{(i)}, \omega) + \lambda \mathcal{R}_{mmd} & E_1 \end{cases} \tag{15}$$

where E_0 and E_1 separately represent the local GAN epoch and the local training epoch. E_2 in Algorithm 3 means the global epoch of FL, which is the times of complete FL rounds, finished by FL clients and the FL server. $\mathcal{L}_{GAN}$ indicates the loss of the local GAN. $\mathcal{L}_C$ denotes the classification loss on the enhanced data $\mathbf{X}^{(i)}$, $\mathcal{R}_{mmd}$ denotes the distances between the source domains, λ is the hyper-parameter.

D. COMPLEXITY ANALYSIS

DG is a problem with high time complexity, and the training model needs time to find the underlying domain invariance. During the FA process, we estimate GAD by generating a small number of samples from each domain. Since the attributes of FDG, each source domain ($S^j, j \in [1, S]$) is in charge of GANs training, so the overall time complexity of procedure 1 and procedure 2 in Algorithm 1 is $\mathcal{O}(SE_0) + \mathcal{O}(S) = \mathcal{O}(SE_0)$. In Algorithm 2, the time complexity consists of FA and the sorting algorithm, $\mathcal{O}(S^2) + \mathcal{O}(S^2) = \mathcal{O}(S^2)$. The time complexity of PCA is $\mathcal{O}(nD^2 + D^3)$, n is a small number of generated data just for FA.

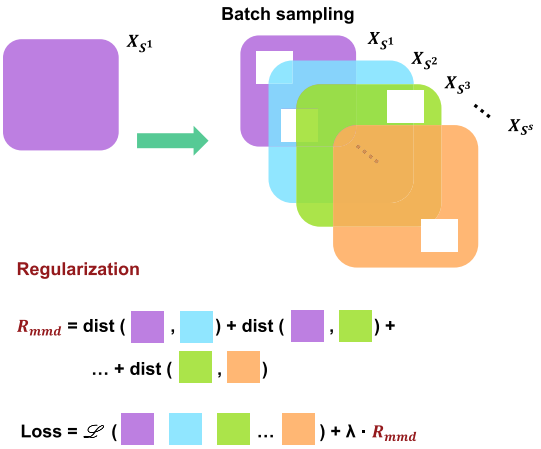


FIGURE 4. An example of data augmentation. X_{S^1} domain (lavender) is the last item in the GAD list for client 1, other domains have more generalization ability. Here, the dist() method is set as MMD distance.

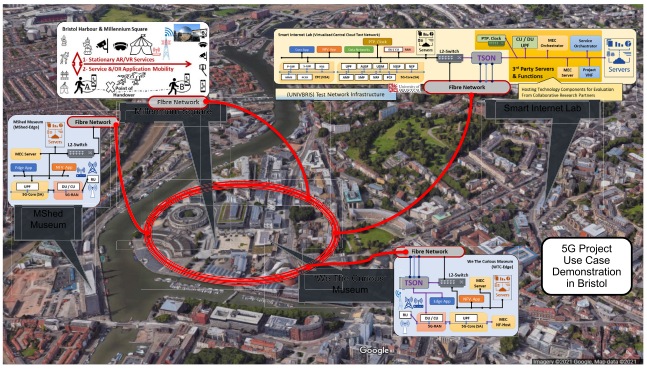


FIGURE 5. Smart Internet Lab Networking Test bed.

V. TEST BED DEPLOYMENT

A. KUBERNETES-BASED FEDERATED LEARNING PLATFORM

Experiments were carried out at the Smart Internet Lab's Networking test bed (Figure 5) which consists of 1 main node (i.e., core/cloud) referenced as Smart Internet Lab, and 3 edge nodes. These nodes are interconnected using fiber and have a control plane in the main node. Implementing FL systems across diverse decentralized clients poses challenges due to their heterogeneity. Cloud technologies like containers and Kubernetes (K8s) enhance computation elasticity and efficiency in deploying distributed ML architectures [28]. We present a K8s-based FL implementation founded on the work by Parra-Ullauri et al. [28], which is deployed in the test bed. This K8s-based implementation, shown in Figure 6, allows us to control the deployment and life-cycle management of FL pipelines while ensuring network isolation and enabling data packet encryption, guaranteeing privacy preservation in cloud-native environments. Figure 7 describes the logical experimental setup. The test bed for running all the experiments is managed by a private OpenStack[1] instance. The physical locations are scattered in different geo-locations to replicate a real FL cross-silo setup. A K8s cluster, consisting of one controller node and three working nodes was deployed. To reduce the need for resources, the controller node also runs the FL server, as part of the domain belonging to the shared central service (using the 10.244.0.0/24 network). This node has 15vCPUs, 25GB of RAM and 100GB of disk space. The worker nodes have 15vCPUs, 15GB of RAM and 100GB of storage. Worker nodes 1, 2, 3 and 4 are individually part of a unique administrative domain, that uses the 10.244.1.0/24, 10.244.2.0/24, 10.244.3.0/24, and 10.244.4.0/24 respectively. Various FL clients can run in the different worker nodes depending on the experiment performed.

To build the FL pipeline, the Flower framework [21] was used and deployed in a K8s cluster. Flower enables the development of FL systems and allows the federation of any workload while being agnostic to the underlying ML framework (e.g., TensorFlow or PyTorch) and programming language (e.g., Python or C++). Here we accelerate the learning with our A30 GPU servers, both for centralized and federated training. The seminal Fed-Avg [2] algorithm is enhanced by the proposed FA-FDG in this paper, to test the model's generalization ability.

[1] Openstack: https://www.openstack.org/.

VI. EXPERIMENTS

A. DATA SETS AND EXPERIMENTS SETTINGS DESCRIPTION

1) DATA SETS

To evaluate our FA-FDG method, three data sets are tested in experiments from the aspect of the classification problem and the regression problem. **1. Digit − Five [12]:** This data set is a collection of five popular digit data sets, MNIST (mn), MNIST-M (mm), Synthetic Digits (syn), SVHN (svhn), and USPS (usps). Each digit data set acts as a domain with different styles of 0-9 digit images. There are five different domains in this scenario with 10 classes. **2. PACS [50]:** In this data set, four different domains (photo (P), art painting (A), cartoon (C), sketch (S)) are contained. The task is classification with seven classes. **3. VNF Profiling [15]**: In this data set, there are three different VNF types: SNORT Inline VNF mode, SNORT Passive VNF mode, and a virtual Firewall (vFW) VNF type, as three different domains. There are eight features and one output. The input variables are CPU utilization (CPUUTP), Memory utilization (MEMUTP), Network latency (RTT), VNF maximum input rate (MIR) and Packet loss (In_RX, Out_Tx). The output variables are one of the VNF resource configurations like CPU cores (CPU), Memory (MEM/MB) and Link Capacity (LC/Mbps). Our goal is to train a model in a federated manner that can generalize well on a new VNF type to predict accurate VNF resource configurations, under the guidance of our FA results with data augmentation.

2) EXPERIMENT SETTINGS

For all data sets, we perform "leave-one-domain-out" experiments [58], where one domain is picked as the target domain,

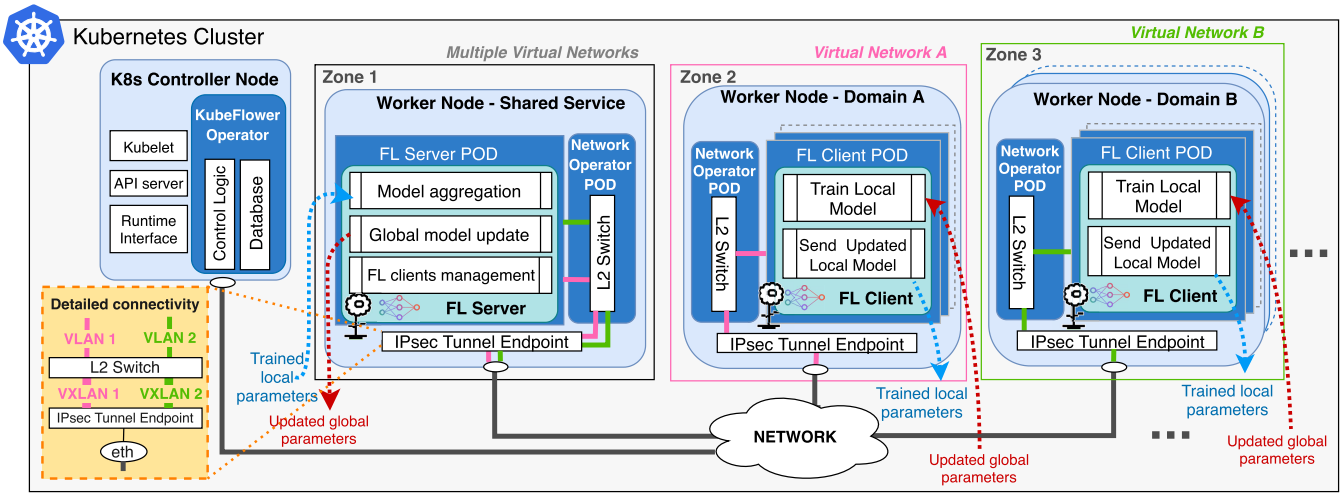


FIGURE 6. **Privacy-Preserving Operator for Kubernetes-based Federated Learning to enable Intelligence Smart Networks [28].**

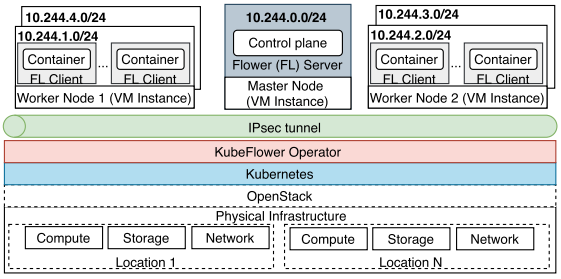


FIGURE 7. **Experiment setup.**

and we train the model on all remaining domains. Then the trained classification or regression model is evaluated in the chosen domain. Each source domain is treated as a FL client. Following standard practice, we use 80% of available data as training data and 20% as validation data. In experiments the well-trained GAN models are firstly packed as .pkl files, then these files are sent to the FA server as well as each FL client. The GAN generators for classification problems are trained by each class or label, so generators combine labels of image data in one domain. For the regression problem from the tabular network profiling data, we choose one column (CPU, MEM or LC) from the original data set as the label to predict. The raw data (9 columns) are used to train the CTGAN. The generated data (9 columns) from CTGAN is used for data augmentation in FL. The trained FL model will be tested on the raw data of the test domain which has no access during training. λ is set as 0.001 in (15). For data set Digit-Five and PACS, all the models are trained with NVIDIA A30 GPUs from our server cluster with $E_0 = 30$, $E_1 = 20$, $E_2 = 6$, considering the previous work about setting E in FDG and DG [9], [50], [58]. DCGAN [48] is chosen as the model of generators for Digit-Five and PACS. For VNF profiling data set, CTGAN [49] and YData Synthetic [2] are selected to generate tabular data and save corresponding CTGAN models with $E_0 = 3000$, $E_1 = 20$, $E_2 = 100$.

[2]YData Synthetic: https://github.com/ydataai/ydata-synthetic.

3) BENCHMARKS

For FA-FDG with distributed data augmentation, we choose **1.Fed − Avg [2]:** which tries to train a shared model across clients by minimizing the overall global loss that is a weighted average of the individual clients' losses. **2. Fed − Prox [6]:** introduces a proximal term that penalizes large changes in weights, this method also helps convergence on highly heterogeneous data. **3. Fed − Yogi [27]:** uses the YOGI adaptive optimizer in a federated version, also tries to generalize more framework for federated optimization. In addition, it is good to mention that q-FedAvg [7] is not suitable for generalization because the larger hyper-parameter q is, the worse-performing clients will dominate the overall loss. In FDG, each domain has its own data characteristic, with four or five clients, q-FedAvg will seriously affect the generalization ability even with little q value. Here we don't show q-FedAvg but with more worthwhile methods like Fed-Avg, Fed-Prox and Fed-Yogi. In Fed-Prox, the hyper-parameter μ is set as one in our experiments, μ shows the weight of the proximal term used in the optimization. If $\mu = 0$, it makes this strategy equivalent to Fed-Avg.

In addition, we also compare with centralized DG methods to show the benefit of data augmentation. For centralized DG, **1. MMD [25]:** MMD aligns the distributions among different domains with generated data. The goal is to learn a representation that minimizes the distance between the different domains. The hyper-parameter λ inhibits the MMD penalty to determine how strongly we would like to confuse the domains. For the domains ranked ahead in the GAD list, λ should be reduced to improve generalization ability and vice versa. **2. CORAL [26]:** Correlation Alignment (CORAL) is an unsupervised method which minimizes domain shift by aligning the second-order statistics of source

TABLE 2. Hyper-parameters for digit-five data set.

Method	mn	mm	svhn	syn	usps
MMD(λ) [25]	1.0	0.5	0.5	1.0	1.0
CORAL(λ) [26]	1.0	0.5	0.5	1.0	1.0
DANN(λ) [59]	1.0	0.1	0.1	0.1	1.0
RSC(λ_1, λ_2) [60]	0.1, 0.1	0.3, 0.1	0.3, 0.2	0.1, 0.3	0.1, 0.1

TABLE 3. Hyper-parameters for PACS data set.

Method	A	C	P	S
MMD (λ) [25]	1.0	1.0	0.5	0.5
CORAL (λ) [26]	0.5	0.1	1.0	0.01
DANN (λ) [59]	0.5	1.0	0.1	0.1
RSC (λ_1, λ_2) [60]	0.1, 0.3	0.3, 0.1	0.1, 0.1	0.1, 0.1

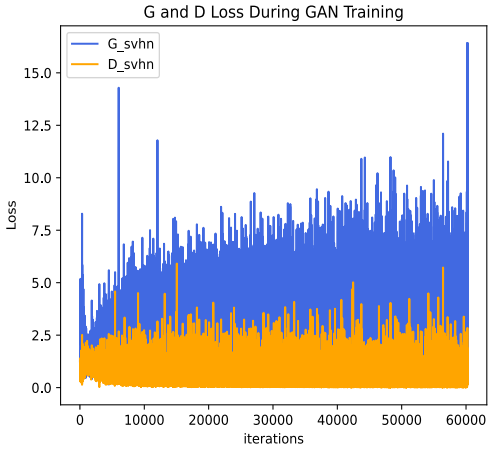


FIGURE 8. The loss of DCGAN during training domain svhn.

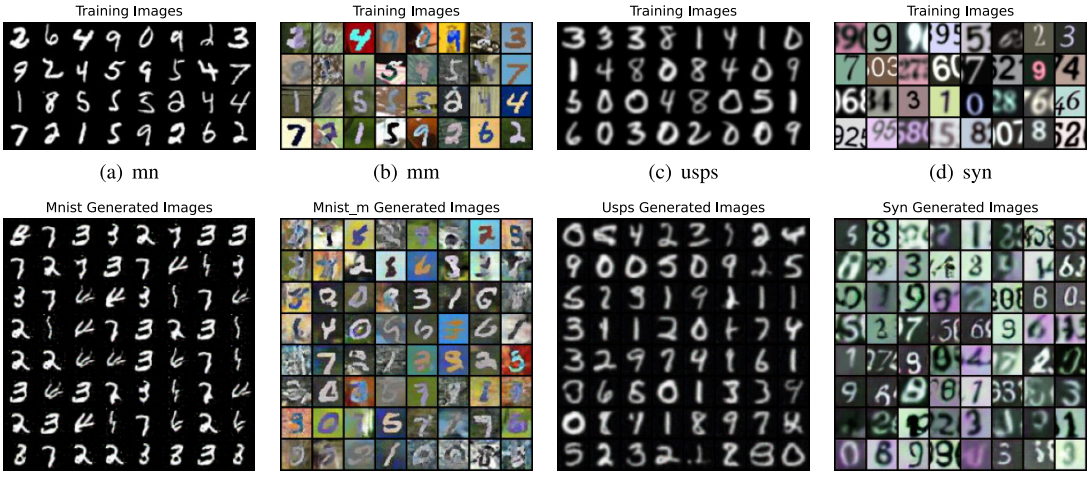
domains' distributions. A CORAL loss is added to minimize the difference in learned feature co-variances across domains, this is similar to minimizing MMD with a polynomial kernel. The function of hyper-parameter λ is nearly the same as MMD. **3. DANN [59]:** Domain-Adversarial Training of Neural Networks (DANN) is a neural network architecture designed to accomplish precise classification of source data while learning invariance feature representations across multiple domains. The parameter λ controls the trade-off between the class classification and domain classification when doing the feature learning. **4. RSC [60]:** Representation Self-Challenging (RSC) iteratively discards the dominant features activated on the training data, and forces the neural network to activate remaining features that correlate with labels. Here are two hyper-parameters λ_1 and λ_2. λ_1 controls the representation to be discarded. The more λ_1 is, the more other features are used to challenge the model. λ_2 guides the percentage of samples to apply RSC in each batch during training. The values of hyper-parameters are shown in Table 2 and Table 3. Table 6 and Table 7 give the comparison of generalization results.

4) PARAMETER SETTINGS

First, each domain trains a domain generator model with the same architecture. Here DCGAN [48] is chosen as the model of generators for Digit-Five and PACS data sets. Domain generators in Digit-Five have four convolutional-transpose layers and output 32×32 images. For the PACS data set, domain generators have five convolutional-transpose layers and generate 64×64 images. The learning rate is 0.0002 with 30 epochs of each class in each domain and $\boldsymbol{z} = 100$. The discriminators have five convolutional layers. For CTGAN in the VNF profiling data set, each VNF domain has the same CTGAN architecture, we set six input variables and three output variables. The epochs are set as 3000 to train CTGAN models which can generate tabular data as close to the raw data.

Then, the well-trained generators are sent to the FA server to obtain insights from the generated data. Considering the data privacy issue, only one class generator with 10 generated samples from each domain is used to estimate the GAD list in the FA server. Here the d^* is set as 200 in Digit-Five and PACS data sets, and d^* is set as 2 in VNF profiling data sets. After having an estimation of the GAD, the domain generators with the insight GAD list will be deployed to each domain client. With the guidance of the GAD list, in each FL client, generators can enhance the domain data to improve the generalization ability of the final training model.

In centralized DG, we evaluate the accuracy fluctuations after each epoch. Whereas in FDG, the test domain is unseen, so we directly give the accuracy of the aggregated FL model on the test domain. For Digit-Five and PACS data sets, in centralized DG, the total maximum number of epochs is 120. In FDG, the local epoch is set as 20 and the number of global aggregations is 6. For FA-FDG, the training round is increased by the same proportion as the samples where $E_1 = 20, E_2 = 7$. Considering the VNF profiling data sets, the local epoch is set as $E_1 = 20$ and the FL round is $E_2 = 100$. For centralized DG, we choose ResNet-18 with the guidance of DomainBed [58] and DeepDG [9], and the SGD optimizer is used in centralized training. For FDG, based on the concept of the unseen domain, we choose Flower [21] and use server-side evaluation. One domain is set as the test unseen domain on the server side for evaluation purposes. The model is trained by other domains in a federated manner. We test the newly aggregated model after each FL round E_2. For the Digit-Five data set, we choose a simple two-layer CNN to train and test in FDG. For PACS, we train ResNet-18 because of the image size. In addition, data from the DomainNet [51] data set are used to train the GAN models in PACS as an auxiliary. For VNF profiling data sets, a 4-layer multilayer perceptron (MLP) acts as the training model of FDG. The optimizer is Adam in FDG. The latter experiment results will show that the FA-FDG can improve the generalization accuracy. Even with simple neural network architecture, the



(a) mn (b) mm (c) usps (d) syn

(e) mn (f) mm (g) usps (h) syn

FIGURE 9. Training data and the generated data from generators of different domains.

proposed FA-FDG can improve the model performance close to the centralized results.

B. RESULTS AND ANALYSIS

1) TRAINING GANS FROM LOCAL DOMAINS

Figure 8 gives the training loss of local DCGAN in the domain svhn. Here we set the local training GANs epoch as 30. It is clear in Figure 8 that the discriminator is not overfitting, and the diversity of the generated samples increases, avoiding the GAN mode collapse. To avoid overfitting, in our DG cases, the GANs training does not have to make the GAN model generate the same data as the raw local data, but generate some data similar to the raw data. GANs in DG are not over-focused on replicating specific features present in the training data. Figure 9 shows the DCGAN result on Digit-Five data sets, and the results on the PACS data set are nearly the same, here we only illustrate the generated image results of Digit-Five. Five domains (mn, mm, syn, svhn, and usps) separately train and save their own GAN models. Figure 9 gives some generated data (Figure 9 (e), (f), (g), (h)) from the well-trained generators in each domain. From Figure 9, we can see that although the generated images may have slight differences compared to the original data like mm (Figure 9 (b), (f)) and syn (Figure 9 (d), (h)). The generated data is still close to the raw data (mn (Figure 9 (a), (e)), usps (Figure 9 (c), (g))). Training with the original data and the generated data together can improve the final model's generalization ability to some extent. Due to the data privacy requirements of FL, we can adopt the idea of keeping the model dynamic while the data remains static. Collecting GAN generator models to enhance and expand the training data. This approach not only prevents the leakage of the raw data but also enriches the training data set, so the final trained model should have competitive performance on the test unseen domain.

For the VNF tabular data set, Figure 10 demonstrates the difference in feature probability distributions between the raw

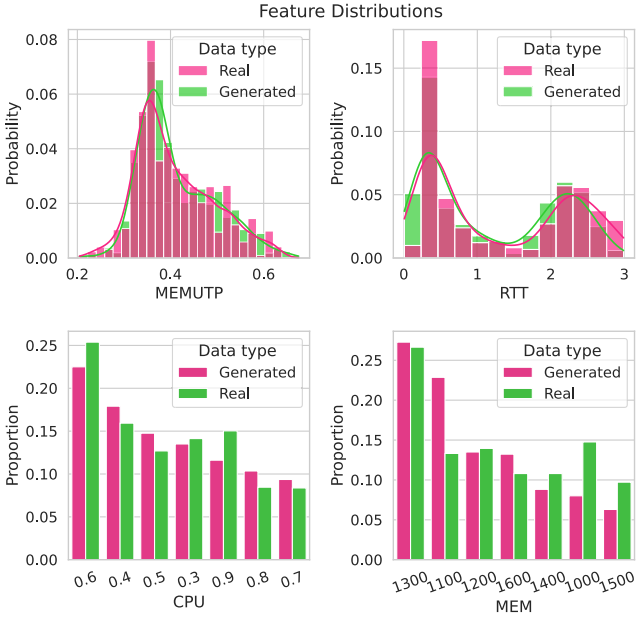


FIGURE 10. Generated data distribution of SNORT-Inline VNF type.

data (red) and generated data (green). The x-axis represents the real values of the feature, and the y-axis represents the probability distribution of the feature or the proportional distribution of the label. Referring to CTGAN [49], the features are set as continuous variables like MEMUTP, and the labels that we need to predict are set as discrete variables like CPU and MEM. The red line is the probability density function curve of one feature from the real data, the green line is the probability density function curve of the same feature from the generated data. Here we only illustrate some feature distributions of Inline VNF, and the results for the rest two VNF types are similar to this figure. The value to predict is the needed CPU, MEM, or LC for a new VNF configuration. For each VNF domain, a CTGAN model is trained separately,

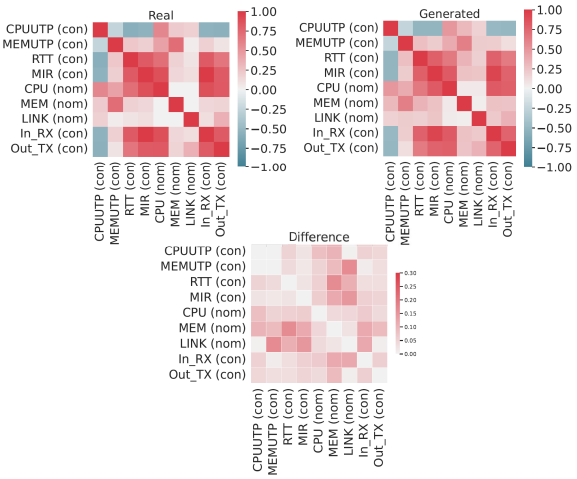


FIGURE 11. Correlation difference between the real data and generated data.

and the generated model files are uploaded to the FA server. It is clear in Figure 10 that the data distribution generated by CTGAN closely approximates the raw data distribution.

Figure 11 also displays the association matrices for the real VNF data set and the generated VNF data set. Pearson's correlation coefficient is used to show the difference between features. The first two figures in Figure 11 present feature correlations in the raw data and generated data. The second-row figure gives the difference. The generated data by CTGAN looks similar to the raw VNF data. However, the deviation between the generated data and the real data (difference in Figure 11) will not affect the final FDG result. Unlike domain adaptation, the generated data does not need to be the exactly same as the raw data because here the main focus is the generalization problem and the test domain is unseen, the local GANs training also avoids overfitting the raw data. On the contrary, we need to utilize this kind of 'deviation' to improve the generalization ability. To some extent, DG tries to make the model work well on the unseen domain, using GANs can be also seen as insights from the AI model, which expresses the generalization insight by the generated data. This is also one of the reasons that we include generative AI in FDG. Our work is also an example of the network for AI and AI for the network. On the network side, the network generates data for the AI model to learn; on the AI side, it generates augmented data for self-training to enhance the accuracy of model predictions in the network. The augmentation of data could improve the model's generalization performance when the model applies to a new VNF type and predicts the needed resources. We believe future network orchestrators also need self-data generation, self-enhancement model training and generalize well on the unseen network scenarios, considering the AI-network mutual enhancement.

2) FEDERATED ANALYTICS FROM THE GENERATED DATA

After training the generator models in each domain, the FA server collects these generators and produces samples for analytics. Firstly, PCA is used to reduce the dimensionality of the generated data in each domain. Then, the GAD value for each domain in FA-FDG is calculated by formula (7). At last, the GAD list is arranged in descending order. The GAD lists obtained by FA are shown in Table 4. From the table, we can get some insights about the generalization ability of domains. For Digit-Five, the mn data set has the strongest generalization ability, followed by mm, and the last is svhn. This is within reason considering the real data distribution. Other domains can be seen as derivatives of the standard mn data set, and mn can be transformed into data sets with different styles. The svhn data set has less generalization ability because of the varied image styles and a large amount of interfering digits. The same idea applies to the PACS data set where other domains can be considered as variations of the real-world P data set, and the S images contain more abstract information. Then P and S data sets have stronger generalization ability compared to A and C data sets. In the VNF profiling data set, the previous work [15] explains that Snort-Passive VNF has similar functions as vFW VNF, but Snort-Passive VNF uses several detection rules to detect malicious network traffic activities. This configuration makes the Snort-Passive VNF have better generalization performance than the simple-setup vFW VNF.

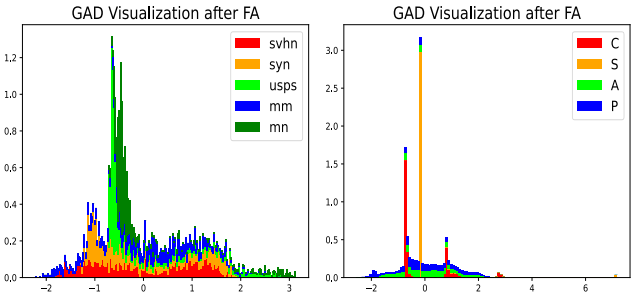
Figure 12 visualizes the GAD results from Table 4. The results are based on the generated data from domains in Digit-Five and PACS. GAD can be seen as an insight after using FA, and Figure 12 (a) (b) mainly wants to show this insight more intuitively. GAD list can represent the generalization ability order of domains in Table 4: mn > mm > usps > syn > svhn. S > P > A > C. In Figure 12, the x-axis represents the data after PCA dimensionality reduction, and the y-axis

TABLE 4. GAD list value of domain data sets.

Domain	mn	mm	usps	syn	svhn
GAD	4.923	3.313	2.533	1.446	1.302
Domain	S	P	A	C	
GAD	4.057	3.787	3.408	1.202	
Domain	SNORT-Passive	SNORT-Inline	vFW		
GAD	1.329	1.246	1.082		

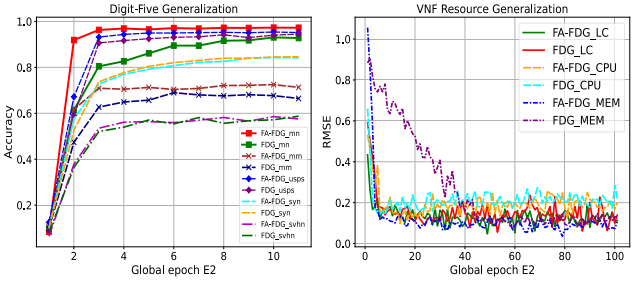
TABLE 5. Computation costs with/without GAN architecture in FL, with the FDG model accuracy improve up to 3.7%.

Method	Metrics	FA(GAN)	FL	Overhead
Traditional FL	Training Time	-	15min17s	-
	CPU usage	-	35.4%	-
	GPU usage	-	6%	-
	RAM used	-	4.589GB	-
FA(GAN)+FL	Training Time	10min2s	20min10s	1.98x
	CPU usage	61.3%	41.9%	2.92x
	GPU usage	28%	8%	6.00x
	RAM used	4.608GB	5.075GB	2.11x



(a) GAD insight in Digit-Five (b) GAD insight in PACS

FIGURE 12. The visualization of GAD insight after FA in Digit-Five (a) and PACS (b).



(a) Accuracy in Digit-Five (b) RMSE in VNF profiling

FIGURE 13. The FDG model accuracy on Digit-Five (a) and RMSE on VNF Snort Inline (b), after data augmentation with GAD from FA.

represents the probability density of the reduced data. Different colours represent different domains. From Figure 12 (a), it is obvious that the probability distributions covered by the mn (green), mm (blue), and usps (lime) data sets are wider. These results lead to a higher probability in the cumulative distribution function when giving random input variables, indicating a stronger generalization ability. On the contrary, the syn (orange) and svhn (red) data sets have lower probability distributions and do not contain rich data components. This distribution causes weaker generalization compared to mn. Figure 12 (b) displays the probability density distribution of the PACS data set. We can see that the P data set has a higher cumulative distribution function value when compared with other data sets. The blue area in Figure 12 (b) shows better generalization ability, the S domain is special because it includes simple line binary images. The rationality of using GAD can both be understood from formula (6) and Figure 12. In formula (6), the larger $\sigma^2_{Q_d}$ can lead high GAD value of S^j, which means the data distribution from domain S^j has a large variance and wider distribution than other domains, like dark green mn domain in Figure 12 (a). When facing the unseen generalization problem, this kind of domain can cover more unknown data distributions whereas the red domain svhn in Figure 12 (a) cannot. Suppose we train a model only from the svhn domain, but the unseen domain distributes from 2 to 3 in Figure 12 (a). In that case, the generalization result will be really bad because not enough data distribution was learned from only the svhn red part, svhn is too stable to generalize. This is the reason we use GAD to measure the stability of a domain. For domains with high GAD values, we enhance the data to improve the overall generalization performance. For the domain with the lowest GAD value, we retain the original data to prevent the generated data from harming the generalization performance. The GAD list can also give guidance about hyper-parameter choosing as explained in **Benchmarks**.

3) FDG RESULTS WITH DATA AUGMENTATION

After data augmentation with the GAD list in each FL client, metrics are used to evaluate the FL model. In classification problems, the accuracy of the FL model on the unseen test domain is recorded after each complete FL round. In regression problems, we use the error-based evaluation metrics for comparing the FDG models. Three indices are Mean Average Error (MAE), Root Mean Squared Error (RMSE), and R-squared values. MAE and RMSE are used to conclude the best model approach. The smaller the value of these error metrics, the higher the accuracy of predictions. R-squared, is used to show how well the data fit the regression model. The closer the R-squared value is to 1, the fit result is better. Considering the GAD list in Table 4, in Digit-Five, FA-FDG sets the well-trained generators of mn, mm, and usps as data augmentation generators. These generators are distributed to each training domain which will have new data from mn, mm, and usps. The amount of newly generated data in each training domain is mn > mm > usps. However, when a domain is selected as the test unseen domain, the generator of this domain will not be distributed, and all this domain data is only used for testing. For example, when the mn data set is unseen, only the generator of mm and usps are deployed to the original syn, svhn, usps and mm domains. In PACS, FA-FDG uses the generators from P and S to realize data augmentation, and the S generator provides a bit more data than the P generator. When the unseen domain is the S, only the P generator is given to A, P, and C.

Figure 13 gives the FDG model accuracy and RMSE, separately on Digit-Five classification and VNF profiling regression. The label 'FA-FDG_mn' in Figure 13 (a) indicates that the mn acts as the unseen test domain, then mm, usps, syn, and svhn participate in FL as four individual clients, with GANs data augmentation under the guidance of GAD. The generated mm data and a small amount of usps data exist in four FL clients under the label 'FA-FDG_mn'. The experiment shows that the 'FA-FDG_mn' has better FDG accuracy on the test unseen mn data set than the 'FDG_mn'. The 'FDG_mn' label in Figure 13 (a) means no GAN architecture with synthetic data added, the mm, usps, syn, and svhn directly do the FL and test the model on the mn (the green square line). The leftover 'FA-FDG_xx' and 'FDG_xx' can be explained the same as above. In Figure 13 (b), VNF Snort Inline acts as the unseen domain, 'FA-FDG_LC' means that we use Snort Passive VNF domain and vFW domain as FL clients, the generated Snort Passive data is added to two FL clients. The enriched data will decrease the FL model RMSE

TABLE 6. The average classification accuracy on digit-five data set.

Paradigm	Method	Backbone	Digital-Five					
			mn	mm	usps	svhn	syn	Avg.
Centralized	MMD [25]	Resnet18	97.84±0.02	69.77±0.11	94.26±0.24	68.32±0.16	**81.96**±0.16	82.37
	CORAL [26]	Resnet18	97.77±0.02	69.85±0.10	94.48±0.08	68.35±0.18	81.07±0.13	82.30
	DANN [59]	Resnet18	96.01±0.03	67.75±0.03	93.18±0.05	**70.11**±0.09	80.60±0.04	81.53
	RSC [60]	Resnet18	97.71±0.02	67.20±0.14	**94.79**±0.08	68.52±0.23	79.52±0.13	81.55
	Ours	Resnet18	**98.10**±0.04	**69.89**±0.16	94.24±0.20	68.96±0.12	81.54±0.04	**82.55**
Federated	Fed-Avg [2]	CNN	90.61	66.08	92.54	**56.29**	82.63	77.63
	Fed-Prox [6]	CNN	89.99	63.24	**93.47**	54.70	82.26	76.73
	Fed-Yogi [27]	CNN	80.87	61.14	82.72	48.96	72.75	69.29
	FA-FDG	CNN	**95.73**	**68.74**	93.11	55.04	**82.91**	**79.11**

on the test unseen Snort Inline domain. LC is the resource configuration of Snort Inline VNF that we need to predict. The label 'FDG_LC' in Figure 13 (b) has no generated data during the FL. The leftover CPU and MEM also be predicted. It is clear in Figure 13 (b) that the data augmentation approach (FA-FDG_xx) has lower RMSE when predicting new VNF resources. In addition, the FA-FDG in Table 8 also indicates that the FL clients already have the data from GANs, whereas Fed-Avg and Fed-Prox are only different aggregation methods without synthetic data. The RMSE, MAE and R-squared in Table 8 show the performance improvement of the regression FL model with the proposed FA-FDG on the unseen domain.

Considering the computation cost, DG or FDG is considered as high computation cost applications, like NVIDIA V100 GPU [61], RTX20280ti [43] and NVIDIA A100GPUs in [50]. Table 5 gives the computation cost result with/without the GAN training, from the aspect of CPU usage, GPU usage, training time (GAN or FL) and used RAM. The computation cost is tested under the mn as the unseen domain, the mm, usps, syn, and svhn act as four FL clients. After 10 global aggregations in FL, the model is tested on the mn data set. The first row of Table 5 is the traditional FL with no GAN architecture, only training with the raw domain data as a benchmark. The second row with the 'FA(GAN)' column of Table 5 however, first training GANs to get the good generators from mm, usps and syn. Then the augmented data are concatenated as tensors to each FL domain data, based on the GAD list. Finally, FL runs with the four clients as the content of the 'FA (GAN)+FL' row with the 'FL' column in Table 5. Traditional FL gets the FDG accuracy of 93.1%, and 'FA(GAN)+FL' have a 96.8% accuracy. The results indicate that the system's computational cost has increased, but the FDG accuracy has improved by 3.7%.

Here we also give the FDG model accuracy on the Digit-Five data set after data augmentation with/without GAD, as shown in Figure 14. The label 'w-GAD_mn' means that mn is set as the test unseen domain, and the rest domains participate in FL after GAD data augmentation. Following each global aggregation E_2 on the FL server, the FDG model is tested on the unseen mn domain. Whereas 'wo-GAD_mn' means the rest four domains except mn have uniform data

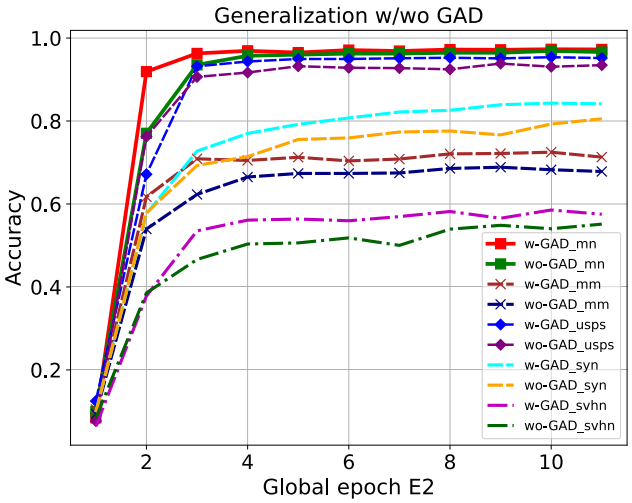


FIGURE 14. The FDG model accuracy on Digit-Five, after data augmentation with/without GAD.

augmentation, with the same amount of generated mm, usps, syn and svhn data in each FL client. Then these four domains do FL and test the aggregated model on the mn. The leftover 'w-GAD_xx' and 'wo-GAD_xx' methods are the same process. It is clear in Figure 14 that the GAD targeted data augmentation improves the FDG model accuracy more than uniform data enhancement. This also suggests that enhancing generalization performance not only simply increases samples, especially when the test domain is unseen. Rather, it should focus on augmenting samples from domains that exhibit strong generalization abilities.

Table 6 and Table 7 show the classification accuracy of the model under leave-one-domain-out testing. In Table 6, when the mn data set is selected as the test unseen domain, this data set is not involved in training, but the remaining domain data sets are used to train the model in a centralized or federated manner. The trained model is tested on the mn data set, as shown in the 'mn' column of Table 6. The same test approach is on the PACS data set. From the test results in Table 6 and 7, the proposed FA-FDG method improves the average classification accuracy in centralized and federated training mode. The reason for the performance improvement

TABLE 7. The average classification accuracy on PACS data set.

Paradigm	Method	Backbone	PACS				
			A	C	P	S	Avg.
Centralized	MMD [25]	Resnet18	81.30±0.25	73.42±0.43	92.93±0.30	79.23±0.84	81.72
	CORAL [26]	Resnet18	80.32±0.28	73.25±0.58	92.99±0.24	80.07±0.51	81.66
	DANN [59]	Resnet18	**82.71±0.19**	75.51±0.34	95.51±0.06	72.92±0.41	81.66
	RSC [60]	Resnet18	80.86±0.93	73.93±0.56	**95.69±0.30**	75.62±0.15	81.53
	Ours	Resnet18	82.24±0.06	**75.85±0.60**	93.93±0.21	**80.66±0.67**	**83.17**
Federated	Fed-Avg [2]	Resnet18	**77.38**	72.90	91.85	**76.47**	79.65
	Fed-Prox [6]	Resnet18	76.85	73.01	90.75	74.05	78.67
	Fed-Yogi [27]	Resnet18	73.28	70.74	91.09	71.23	76.59
	FA-FDG	Resnet18	77.19	**73.78**	**93.69**	74.26	**79.73**

TABLE 8. Evaluation metrics for generalized configuration of VNF resources.

	Method	Fed-Avg			Fed-Prox			**FA-FDG**		
VNF Types	Metrics	CPU	MEM	LC	CPU	MEM	LC	CPU	MEM	LC
SNORT Inline	RMSE	0.211	0.141	0.207	0.219	0.124	0.155	**0.193**	**0.096**	**0.120**
	MAE	0.187	0.125	0.116	0.228	0.081	0.134	**0.128**	**0.066**	**0.081**
	R-squared	0.408	0.203	0.204	0.361	0.298	0.301	**0.517**	**0.424**	**0.429**
SNORT Passive	RMSE	0.102	0.078	0.089	0.095	0.071	0.088	0.113	0.077	0.081
	MAE	0.088	0.063	0.077	0.087	0.069	0.060	0.086	0.068	0.067
	R-squared	0.781	0.626	0.660	0.753	0.689	0.695	0.769	0.679	0.677
vFW	RMSE	0.220	0.087	0.117	0.184	0.086	0.156	**0.174**	0.092	**0.095**
	MAE	0.196	0.075	0.092	0.161	0.079	0.142	0.162	0.077	**0.079**
	R-squared	0.353	0.499	0.529	0.427	0.588	0.446	**0.498**	0.461	**0.666**

TABLE 9. Evaluation metrics of VNFs.

	Method	Fed-Avg		
VNF Types	Metrics	CPU	MEM	LC
SNORT Inline	RMSE	0.152	0.112	0.127
	MAE	0.147	0.083	0.120
	R-squared	0.526	0.464	0.329
SNORT Passive	RMSE	0.086	0.078	0.111
	MAE	0.075	0.061	0.091
	R-squared	0.758	0.530	0.548
vFW	RMSE	0.184	0.075	0.063
	MAE	0.163	0.046	0.057
	R-squared	0.554	0.618	0.781

is that we augment the data from domains with strong generalization ability in a targeted manner.

Meanwhile, the FL results in Table 6 indicate that even though sometimes the trained model for FDG is simple, the performance of FL can still get close to the centralized training accuracy by optimizing the data structure through insights from FA. This experiment is also a good example of how FA can improve the performance of FL. Considering the training time, federated CNN costs less time than the centralized ResNet-18 model. Sometimes centralized training can be very time-consuming, whereas FL distributes data across different clients, only transmitting model parameters for aggregation. This decentralized approach can achieve competitive results with simple models in a shorter time. In FDG, the unseen test domain is placed on the server side. After each round of aggregation, the model is directly tested on the unknown domain. Among the selected FL aggregation strategies (Fed-Avg, Fed-Prox, Fed-Yogi), we also find that Fed-Yogi and Fed-Prox give sub-optimal prediction results. This is because the momentum hyper-parameter of the Yogi optimizer (momentum, second moment) needs fine-tuning, and the proximal term of Fed-Prox also requires adjustment based on the data and model.

In VNF profiling scenarios, Figure 15 demonstrates the setup with a Snort VNF instance [15]. This figure gives the connection between the profiled VNF, the traffic generator tool and server end-point machines (iPerf client and iPerf server). iPerf yields a series of input traffic records for each VNF, which depends on the specific VNF types. Then iPerf records different profiling data sets for Snort Inline mode, Snort Passive mode, and vFW.

Table 8 shows the results of FDG considering the regression problem. The leave-one-domain-out strategy is still used, with two VNFs as FL clients with the augmented tabular data, then the trained model is tested on the rest unseen VNF. In Table 9, all three types of VNF participate in FL,

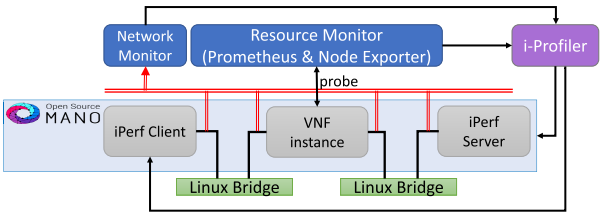


FIGURE 15. **VNF setup.**

the aggregated model is tested on the corresponding VNF without FA-FDG. In Table 8, models generated by different FL aggregation methods are tested on the target VNF type. Considering the Table 4, we only train the data generators of Snort Passive and Snort Inline modes. For example, for Inline VNF, only Snort Passive VNF and vFW VNF participate in FL. After several rounds of iterations, the aggregated model is tested on Inline VNF data. The same testing method is used for the other two VNFs.

From the CPU columns and LC columns of different FL aggregation methods in Table 8, the proposed FA-FDG method outperforms the Fed-Avg and Fed-Prox in the generalized prediction of CPU and LC. FA-FDG also achieved competitive results in generalization to predict MEM. For the row of SNORT Passive, the generalization of SNORT Passive VNF does not reach optimal results. This is because vFW and SNORT Passive are similar in function [15], and vFW data is not enhanced according to the GAD list in Table 4. Only data augmentation of the SNORT Inline mode does not significantly improve the model's generalization performance when testing on the Snort Passive mode. Additionally, by comparing the results of Fed-Avg in Table 8 and Table 9, it can be observed that the model jointly trained by three VNF types in Table 9 performs slightly better than the model trained by the other two VNF types for generalization in Table 8. This phenomenon also indicates that the richness of data indirectly affects the performance of FDG model. When the data distribution of the unseen domain is close to the training domains, the generalization results may be better. The selection of federated users with rich data is also a promising research area in terms of FDG.

VII. CONCLUSION

Future network optimizers require self-generation of data, self-training of models and automatic performance improvement. FL and generative AI will play an important role in the next generation of networks to enhance the generalization ability of AI models when facing unseen scenarios. In this article, a new distributed data augmentation method called FA-FDG is proposed. First, FA-FDG collects data generators from local GANs. Then, a domain relationship evaluation called GAD was designed by FA. In the end, generators with the GAD list are deployed to the FL clients to realize targeted data augmentation. The experimental results showed that our method was able to efficiently get the data insight and improve the generalization ability of the FL model. The proposed FA-FDG also gives an option of optimizing test-domain validation set (oracle) in [58] because we can generate some 'raw data' from the target domain to find some latent combination of hyper-parameters of models. However, some limitations of this work in FDG still need to be improved.

1) The right selection of the GANs for FA is important. CGAN [62], InfoGAN [63] and generative diffusion model [64] can be expanded. More privacy data augmentation methods for FL could be explored [65], [66].

2) CausIRL [67] regularization can be extended in MMD and CORAL in federated deep domain generalization.

3) More data from some network infrastructures may collected as time series data, TimeGAN [68] could be explored more in FDG for future networks. More complicated work is ongoing.

REFERENCES

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, A. Singh and J. Zhu. Eds., Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.

[3] R. Daniel and M. Stefano. (May 27, 2020). *Federated Analytics: Collaborative Data Science Without Data Collection. Google Research*. [Online]. Available: https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html

[4] C. Beauville. (Jan. 24, 2023). *Federated Analytics With Flower and Pandas*. [Online]. Available: https://flower.dev/blog/2023-01-24-federated-analytics-pandas/

[5] D. Wang, S. Shi, Y. Zhu, and Z. Han, "Federated analytics: Opportunities and challenges," *IEEE Netw.*, vol. 36, no. 1, pp. 151–158, Jan. 2022.

[6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[7] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," 2019, *arXiv:1905.10497*.

[8] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[9] J. Wang et al., "Generalizing to unseen domains: A survey on domain generalization," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8052–8072, May 2022.

[10] M. Akrout, A. Feriani, F. Bellili, A. Mezghani, and E. Hossain, "Domain generalization in machine learning models for wireless communications: Concepts, State-of-the-art, and open issues," 2023, *arXiv:2303.08106*.

[11] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4396–4415, Apr. 2023.

[12] Y. Li et al., "Federated domain generalization: A survey," 2023, *arXiv:2306.01334*.

[13] R. Zhang, Q. Xu, J. Yao, Y. Zhang, Q. Tian, and Y. Wang, "Federated domain generalization with generalization adjustment," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 3954–3963.

[14] N. Truong, K. Sun, S. Wang, F. Guitton, and Y. Guo, "Privacy preservation in federated learning: An insightful survey from the GDPR perspective," *Comput. Secur.*, vol. 110, Nov. 2021, Art. no. 102402.

[15] S. Moazzeni et al., "A novel autonomous profiling method for the next-generation NFV orchestrators," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 642–655, Mar. 2021.

[16] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data," 2018, *arXiv:1811.11479*.

[17] R. Venugopal, N. Shafqat, I. Venugopal, B. M. J. Tillbury, H. D. Stafford, and A. Bourazeri, "Privacy preserving generative adversarial networks to model electronic health records," *Neural Netw.*, vol. 153, pp. 339–348, Sep. 2022.

[18] Z. Li, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Federated learning with GAN-based data synthesis for non-IID clients," in *Proc. Int. Workshop Trustworthy Federated Learn.* Cham, Switzerland: Springer, 2022, pp. 17–32.

[19] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Supplementary material geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Providence, RI, USA: IEEE Computer Society and the Computer Vision Foundation, Jun. 2012. [Online]. Available: https://ieeexplore.ieee.org/xpl/conhome/6235193/proceeding

[20] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2066–2073.

[21] D. J. Beutel et al., "Flower: A friendly federated learning research framework," 2020, *arXiv:2007.14390*.

[22] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang, "Fate: An industrial grade platform for collaborative learning with data protection," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 10320–10325, Jan. 2021.

[23] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" 2019, *arXiv:1911.07963*.

[24] A. Ziller et al., "PySyft: A library for easy federated learning," in *Federated Learning Systems: Towards Next-Generation AI*, M. H. U. Rehman and M. M. Gaber, Eds. Cham, Switzerland: Springer, 2021, pp. 111–139.

[25] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," 2014, *arXiv:1412.3474*.

[26] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *Proc. Comput. Vis. (ECCV)*, Amsterdam, The Netherlands. Cham, Switzerland: Springer, Oct. 2016, pp. 443–450.

[27] S. Reddi et al., "Adaptive federated optimization," 2020, *arXiv:2003.00295*.

[28] J. M. Parra-Ullauri et al., "Privacy preservation in kubernetes-based federated learning: A networking approach," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2023, pp. 1–7.

[29] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[30] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang, and D. Niyato, "Federated learning for 6G communications: Challenges, methods, and future directions," *China Commun.*, vol. 17, no. 9, pp. 105–118, Sep. 2020.

[31] Z. Yang, M. Chen, K. K. Wong, H. V. Poor, and S. Cui, "Federated learning for 6G: Applications, challenges, and opportunities," *Engineering*, vol. 8, pp. 33–41, Jan. 2022.

[32] A. R. Elkordy et al., "Federated analytics: A survey," *APSIPA Trans. Signal Inf. Process.*, vol. 12, no. 1, pp. 1–33, 2023.

[33] Z. Wang, Y. Zhu, D. Wang, and Z. Han, "Federated analytics informed distributed industrial IoT learning with non-IID data," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2924–2939, Oct. 2023.

[34] W. Zhu, P. Kairouz, B. McMahan, H. Sun, and W. Li, "Federated heavy hitters discovery with differential privacy," in *Proc. 23rd Int. Conf. Artif. Intell. Stat.*, 2020, pp. 3837–3847.

[35] X. Zhang, A. Mavromatics, A. Vafeas, R. Nejabati, and D. Simeonidou, "Federated feature selection for horizontal federated learning in IoT networks," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 10095–10112, Jun. 2023.

[36] Z. Wang, Y. Zhu, D. Wang, and Z. Han, "FedACS: Federated skewness analytics in heterogeneous decentralized data environments," in *Proc. IEEE/ACM 29th Int. Symp. Quality Service (IWQOS)*, Jun. 2021, pp. 1–10.

[37] J. M. Parra-Ullauri et al., "Federated analytics for 6G networks: Applications, challenges, and opportunities," *IEEE Netw.*, early access, Jan. 17, 2024, doi: 10.1109/MNET.2024.3355218.

[38] M. Rasouli, T. Sun, and R. Rajagopal, "FedGAN: Federated generative adversarial networks for distributed data," 2020, *arXiv:2006.07228*.

[39] W. Li, J. Chen, Z. Wang, Z. Shen, C. Ma, and X. Cui, "IFL-GAN: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10502–10515, Dec. 2023.

[40] J. Chen, M. Jiang, Q. Dou, and Q. Chen, "Federated domain generalization for image recognition via cross-client style transfer," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2023, pp. 361–370.

[41] R. Bai, S. Bagchi, and D. I. Inouye. (2023). *Benchmarking Algorithms for Domain Generalization in Federated Learning*. [Online]. Available: https://openreview.net/forum?id=IsCg7qoy8i9

[42] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 1013–1023.

[43] L. Zhang, X. Lei, Y. Shi, H. Huang, and C. Chen, "Federated learning for IoT devices with domain generalization," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 9622–9633, Jun. 2023.

[44] J. Zhang, J. Zhang, J. Chen, and S. Yu, "GAN enhanced membership inference: A passive local attack in federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[45] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–7.

[46] E. Ekblom, E. L. Zec, and O. Mogren, "EFFGAN: Ensembles of fine-tuned federated GANs," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2022, pp. 884–892.

[47] M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan, "Multi-component image translation for deep domain generalization," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 579–588.

[48] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[49] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–9.

[50] A. T. Nguyen, P. Torr, and S. N. Lim, "FedSR: A simple and effective domain generalization method for federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 38831–38843.

[51] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1406–1415.

[52] C. Hardy, E. Le Merrer, and S. Bruno, "MD-GAN: Multi-discriminator generative adversarial networks for distributed datasets," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, May 2019, pp. 866–877.

[53] M. Rezaei, J. J. Näppi, C. Lippert, C. Meinel, and H. Yoshida, "Generative multi-adversarial network for striking the right balance in abdominal image segmentation," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 15, no. 11, pp. 1847–1858, Nov. 2020.

[54] Z. Zhang, H. Chen, S. Li, Z. An, and J. Wang, "A novel geodesic flow kernel based domain adaptation approach for intelligent fault diagnosis under varying working condition," *Neurocomputing*, vol. 376, pp. 54–64, Feb. 2020.

[55] A. Samat, P. Gamba, J. Abuduwaili, S. Liu, and Z. Miao, "Geodesic flow kernel support vector machine for hyperspectral image classification by unsupervised subspace feature transfer," *Remote Sens.*, vol. 8, no. 3, p. 234, Mar. 2016.

[56] J. Wang and Y. Chen, *Introduction to Transfer Learning: Algorithms and Practice*. Cham, Switzerland: Springer, 2023.

[57] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Co-clustering based classification for out-of-domain documents," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2007, pp. 210–219.

[58] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," 2020, *arXiv:2007.01434*.

[59] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 2. Lile, France: International Machine Learning Society (IMLS), Jul. 2015, pp. 1180–1189.

[60] Z. Huang, H. Wang, E. P. Xing, and D. Huang, "Self-challenging improves cross-domain generalization," in *Proc. Comput. Vis. ECCV, 16th Eur. Conf.*, Glasgow, U.K. Cham, Switzerland: Springer, Aug. 2020, pp. 124–140.

[61] X. Jiang, J. Huang, S. Jin, and S. Lu, "Domain generalization via balancing training difficulty and model capability," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 18993–19003.

[62] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1125–1134.

[63] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–4.

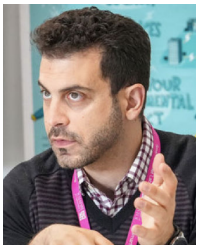
[64] F. V. Stanley Jothiraj and A. Mashhadi, "Phoenix: A federated generative diffusion model," 2023, *arXiv:2306.04098*.

[65] B. Xin, W. Yang, Y. Geng, S. Chen, S. Wang, and L. Huang, "Private FL-GAN: Differential privacy synthetic data generation based on federated learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Aug. 2020, pp. 2927–2931.
[66] M. Shin, C. Hwang, J. Kim, J. Park, M. Bennis, and S.-L. Kim, "XOR mixup: Privacy-preserving data augmentation for one-shot federated learning," 2020, *arXiv:2006.05148*.
[67] M. Chevalley, C. Bunne, A. Krause, and S. Bauer, "Invariant causal mechanisms through distribution matching," 2022, *arXiv:2206.11646*.
[68] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.

XUNZHENG ZHANG (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in communication engineering from Shandong University, China. He is currently pursuing the Ph.D. degree with the Smart Internet Laboratory, High-Performance Networks Research Group, University of Bristol, U.K. He has experience with the Internet of Things and network test-bed deployment under Industry 4.0. He was a recipient of the Best Paper Award from the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications 2020, London, U.K. He has also served as a reviewer for IEEE Internet of Things Journal, *IEEE Internet of Things Magazine*, and IEEE Transactions on Knowledge and Data Engineering. His current research interests include 6G Internet of Things, federated learning, and future network optimization with generative AI.

JUAN MARCELO PARRA-ULLAURI (Member, IEEE) received the bachelor's degree in electronics and telecommunications engineering from the University of Cuenca, Ecuador, in 2017, and the Ph.D. degree in computer science from Aston University, U.K., in 2023. He is currently a Senior Research Associate with the Smart Internet Laboratory. His research interests include the Internet of Things, distributed machine learning, cloud computing, explainability in autonomous systems, and data engineering.

SHADI MOAZZENI (Member, IEEE) received the M.Sc. degree in computer architecture engineering from the Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2010, and the Ph.D. degree in computer architecture engineering from the University of Isfahan, Iran, in 2018. From July 2016 to February 2017, she was a Ph.D. Visiting Researcher with the University of Bologna, Italy. She is currently a Research Fellow with the University of Bristol, Bristol, U.K., where she is a member of the Smart Internet Laboratory and the High-Performance Networks Research Group. She is also the Cluster Lead Researcher of the EU Horizon 2020 5G-VICTORI project and the Deputy Leader of WP5 in the U.K.-funded Project REASON. Her current research interests include next-generation intelligent network orchestration, multi-access edge computing, 6G networks intelligent profiling, and orchestration toward zero-touch network and service management.



XENOFON VASILAKOS (Member, IEEE) received the M.Sc. degree in parallel and distributed computer systems from Vrije Universiteit Amsterdam and the Ph.D. degree in informatics from Athens University of Economics and Business (AUEB), Athens. He is currently a Lecturer in AI for digital infrastructures with the University of Bristol and a member of Bristol Digital Futures Institute (BDFI) and the Smart Internet Laboratory. He has participated in various EU and national research projects, as well as industry-funded projects. His current research interests include 6G architectures aiming zero touch networking.

REZA NEJABATI (Senior Member, IEEE) is currently the Chair Professor of Intelligent Networks and the Head of the High-Performance Network Group, Department of Electrical and Electronic Engineering, University of Bristol, U.K. He is also a Visiting Professor and the Cisco Chair of the Cisco Center for Intent-Based Networking, Curtin University, Australia. He has established successful and internationally recognized experimental research activities in "Autonomous and Quantum Networks." Building on his research, he co-founded a successful start-up company (Zeetta Networks Ltd.) with 25 employees and £6 million funding. His research received the prestigious IEEE Charles Kao Award in 2016 and has made important contributions to 5G, smart city, quantum communication, and future Internet experimentation.

DIMITRA SIMEONIDOU (Fellow, IEEE) is currently a Full Professor with the University of Bristol, the Co-Director of Bristol Digital Futures Institute, and the Director of the Smart Internet Laboratory. She has been the Technical Architect and the CTO of the Smart City Project Bristol is Open. She is currently leading Bristol City/Region 5G Urban Pilots. She has authored and coauthored over 600 publications, numerous patents, and several major contributions to standards. She has been the co-founder of two spin-out companies, the latest being the University of Bristol VC-funded spin-out Zeetta Networks, delivering SDN solutions for enterprise and emergency networks. Her research interests include high-performance networks, programmable networks, wireless-optical convergence, 5G/6G, and smart city infrastructures. She is increasingly working with social sciences on topics of digital transformation for society and businesses. She is a fellow of the Royal Academy of Engineering and a Royal Society Wolfson Scholar.