FedAlign: Federated Domain Generalization with Cross-Client Feature Alignment

Sunny Gupta, Vinay Sutar, Varunav Singh, Amit Sethi
Indian Institute of Technology Bombay, Powai, Mumbai
{sunnygupta, 21d070078, 21d070086, asethi}@iitb.ac.in

Abstract

*Federated Learning (FL) offers a decentralized paradigm for collaborative model training without direct data sharing, yet it poses unique challenges for Domain Generalization (DG), including strict privacy constraints, non-i.i.d. local data, and limited domain diversity. We introduce **FedAlign**, a lightweight, privacy-preserving framework designed to enhance DG in federated settings by simultaneously increasing feature diversity and promoting domain invariance. First, a cross-client Representation enrichment module broadens local domain representations through domain-invariant feature perturbation and selective cross-client feature transfer, allowing each client to safely access a richer domain space. Second, a dual-stage alignment module refines global feature learning by aligning both feature embeddings and predictions across clients, thereby distilling robust, domain-invariant features. By integrating these modules, our method achieves superior generalization to unseen domains while maintaining data privacy and operating with minimal communication overhead. We present state-of-the-art results on popular and diverse Domain Generalization datasets. The code is available at https://github.com/sunnyinAI/FedALign.*

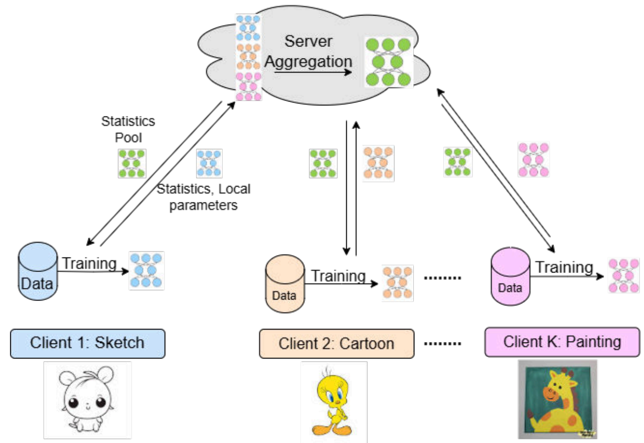

Figure 1. Illustration of the typical scenario in FL. Each client contains data from a unique domain, and the test domain (Photo) differs from all domains present on the clients.

1. Introduction

Conventional machine learning techniques are built on the assumption that training and test data are identically and independently distributed (IID). However, this assumption is often violated in real-world applications where models frequently encounter Out-of-Distribution (OOD) data, leading to significant performance degradation on unseen domains [16]. For instance, a model trained on cartoon images may fail to generalize to sketches due to domain shifts. **Domain Generalization (DG)** aims to address this limitation by equipping models with the ability to generalize effectively to unseen data distributions [36].

Despite the promise of DG, many existing approaches depend on centralized datasets, a condition that is infeasible in scenarios where data is distributed across multiple clients. **Federated Learning (FL)** [10] provides a decentralized alternative by enabling collaborative model training without exposing raw data. However, integrating DG within FL poses unique challenges, including limited domain diversity at the client level and stringent privacy constraints inherent in decentralized environments.

Federated Domain Generalization (FDG) focuses on learning domain-invariant features—label-relevant attributes that remain stable across diverse domains. Current FDG approaches predominantly employ two main strategies: *Federated Adversarial learning* [11, 15, 27, 33] and *Federated Representation Alignment* [13, 30, 33].

The first method uses adversarial objectives to align representations, but they often incur high computational overhead and can suffer from training instabilities such as model collapse [1]. On the other hand, by aligning features across clients in representation alignment, this line of work aims to mitigate domain discrepancies. However, limited domain

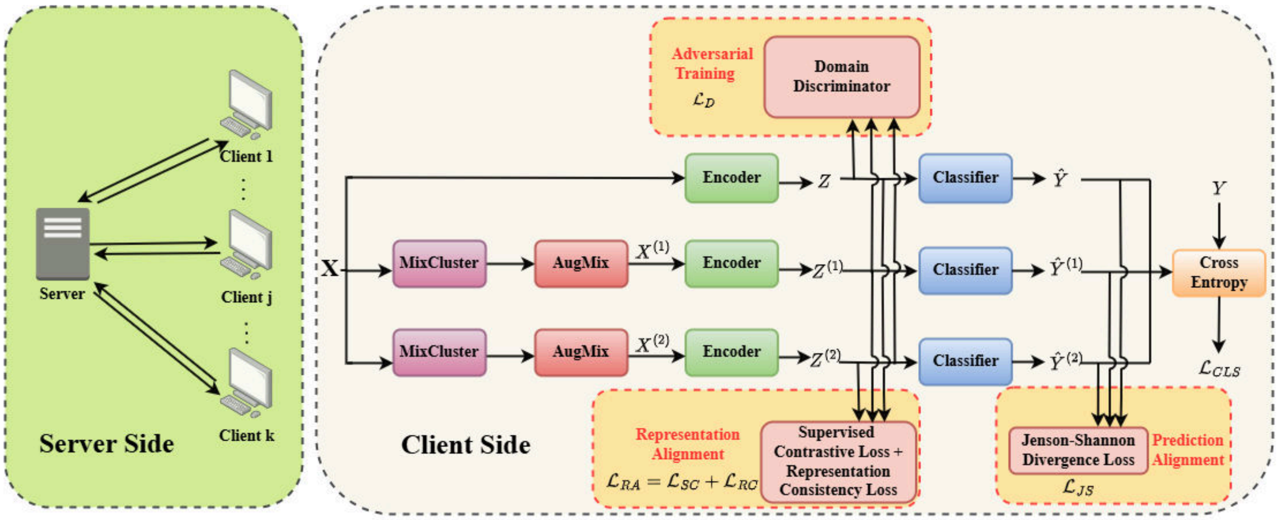

Figure 2. Overview of FedAlign: Clients share local model parameters and sample statistics with the server, which aggregates and redistributes them. Local training incorporates style transfer and feature augmentation followed by representation alignment and prediction alignment to enhance domain-invariant feature learning, followed by cross-entropy loss incorporated for general classification, and Domain Discriminator for adversarial training on the augmented representations.

diversity at the client level and strict privacy constraints can hinder alignment effectiveness at scale.

An alternative solution space involves *Federated Style Transfer* [29, 31], which augments local data diversity via techniques like AdaIN [7] and CycleGAN [37]. While effective at generating domain-varied samples, these approaches often demand additional models for feature extraction and high-dimensional embedding exchanges.

To address these challenges, we propose **FedAlign**, a federated domain generalization framework: FedAlign introduces a novel feature-sharing mechanism that enriches each client's domain exposure without revealing raw data. This strategy perturbs domain-invariant features and redistributes them across clients in a privacy-preserving manner, broadening the effective training distribution while upholding confidentiality.

A two-step alignment process ensures consistent performance across varied domains: Supervised Contrastive Loss encourages representations of samples with identical labels to converge, effectively reducing intra-class variance across domains. Jensen–Shannon Divergence enforces prediction consistency by aligning output distributions for both original and perturbed data, further bolstering out-of-distribution robustness.

By focusing on privacy-preserving feature transfers and a dual-stage alignment of representations and predictions, FedAlign addresses the critical limitations of existing FDG methods specifically, the interplay of limited local data, insufficient domain diversity, and strict privacy constraints.

2. Related Work

2.1. Representation Alignment

A prominent line of research in **Domain Generalization (DG)** focuses on representation alignment—reducing domain-specific variations by aligning feature distributions across multiple domains. Notable examples include:

Approaches like DANN [3–5] deploy a domain classifier to guide alignment, ensuring the extracted features are domain-invariant. By training the feature extractor and domain classifier in an adversarial manner, these methods successfully mitigate domain discrepancies.

CORAL [19] aligns second-order statistics (e.g., covariance matrices) between source and target feature distributions, thereby reducing mismatches in feature representations. Methods grounded in Maximum Mean Discrepancy (MMD) [20, 24, 25] leverage kernel-based metrics to align representations across domains, promoting feature embeddings.

Although these alignment techniques have demonstrated improved performance on unseen domains, they commonly assume centralized access to all training domains conflicting with the **FL** framework.

2.2. Style Transfer

A range of style transfer-based domain generalization (DG) methods [22, 23, 28] aim to enrich domain diversity, thereby improving model robustness on unseen target domains. These approaches can be broadly separated into two main categories:

In the first category, generative models are employed to synthesize data with diverse styles [14, 17]. By enhancing variability in color, texture, and other visual attributes, these methods reduce reliance on domain-specific features. However, generative modeling often demands substantial computational resources and can encounter training instability—including model collapse in adversarial training—thereby jeopardizing convergence and overall performance.

A second line of work leverages data augmentation techniques such as MixStyle [35] and Mixup [32]. Instead of synthesizing entirely new samples, these methods manipulate existing data to boost intra-batch diversity. Specifically: MixStyle interpolates channel-wise style statistics within a batch, promoting domain-invariant feature learning. Mixup merges data samples and their corresponding labels to expand the decision boundary.

Compared to generative approaches, augmentation-based methods are more computationally efficient and inherently free from adversarial training instability, rendering them particularly suitable for large-scale DG applications.

2.3. Federated Domain Generalization

Most Existing *Federated Domain Generalization (FDG)* methods focus on learning domain-invariant representations across heterogeneous clients, but face significant challenges:

Approaches like FedADG [33] use a global discriminator to extract universal features while preserving privacy. However, they often suffer from high computational costs and training instability, including potential model collapse. Methods such as FedSR [13] employ L2-norm and conditional mutual information regularization to align feature distributions. Yet, they struggle in large-scale settings due to limited domain diversity at individual clients, hindering out-of-distribution generalization. To address limited local diversity, CCST [2] uses cross-client style transfer based on AdaIN [7]. While it expands training distributions, it relies on pre-trained VGG networks [18], introducing high communication overhead and privacy risks [9, 12]. Additionally, pre-trained networks may inadvertently include target domains, conflicting with domain generalization principles. Alternative approaches include: FedIIR [6], which aligns client gradients to learn domain-invariant relationships, FedCCRL [26], which uses feature alignment to improve domain-invariant feature learning and GA [34], which dynamically adjusts aggregation weights to minimize client performance disparities.

Despite these advancements, they lack consistent superiority in empirical evaluations, highlighting ongoing performance and scalability challenges in FDG.

Overall, existing methods often overlook the dual constraints of limited data volume and domain diversity at the client level. Their reliance on high-dimensional data exchange or costly adversarial training underscores the need for efficient, privacy-preserving, and robust FDG solutions—a gap that FedAlign aims to address.

3. Methodology

3.1. Preliminary

Federated Domain Generalization (FDG) aims to train models collaboratively across multiple clients, where each client holds data from distinct domains. The goal is to develop a global model that generalizes effectively to unseen target domains without direct access to their data. Let X and Y denote the input images and target domains, respectively. Consider M source domains:

$$S_{\text{source}} = \{S_i \mid i = 1, 2, \dots, M\}, \tag{1}$$

with each domain sampled from a unique joint distribution $P_i(x, y)$, where $x \in X$ and $y \in Y$. These distributions differ significantly across domains, such that $P_i(x, y) \neq P_j(x, y)$ for $i \neq j$, reflecting real-world domain shifts in data distribution.

In a federated learning setting, data from each domain S_i is distributed across K clients, denoted as $D_k \subset S_i$. Each client performs local training on its private dataset and communicates only model updates or minimal statistics with a central server to preserve privacy. The objective of FDG is to collaboratively train a global model $f : X \to Y$ across $K.M$ data subsets that minimizes the prediction error on an unseen target domain S_{target}:

$$\min_f \mathbb{E}_{(x,y)\sim S_{\text{target}}} \left[\ell(f(x), y)\right], \tag{2}$$

where $\ell(\cdot)$ is a task-specific loss function, such as cross-entropy. Importantly, the unseen target domain S_{target} is inaccessible during training, and its joint distribution $P_{\text{target}}(x, y)$ differs from all source domain distributions $P_i(x, y)$, i.e., $P_{\text{target}}(x, y) \neq P_i(x, y)$ for all $i \in \{1, 2, \dots, M\}$.

To ensure privacy, the server is not allowed to directly access the data stored on the clients. Instead, after local training, each client uploads its model parameters to the server. The server then aggregates these parameters and distributes the updated model back to the clients.

Let θ_i^t represent the model parameters of client i at training round t. The server performs aggregation using the following weighted averaging formula:

$$\theta^{t+1} = \frac{1}{N} \sum_{i=1}^{K\cdot M} n_i \theta_i^t$$

where $N = \sum_{i=1}^{K\cdot M} n_i$ denotes the total number of data samples across all participating clients, and n_i represents the number of samples available on client i. Once aggregation is complete, the server transmits the updated parameters θ^{t+1} back to each client.

Algorithm 1 FedAlign

Require: Client datasets $\{D_k \mid k = 1, \dots, K\}$ where $D_k = \{(x_i, y_i)\}_{i=1}^{n_k}$, global model $f = g \circ h$ with encoder h and classifier g, communication rounds T, local epochs E, learning rate η

SERVER SIDE:
for $t = 1, \dots, T$ **do**
  Select a subset of clients C_t to participate.
  Broadcast global parameters θ_t to selected clients.
  for $k \in C_t$ **do**
    Receive updated client parameters $\theta_{k,t+1}$.
  end for
  Update global parameters: $(N = \sum_{k \in C_t} n_k)$

$$\theta_{t+1} = \frac{1}{N} \sum_{k \in C_t} n_k \theta_{k,t+1}$$

end for
Client Side:
Input global parameters θ_t.
for $e = 1, \dots, E$ **do**
  for a batch $X \in \mathbb{R}^{B \times C \times H \times W}$ **do**
    Generate augmented batches:

$$X^{(1)} = M(X), \quad X^{(2)} = M(X).$$

    Compute representations and predictions:

$$Z, Z^{(1)}, Z^{(2)} = h(X), h(X^{(1)}), h(X^{(2)}),$$

$$\hat{Y}, \hat{Y}^{(1)}, \hat{Y}^{(2)} = g(Z), g(Z^{(1)}), g(Z^{(2)}).$$

    Obtain domain features and domain labels:

$$Z_{\text{dom}} = \text{concat}(Z,\ Z^{(1)},\ Z^{(2)}).$$

$$d_{\text{orig}} = \mathbf{0}_{B \times 1}, \quad d_{\text{aug}} = \mathbf{1}_{2B \times 1},$$

$$d = \text{concat}(d_{\text{orig}},\ d_{\text{aug}}).$$

    Compute domain predictions: $\hat{d} = D(Z_{\text{dom}})$.
    Compute domain loss using BCE with logits:

$$L_{\text{D}} = \text{BCEWithLogitsLoss}(\hat{d},\ d).$$

    Compute total loss: $L = L_{\text{CLS}} + \lambda_1 L_{\text{RA}} + \lambda_2 L_{\text{JS}}$.
    Update local model on L using η.
  end for
end for
Return $\theta_{k,t+1}$ to the server.

Output: Global model θ_T after T rounds of communication.

3.2. Framework Overview

An illustration of the proposed FedAlign framework is provided in Fig. 2, and the detailed algorithmic steps can be found in Algorithm 1. Our approach integrates MixStyle-based cross-client feature augmentation along with Domain Adversarial Training for multi-level alignment objectives.

Client-Side Processing and Augmentation

Given a batch of samples $X \in \mathbb{R}^{B \times C \times H \times W}$, where B denotes the batch size, C the number of channels, and H and W the image height and width, respectively, we first apply Cross-Client Representation Enrichment (Sec. 3.3) to generate two additional augmented batches:

$$X^{(1)} = M(X), \quad X^{(2)} = M(X), \tag{3}$$

where $M(\cdot)$ represents the composition of MixCluster module (described in Algorithm 2) and AugMix (described in Algorithm 3) in the Cross-Client Representation Enrichment Module. The MixCluster module interpolates channel-wise statistics (mean and standard deviation) between two selected samples and AugMix applies multiple augmentation chains, each with diverse operations.

After Cross-Client Representation Enrichment, the FedAlign framework extracts representations and generates predictions for each of the augmented batches. Let Z represent the latent feature space. We decompose the model f into two components:

$$f = g \circ h, \tag{4}$$

where h represents the representation encoder and g is the classifier.

$$h : X \to Z, \quad g : Z \to \hat{Y}$$

The extracted features representations Z^1 and Z^2 and the output predictions $\hat{Y}^1$ and $\hat{Y}^2$ are as follows:

$$Z = h(X), \quad Z^{(1)} = h(X^{(1)}), \quad Z^{(2)} = h(X^{(2)}), \tag{5}$$

$$\hat{Y} = g(Z), \quad \hat{Y}^{(1)} = g(Z^{(1)}), \quad \hat{Y}^{(2)} = g(Z^{(2)}). \tag{6}$$

3.3. Cross-Client Representation Enrichment Module

To address the challenge of limited domain diversity in federated learning, we introduce an enhanced version of MixStyle, called MixCluster, a computationally lightweight strategy for capturing domain-specific features. While AugMix perturbs style information (e.g., color and texture). Together, these Representation Enrichment Modules effectively simulate additional, previously unseen domains, expanding the training data distribution and improving model resilience against domain shifts.

Consider a batch of input samples $X = \{x_i \mid i = 1, \dots, B\}$, where B is the batch size. For each sample x, MixStyle computes the channel-wise mean $\mu(x)$ and standard deviation $\sigma(x)$ as:

$$\mu(x)_c = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{c,h,w}, \tag{7}$$

$$\sigma(x)_c = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{c,h,w}-\mu(x)_c)^2}, \tag{8}$$

where H and W are the height and width of the feature map, and c indexes the channels. Along with the locally trained model parameters, each client also uploads the channel-wise statistics, namely the mean and the standard deviation (Eqs. (7), (8)).

Each client i uploads a certain fraction $0 < r < 1$ of sample statistics denoted as:

$$P_i = \{(\mu(x_j), \sigma(x_j)) \mid x_j \in D_i,\ j = 1, 2, \ldots, \lceil rn_i \rceil\}. \tag{9}$$

After collecting statistics from all clients, the server aggregates these statistics to form the statistics pool:

$$P_{\text{pool}} = \{P_i \mid i = 1, 2, \ldots, K \cdot M\}, \tag{10}$$

and subsequently distributes as $P^i_{\text{pool}} = P_{\text{pool}} \setminus P_i$ To each client i, the available statistics comprise both its own statistics and those collected from other clients, forming the complete statistics pool.

Following this, each client applies the Cross-Client Representation Enrichment Module, which consists of MixCluster (Algorithm2) and AugMix (Algorithm3).

The MixCluster algorithm enhances feature stylization through dynamic statistic sampling and adaptive mixing. The statistics pool (Eq.10) undergo K-means clustering into K groups, with sampling weights $\mathbf{w}$ computed per cluster based on intra-cluster variance to prioritize diverse patterns.

For each batch, features are instance-normalized ($\tilde{X}$), then supplemented by cluster-aware sampling: statistics are drawn proportionally to cluster sizes, with allocations $b_k = \lfloor B \cdot |\mathcal{P}_k|/|\mathcal{P}| \rfloor$ ensuring representative sampling. An adaptive mixing ratio λ combines source (μ_X, σ_X) and sampled (μ_2, σ_2) statistics, modulated by a difficulty score d derived from feature activation uncertainty ($d = 1 - \max(\text{softmax}(\mathbb{E}[\tilde{X}]))$). Final outputs combine normalized features with mixed statistics:

AugMix enhances the original sample x by passing it through multiple augmentation chains, where each chain applies a series of simple augmentation operations. The outputs from these chains are then blended using an element-wise convex combination, ensuring a diverse yet coherent transformation.

3.4. Adversarial Training

The domain discriminator D is trained to distinguish between original features (Z) and augmented features ($Z^{(1)}, Z^{(2)}$) through the following adversarial objective:

Algorithm 2 MixCluster

Require: Input batch $X \in \mathbb{R}^{B\times C\times H\times W}$, statistic pool $\mathcal{P}$, clusters K, $\alpha > 0$

Ensure: Stylized batch $\hat{X}$

MAINTAINSTATISTICPOOL($X, \mathcal{P}$)

$\{\mathbf{c}_k\}_{k=1}^{K}, \mathbf{L} \leftarrow$ KMEANS($\mathcal{P}_\mu \cup \mathcal{P}_\sigma$)

procedure COMPUTECLUSTERWEIGHTS($\mathcal{P}$, $\mathbf{L}$)

  $\mathbf{S} \leftarrow \text{concat}(\mathcal{P}_\mu, \mathcal{P}_\sigma)$

  $\mathbf{v} \leftarrow \text{Var}(\mathbf{S}, \text{dim} = 1)$

  $\mathbf{w}_k \leftarrow \mathbf{v}[\mathcal{I}_k]/\sum \mathbf{v}[\mathcal{I}_k]$ for $\mathcal{I}_k = \{i \mid \mathbf{L}_i = k\}$

  return $\mathbf{w}$

end procedure

$\mathbf{w} \leftarrow$ COMPUTECLUSTERWEIGHTS($\mathcal{P}, \mathbf{L}$)

$\mu_X, \sigma_X \leftarrow \mathbb{E}_{H,W}[X], \sqrt{\mathbb{V}_{H,W}[X] + \epsilon}$

$\tilde{X} \leftarrow (X - \mu_X)/\sigma_X$

procedure CLUSTERSAMPLE($\mathcal{P}$, K, $\mathbf{L}$, $\mathbf{w}$, B)

  $b_k \leftarrow \lfloor B \cdot |\{i : L_i = k\}|/|\mathcal{P}| \rfloor$ ▷ Proportional allocation

  return $\sum_{k=1}^{K}$ MultinomialSample($\mathcal{P}_k, \mathbf{w}_k, b_k$)

end procedure

$\mu_2, \sigma_2 \leftarrow$ CLUSTERSAMPLE($\mathcal{P}, K, \mathbf{L}, \mathbf{w}, B$)

$d \leftarrow 1 - \max(\text{softmax}(\mathbb{E}_{H,W}[\tilde{X}]))$ ▷ Difficulty score

$\lambda \leftarrow \sigma(kd)\text{Beta}(\alpha, \alpha)$

$\mu_{\text{mix}} \leftarrow \lambda\mu_X + (1-\lambda)\mu_2$

$\sigma_{\text{mix}} \leftarrow \lambda\sigma_X + (1-\lambda)\sigma_2$

$\hat{X} \leftarrow \tilde{X}\sigma_{\text{mix}} + \mu_{\text{mix}}$

Algorithm 3 AugMix

Require: A batch of samples X, operations base $\mathcal{O}$, mixing parameter $\beta > 0$

Ensure: A batch of augmented samples X_{augmix}

$\hat{X} \leftarrow [\,]$

for $x \in X$ **do**

  Initialize x_{aug} with zeros

  Uniformly sample $k \sim \{1, 2, 3\}$

  Sample $(w_1, w_2, ..., w_k) \sim \text{Dirichlet}(\beta, ..., \beta)$

  for $i = 1, ..., k$ **do**

    Sample operations $\text{op}_1, \text{op}_2, ..., \text{op}_m \sim \mathcal{O}$ with probabilities $\mathbf{p}_{\text{ops}}$

    Apply chain: $x_{\text{chain}} = (\text{op}_m \circ ... \circ \text{op}_1)(x)$

    Combine augmentations: $x_{\text{aug}} = x_{\text{aug}} + w_i \cdot x_{\text{chain}}$

  end for

  Compute difficulty score: $d = 1 - \max(\text{softmax}(\mathbb{E}[x]))$

  Sample mixing weight $m = \sigma(kd) \cdot \text{Beta}(\beta, \beta)$

  Combine: $\hat{x} = mx + (1-m)x_{\text{aug}}$

end for

$$\min_h \max_D \mathbb{E}_{z\sim Z}\left[\log D(z)\right] + \mathbb{E}_{z\sim\{Z^{(1)}, Z^{(2)}\}}\left[\log(1 - D(z))\right] \tag{11}$$

Implementation Details:

1. *Domain Feature Construction:* Concatenate representations from original and augmented views:

$$Z_{\text{dom}} = \text{concat}(Z, Z^{(1)}, Z^{(2)}) \in \mathbb{R}^{3B \times d} \tag{12}$$

2. *Label Assignment:* Create domain labels where original samples are labeled 0 and augmented samples 1:

$$d = \text{concat}(\mathbf{0}_{B\times 1}, \mathbf{1}_{2B\times 1}) \tag{13}$$

3. *Adversarial Loss:* Train D using binary cross-entropy with logits:

$$\mathcal{L}_D = \frac{1}{3B} \sum_{i=1}^{3B} d_i \log \sigma(\hat{d}_i) + (1 - d_i) \log(1 - \sigma(\hat{d}_i)) \tag{14}$$

where $\hat{d} = D(Z_{\text{dom}})$ are logits(raw model output before applying sigmoid) and σ is the sigmoid function.

The discriminator D employs dropout and batch normalization in its fully-connected layers to prevent overfitting to transient feature patterns during adversarial training.

3.5. Representation Alignment

The method encourages the alignment of feature representations Z, Z^1, Z^2 for samples that share the same class label. Supervised Contrastive Loss leverages label information to cluster samples from the same class closer together while simultaneously pushing samples from different classes apart in the latent space. This process fosters the learning of representations that are both discriminative and domain-invariant, enhancing generalization across diverse distributions.

Given two batches of feature representations Z', $Z'' \in \mathbb{R}^{B\times V}$ and their corresponding labels Y' and Y'' (Eq. 5, 6), we concatenate these representations and labels to construct the overall representation Z and overall label set Y for the full batch of size $2B$:

$$Z = \text{cat}(Z', Z'') \in \mathbb{R}^{2B\times V}, \tag{15}$$

$$Y = \text{cat}(Y', Y''). \tag{16}$$

Here, V represents the dimensionality of the feature representations. We define the index set as

$I = \{1, 2, \dots, 2B\}$, and for each index i, the set $A(i)$ is given by: $A(i) = I \setminus \{i\}$. This ensures that for each sample in the batch, $A(i)$ includes all other samples except the one at index i. The supervised contrastive loss $\bar{\mathcal{L}}_{\text{SC}}(Z)$ for the combined representation Z is then computed as follows:

$$\sum_{i\in I} -\frac{1}{|P(i)|} \sum_{p\in P(i)} \log \left(\frac{\exp(\text{sim}(Z_i, Z_p)/\tau)}{\sum_{a\in A(i)} \exp(\text{sim}(Z_i, Z_a)/\tau)} \right) \tag{17}$$

where:

- $P(i) = \{p \in A(i) \mid y_p = y_i\}$, is the set of indices for samples having the same label as i.
- $s(Z_i, Z_p) = \exp\left(\text{sim}(Z_i, Z_p)/(\tau)\right)$, $\text{sim}(z_i, z_p)$ indicates the cosine similarity between z_i and z_p.
- τ is a temperature parameter used to control the concentration of the distribution.

By maximizing similarity for positive pairs (z_i, z_p) while minimizing similarity for negative pairs, L_{SC} encourages class-aligned and discriminative representations.
The final averaged supervised contrastive loss using the augmented samples used in FedAlign is defined as:

$$\mathcal{L}_{\text{SC}} = \frac{1}{2}\left(\bar{\mathcal{L}}_{\text{SC}}(\bar{Z}^{(1)}) + \bar{\mathcal{L}}_{\text{SC}}(\bar{Z}^{(2)}\right) \tag{18}$$

where:

$$\bar{Z}^{(1)} = \text{cat}(Z^{(1)}, Z), \tag{19}$$

$$\bar{Z}^{(2)} = \text{cat}(Z^{(2)}, Z) \tag{20}$$

Representation Consistency Loss (L_{RC})

To further ensure stability and consistency in the feature space, we incorporate a Mean Squared Error (MSE) term between original and augmented representations:

$$L_{RC} = \frac{1}{|\text{mix\_feat}|} \sum_{i=1}^{2} \|Z - Z^{(i)})\|^2, \tag{21}$$

'mix_feat' refers to the set of feature maps selected for MixCluster augmentations. By minimizing L_{RC}, the model maintains consistency in latent representations, even when subjected to domain-altering transformations.

We combine these objectives into a single representation alignment loss, which balances both discriminative class alignment and robust consistency:

$$L_{RA} = L_{SC} + L_{RC}. \tag{22}$$

3.6. Prediction Alignment

In addition to feature-level alignment, FedAlign imposes consistency on the model's predictions through Jensen–Shannon (JS) Divergence, which measures the stability of predictions across original and augmented samples. Formally, for predictions Y, $Y^{(1)}$, and $Y^{(2)}$, the JS Divergence loss is defined as:

$$L_{JS} = \frac{1}{3}\left[KL(Y\|\bar{Y}) + KL(Y^{(1)}\|\bar{Y}) + KL(Y^{(2)}\|\bar{Y})\right], \tag{23}$$

where:

- $\bar{Y} = \frac{1}{3}(Y + Y^{(1)} + Y^{(2)})$ is the mean prediction distribution across the original and augmented samples.
- KL denotes the Kullback–Leibler Divergence, quantifying how one probability distribution diverges from a second, reference distribution.

By enforcing prediction consistency, L_{JS} encourages the network to produce stable outputs. It always has a finite value, making it a more stable alternative to KL divergence when used as a loss function.

Algorithm	PACS					OfficeHome					Caltech-10					miniDomainNet				
	P	A	C	S	Avg.	A	C	P	R	Avg.	A	C	D	W	Avg.	C	P	R	S	Avg.
FedAvg	90.30	69.95	75.70	71.80	76.94	61.50	49.00	76.50	77.04	66.01	**95.30**	87.89	98.73	87.46	92.35	64.13	56.89	68.23	50.47	59.93
FedProx	91.21	69.84	73.15	70.87	76.27	62.43	51.09	75.47	76.89	66.47	94.57	88.33	97.45	85.76	91.53	62.57	55.96	67.11	49.71	58.83
FedADG	89.52	63.54	71.45	64.32	72.20	60.58	47.15	72.19	75.84	63.94	95.09	88.07	99.36	93.56	**94.02**	60.17	56.34	68.45	49.82	58.69
GA	92.98	66.01	76.43	69.23	76.16	62.05	49.27	76.10	76.91	66.08	94.99	87.44	97.45	91.53	92.85	62.89	56.12	65.05	48.41	58.11
FedSR	91.42	72.68	76.01	70.73	77.71	**63.95**	50.12	76.58	77.84	67.12	93.63	87.80	96.82	87.12	91.34	67.59	61.09	68.78	57.11	63.64
FedIIR	91.53	71.25	78.61	71.78	78.29	61.64	50.14	75.12	76.83	65.93	94.57	88.16	98.73	90.17	92.91	63.01	57.23	64.79	47.58	58.15
CCST	89.93	76.18	75.97	78.34	80.10	60.94	50.13	76.74	77.65	66.36	93.32	83.70	92.99	88.81	89.71	60.18	57.34	67.73	49.72	58.74
FedCCRL	93.83	79.15	77.90	78.95	82.46	63.08	55.67	76.77	77.71	68.31	94.57	88.6	98.73	91.53	93.36	66.33	59.68	68.03	54.50	62.14
FedAlign	**93.11**	**80.57**	**77.94**	**80.20**	**82.96**	62.68	**57.17**	**78.04**	**78.37**	**69.03**	94.78	**89.85**	**98.73**	**91.53**	93.72	**67.83**	**61.18**	**69.23**	**56.80**	**63.76**

Table 1. Test accuracy on each dataset. These experiments were conducted with an upload ratio of $r = 0.1$. Each algorithm was evaluated three times, and the final results represent the average test accuracy.

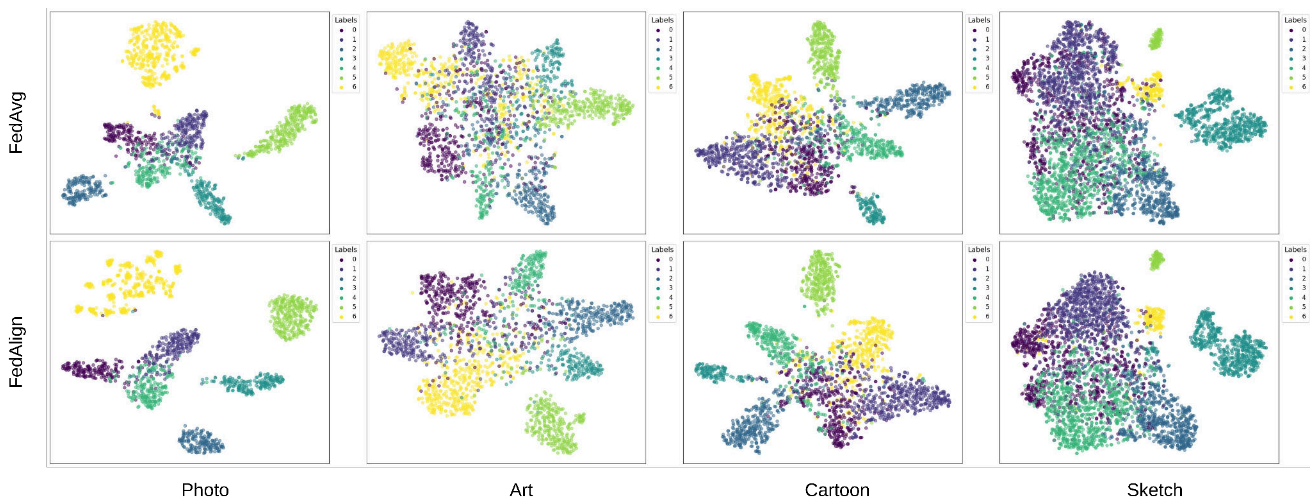

Figure 3. t-SNE visualization of the representation distribution using FedAvg. The representations show domain-specific clusters with noticeable overlaps, highlighting the limitations of FedAvg in learning robust domain-invariant features.

3.7. Total Loss Function

The final loss function for FedAlign combines the primary classification objective with both representation and prediction alignment terms:

$$L = L_{CLS} + \lambda_1 L_{RA} + \lambda_2 L_{JS}, \tag{24}$$

where: L_{CLS} is the averaged cross-entropy loss over Y, $Y^{(1)}$, $Y^{(2)}$ taken pairwise for classification and λ_1 and λ_2 are hyperparameters that balance the influence of the representation and prediction alignment terms, respectively.

L_{D} is not directly added to the total loss but used to update the domain discriminator separately.

The feature extractor h is simultaneously optimized to:

- Minimize the primary classification loss $\mathcal{L}_{\mathrm{CLS}}$
- Reduce representation discrepancy through $\mathcal{L}_{\mathrm{RA}}$
- Align prediction distributions via $\mathcal{L}_{\mathrm{JS}}$
- Adversarially *maximize* $\mathcal{L}_D$ through gradient reversal

3.8. Privacy Concern

Note that only the statistical summaries of data samples are shared by the client with the server and subsequently distributed to other clients, not the raw data itself. Since the mean and standard deviation do not contain sufficient information to reconstruct the original data samples, the privacy of each client remains safeguarded, aligning with the principles of the Federated Learning paradigm.

Dataset	Domains	Cls.	Samples	Characteristics
PACS [8]	Art, Cartoon, Photo, Sketch	7	9,991	High inter-domain variability
OfficeHome [21]	Art, Clipart, Product, Real	65	15,588	Diverse object appearances
miniDomainNet [35]	Clipart, Infograph, Painting, Real	126	140k	Large-scale, heterogeneous
Caltech-10 [24]	Amazon, Caltech-10, DSLR, Webcam	101	2,523	Combination of 2 Datasets

Table 2. Overview of datasets used for evaluation.

4. Experiments

Datasets: We evaluate **FedAlign** on four widely used domain generalization benchmarks, each offering distinct characteristics and posing unique challenges (Table 2)

Evaluation Protocol: We use the leave-one-domain-out protocol to assess generalization. For each dataset, one domain serves as the test set while the others form the training

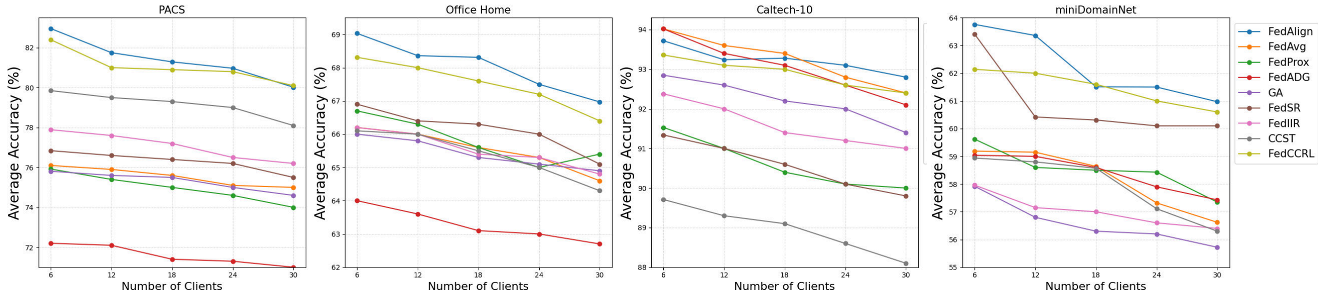

Figure 4. Average test accuracy (%) versus the number of participating clients.

set. This process repeats for all domains.

Data Partitioning: Datasets are split among clients following Section 3.1, simulating non-IID distributions to evaluate robustness to data heterogeneity in federated learning.

Model Architecture: All methods, including FedAlign, use MobileNetV3-Large as the backbone. The final fully connected layer acts as the classifier g, while preceding layers form the encoder h.

Training Configuration: Training runs for 10 communication rounds with 3 local epochs per round. We use Adam optimizer with an initial learning rate of 0.001, decayed via cosine scheduling. The supervised contrastive loss uses $\tau = 0.1$. Input images are resized to 224×224 for PACS, OfficeHome, and Caltech-10, and 128×128 for miniDomainNet.

4.1. Performance and Comparative Analysis

As shown in Table 1, FedAlign consistently outperforms the baseline methods across the evaluated datasets, achieving a better overall average accuracy.

In addition to superior average accuracy, Figure 4 illustrates the resilience of FedAlign under varying client sizes, encompassing both small- and large-scale client settings. While the performance of all baseline methods deteriorates markedly as the number of participating clients increases, FedAlign maintains a consistent advantage.

The decline in accuracy with an increasing number of clients can largely be attributed to data heterogeneity. When more clients participate, the diversity in data distributions across clients grows, leading to statistical heterogeneity. This heterogeneity causes inconsistencies in local model updates, which can conflict during global aggregation, ultimately slowing convergence and reducing overall model accuracy. We further discuss the effect of hyperparameters and upload ratio in `supplementary materials` and also present ablation study to explore the individual effects of MixCluster, along with Supervised Contrastive, Representation Consistency and JS Divergence components of Loss Function.

4.2. Representation Distribution via t-SNE

To evaluate FedAlign's effectiveness, we analyze learned representations using t-SNE visualizations. Figure3 compares representation spaces across four domains—Photo, Art, Cartoon, and Sketch—for FedAvg (baseline) and FedAlign. FedAlign produces more distinct and compact clusters, particularly in challenging domains like Photo and Art, indicating stronger domain alignment. It also improves intra-class coherence, with samples from the same class tightly grouped, and enhances inter-class separability, reducing overlap between classes. These results demonstrate FedAlign's ability to handle domain shifts effectively, offering more robust and generalized feature representations compared to traditional federated learning baselines.

5. Conclusion

In this paper, we present FedAlign, a novel framework for Federated Domain Generalization (FDG) that addresses the critical challenges of limited local data availability and client heterogeneity. Our approach introduces (1) an efficient cross-client representation enrichment module that diversifies representations through cluster-aware augmentation, and (2) adversarial training for domain alignment (3) a dual-stage alignment strategy combining adversarial domain invariance with prediction consistency optimization to extract robust domain-agnostic features. Extensive evaluations across multiple benchmark datasets demonstrate that FedAlign consistently outperforms state-of-the-art methods, achieving superior accuracy while maintaining strong scalability across varying client populations.

These advancements position FedAlign as a practical solution for real-world federated learning scenarios, particularly in healthcare and IoT applications where data heterogeneity and privacy constraints are paramount. Our results suggest a promising direction for future research in federated representation learning, emphasizing the importance of joint optimization of feature diversity and domain invariance in distributed environments.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[2] Junming Chen, Meirui Jiang, Qi Dou, and Qifeng Chen. Federated domain generalization for image recognition via cross-client style transfer. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 361–370, 2023.

[3] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[4] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[5] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2477–2486, 2019.

[6] Yaming Guo, Kai Guo, Xiaofeng Cao, Tieru Wu, and Yi Chang. Out-of-distribution generalization of federated learning via implicit invariant relationships. In *International Conference on Machine Learning*, pages 11905–11933. PMLR, 2023.

[7] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[8] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[9] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3347–3366, 2021.

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

[11] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. *Advances in Neural Information Processing Systems*, 32, 2019.

[12] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[13] A Tuan Nguyen, Philip Torr, and Ser Nam Lim. Fedsr: A simple and effective domain generalization method for federated learning. *Advances in Neural Information Processing Systems*, 35:38831–38843, 2022.

[14] Ragja Palakkadavath, Thanh Nguyen-Tang, Hung Le, Svetha Venkatesh, and Sunil Gupta. Domain generalization with interpolation robustness. In *Asian Conference on Machine Learning*, pages 1039–1054. PMLR, 2024.

[15] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019.

[16] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

[17] Alexander Robey, George J Pappas, and Hamed Hassani. Model-based domain generalization. *Advances in Neural Information Processing Systems*, 34:20210–20229, 2021.

[18] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 443–450. Springer, 2016.

[20] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[21] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017.

[22] Riccardo Volpi and Vittorio Murino. Addressing model vulnerability to distributional shifts over image transformation sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7980–7989, 2019.

[23] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31, 2018.

[24] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S Yu. Visual domain adaptation with manifold embedded distribution alignment. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 402–410, 2018.

[25] Jindong Wang, Yiqiang Chen, Wenjie Feng, Han Yu, Meiyu Huang, and Qiang Yang. Transfer learning with dynamic distribution adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(1):1–25, 2020.

[26] Xinpeng Wang, Yongxin Guo, and Xiaoying Tang. Fedccrl: Federated domain generalization with cross-client representation learning. *arXiv preprint arXiv:2410.11267*, 2024.

[27] Qinwei Xu, Ruipeng Zhang, Ya Zhang, Yi-Yan Wu, and Yanfeng Wang. Federated adversarial domain hallucination for privacy-preserving domain generalization. *IEEE Transactions on Multimedia*, 26:1–14, 2023.

[28] Zhenlin Xu, Deyi Liu, Junlin Yang, Colin Raffel, and Marc Niethammer. Robust and generalizable visual representation learning via random convolutions. *arXiv preprint arXiv:2007.13003*, 2020.

[29] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4085–4095, 2020.

[30] Chun-Han Yao, Boqing Gong, Hang Qi, Yin Cui, Yukun Zhu, and Ming-Hsuan Yang. Federated multi-target domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1424–1433, 2022.

[31] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. *arXiv preprint arXiv:2107.00233*, 2021.

[32] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[33] Liling Zhang, Xinyu Lei, Yichun Shi, Hongyu Huang, and Chao Chen. Federated learning with domain generalization. *arXiv preprint arXiv:2111.10487*, 2021.

[34] Ruipeng Zhang, Qinwei Xu, Jiangchao Yao, Ya Zhang, Qi Tian, and Yanfeng Wang. Federated domain generalization with generalization adjustment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3954–3963, 2023.

[35] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30:8008–8018, 2021.

[36] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4396–4415, 2022.

[37] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.