

# Federated Domain Generalization for Image Recognition via Cross-Client Style Transfer

Junming Chen<sup>1\*</sup> Meirui Jiang<sup>2\*</sup> Qi Dou<sup>2</sup> Qifeng Chen<sup>1</sup>  
<sup>1</sup>HKUST <sup>2</sup>CUHK  
 {jchenfo, cqf}@ust.hk {mrjiang, qdou}@cse.cuhk.edu.hk

## Abstract

*Domain generalization (DG) has been a hot topic in image recognition, with a goal to train a general model that can perform well on unseen domains. Recently, federated learning (FL), an emerging machine learning paradigm to train a global model from multiple decentralized clients without compromising data privacy, has brought new challenges and possibilities to DG. In the FL scenario, many existing state-of-the-art (SOTA) DG methods become ineffective because they require the centralization of data from different domains during training. In this paper, we propose a novel domain generalization method for image recognition under federated learning through cross-client style transfer (CCST) without exchanging data samples. Our CCST method can lead to more uniform distributions of source clients, and make each local model learn to fit the image styles of all the clients to avoid the different model biases. Two types of style (single image style and overall domain style) with corresponding mechanisms are proposed to be chosen according to different scenarios. Our style representation is exceptionally lightweight and can hardly be used to reconstruct the dataset. The level of diversity is also flexible to be controlled with a hyper-parameter. Our method outperforms recent SOTA DG methods on two DG benchmarks (PACS, OfficeHome) and a large-scale medical image dataset (Camelyon17) in the FL setting. Last but not least, our method is orthogonal to many classic DG methods, achieving additive performance by combined utilization. Our code is available at: <https://chenjunming.ml/proj/CCST>.*

## 1. Introduction

Federated learning (FL) aims to train a machine learning model on multiple decentralized local clients without explicitly exchanging data samples. This emerging technique has triggered increasing research interest in recent years,

owing to its significant applications in many real-world scenarios such as finance, healthcare, and edge computing [22]. The paradigm works in a way that each local client (e.g., hospital) learns from their local data and only aggregates the model parameters at a specific frequency on the central server to yield a global model.

One of the biggest challenges in FL is tackling the non-identically and independently distributed (non-IID) data across different clients. Although much progress has been made on addressing non-IID issues in FL [29, 28], most of them only focus on improving the performance of internal clients. Few papers focus on domain generalization in FL, which is a crucial scenario considering the model generalization ability on a new client with unseen data distribution. For example, it is important that a federated trained disease diagnose model by multiple hospitals can be directly utilized by other new hospitals with a high accuracy, especially when they have few annotated data to train a good model. DG aims to improve the test performance on unseen target domains with the model trained on multi-source data. A prior work FedDG [33] proposes to exchange the amplitude information in frequency domain cross clients and utilize episodic learning to improve the performance further. However, they are specific to medical image segmentation tasks and consider the distribution shift across medical imaging protocols, which remains unexplored for larger domain gaps in the wild. In contrast, we aim to improve the model generalization ability for image recognition tasks, and our method is able to handle domain shifts from small (cross-site medical images) to more significant ones like photos and sketches in the PACS dataset.

The FL scenario poses particular and new challenges to DG: regarding each client as a domain with a specific style, the data from each domain cannot be put together during training, which violates the implicit requirement of many DG methods. For example, meta-learning [26] and adversarial domain invariant feature learning [27] both require access to all the source domains at the same time, which is not directly applicable in federated learning. In addition, straightforward aggregating the parameters of local models

\*Joint first authors.

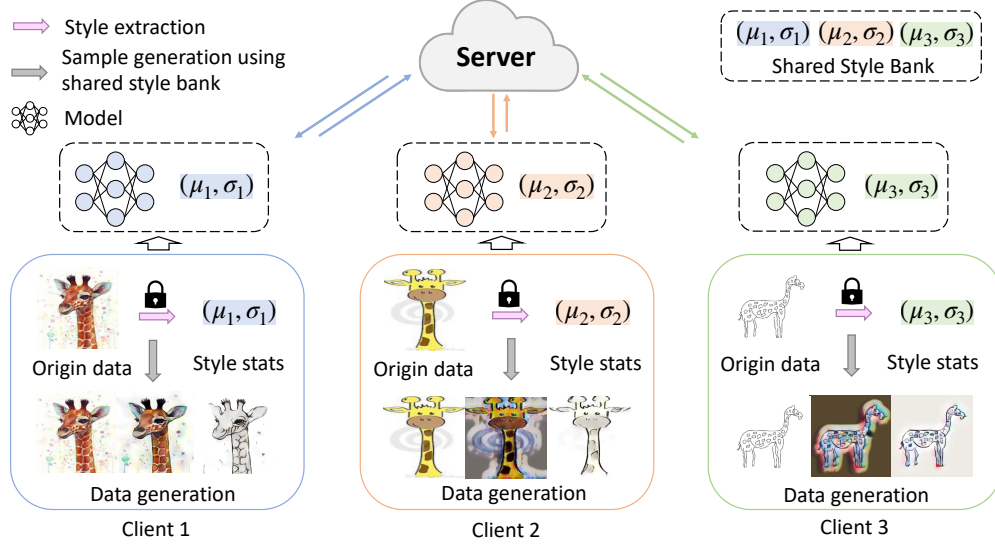


Figure 1: Overview of our framework with style transfer across source clients using three different source styles on the PACS dataset. We augment each source client data with styles of other two source clients.

may lead to a sub-optimal global model because the local models are biased to different client styles. To solve those problems, we propose a data-level cross-domain style transfer (CCST) method that augments the data by using other source domain styles with the style transfer technique. In this way, each client will have styles of all the other source clients, and thus all the local models will have the same goal to fit images with all the source styles, which avoids the different local model biases that may compromise the global model performance. Moreover, CCST is orthogonal to other DG methods, and thus existing methods on centralized DG can also benefit from a further accuracy boost.

Our CCST method for federated domain generalization is general and compatible with any style transfer method that satisfies two requirements: First, the style information in the style transfer algorithm cannot be utilized to reconstruct the dataset; Second, this style transfer method should be an arbitrary style transfer per model method, which means the style transfer model should be ready to transfer a content image to arbitrary styles. Since there can be many clients in federated learning, the style transfer model should better have the ability to transfer all those styles without retraining. Otherwise, the deployment cost will significantly increase since each client has to store various models locally for different styles and even require retraining for unseen styles. In our paper, we choose AdaIN [15], an effective real-time arbitrary style transfer model to demonstrate the effectiveness of our CCST framework. The style information used in AdaIN is the moments (i.e., mean and variance) of each pixel-level feature channel at a specific VGG layer, which are extremely lightweight (two 512-dimensional vectors) and do not contain spatial structural information about

the image content. Therefore, such style information is efficient to be shared across clients and can hardly lead to the reconstruction of the dataset. Further analysis could be found in Section 4.4.

The overall framework of our method is shown in Figure 1. Each client is regarded as a domain with a domain-specific style. Before training the image recognition model, we first compute the style information of images in each source client. We design two types of styles that can be shared: single image style and overall domain style, which will be illustrated in detail in Section 3. Then the source clients will upload their style information to the global server and share them with all the source clients, which we call style bank. Each source client utilizes the shared style bank to perform style transfer on their local data, during which a hyperparameter  $K$  is introduced to control the diversity level of our CCST process. Federated training will begin after each source client finishes data augmentation, and then the trained model will be directly tested on the unseen target client. Our contributions are summarized below:

(a) We propose a simple yet effective framework named cross-client style transfer (CCST). Our approach achieves new state-of-the-art generalization performance in FL setting on two standard DG benchmarks (PACS [24], Office-Home [39]) and a large-scale medical image dataset (Cameleon17 [3]). (b) Two types of styles with corresponding sharing mechanisms are proposed, named *overall domain style* and *single image style*, which can be chosen according to different circumstances. The diversity level of our method is also flexible to be adjusted. (c) The proposed method is orthogonal to many other SOTA DG methods. Therefore, our method can be readily applied to those DG

methods to have a further performance boost. We also study the effectiveness of several SOTA DG methods when they are applied in the FL setting for image recognition. **(d)** We give an intuitive (Section 4.4) and experimental analysis (Section A) on the privacy-preserving performance of our style vectors to demonstrate that one can hardly reconstruct the original images merely from the style vectors using the generator from a SOTA GAN [32] in FL setting.

## 2. Related Work

**Domain generalization.** Domain generalization is a popular research field that aims to learn a model from multiple source domains such that the model can generalize on the unseen target domain. Many works are proposed towards solving the domain shifts from various directions under the centralized data setting. Those methods can be divided into three categories [42], including manipulating data to enrich data diversity [18, 44, 37, 47], learning domain-invariant representations or disentangling domain-shared and specific features to enhance the generalization ability of model [1, 36, 4, 46] and exploiting general learning strategies to promote generalizing capability [26, 17, 7, 8].

However, many of these methods require centralized data of different domains, violating the local data preservation in federated learning. Specifically, access for more than one domain is needed to augment data or generate new data in [37, 18], domain invariant representation learning or decomposing features is performed under the comparison across domains [1, 36, 46] and some learning strategy based methods utilize extra one domain for meta-update [26, 7, 8]. Nevertheless, some methods do not explicitly require centralized domains or can be adapted into federated learning with minor changes. For example, MixStyle [47] can optionally conduct the style randomization in a single domain to augment data; [44] uses Fourier transformation to augmentation that is free of sharing data; JiGen [4] proposes a self-supervised task to enhance representation capability; RSC [17] designs a learning strategy based on gradient operations without explicit multi-domain requirements.

**Federated / decentralized domain generalization.** Despite many works on centralized domain generalization and tackling non-IID issues in FL, there are few works addressing the DG problem in FL. FedDG [33] exchanges the amplitude information across images from different clients and utilizes episodic learning to improve performance further. However, it only focuses on the segmentation task with superficial domain shift in data, and its performance on image recognition with larger domain shift remains unexplored. COPA [43] propose only aggregating the weights for domain-invariant feature extractor and maintaining an assemble of domain-specific classifier heads to tackle the decentralized DG. However, since COPA has to share clas-

sifier heads of all the clients locally and globally, it may lead to privacy issues, heavier communication, and higher test-time inference cost.

**Neural style transfer.** Neural style transfer (NST) aims to transfer the style of an image to another content image with its semantic structure reserved. The development of NST has roughly gone through three stages: per-style-per-model (PSPM), multiple-style-per-model (MSPM) and arbitrary-style-per-model (ASPM) methods [20]. PSPM methods [11, 21, 38] can only transfer a single style for each trained model. MSPM methods [9, 5, 45, 31] are able to transfer multiple styles with a single trained model. However, PSPM and MSPM are expensive to deploy when too many styles are required to be transferred in our setting. ASPM [6, 16, 12, 30] can transfer arbitrary styles to any content images and is often faster than PSPM and MSPM, which is more suitable for our scenario.

The first ASPM method is proposed by Chen and Schmidt [6], but it cannot achieve real-time. AdaIN [16] is the first real-time arbitrary style transfer method, which utilizes the channel-wise mean and variance as style information. It performs de-stylization by normalizing the VGG feature with its own style and then stylizes itself by affine transformation with the mean and variance of the style image feature. Another real-time ASPM method [12] is a follow-up work of CIN [10]. They change the MSPM method CIN into an ASPM method by predicting the affine transformation parameters for each style image through another style prediction network. However, the level of style-content disentanglement of the predicted style vector remains unknown, which may have privacy issue in FL setting. Later, Li et al. [30] propose a universal style-learning free ASPM method, which utilizes ZCA whitening transform for de-stylization and coloring transform for style transfer. However, this method is much slower than previous methods in practice. Therefore, we choose the neatest and efficient real-time ASPM method AdaIN as our style transfer model in our framework.

## 3. Method

The core idea of our method is to let the distributed clients have as similar data distribution as possible by introducing styles of other clients into each of them via cross-client style transfer without dataset leakage. Figure 3 shows the data distribution before and after our CCST method. In this way, we can make the trained local models learn to fit all the source client styles and avoid aggregating the local models biased to different styles. As a result, each client can be regarded as a deep-all [4] setting, and the local models will have the same goal to fit styles from all the source clients. We propose two types of styles that can be chosen to transfer: one is overall domain style, the other is single image style. In the following sections, we will introduce

our reorganized style transfer framework and the process of cross-client style transfer.

Our CCST method for federated domain generalization is general and compatible with any style transfer method that satisfies two requirements: For a style transfer model to be utilized in our general CCST framework, at least two requirements should be satisfied: 1) The style information shared among clients cannot be utilized to reconstruct the dataset; 2) The style transfer method should be a real-time arbitrary style transfer model to allow efficient and straightforward style transfer. AdaIN [16] is the first real-time arbitrary style transfer model, and it perfectly satisfies both requirements. Moreover, it is extremely difficult to recover the dataset only from the style information utilized in AdaIN. We give an intuitive analysis in Section 4.4 about the privacy issue. Therefore, we choose AdaIN as our style transfer model to demonstrate the effectiveness of our CCST framework. Formally, for content image  $I_c$  and style image  $I_s$ , the corresponding VGG features  $F_c$  and  $F_s$  are:

$$F_c = \Phi(I_c), \quad F_s = \Phi(I_s), \quad (1)$$

where  $\Phi$  is the VGG encoder.

The AdaIN module takes the VGG features  $F_c$  and  $F_s$  of content and style images as input, which first normalize (de-stylization) with moments of  $F_c$  and then perform an affine transformation (stylize) with the moments of  $F_s$ . Formally,

$$AdaIN(F_c, F_s) = \sigma(F_s) \left( \frac{F_c - \mu(F_c)}{\sigma(F_c)} \right) + \mu(F_s), \quad (2)$$

where  $\mu(\cdot)$  and  $\sigma(\cdot)$  compute channel-wise mean and standard variance of image features. Assume  $F_c$  have the shape of  $x \times h \times w$  given channel dimension  $x$  and feature map resolution  $h \times w$ , then  $\mu(F_c)$  and  $\sigma(F_c)$  have the same shape of  $x \times 1$ . Normally,  $x = 512$  in our work.

Then, the transferred feature  $F_{c \leftarrow s}$  derived from  $AdaIN(F_c, F_s)$ , is passed to a decoder  $\Psi$  to generate the final stylized image  $I_{c \leftarrow s}$ :

$$I_{c \leftarrow s} = \Psi(AdaIN(F_c, F_s)). \quad (3)$$

### 3.1. Preliminaries

As shown in Figure 2, we divide the workflow of AdaIN into two parts. The first part is the style extractor  $SE$  with style image as input. Denote the style as  $S$ , then:

$$S = SE(I_s) = (S_\mu, S_\sigma) = (\mu(\Phi(I_s)), \sigma(\Phi(I_s))). \quad (4)$$

The second part is an image generator  $G$  with content image and style vector as input:

$$I_{c \leftarrow s} = G(I_c, S) = \Psi(S_\sigma \left( \frac{\Phi(I_c) - \mu(\Phi(I_c))}{\sigma(\Phi(I_c))} \right) + S_\mu). \quad (5)$$

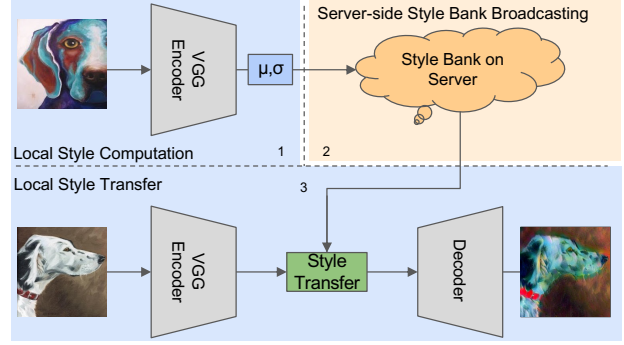


Figure 2: The re-organized AdaIN [16] framework utilized for cross-client style transfer in federated learning. The VGG encoder is shared between the style extraction and image generation stage. Dash lines separate the three stages of our method: 1. Local style computation; 2. Server-side style bank broadcasting; 3. Local style transfer.

## 3.2. Cross-Client Style Transfer

We regard each client in federated learning as a domain. The workflow of the data augmentation process is shown in Algorithm 1. Assume there are  $N$  clients  $\{C_1, C_2, \dots, C_N\}$  and there is a central server. Note that we will introduce two types of styles and corresponding mechanisms in parallel in the following illustration. As shown in Figure 2, our method has three stages:

### 3.2.1 Local style computation and sharing.

In the beginning, each client needs to compute their style and upload them to the global server. Two types of styles can be chosen to share across clients:

**Single image style.** Image style is calculated as the pixel-level channel-wise mean and standard variance of the VGG feature of image. Formally, for a randomly chosen image with index  $i$  and VGG feature  $F_i^{C_n}$  at client  $C_n$ , the single image style  $S_{single(i)}^{C_n}$  would be:

$$S_{single(i)}^{C_n} = (\mu(F_i^{C_n}), \sigma(F_i^{C_n})). \quad (6)$$

If single image styles are utilized for style transfer, multiple styles of different images should be uploaded to the server for this client to avoid single image bias and increase diversity, which forms a local image style bank  $S_{bank}^{C_n}$ . Formally, denote randomly selected  $J$  image styles to be uploaded by  $C_n$  as

$$S_{bank}^{C_n} = \{S_{single(i_1)}^{C_n}, \dots, S_{single(i_J)}^{C_n}\}, \quad (7)$$

where  $\{i_1, \dots, i_J\}$  are randomly sampled image indices from client  $C_n$ . Sharing single image styles consumes relatively low computation but can lead to high communication costs for uploading multiple styles.



**Overall domain style.** Domain style is the domain-level channel wise mean and standard variance, which considers all the images (pixels) in a client. Formally, assume client  $C_n$  has  $M$  training images with corresponding VGG features  $\{F_1^{C_n}, F_2^{C_n}, \dots, F_M^{C_n}\}$ , the overall style  $S_{overall}^{C_n}$  of this client is:

$$\begin{aligned} S_{overall}^{C_n} &= (\mu(F_{all}^{C_n}), \sigma(F_{all}^{C_n})), \\ F_{all}^{C_n} &= Stack(F_1^{C_n}, F_2^{C_n}, \dots, F_M^{C_n}). \end{aligned} \quad (8)$$

The computation cost of the overall domain style is relatively high compared to just computing several single image styles. However, since each domain only has one domain style  $S_{overall}^{C_n}$ , choosing to upload the overall domain style to the server is more communication efficient.

Besides, the overall domain style can represent the domain style more robustly, while single image styles give more diversity and randomness to the style bank.

### 3.2.2 Server-side style bank broadcasting.

When the server receives all the styles of each client, it will concatenate all the styles as a style bank  $\mathbf{B}$  and broadcast it back to all the clients. In the two different style sharing modes, the style bank will also be different:

- Style bank  $\mathbf{B}_{single}$  for single image styles:

$$\mathbf{B}_{single} = \{S_{bank}^{C_n} | n = 1, 2, \dots, N\}. \quad (9)$$

- Style bank  $\mathbf{B}_{overall}$  for overall domain style:

$$\mathbf{B}_{overall} = \{S_{overall}^{C_n} | n = 1, 2, \dots, N\}. \quad (10)$$

Similarly, here  $\mathbf{B}_{single}$  cost more memory than  $\mathbf{B}_{overall}$ . Therefore, the latter one is more communication-friendly.

### 3.2.3 Local style transfer.

When client  $C_n$  receives the style bank  $\mathbf{B}$ , the local data can be augmented by transferring styles in  $\mathbf{B}$  to existing images, which introduces styles of other domains into this client. A hyperparameter  $K \in \{1, 2, \dots, N\}$  called *augmentation level*, is set to choose  $K$  styles from style bank  $\mathbf{B}$  for the augmentation of each image, indicating the diversity of final augmented data set. Suppose the size of the original dataset is  $d$ , then after cross-client style transfer, the size of the augmented data set will become  $d \times K$ .

The workflow for local style transfer with two choices of styles is illustrated in Algorithm 1. First, for each image  $I$  in client  $C_n$ ,  $K$  random domains are selected, and each selected domain should have one style vector  $S$  to be input into image generator  $G$ . If transfer the overall domain styles, simply choose the corresponding style  $S_{overall}$

---

#### Algorithm 1 Local Cross-Client Style Transfer at Client $C_n$

---

**Input:** Training image set  $\mathbf{I}^{C_n}$ , global style bank  $\mathbf{B}$ .

**Parameter:** Augmentation level  $K$ , style type  $T$ .

**Output:** Augmented dataset  $\mathbf{D}^{C_n}$

```

1:  $\mathbf{D}^{C_n} = []$  ▷ Augmented dataset
2: for  $i = 1, 2, \dots, m$  do ▷  $m = size(\mathbf{I})$ 
3:    $S = random.choice(\mathbf{B}, K)$ 
4:   for  $S^{C_n}$  in  $S$  do
5:     if  $C_n$  is current client then
6:        $\mathbf{D}^{C_n}.append(I_i)$ 
7:     else if  $T$  is single mode then
8:        $\mathbf{D}^{C_n}.append(G(I_i, random.choice(S^{C_n}, 1)))$ 
▷ Eq. 5
9:     else if  $T$  is overall mode then
10:       $\mathbf{D}^{C_n}.append(G(I_i, S^{C_n}))$ 
11: return  $\mathbf{D}^{C_n}$ 

```

---

from  $\mathbf{B}_{overall}$ ; otherwise if transfer single images styles, one style  $S_{single}$  will be randomly chosen from  $S_{bank}^{C_n}$  as the style of the selected domain in  $\mathbf{B}_{single}$ . In both style modes, if the domain itself is chosen, this image will be directly put into the augmented data set.

## 4. Experiments

### 4.1. Datasets

We evaluate our method on two standard domain generalization datasets (PACS [25], Office-Home [40]) that consist of various image styles as domains and a real-world medical image dataset (Camelyon17 [3]). Specifically, PACS is a 7-class image recognition benchmark including 9,991 images with four different image style domains, including photo, art, cartoon, and sketch. Office-Home is another image recognition dataset that includes 15,588 images of 65 classes from four different domains (art, clipart, product, and real-world). Camelyon17 is a public tumor classification dataset, which has histology images from 5 hospitals.

### 4.2. Experimental Settings

**Experiment setup.** We take each domain as a single client and conduct the leave-one-domain-out experiments on PACS and Office-Home datasets. Specifically, we select one client as the target test domain and train our model on the other clients. For the medical dataset, following the setting of source/target domains in literature [3, 23], we apply the leave-one-domain-out setting to hospital 4 and hospital 5. For the PACS dataset, we follow the JiGen [4] to split 90% data of each client as the training set and 10% of that as the validation set for source clients, while for unseen target clients, the entire data is used for testing. For OfficeHome and Camelyon17, which have more data samples, the ratio between train and validation set is 4:1 for each source client,

Table 1: Accuracy comparison of image recognition on the PACS and Office-Home dataset, each single letter column represents an unseen target client. Our CCST with the overall domain style (K=3) outperforms other methods. We use FedAvg as our base FL framework. Jigen, RSC, and Mixstyle are applied within each client. The backbone networks utilized in PACS and Office-Home are ImageNet-pretrained ResNet50 and ResNet18 respectively. (†: Since COPA did not release their code, we copy the results from their paper here. But it cannot directly compare with our results due to the setting difference.)

Method	PACS					Office-Home					
	P	A	C	S	Avg.	A	C	P	R	Avg.	Avg.
FedAvg (AISTATS'17) [34]	95.51	82.23	78.20	73.56	82.37	60.08	45.59	69.48	72.82	61.99	72.18
Jigen (CVPR'19) [4]	95.99	84.72	77.09	72.16	82.49	60.29	46.16	69.26	72.59	62.07	72.28
RSC (ECCV'20) [17]	95.21	83.15	78.24	74.62	82.81	58.23	46.05	70.27	<b>73.39</b>	61.99	72.40
MixStyle (ICLR'21) [47]	95.93	85.99	<b>80.03</b>	75.46	84.35	58.44	<b>50.29</b>	70.61	70.64	62.49	73.42
FedDG (CVPR'21) [33]	96.23	83.94	79.27	73.30	83.19	<b>60.70</b>	45.82	71.51	73.05	62.77	72.98
COPA-Res18† (ICCV'21) [43]	94.60	83.30	79.80	82.50	85.10	59.40	55.10	74.80	75.00	66.10	75.60
CCST (Overall, K=3)	<b>96.65</b>	<b>88.33</b>	78.20	<b>82.90</b>	<b>86.52</b>	59.05	50.06	<b>72.97</b>	71.67	<b>63.56</b>	<b>75.04</b>

and 20% data is utilized as the test set on the unseen target client. We compare our method with **FedDG** [33], which aims to solve DG problems in federated learning for medical image segmentation. We also test the performance of three centralized DG methods under FL setting (FedAvg), including **JiGen** [4], **RSC** [17] and **MixStyle** [47]. For **COPA**, due to its re-designed layers of ResNet18 and unknown train-validate-test split, we copy the results for reference only. We regard every single client as a centralized dataset and apply these methods locally in FL. We report the test accuracy on each unseen client by choosing the best validation model.

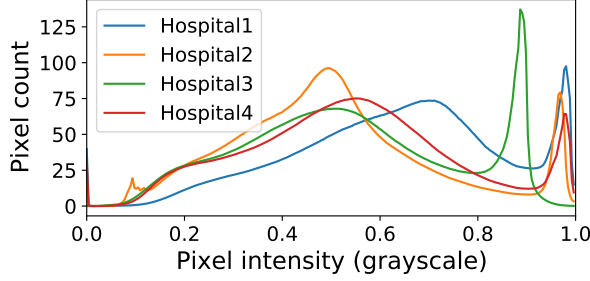
**Implementation details.** We utilize the pre-trained AdaIN [16] to perform style transfer. Following [17], we choose ResNet [13] pre-trained on ImageNet as our backbone for PACS and Office-Home datasets. For the Camelyon17 dataset, we follow [19] to use the DenseNet121 [14]. We use FedAvg [34] as our FL framework and train the model using SGD optimizer with  $1e^{-3}$  learning rate for 500 communication rounds with one local update epoch on the PACS and Office-Home dataset. For Camelyon17, we train 100 communication rounds considering its large data amount. The JiGen and RSC can be directly integrated into the FedAvg without further modifications. We adapt the MixStyle into an intra-client version that shuffles styles inside each batch of data to fit the federated setting. All hyper-parameters of compared methods are chosen based on corresponding papers. We follow the standard training procedure of FedAvg [34] for federated training. The value of M in is  $\lceil \text{dataset\_size}/32 \rceil$  in our experiment. The framework is implemented with PyTorch and is trained on a single NVIDIA RTX 2080 Ti GPU.

### 4.3. Results

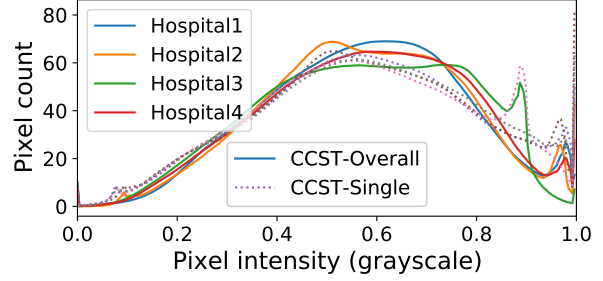
**Comparison with state-of-the-arts.** We compare our approach with three centralized DG methods and a feder-

ated DG method for on standard DG benchmarks PACS and OfficeHome as well as a real-world medical image dataset Camelyon17. Table 1 presents the quantitative results of the image recognition task for different target clients on both PACS and Office-Home datasets. Each single letter column shows the test accuracy of the global model with the best validation accuracy on an unseen client. Although all DG methods can have better performance based on FedAvg, our approach demonstrates a significant boost over others on both datasets. On the PACS benchmark, our method achieves the average accuracy of 86.52%, which is 3.47% better than the second best method FedDG. Especially for the unseen client S (Sketch), CCST outperforms other methods by more than 7%. When photo is the target domain, all the methods perform similarly because we start training based on the ImageNet-pretrained model, which already has very high performance on photo images. Besides the PACS benchmark, the performance of our approach on the Office-Home dataset also has consistent results. Specifically, CCST outperforms other methods on average with a testing accuracy of 63.56%. Due to the small discrepancy in domain styles, all those domain generalization methods bring smaller improvements (less than 1%) than on the PACS. Overall, CCST outperforms other DG methods by a large margin. Figure 4a shows the results on the Camelyon17 dataset, our method outperforms other DG methods both when hospital 4 and hospital 5 as the target client. Some DG method, such as JiGen, is even harmful when applied in FL setting on the Camelyon17 dataset.

Our CCST enables each local client model to update under more diversified images, thus providing a wide range of image styles. The more diversified intra-client distribution helps reduce the bias towards fitting a specific distribution and update the model with a more general direction. As shown in Figure 3, the distribution of source clients become much more uniform after applying our CCST method. In



(a) Camelyon17 distribution before CCST.



(b) Camelyon17 distribution after CCST.

Figure 3: The distribution of source clients on the Camelyon17 before and after our CCST when the target client is hospital 5. The y-axis is the average pixel count per image, and the x-axis is the greyscale values. Note that we convert the RGB into grey images for distribution visualization. **(a)** Before CCST, the distributions of source clients are not uniform. **(b)** After CCST with either single image style or overall style, the distributions of source clients become much more uniform.

contrast, for domain generalization methods aim to learn more general features (e.g., Jigen, RSC), they suffer from limited intra-client distribution and different inter-client distribution, only achieving marginal improvements. Similar to CCST, MixStyle and FedDG also aim to increase the model generalization ability by diversifying the feature distribution. While in the federated setting, MixStyle can only perform intra-client feature diversification by assuming every single image has a unique style, which set a limit on its diversity level. FedDG utilizes amplitude information in frequency space of an image as a kind of style information, and perform amplitude exchange to diversify the data distribution across clients. Nevertheless, the amplitude exchange leads to minor appearance change, which may not enough for images with a large domain gap, while CCST utilizes style transfer to give a more thorough style exchange across clients, which leads to better results. We give visualization results after style transfer in the Section F of the supplementary. The overall domain style usually represents a more general and accurate client style, while the single image style brings more randomness.

**Control experiments on CCST.** We conduct control experiments to investigate two types of image style with different augmentation levels. In Table 2a, *Single* and *Overall* represents single image style and overall domain style mentioned in Section 3 respectively. Different augmentation level  $K$  indicates the intensity of augmentation. We evaluate the four settings on the PACS benchmark with ResNet50. For each kind of style, larger augmentation level  $K$  leads to a better performance. It is worth mentioning that the performance achieved by  $K=2$  is similar with  $K=3$ , which indicates that our method can already achieve good performance with a relatively large  $K$ . For different types of style, overall domain style shows more improvement than single image style because the overall style is able to represent a more general and accurate domain statistics, while single image styles may differ a lot due to randomness.

**Orthogonality.** Our method is orthogonal to many other DG methods and can lead to additive performance via combined utilization. As many traditional domain generalization methods require centralized data and need to make use of various styles to achieve a domain robust model, CCST can serve as an initial step to benefit traditional DG methods with diversified styles. As shown in Figure 4b, we plot the average test accuracy of FedDG and three centralized DG methods on the PACS benchmark before and after applying our CCST. From the average accuracy, we can see all DG methods benefit a further boost with the help of cross-client style transfer (CCST). Interestingly, we find the performance when tested on sketch client (S) gains the largest improvement with CCST. Besides, we also extend our method with Tent [41] which uses entropy to update parameters in batch normalization layers at test time. As shown in Table 2b, with this method combined in federated setting, our method surpasses the state-of-the-art DG method EoA [2] on the PACS benchmark in centralized setting with an average test accuracy of 90.47%.

#### 4.4. Discussions

**Computation and communication trade-off.** To reduce the communication cost that is usually known as the bottleneck of FL, we make a trade-off of using extra local computational cost in the overall style computation stage. Specifically, for the overall style, it requires a lot of local computation. A possible solution could be only choosing a relatively large portion of images from each class to approximate the overall domain style. After local style computation, only lightweight style vectors (two 512 dimensional vectors for each client) will be communicated between local clients and the central server for once. The extra computation cost of our CCST method is very low, we give a quantitative analysis in the Section B of the supplementary.

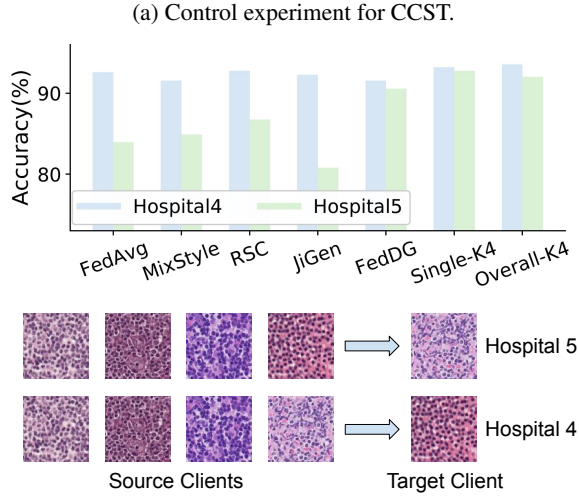
**Analysis for the privacy issue of the shared style vector.** Like all federated learning methods, our method

Table 2: **(a)** Performance of our approach using ResNet50 as the backbone with four different image style transfer settings compared with the baseline of FedAvg on the PACS benchmark. Each column represents a single unseen target client. **(b)** Performance of our approach with test time adaptation (Tent) [41] on the PACS benchmark using ResNet50.

Setting	Unseen client				Average
	P	A	C	S	
FedAvg [34]	95.51	82.23	78.20	73.56	82.37
Single (K=1)	95.75	87.5	74.66	76.56	83.62
Single (K=2)	<b>96.77</b>	86.23	75.73	80.12	84.71
Single (K=3)	96.65	86.63	74.53	81.85	84.84
Overall (K=1)	95.69	86.67	75.85	77.37	83.90
Overall (K=2)	96.41	<b>88.72</b>	78.03	80.91	86.02
Overall (K=3)	96.65	88.33	<b>78.20</b>	<b>82.90</b>	<b>86.52</b>

Setting	Unseen client				Average
	P	A	C	S	
EoA [2]	98.00	90.50	83.40	82.50	88.60
Single (K=1)	97.78	89.55	84.51	82.79	88.66
Single (K=2)	97.54	89.40	84.43	85.42	89.20
Single (K=3)	98.08	89.75	86.05	86.03	89.98
Overall (K=1)	97.78	<b>90.92</b>	86.01	83.66	89.59
Overall (K=2)	<b>98.38</b>	90.72	86.47	85.62	90.30
Overall (K=3)	98.14	90.87	<b>86.77</b>	<b>86.10</b>	<b>90.47</b>



(a) Results on the Camelyon17.



(b) Orthogonality on the PACS.

Figure 4: **(a)** The results on the Camelyon17 dataset. Our method outperforms other DG methods when tested on hospitals 4 and 5. **(b)** Extra performance boost on other domain generalization methods with our cross-domain style transfer (CCST) with overall style on the PACS dataset. Each x-tick represents the single unseen client in a leave-one-client-out experiment, and Avg. is abbreviated for the average accuracy.

may also suffer from privacy issue. However, it is difficult to reconstruct the original datasets with only a 1024-dimensional style vector shared. First, different datasets can have the same style vector. Second, the style vector is the statistics of the pixel feature set, which has no order. Even if we assume that the original pixel feature set can be reconstructed, it is hard to rearrange those pixels into proper order. This is even more difficult for the overall style vector because it is the set statistics of pixels features from many images. Therefore, one can hardly recover the whole datasets only with our shared style vectors. We further experimented with reconstructing the original images from style vectors using a state-of-the-art GAN generator [32]. The experiment shows that neither can a malicious source client recover other clients' images only from style vectors, nor an outside attacker can reconstruct source images using a pre-trained generator. Please refer to the Section A in the supplementary for more details.

## 5. Conclusion

In this paper, we propose a novel DG method for image recognition under the FL setting. Our method utilizes cross-client style transfer to introduce all the source client styles into each client while not violating the original data-sharing restriction of FL. Two types of styles and corresponding sharing mechanisms are proposed to be utilized accordingly. The state-of-the-art generalization performance of our method under the FL setting is demonstrated on two standard DG datasets and a real-world medical image dataset with comprehensive experiments. The data style transfer strategy opens a new way for heterogeneous clients to access diversified distributions without sharing original data, which helps the global model generalize better in FL. Moreover, our method is orthogonal to many other SOTA DG methods and can be combined together to have further performance boost.



## References

- [1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. [3](#)
- [2] Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *arXiv preprint arXiv:2110.10832*, 2021. [7](#), [8](#)
- [3] Péter Bándi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermesen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, Quanzheng Li, Farhad Ghazvinian Zanjani, Svitlana Zinger, Keisuke Fukuta, Daisuke Komura, Vlado Ovtcharov, Shenghua Cheng, Shaoqun Zeng, Jeppe Thagaard, Anders B. Dahl, Huangjing Lin, Hao Chen, Ludwig Jacobsson, Martin Hedlund, Melih Çetin, Eren Halıcı, Hunter Jackson, Richard Chen, Fabian Both, Jörg Franke, Heidi Küsters-Vandeveld, Willem Vreuls, Peter Bult, Bram van Ginneken, Jeroen van der Laak, and Geert Litjens. From detection of individual metastases to classification of lymph node status at the patient level: The camelyon17 challenge. *IEEE Transactions on Medical Imaging*, 38(2):550–560, 2019. [2](#), [5](#)
- [4] Fabio M Carlucci, Antonio D’Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019. [3](#), [5](#), [6](#), [11](#)
- [5] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017. [3](#)
- [6] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016. [3](#)
- [7] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *NeurIPS*, 2019. [3](#)
- [8] Yingjun Du, Jun Xu, Huan Xiong, Qiang Qiu, Xiantong Zhen, Cees GM Snoek, and Ling Shao. Learning to learn with variational information bottleneck for domain generalization. In *ECCV*, 2020. [3](#)
- [9] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016. [3](#)
- [10] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017. [3](#)
- [11] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. [3](#)
- [12] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *BMVC*, 2017. [3](#)
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. [6](#), [13](#)
- [14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. [6](#)
- [15] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. [2](#)
- [16] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. [3](#), [4](#), [6](#), [13](#)
- [17] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *ECCV*, 2020. [3](#), [6](#), [11](#)
- [18] Philip TG Jackson, Amir Atapour Abarghouei, Stephen Bonner, Toby P Breckon, and Boguslaw Obara. Style augmentation: data augmentation via style randomization. In *CVPR Workshops*, 2019. [3](#)
- [19] Meirui Jiang, Zirui Wang, and Qi Dou. Harmofl: Harmonizing local and global drifts in federated learning on heterogeneous medical images. In *AAAI*, 2022. [6](#)
- [20] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3365–3385, 2020. [3](#)
- [21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *ArXiv*, abs/1603.08155, 2016. [3](#)
- [22] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019. [1](#)
- [23] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanus Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021. [5](#)
- [24] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017. [2](#), [11](#)
- [25] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017. [5](#)
- [26] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *CVPR*, 2019. [1](#), [3](#)
- [27] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018. [1](#)
- [28] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018. [1](#)
- [29] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *ICLR*, 2021. [1](#)
- [30] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeurIPS*, 2017. [3](#)

- [31] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified texture synthesis with feed-forward networks. *CVPR*, 2017. 3
- [32] Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed El-gammal. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *ICLR*, 2021. 3, 8, 11
- [33] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *CVPR*, 2021. 1, 3, 6, 11
- [34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017. 6, 8, 11, 15
- [35] Henri J Nussbaumer. The fast fourier transform. In *Fast Fourier Transform and Convolution Algorithms*, pages 80–111. Springer, 1981. 11
- [36] Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In *ICML*, 2020. 3
- [37] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018. 3
- [38] Dmitry Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempit-sky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 3
- [39] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017. 2, 13
- [40] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017. 5
- [41] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 7, 8
- [42] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Wenjun Zeng, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. In *IJCAI*, 2021. 3
- [43] Guile Wu and Shaogang Gong. Collaborative optimization and aggregation for decentralized domain generalization and adaptation. In *ICCV*, 2021. 3, 6
- [44] Qinwei Xu, Ruipeng Zhang, Ya Zhang, Yanfeng Wang, and Qi Tian. A fourier-based framework for domain generalization. In *CVPR*, 2021. 3
- [45] Hang Zhang and K. Dana. Multi-style generative network for real-time transfer. *ArXiv*, abs/1703.06953, 2018. 3
- [46] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. *NeurIPS*, 2020. 3
- [47] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021. 3, 6, 11

Setting	Unseen client				Average
	P	A	C	S	
FedAvg (AISTATS'17) [34]	95.21	82.91	78.80	73.99	82.73
Jigen (CVPR'19) [4]	95.63	83.25	81.10	71.95	82.98
RSC (ECCV'20) [17]	94.55	83.20	79.99	72.79	85.31
MixStyle (ICLR'21) [47]	96.47	86.89	81.06	76.81	82.63
FedDG (CVPR'21) [33]	95.93	84.28	79.44	73.89	83.89
CCST (Overall, K=3)	<b>96.65</b>	<b>88.33</b>	<b>78.20</b>	<b>82.90</b>	<b>86.52</b>

Table 3: Compare the results of our CCST (Overall, K=3) with baselines that are trained with local iterations=3.

## A. Image reconstruction from style vectors

To evaluate the safety of the style vectors, we train a generator from a SOTA GAN [32] to reconstruct the image from its style vector. We train the generator until the validation loss converges sufficiently. The best model is selected with the highest validation average PSNR. The results are shown in Figure 5. We consider three scenarios:

- First, intra-client reconstruction (train and test on data from the same client). Note that this scenario is impossible in FL unless the client has already leaked their data. This is an extreme case to examine the best results the generator can achieve.

In Figure 5a, the diagonal image pairs show intra-client reconstruction results on PACS, and the bottom two pairs show intra-client reconstruction results on Camelyon17. We can see that the generator fails on intra-client reconstruction on the Camelyon17 dataset. For PACS, although the generator is possible to overfit the data within a single domain, this is only vulnerable when a large amount of data is leaked.

- Second, inter-client reconstruction (malicious client). It is possible that there exists a malicious client who wants to use its own data to reconstruct the images of other clients from the shared style vectors. From the results, we can hardly infer any content information except for overall color. Although (P, Photo) shows a rough shape of GT, it belongs to the intra-client scenario, which violates the FL setting.

In Figure 5a, image pairs that are not lie at diagonal line are inter-client reconstruction results. The figure shows that the inter-client reconstruction fails on both PACS and Camelyon17 datasets.

- Third, third-party reconstruction (pre-trained on large-scale images). More generally, if an outside attacker has compromised the style vectors and wants to reconstruct the images from the shared style vectors, they can train the reconstructor on a large-scale image dataset.

We train the generator on ImageNet and visualize the reconstruction results in Figure 5b. According to the

results, the pre-trained generator totally fails to reconstruct the target images.

Therefore, in the real FL scenarios, one can hardly reconstruct the original images merely from the shared style vectors.

## B. Time cost of extra computation

The extra computation time cost of our method is very low. Specifically, for the overall style computation, it takes 7 seconds for 2048 images with  $256 \times 256$  resolution. For image stylization, it takes 54 seconds to stylize 2048 images of  $256 \times 256$  resolution under either “Overall, K=3” or “Single, K=3” mode. The results are tested on an NVIDIA RTX 2080Ti GPU using PyTorch 1.11.0 with CUDA11.

## C. Training budget

To be fairer in the training budget, we increase the local training iterations of baselines methods from 1 to 3 to compare with our overall (K=3) method. The results are shown in Table 3. According to the results, more local iterations do not lead to obvious accuracy improvement for baseline methods, and our CCST (Overall, K=3) still outperforms all the baseline methods.

## D. Visualization of the FFT amplitude exchange results on the PACS

As shown in Figure 6, we visualize the results after the FFT amplitude exchange using single and overall amplitude on the PACS dataset [24]. We can see that the FFT amplitude exchange does not make a noticeable change to appearance or artistic style but only adds to some spatial repetitive texture and color patterns. This could be one of the reasons why FFT cannot outperform our CCST method. Because in the PACS dataset, we have large domain gaps such as that between photos and sketches. Simple changes in color, brightness, or background texture cannot make up the gap very well, while AdaIN style transfer can perform better by producing visually plausible artistic style transfer.

## E. AdaIN style transfer vs FFT amplitude exchange

In FedDG [33], the amplitude information in the frequency space of an image can also serve as a kind of style, while we utilize the IN statistics of each feature channel as style information. To explore the differences between the FFT [35] amplitude and IN statistics as style, we made a thorough comparison under our augmentation framework. The amplitude exchange alone without episodic learning in FedDG is equivalent to our framework with the setting of the single style when K=1.

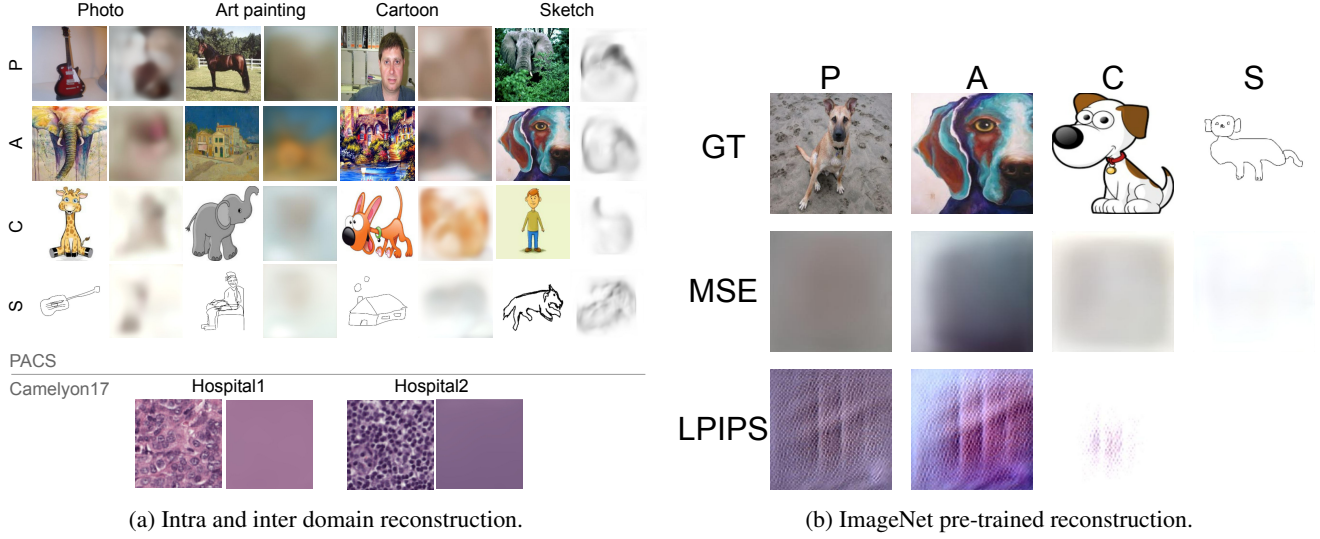


Figure 5: **(a)** Image reconstruction from the style vectors. (Ground truth, reconstructed image) pairs are shown. For PACS, the x-axis represents which domain the generator is trained on, the y-axis represents which domain the input style vectors are from. Diagonal image pairs show intra-client results. For Camelyon17, we show the intra-client reconstruction results only. The generator is trained with MSE loss. **(b)** We utilize the ImageNet pre-trained reconstructor to recover the PACS images from their style vectors. From top to bottom rows are the ground truth images, reconstruction results using generator trained with MSE loss, and reconstruction results using generator trained with LPIPS loss.

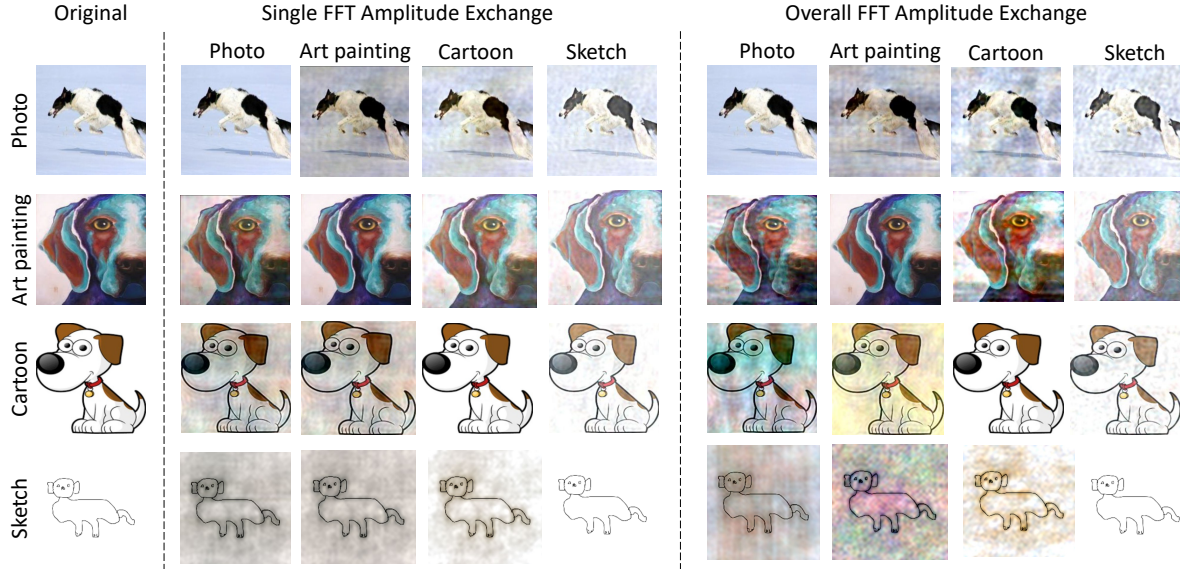


Figure 6: Visualization of images after the FFT amplitude exchange on the PACS dataset. Similar with Figure 7, we duplicate the content image if it is the same as the amplitude target image.

We show the comparison results in Table 4. Compared with our proposed method (using IN statistics as style), the FFT-based amplitude exchange method consistently performs worse under the same setting except for the setting of single style when  $K=3$ . Moreover, for the FFT-based style, the overall amplitude<sup>1</sup> of a domain fails to result in better

results than a single image FFT amplitude. In contrast, our framework has a significant boost when using overall style. With the help of our framework, the best result (single,  $k=3$ )

<sup>1</sup>We compute the overall amplitude by averaging the amplitudes of all images in this domain.



Table 4: Comparison between using FFT amplitude and IN statistics as style in our proposed cross-client style transfer framework with single and overall style under different K values. This table reports the performance with varying hyper-parameters of our framework on the PACS dataset with ResNet50 as the backbone.

Method	CCST (Ours)					FFT Amplitude Exchange				
	P	A	C	S	Avg.	P	A	C	S	Avg.
Single(K=1)	95.75	87.5	74.66	76.56	83.62	96.71	85.69	76.19	73.76	83.09
Single(K=2)	<b>96.77</b>	86.23	75.73	80.12	84.71	96.89	87.16	79.31	74.32	84.42
Single(K=3)	96.65	86.63	74.53	81.85	84.84	96.65	<b>86.87</b>	<b>79.74</b>	77.3	<b>85.14</b>
Overall(K=1)	95.69	86.67	75.85	77.37	83.90	96.89	86.38	78.92	72.36	83.64
Overall(K=2)	96.41	<b>88.72</b>	78.03	80.91	86.02	93.95	79.79	72.18	77.96	80.97
Overall(K=3)	96.65	88.33	<b>78.20</b>	<b>82.90</b>	<b>86.52</b>	95.21	81.25	73.34	<b>80.27</b>	82.52

of the FFT-based method can have a 2% improvement compared with the original version in FedDG (single, K=1). In general, our second-best result (overall, K=2) outperforms the best result of FFT amplitude exchange (single, K=3) by 0.9%; our best result (overall, K=3) outperforms the best result of FFT amplitude exchange (single, K=3) by 1.4%.

The experiments show that it is only practical to use the single image amplitude for the FFT amplitude exchange method. Utilizing the single image style mode makes the communication cost high due to the uploading and downloading of the style bank, leading to inflexibility. However, our CCST method can flexibly choose between single image style and overall domain style accordingly, especially the choice of using overall domain style to decrease the communication cost.

## F. PACS visualization

Figure 7 shows the visual results of cross-client style transfer with two types of styles. The overall domain style represents a more general and accurate client style, while the single image style brings more randomness.

## G. Visualization of style transfer results on the Office-Home

In this section, we show the qualitative results by visualizing images before and after the AdaIN [16]-based style transfer. In Figure 8, we show images of four different target domains in the Office-Home dataset [39]. Except for the art domain, samples from the other three domains show less domain gap. For each domain, we visualize the generated images using both random single image style and overall domain style. According to our experiment results, the overall style is usually more effective than using the single

image style. Random single image style sometimes may choose an image that is not representative for the whole domain. For example, in Figure 8, when transferring the clock image with the Clipart style into real-world style, the stylized image with overall style has a more colorful and representative style than that using random single image style.

## H. Additional experimental results

We show the results of our CCST with ResNet [13] as the backbone network on the PACS and Office-Home dataset in Table 5. For PACS dataset, using ResNet18 (Table 5) and ResNet50 (Table 2a) as backbone have consistent results: the overall style with K=3 leads to the best performance. When using ResNet18 as the backbone, the improvement upon baseline is more significant than that of using ResNet50.

For CCST results on the Office-Home dataset, besides  $K = 1$ , all other settings outperform the FedAvg in terms of the average accuracy. To explore the reason for the failure of CCST with  $K = 1$ , we visualize the images with style transfer as shown in Figure 8. From the visualization results, we can observe that the domain shift among domains of the Office-Home is smaller than that of the PACS dataset. For example, the product images collected on websites are similar to the real-world object images taken by a regular camera. Due to the slight differences between different domain styles and the randomness in single image style transfer, Single(K=1) achieves a lower accuracy than FedAvg on average. However, the overall domain style still shows a stronger representation capability and only has a minor performance gap compared with FedAvg. Overall, our best CCST results outperform the FedAvg baseline even with less domain shift in the Office-Home dataset.

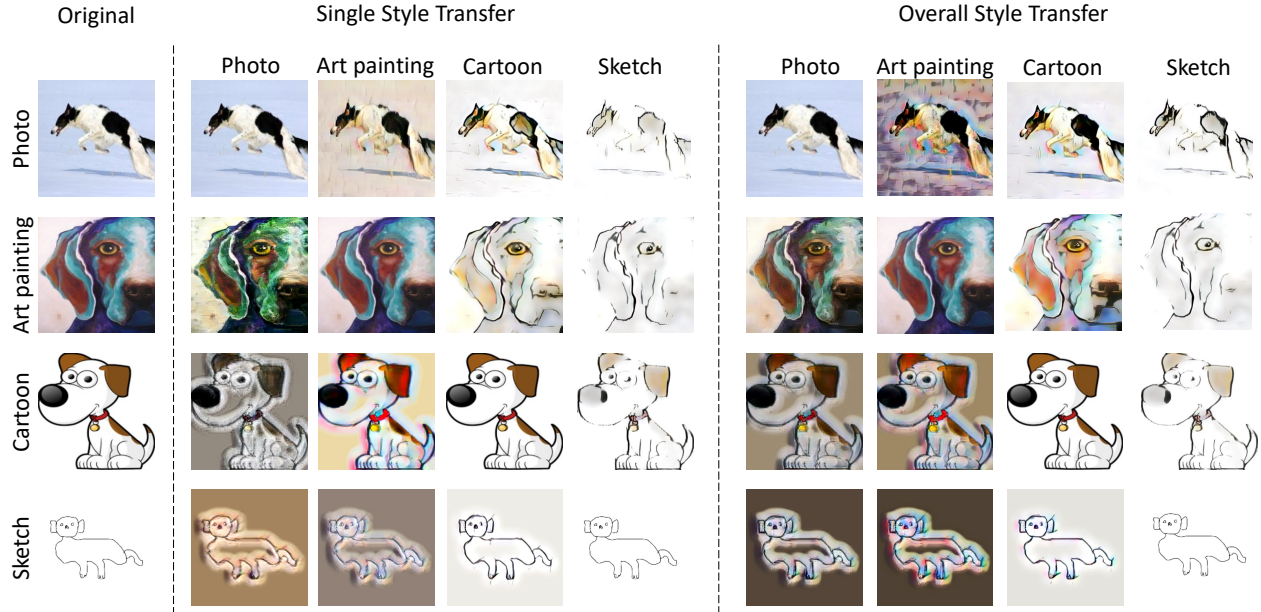


Figure 7: Visualization of stylized images on PACS. Note that if the content image is from the same domain of the style statistics, we directly copy the content image to the augmented dataset instead of transferring the same style to it.



Figure 8: Visualization of the AdaIN stylized images on the OfficeHome dataset. Note that if the content image is from the same domain as that of style statistics, we directly copy the content image to the augmented dataset instead of transferring the same style to it.

Table 5: Results of our CCST with different image style types and K values under PACS and Office-Home dataset. The backbone network is ResNet18. Each column represents a single unseen target client.

Method	PACS					Office-Home				
	P	A	C	S	Avg.	A	C	P	R	Avg.
FedAvg [34]	91.44	75.98	73.21	61.08	75.43	60.08	45.59	69.48	<b>72.82</b>	61.99
Single(K=1)	94.07	77.73	70.99	72.82	78.90	55.14	43.64	68.58	68.92	59.07
Single(K=2)	<b>95.27</b>	79.05	72.82	77.88	81.26	57.61	48.68	71.17	71.44	62.23
Single(K=3)	94.79	80.27	71.72	<b>80.86</b>	81.91	58.44	45.70	72.30	71.56	62.00
Overall(K=1)	94.19	79.88	72.14	75.41	80.41	59.47	47.88	67.91	70.87	61.53
Overall(K=2)	93.95	79.79	72.18	77.96	80.97	57.82	<b>50.52</b>	71.28	70.99	62.65
Overall(K=3)	95.21	<b>81.25</b>	<b>73.34</b>	80.27	<b>82.52</b>	59.05	50.06	<b>72.97</b>	71.67	<b>63.44</b>