ConstStyle: Robust Domain Generalization with Unified Style Transformation

Nam Duong Tran [1] Nam Nguyen Phuong [1] Hieu H. Pham [2] Phi Le Nguyen [1*] My T. Thai [3*]
[1] Institute for AI Innovation and Societal Impact, Hanoi University of Science and Technology
[2] VinUniversity [3] University of Florida

Abstract

Deep neural networks often suffer performance drops when test data distribution differs from training data. Domain Generalization (DG) aims to address this by focusing on domain-invariant features or augmenting data for greater diversity. However, these methods often struggle with limited training domains or significant gaps between seen (training) and unseen (test) domains. To enhance DG robustness, we hypothesize that it is essential for the model to be trained on data from domains that closely resemble unseen test domains—an inherently difficult task due to the absence of prior knowledge about the unseen domains. Accordingly, we propose ConstStyle, a novel approach that leverages a unified domain to capture domain-invariant features and bridge the domain gap with theoretical analysis. During training, all samples are mapped onto this unified domain, optimized for seen domains. During testing, unseen domain samples are projected similarly before predictions. By aligning both training and testing data within this unified domain, ConstStyle effectively reduces the impact of domain shifts, even with large domain gaps or few seen domains. Extensive experiments demonstrate that ConstStyle consistently outperforms existing methods across diverse scenarios. Notably, when only a limited number of seen domains are available, ConstStyle can boost accuracy up to 19.82% compared to the next best approach. [1]

1. Introduction

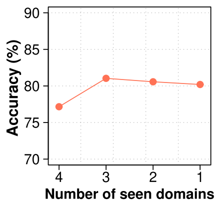
Deep neural networks (DNNs) often experience significant performance degradation when deployed on unseen test domains, which differ in distribution from the training data. This issue, known as domain shift, poses a fundamental challenge in real-world applications where data distributions are inherently diverse and unpredictable. Since it is impractical to collect data representative of all possible domains, bridging the gap between training (seen) and testing (unseen) domains is crucial for achieving robust performance—yet remains a major hurdle.

Domain generalization (DG) addresses this issue by training models that generalize well to unseen domains without relying on their data during training [20, 30]. Existing DG methods primarily focus on two strategies: (1) learning domain-invariant features, and (2) augmenting training data to enhance domain diversity. Invariant representation learning methods extract shared features across domains, minimizing the impact of domain-specific variations [2, 3, 11, 15]. However, these approaches typically require numerous diverse domains to effectively capture invariance, making them costly and often impractical. Alternatively, data augmentation methods increase domain diversity [10, 13, 26, 29], fundamentally based on the assumption that training with more data from diverse domains will produce better performance. Yet, our empirical analysis shows that increasing the number of seen domains does not always improve performance on unseen domains. In fact, training on fewer but carefully selected domains can sometimes yield better generalization, such as resulting in higher accuracy, as shown in Figure 1 or having greater class separation, as shown in Figure 2.

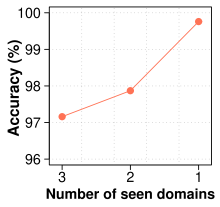
Furthermore, existing methods typically emphasize the training process, focusing on seen domains while neglecting the testing phase, where the domain gap becomes most pronounced. This oversight reveals another key limitation in current approaches, as they do not adequately address the unseen domain at inference time. This leads to a critical question: How can we enhance model generalization on unseen domains that are unknown during training but must be handled effectively during inference?

To overcome these challenges, we propose ConstStyle, a novel DG framework that unifies the treatment of both training and testing processes. ConstStyle leverages our newly introduced concept of *unified domain*, which acts as a common representation space that minimizes style discrepancies between different domains, both seen and unseen. Specifically, during training, all samples are aligned with this unified domain to extract consistent features. At inference, test samples undergo a style transformation to match the unified domain. By aligning data in this manner, ConstStyle reduces the impact of domain shifts, even when there

*Corresponding Authors
[1] Source code: `https://github.com/nduongw/ConstStyle`

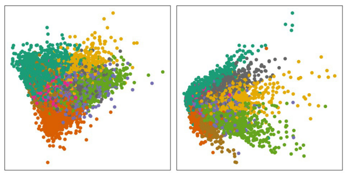


(a) Digits5 dataset.



(b) VLCS dataset.

Figure 1. Accuracy of ConvNet with varying numbers of seen domains during training. In some cases, using fewer seen domains leads to improved performance.


(a) Training ConvNet with 4 seen domains (left) vs single seen domain (right) on Digits5 dataset.

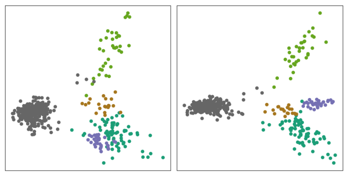

(b) Training ResNet18 with 3 seen domains (left) vs single seen domain (right) on VLCS dataset.

Figure 2. The illustrations indicate that training on a single seen domain can sometimes produce more defined class boundaries than training across multiple domains. Each dot represents an instance, with colors signifying the labels.

is a significant gap between seen and unseen domains. Our approach is grounded in a theoretical framework that guides the selection of the unified domain to optimize generalization, providing a robust solution to the DG problem.

The main contributions of this paper are as follows:

- We propose ConstStyle, a novel domain generalization framework that projects all data into a unified domain, addressing both training and testing phases to improve generalization on unseen domains.
- We present a theoretically grounded algorithm for defining such the unified domain, ensuring its reliability and effectiveness in reducing domain shifts.
- We introduce an alignment algorithm that projects unseen samples onto the unified domain during testing, with the goal of reducing information loss and closing the domain gap between the testing and training domains. This algorithm is backed by a theoretical analysis that establishes performance bounds for the model on unseen domains.
- Extensive experiments in various scenarios with benchmark datasets: PACS, Digit5, and Duke-Market101, show that ConstStyle improves precision up to 19.82%.

2. Related works

Domain Generalization (DG) is a pivotal research area aimed at enhancing model robustness and reliability, especially in high-stakes applications. DG focuses on out-of-distribution generalization, enabling models to perform well on unseen domains by training on one or more source domains. Various methods have been proposed to address domain shifts, generally categorized into the following approaches.

Invariant Representation Learning. Invariant representation learning seeks to extract features consistent across domains. Domain alignment methods, such as CIDDG [11], minimize distributional differences across domains. Rather than enforcing invariance across all features, disentangled feature learning approaches [1, 15] separate features into domain-specific and domain-invariant components, then learn them simultaneously. To further enhance this, authors in [2] introduce RIDG, a method that learns to ensure representations for samples within the same class remain consistent across domains by utilizing a rationale matrix and rationale invariant loss function, fostering improved generalization. Additionally, normalization techniques [3, 14] remove style information to produce invariant representations. Despite their promising results, these methods require a large number of domains to effectively extract invariant features, posing challenges when deploying the model in real-world environments.

Data Augmentation. Numerous strategies, based on image level, such as AugMix [7] and CutMix [23] are developed to achieve robust augmentation. Mixup [24] goes further by using pairwise linear interpolation in both image and label spaces, while Manifold Mixup [18] extends this to the feature level, boosting generalization. Additionally, [19] enhances the robustness by generating adversarial examples from hypothetical target domains, thereby strengthening the robustness of the model. Taking a different approach based on the observation that style statistics (mean and covariance) capture essential style information specific to each domain, style augmentation increases the training data quantity. For instance, StyDeSty [12] uses the stylization module to generate various style versions given a source domain. TF-Cal [27] utilizes linear combinations of two seen styles and combinations of a seen style with the representative style, while MixStyle [29] mixes the seen style within batches to increase the diversity of the source domain. DSU [10] further estimates feature statistic uncertainty to sample new style features, simulating out-of-distribution domains, while CSU [26] incorporates feature correlation in style mixing to retain semantic consistency, and Style Neophile [9] selects style prototypes from a style queue based on Maximum Mean Discrepancy, capturing source style distributions. However, as shown in our analysis 4.2.5, training with numerous domains does not always enhance model performance due to the dissimilarity between source and unseen domains.

To address the aforementioned challenges, we introduce ConstStyle, a novel approach that alleviates the domain shift problem by projecting all domains into a unified space. This reduces the impact of domain limitations while en-

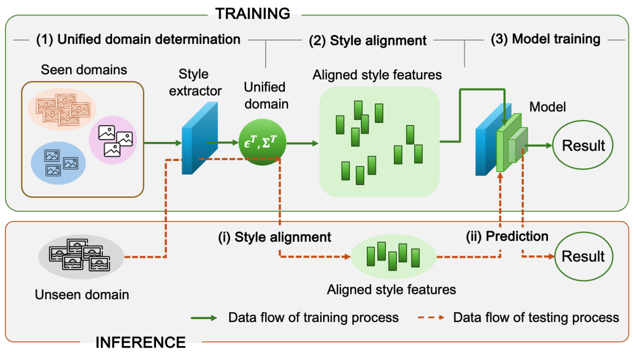


Figure 3. Overview of ConstStyle.

abling the model to learn consistent features, thereby enhancing its generalizability.

3. Our Proposed ConstStyle

3.1. Preliminaries

Notations and Definitions. Throughout this paper, we assume that there are N seen domains $\mathcal{S}_1, ..., \mathcal{S}_N$ and M unseen domains $\mathcal{U}_1, ..., \mathcal{U}_M$ (where N and M are not known in advance). We focus on the classification task. Let ω denote the model of interest, which consists of two components: a representation learning module, denoted by θ, and a classifier head, denoted by ζ. As noted in previous studies [29], the intermediate layers of θ often capture domain-specific style information. Thus, θ can be decomposed into two parts: $\theta = \theta_f(\theta_s(x))$, where θ_s serves as the style extractor, and θ_f generates the final representation of the image [8].

Definition 1 (Instance Style) *Given an input image x, let $z_x \in \mathbb{R}^{C\times H\times W}$ be the style feature of x, i.e., $z_x = \theta_s(x)$. We define by ϵ_x the style statistic of x, which captures the channel-wise mean and variance of z_x. Specifically, we express ϵ_x as $\epsilon_x = concat(\mu_x, \sigma_x)$, where $\mu_x \in \mathbb{R}^C$ and $\sigma_x \in \mathbb{R}^C$ are defined as follows:*

$$\mu_{x_c} = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} z_{x_{c,h,w}},$$

$$\sigma_{x_c} = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(z_{x_{c,h,w}} - \mu_{x_c})^2}.$$

Definition 2 (Domain Style) *Let $\mathcal{S}$ be a domain and $D_\mathcal{S}$ a set of data samples belonging to $\mathcal{S}$, $D_\mathcal{S} = \{(x_1, y_1), ..., (x_{|D_\mathcal{S}|}, y_{|D_\mathcal{S}|})\}$, and let ϵ_{x_i} be the style statistic of instance x_i. We define the style of domain $D_\mathcal{S}$, denoted as $\mathcal{P}_\mathcal{S}$ as multivariate normal distribution representing the style statistics of $D_\mathcal{S}$'s elements, i,e,., $\{\epsilon_{x_i}\}_{i=1}^{|D_\mathcal{S}|}$. The mean $\epsilon_\mathcal{S}$ and variance $\Sigma_\mathcal{S}$ of $\mathcal{P}_\mathcal{S}$ is calculated as follows:*

$$\epsilon_\mathcal{S} = \frac{1}{|D_\mathcal{S}|}\sum_{i=1}^{|D_\mathcal{S}|}\epsilon_{x_i}, \Sigma_\mathcal{S} = \frac{1}{|D_\mathcal{S}|}\sum_{i=1}^{|D_\mathcal{S}|}(\epsilon_{x_i} - \epsilon_\mathcal{S})^T(\epsilon_{x_i} - \epsilon_\mathcal{S}).$$

Problem Formulation. Let $(\mathcal{X}, \mathcal{Y})$ denote the space of inputs and labels. The DG problem asks us to identify a model ω^*, which is trained on all seen domains $\{\mathcal{S}_k\}_{k=1}^N$, and performs well on both seen domains $\{\mathcal{S}_k\}_{k=1}^N$ and unseen domains $\{\mathcal{U}_j\}_{j=1}^M$ ($j = 1, ...M$). This can be formulated as follows:

$$\omega^* = \arg\min_{\omega}\left(\sum_{j=1}^{M}\mathop{\mathbb{E}}_{(u,y)\in\mathcal{U}_j}[l(\omega(u), y)] + \sum_{k=1}^{N}\mathop{\mathbb{E}}_{(x,y)\in\mathcal{S}_k}[l(\omega(x), y)]\right),$$

where l is the loss function, which is varied depending on the task (e.g. cross entropy loss for the classification task).

3.2. Overview

Figure 3 presents the workflow of ConstStyle, designed to enhance accuracy across both seen and unseen domains by addressing both training and testing phases. In the training phase, ConstStyle follows three main steps: (i) determining the style of the unified domain; (ii) transforming the style of all samples in the training data set (from the seen domains) to match the unified domain's style; and (iii) training the model using these style-aligned samples. In the testing phase, samples from the unseen domain are first adjusted to align with the unified domain style before being processed by the trained model for inference.

A central challenge is defining an appropriate unified domain, especially since unseen domains are unknown during training. To tackle this, we introduce an algorithm that constructs the unified domain using only information from seen domains, optimizing its selection for better performance on seen data. However, since the unified domain is defined solely using information from seen domains, a significant gap may exist between the unified and unseen domains. This can lead to substantial data distortion when mapping unseen samples to the unified domain, potentially decreasing model performance. To address this issue, we propose a partial alignment algorithm that efficiently aligns the test samples to the unified domain. This solution balances the alignment of the testing sample style with the trained style (thus addressing the domain shift) while preserving the essential information of the testing data (thus avoiding performance degradation due to excessive data distortion).

The details of the algorithm for determining the style of the unified domain are presented in Section 3.3, while the algorithms for the training and testing process are detailed in Sections 3.4 and 3.5.

3.3. Unified Domain Determination

The unified domain is designed to maximize the model's performance on seen domains. To this end, our unified domain is defined as follows.

Definition 3 (Unified Domain) *Given N seen domains $\{S_k\}_{k=1}^N$, each S_k has an associated domain style $\mathcal{P}_{S_k}$ follows a distribution $\mathcal{N}(\epsilon_{S_k}, \Sigma_{S_k})$. The unified domain $\mathcal{T}$ is the one with the domain style that serves as the Barycenter of $\{\mathcal{P}_{S_k}\}_{k=1}^N$ (denoted as $\mathcal{B}$). Specifically, the Barycenter [31] is a distribution, i.e., $\mathcal{B} \sim \mathcal{N}(\epsilon_B, \Sigma_B)$, that intuitively minimizes the total distance to $\{\mathcal{P}_k\}_{k=1}^N$, where ϵ_B is determined as follows:*

$$\epsilon_B = \frac{1}{N}\sum_{k=1}^{N} \epsilon_{S_k}.$$

The covariance matrix Σ_B is obtained by solving an iterative optimization problem. Starting with an initial covariance matrix $\Sigma_{B_0} = \frac{1}{N}\sum_{k=1}^N \Sigma_{S_k}$, the update formula at each iteration i is given by:

$$\Sigma_{B_{i+1}} = \frac{1}{N}\sum_{k=1}^{N} \left(\Sigma_{B_i}^{\frac{1}{2}} \Sigma_{S_k} \Sigma_{B_i}^{\frac{1}{2}}\right)^{\frac{1}{2}}.$$

Next, we explain the rationale behind our unified domain and present a theoretical analysis of the model's performance when trained with data projected onto this domain.

Theoretical Analysis. Let D_k denote the dataset from domain S_k, and D_k^T represent D_k after it has been mapped to the unified domain. Naturally, the model ω often achieves optimal performance on the seen domain S_k when trained directly on the original dataset D_k. Therefore, we aim at designing a unified domain such that training ω on D_k^T achieves performance comparable to training on D_k.

Lemma 1 *Let ω^* and ω^T be the models trained using D_k and D_k^T, respectively. Let us denote by $L^{\mathcal{S}_k}$ and $L^{\mathcal{S}_k^T}$ the empirical losses of ω^* and ω^T calculated over D_k and D_k^T, respectively. Then, the gap between $L^{\mathcal{S}_k}$ and $L^{\mathcal{S}_k^T}$ is bounded as follows:*

$$L^{\mathcal{S}_k^T} - L^{\mathcal{S}_k} \leq \beta \times (\mathcal{D}_\mu(\mathcal{T}, \mathcal{S}_k) + \mathcal{D}_\sigma(\mathcal{T}, \mathcal{S}_k)), \tag{1}$$

where $\mathcal{D}_\mu(\mathcal{T}, \mathcal{S}_k)$ and $\mathcal{D}_\sigma(\mathcal{T}, \mathcal{S}_k)$ denote the distances between the means and standard deviations of distributions of unified domain $\mathcal{T}$ and seen domain $\mathcal{S}_k$, respectively; β is the upper bound of Lipschitz coefficient of the loss function on all seen domains. (The proof is shown in Suppl. B.1).

Theorem 1 *The disparity in empirical losses computed across all the seen data, between the model trained on data projected onto the unified domain and those trained on the original data from the seen domains $\{S_k\}_{k=1}^N$, is bound by the following inequality:*

$$\sum_{k=1}^{N}\left(L^{\mathcal{S}_k^T} - L^{\mathcal{S}_k}\right) \leq \beta \times \sum_{k=1}^{N} (\mathcal{D}_\mu(\mathcal{T}, \mathcal{S}_k) + \mathcal{D}_\sigma(\mathcal{T}, \mathcal{S}_k)). \tag{2}$$

(The proof is presented in Suppl. B.2)

Theorem 1 suggests that the unified domain should be selected to minimize the distance on the right side of (2).

Practical Algorithm. Now, assuming that we have n training samples: $\{(x_i, y_i)\}_{i=1}^n$, with each x_i associated with a style statistic ϵ_{x_i}. If the seen domains $\{S_k\}_{k=1}^N$ related to these samples are known, Definition 3 could be applied directly to determine the unified domain style.

In practical scenarios, however, we usually have only the training samples without specific domain labels. To address this, we introduce an efficient algorithm to estimate the style statistics of the seen domains before applying Definition 3. Specifically, we utilize a Gaussian Mixture Model (GMM) to capture the distribution of the style statistics $\{\epsilon_{x_i}\}_{i=1}^n$. The GMM Expectation-Maximization Algorithm is then employed to cluster $\{\epsilon_{x_i}\}_{i=1}^n$ into N' distinct groups. The number of clusters N' is treated as a hyperparameter (which can be set to the number of seen domains if this information is known). Since samples from the same domain typically exhibit similar style statistics, each cluster approximately represents style statistics from the same domain. Consequently, the distribution of each cluster can be considered as an approximate style statistic for that domain. Finally, Definition 3 is applied to the normal distributions associated with these clusters to establish the unified domain's style statistic.

In practice, calculating the exact Barycenter can be computationally intensive. Thus, a straightforward yet effective approximation involves defining domain style of the unified domain, i.e., $\mathcal{P}_T = \mathcal{N}(\epsilon^T, \Sigma^T)$, by averaging the style distribution of the clusters as follows:

$$\epsilon^T = \frac{1}{N'}\sum_{k=1}^{N'} \epsilon_{\mathcal{C}_k}, \Sigma^T = \frac{1}{N'}\sum_{k=1}^{N'} \Sigma_{\mathcal{C}_k}, \tag{3}$$

where N' is the number of clusters, and $\epsilon_{\mathcal{C}_k}$, $\Sigma_{\mathcal{C}_k}$ are mean and covariance matrix of the Gaussian distribution associated with cluster $\mathcal{C}_k$, respectively. Details of the unified domain determination is provided in Algorithm 2 (Supplementary).

3.4. Training Process

Style transformation involves adapting the original style of training samples to match the style of the unified domain. This process relies on a trained model to extract the style features from the samples. To achieve this, we split the ConstStyle training process into two stages: *Initial Training* and *Unified-Style Training*. In the initial training phase, we train a model using the original training data to develop a feature extractor capable of capturing the style features of the samples. In the subsequent phase, the feature extractor is used to transform all training data to match the style of the unified domain. The model is then trained using these style-aligned samples. To save training costs, we perform the initial training phase for only a few epochs (instead of training until convergence), to establish the initial unified domain. Afterward, we utilize the feature extractor from unified-style training.

Initial Training. Initially, the model ω is trained on the original training dataset using the traditional Empirical Risk Minimization (ERM) approach. Assume that the model obtained after the first training epoch is $\omega^o = \zeta^o(\theta_f^o(\theta_s^o))$, where θ_s^o denotes the style feature extractor, θ_f^o the remaining encoder component, and ζ^o the classification head. Then, each training sample x_i is fed through $\theta_s^o(.)$ to get its original style feature $z_{x_i}^o = \theta_s^o(x_i)$. These style features are used to determine the initial unified domain $\mathcal{T}^o$ as described in Section 3.3. Once the unified domain is acquired, the training process shifts to the second phase, where the model is trained with data transformed to the unified domain through a style transformation procedure.

Unified-Style Training. ω^o and $\mathcal{T}^o$ are used as the initial points for this training process. At each training epoch during this phase, each image x_i in the training batch is fed through the current style extractor $\theta_s(.)$ to get its original style feature $z_{x_i} = \theta_s(x_i)$. This original style statistic is then aligned with a random style statistic $\epsilon_s = (\mu_s, \sigma_s)$ sampled from the unified domain's style distribution $\mathcal{P}_{\mathcal{T}} = \mathcal{N}(\epsilon_T, \Sigma_T)$ as follows:

$$z_{x_i}^T = \sigma_s \times \frac{z_{x_i} - \mu_x}{\sigma_x} + \mu_s. \quad (4)$$

After transformation, the new style feature $z_{x_i}^T$ is fed into the model ω for further training, i.e., $\omega = \zeta(\theta_f(z_{x_i}^T))$. Backpropagation is performed across all components of ω, including the feature extractor $\theta_s(.)$. The unified domain $\mathcal{T}$ is periodically updated every γ epochs, a strategy that improves unified domain quality while preserving training stability. It is worth noting that, through this projection process, the model is trained on $E * D$ different variations of style features (where E is the number of training epochs and D is the total seen data), all aligned with the unified domain. This strategy not only expands the training dataset but also enriches the diversity of style features specific to the unified domain, thereby enhancing the model's adaptability to it. Details of the training process are presented in Algorithm 1 in Supplementary.

3.5. Inference Process

To bridge the domain gap between test and training data, we introduce a novel approach that aligns the style of test samples with the unified domain prior to inference. The challenge here is that since the unified domain is entirely defined based on the seen domains, the gap between the unified domain and the test domains (unseen domains) can be quite large, leading to the potential loss of original characteristics in the data after being aligned with the unified domain[2]. To address this issue, we employ a partial projection strategy that balances transforming the style of test samples to match the unified domain while preserving their original characteristics.

Specifically, let $\omega^* = \zeta^*(\theta_f^*(\theta_s^o))$ be the model obtained after the training phase. In the inference phase, each testing sample u is firstly fed into θ_s^o to get the style feature z_u^o. This style feature is then partially aligned with the unified domain's style to generate a new style feature, denoted as z_u^T. This z_u^T is subsequently input into $\zeta^*(\theta_f^*(.))$ to yield the final prediction result. The formula below is used to align the style z_u^o of a test sample u:

$$z_u^T = (\alpha \times \sigma_u + (1-\alpha) \times \sigma_T)\,\frac{z_u^o - \mu_u}{\sigma_u} + (\alpha \times \mu_u + (1-\alpha) \times \mu_T)\,, \quad (5)$$

where (μ_u, σ_u) is the original style statistic of u, and $\epsilon_T = (\mu_T, \sigma_T)$ is the mean; α is a hyperparameter in the range of $(0, 1)$ controlling the extent to which the original feature is preserved. Specifically, $\alpha = 0$ indicates that the testing data is completely mapped to the unified domain, while $\alpha = 1$ keeps the test data in its original state. We perform an ablation study with different values of α to examine its effects and assess our alignment algorithm (see Supplementary E.4). Additionally, we present below a theorem that establishes a bound on the distance between the empirical losses of the model on unseen and seen domains.

Theorem 2 *Let $\mathcal{S}$ and $\mathcal{U}$ be the set of data from all seen and unseen domains, respectively; D_{ST} and D_{UT} be the set of seen data and unseen data after projected onto the unified domain using our algorithm; $L^{\mathcal{U}^T}$ and $L^{\mathcal{S}^T}$ be the empirical losses of models trained by D_{ST} and D_{UT} calculated over the seen and unseen data, respectively. The difference of $L^{\mathcal{U}^T}$ and $L^{\mathcal{S}^T}$ is bounded by the following inequality:*

$$L^{\mathcal{U}^T} - L^{\mathcal{S}^T} \leq \alpha \times \beta \times (\mathcal{D}_\mu(\mathcal{U}, \mathcal{T}) + \mathcal{D}_\sigma(\mathcal{U}, \mathcal{T})) + \epsilon \times \sqrt{2.Tr(I)}, \quad (6)$$

where $\mathcal{D}_\mu(\mathcal{T}, \mathcal{U})$ and $\mathcal{D}_\sigma(\mathcal{T}, \mathcal{U})$ represent the distances between the mean and variance of the distributions of $\mathcal{T}$ and $\mathcal{U}$. $Tr(I)$ represents the trace of the identity matrix I, which is the sum of the diagonal elements of the matrix. I is the identity matrix with dimensions $C \times H \times W$, where C, H, W are the channel, height, and width dimensions of the output of θ_s. (The proof is provided in Suppl. B.3).

We provide details of the inference process in Algorithm 3 in Supplementary.

Method	Venue	In-domain combinations				Avg
		A, C, S	P, C, S	P, A, S	P, A, C	
ERM	-	95.02	95.91	95.75	97.08	95.94
MixStyle [29]	ICLR 2021	94.66	95.78	96.01	96.59	95.76
DSU [10]	ICLR 2022	94.78	96.65	**96.39**	**97.40**	96.30
CSU [26]	WACV 2024	94.78	96.52	96.13	97.07	96.12
ConstStyle	**Ours**	**95.37**	**97.02**	**96.39**	97.24	**96.50**

Table 1. In-domain performance of ConstStyle compared with baselines on PACS datasets. P, A, C, S denote Photo, Art, Cartoon, Sketch. The best result is colored **purple** and the second best result is colored blue.

4. Experimental Evaluation

We conducted a series of experiments to assess ConstStyle's effectiveness across various scenarios. The evaluation covered three primary tasks: image classification, image corruption, and instance retrieval, offering a thorough analysis of the method's robustness under different conditions.

4.1. Settings

Image Classification. We address the style-shift problem and evaluate our method on the PACS dataset [4], a Domain Generalization benchmark with four styles (Photo, Art, Cartoon, Sketch). Following [22], we conduct experiments under two scenarios: (1) a single unseen domain, where the model trains on three domains and tests on the fourth [10, 26, 29], and (2) multiple unseen domains, where training is further restricted to assess generalization. Additionally, we evaluate the Digits5 dataset [21] to improve robustness across five domains.
Image Corruption. We further assess the robustness of our method against image corruption using the CIFAR10-C dataset [6], which includes 19 types of corruption at five severity levels. Higher levels indicate stronger corruption. CIFAR10 serves as the source domain and CIFAR10-C as the target domain.
Instance Retrieval. For the instance retrieval task, we evaluate re-ID methods, matching individuals across camera views. Market1501 [28] and Duke [16] are used interchangeably for training and testing. Performance is assessed using the ranking accuracy and mean average precision (mAP). In all experiments, we set $\alpha = 0.6$ for PACS dataset and $\alpha = 0.5$ for Digit5 dataset and report the results with ERM refers to the approach that trains the model using Empirical Risk Minimization loss. Details of experimental settings are provided in Suppl. C.

4.2. Image Classification

4.2.1. In-domain Performance

We first evaluate ConstStyle's performance on the seen domains. As shown in Table 1, ConstStyle outperforms existing methods in three out of four cases and achieves the highest overall accuracy. Compared to the standard ERM approach, ConstStyle improves performance by 0.56% and further exceeds the second-best method by an additional 0.2% in accuracy.

Method	Venue	Domains				Avg
		Art	Cartoon	Photo	Sketch	
ERM	-	77.10	77.77	96.40	68.17	79.86
Crossgrad [17]	ICLR 2018	78.12	77.90	96.64	70.64	80.82
Mixup [24]	ICLR 2018	78.71	74.53	96.16	66.24	78.91
Cutmix [23]	ICCV 2019	77.49	73.33	96.34	69.80	79.24
EDFMix [25]	CVPR 2022	83.05	81.05	96.64	76.50	84.31
RIDG [2]	ICCV 2023	80.17	78.32	96.82	72.32	81.90
MixStyle [29]	ICLR 2021	81.25	80.03	96.82	72.17	82.57
DSU [10]	ICLR 2022	83.94	81.10	96.23	79.05	85.08
CSU [26]	WACV 2024	84.62	82.21	96.41	78.11	85.33
ConstStyle	**Ours**	**85.45**	**82.42**	**96.89**	**82,32**	**86,77**

Table 2. Performance of ConstStyle compared with baselines on PACS datasets.The best result is colored **purple** and the second best result is colored blue.

Method	Venue	Domains					Avg
		MNISTM	SVHN	SYN	USPS	MNIST	
ERM	-	67.45	77.16	86.43	96.90	98.30	85.24
Crossgrad[17]	ICLR 2018	69.03	77.20	86.93	96.9	69.03	85.65
Mixup [24]	ICLR 2018	64.08	79.32	81.34	94.51	98.30	83.50
Cutmix[23]	ICCV 2019	65.45	79.55	84.60	95.90	98.30	84.36
EDFMix[25]	CVPR 2022	70.95	77.55	86.94	96.98	98.30	86.14
RIDG[2]	ICCV 2023	67.35	79.18	86.86	**97.04**	98.30	85.95
MixStyle[29]	ICLR 2021	61.48	57.18	64.17	87.58	96.90	73.06
DSU[10]	ICLR 2022	67.84	77.01	87.21	96.55	98.30	85.38
CSU[26]	WACV 2024	68.25	78.60	86.70	96.40	98.30	85.65
ConstStyle	**Ours**	**71.51**	**79.9**	**87.90**	96.80	**98.30**	**86.88**

Table 3. Performance of ConstStyle compared with baselines on Digits5 datasets. The best result is colored **purple** and the second best result is colored blue.

4.2.2. Generalization on Multi-domain Classification

Tables 2 and 3 provide a summary of the comparison between ConstStyle and existing approaches on the image classification task. As shown, ConstStyle surpasses all other methods on the PACS dataset and achieves the highest performance on four out of five unseen domains in the Digits5 dataset. Regarding the average accuracy across unseen domains, ConstStyle leads on both datasets. For particularly challenging domains such as Art, Cartoon, and Sketch, ConstStyle improves performance by 0.21% to 4.21% over the second-best method. In the Sketch domain specifically, where the unseen domain style statistics vary significantly from the training domains, ConstStyle achieves the largest improvement over other methods, underscoring its capability to handle substantial domain gaps. On the Digits5 dataset, ConstStyle also shows superior performance over baseline methods, with the greatest improvement observed in the MNISTM domain, which presents the greatest challenge. These results highlight ConstStyle's strong adaptability and robustness in diverse, difficult settings.

[2]This issue does not arise with data from seen domains, as the unified domain is designed to closely align with them.

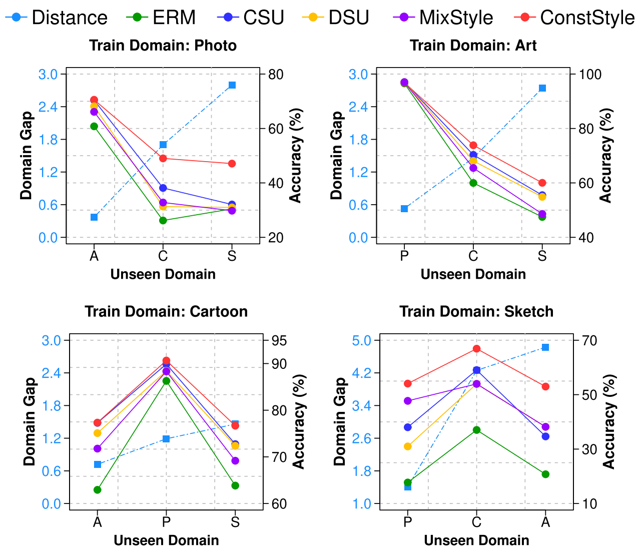


Figure 4. Effect of domain gap. ConstStyle achieves significantly better performance than other methods in handling severe domain gaps.

4.2.3. Robustness Against the Numbers of Unseen Domains

Previous studies have generally evaluated their methods with limited setups, often using scenarios with only a single unseen domain. To offer a more thorough evaluation, we conduct experiments with various numbers of unseen domains. Specifically, we incrementally increase the number of unseen domains to two for the PACS dataset and three for the Digits5 dataset. Detailed results for the PACS dataset are shown in Table 4, while those for the Digits5 dataset are provided in the Suppl. D.1. As shown, ConstStyle continues to achieve the highest accuracy even when the number of seen domains is reduced and the number of unseen domains is increased, demonstrating its strong generalization capability. Specifically, ConstStyle improves accuracy by 1.36% when the test domains are Art and Cartoon, and achieves a notable 5.91% increase over the state of the art when Cartoon and Sketch are the unseen domains. On average, ConstStyle increases overall performance by 2.43%, underscoring its consistent effectiveness as the number of training domains varies.

4.2.4. Impacts of Domain Gap

We conduct experiments to evaluate the performance of ConstStyle when confronted with unseen domains that may exhibit varying degrees of distance from the seen domains. Specifically, we train the model on a specific seen domain, then perform inference on various unseen domains, and investigate how the model's accuracy changes. We utilize the Fréchet distance [5] (also known as the 2-Wasserstein distance) to model the gap between domains' style distributions. The results are shown in Figure 4. In general, as the distance between domains increases, the performance

Method	Venue	A,P	C,P	P,S	A,C	A,S	C,S	Avg
ERM	-	74.50	84.92	76.27	64.79	69.31	52.18	70.32
Crossgrad[17]	ICLR 2018	74.26	85.12	76.92	64.16	69.96	51.27	70.28
Mixup[24]	ICLR 2018	76.46	82.78	73.24	64.34	66.23	51.27	68.51
Cutmix[23]	ICCV 2019	73.66	82.63	76.38	64.43	69.76	49.09	69.32
EFDMix[25]	CVPR 2022	78.56	86.72	80.90	71.10	76.28	54.28	74.64
RIDG[2]	ICCV 2023	75.49	84.57	77.00	67.75	70.03	53.42	71.37
MixStyle[29]	ICLR 2021	75.60	86.79	80.54	72.10	73.34	55.58	73.99
DSU[10]	ICLR 2022	80.36	86.14	83.56	73.49	77.49	62.40	77.24
CSU[26]	WACV 2024	82.92	**87.49**	83.52	74.59	77.42	64.30	78.38
ConstStyle	**Ours**	**84.64**	**87.49**	**85.28**	**75.95**	**81.29**	**70.21**	**80.81**

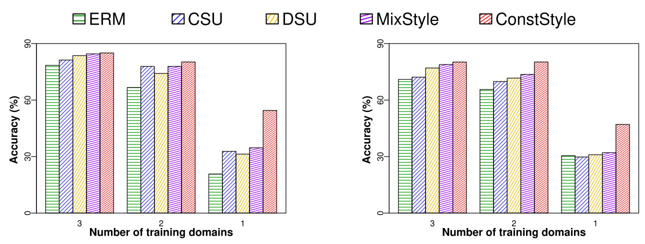
Table 4. Performance comparison on the PACS dataset across six scenarios with different combinations of unseen domains. The best result is colored **purple** and the second best result is colored blue.

of the methods tends to decrease. However, ConstStyle consistently delivers the highest accuracy and the slowest rate of performance decline across all scenarios. Specifically, when the training domain is Photo, ConstStyle outperforms CSU, with the performance gap ranging from 0.14% to 15.03% as the distance between seen and unseen domains grows. Additionally, ConstStyle shows a performance gap of up to 4.54% when the training domain is Art and 3.95% when the training domain is Cartoon. These results highlight the importance of projecting data onto a common domain, which helps mitigate domain gaps and enables the extraction of the most relevant cross-domain features.

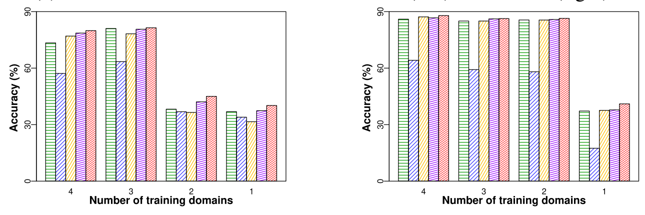
4.2.5. Impacts of the Number of Seen Domains

Performance comparison between the models. We compare the performance of the methods with the varying number of seen domains (Figure 5). As shown, reducing the number of training domains tends to negatively impact the models' generalizability due to the less diverse features learned from the data. However, there are some special cases where training the model with fewer domains is more effective compared with more domains, which will be discussed later. Despite the significant decline in performance across methods, ConstStyle exhibits the slowest degradation, which can be observed when comparing with the second-best method, CSU. Specifically, ConstStyle maintains an accuracy advantage of up to 19.82% when trained on Sketch and tested on Art, and up to 15.02% when trained on Cartoon and tested on Sketch, outperforming CSU. In the Digit5 dataset, ConstStyle achieves a performance gap of up to 2.77% when the test domain is SVHN and up to 3.19% when the test domain is SYN, compared to CSU.

When does training with fewer domains result in better performance? As previously discussed, training with a wide range of domains does not necessarily yield better performance. To explore this further, we conducted experiments on PACS dataset, varying the number of training domains. In each experiment, we set Sketch as the test domain and began by training the model with Art as the only seen domain. We then gradually added more domains to the training set, calculating the distance between the seen and



(a) PACS dataset with Test domains are Art (left) and Sketch (right).



(b) Digit5 dataset with Test domains are SVHN (left) and SYN (right).

Figure 5. Effects of the number of training domains. ConstStyle consistently delivers the best performance across all scenarios.

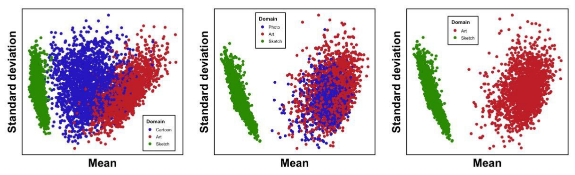


Figure 6. Style statistics for different domain combinations when training ResNet18 with Sketch as the test domain. Accuracies for each domain combination are 67.44%, 44.57%, and 46.23%.

unseen domains after each addition. These results, illustrated in Figure 6, reveal that model performance improves only when the added domains decrease the domain gap to the unseen domain.

4.3. Robustness Against Image Corruption

Table 5 presents the results, revealing that as corruption levels increase, the performance advantage of ConstStyle over existing methods becomes more pronounced. ConstStyle secures the top performance in four out of five cases, with the only exception occurring at the lowest corruption level, where unseen domains closely match the seen domains. On average, ConstStyle achieves the highest performance with a 1.04% improvement over the second-best method. Compared to the ERM method, which performs best at the lowest corruption level, ConstStyle surpasses it by 0.36% to 15.83% across higher corruption levels. Moreover, ConstStyle outperforms other style-based methods with improvements ranging from 0.75% to 4.96%, emphasizing its resilience in handling heavily corrupted datasets. These results underscore ConstStyle's robustness in handling highly corrupted datasets.

Method	Level of corruption					Avg
	level 1	level 2	level 3	level 4	level 5	
ERM	**87.33**	80.29	70.57	64.81	47.80	70.16
MixStyle[29]	87.32	79.90	71.83	66.54	58.67	73.13
DSU[10]	86.91	79.40	68.68	64.14	49.44	69.71
CSU[26]	86.73	79.87	70.66	65.51	60.88	72.73
ConstStyle	86.59	**80.65**	**72.03**	**67.96**	**63.63**	**74.17**

Table 5. Comparison of methods under different corruption levels, conducted on the CIFAR10-C datasets. The best result is colored **purple** and the second best result is colored blue.

Method	Market → Duke				Duke → Market			
	mAP	R1	R5	R10	mAP	R1	R5	R10
ERM	13.8	27.1	40.5	46.9	21.9	45.9	65.6	72.7
MixStyle[29]	19.5	37.0	52.0	58.5	24.6	52.2	70.9	78.4
DSU[10]	21.6	40.2	54.0	59.8	25.5	55.7	73.0	79.2
CSU[26]	24.1	44.2	59.2	65.1	24.9	55.0	72.2	78.2
ConstStyle	**26.1**	**44.7**	**61.1**	**67.6**	**27.0**	**56.5**	**75.0**	**81.1**

Table 6. Performance of the methods on Instance retrieval task. The best result is colored **purple** and the second best result is colored blue.

4.4. Generalization in Instance Retrieval

The results shown in Table 6 highlight that ConstStyle substantially outperforms all other methods in both settings. While other methods encounter difficulties with new domains due to their dependence on generating instances from the existing dataset, ConstStyle addresses this challenge through its unified domain approach. Compared to the second-best method, ConstStyle achieves improvements in mean average precision (*mAP*) ranging from 0.1% to 2.8%.

5. Conclusion

This paper introduces ConstStyle, a novel approach for addressing the domain shift problem. The key concept of ConstStyle is to align data to a unified domain prior to both training and testing, enabling the capture of domain-invariant features and reducing discrepancies with unseen domains. This alignment approach mitigates domain shift effects and maintains performance stability, even with fewer training domains. ConstStyle consistently outperforms existing methods on style-shift datasets, achieving up to a 19.82% accuracy improvement over the next best approach. Additionally, we provide a theoretical analysis on the performance bounds for both seen and unseen domains. Future work will explore addressing other types of domain shift, such as feature shift.

6. Acknowledgement

This research is funded by Hanoi University of Science and Technology (HUST) under grant number T2024-TD-002.

References

[1] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 301–318. Springer, 2020. 2

[2] Liang Chen, Yong Zhang, Yibing Song, Anton van den Hengel, and Lingqiao Liu. Domain generalization via rationale invariance. In *ICCV*, 2023. 1, 2, 6, 7

[3] Sungha Choi, Sanghun Jung, Huiwon Yun, Joanne T Kim, Seungryong Kim, and Jaegul Choo. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021. 1, 2

[4] Li Da, Yang Yongxin, Song Yi-Zhe, and M. Hospedales Timothy. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 6

[5] DC Dowson and BV666017 Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 1982. 7

[6] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. 6

[7] Dan Hendrycks*, Norman Mu*, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple method to improve robustness and uncertainty under data shift. In *International Conference on Learning Representations*, 2020. 2

[8] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 3

[9] Juwon Kang, Sohyun Lee, Namyup Kim, and Suha Kwak. Style neophile: Constantly seeking novel styles for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7130–7140, 2022. 2

[10] Xiaotong Li, Yongxing Dai, Yixiao Ge, Jun Liu, Ying Shan, and LINGYU DUAN. Uncertainty modeling for out-of-distribution generalization. In *International Conference on Learning Representations*, 2022. 1, 2, 6, 7, 8

[11] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 624–639, 2018. 1, 2

[12] Songhua Liu, Xin Jin, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Stydesty: Min-max stylization and destylization for single domain generalization. In *Forty-first International Conference on Machine Learning*, 2024. 2

[13] Oren Nuriel, Sagie Benaim, and Lior Wolf. Permuted adain: Reducing the bias towards global statistics in image classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9482–9491, 2021. 1

[14] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the european conference on computer vision (ECCV)*, 2018. 2

[15] Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In *International conference on machine learning*, 2020. 1, 2

[16] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, 2016. 6

[17] Shankar Shiv, Piratla Vihari, Chakrabarti Soumen, Chaudhuri Siddhartha, Jyothi Preethi, and Sarawagi Sunita. Generalizing across domains via cross-gradient training. In *ICLR*, 2018. 6, 7

[18] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pages 6438–6447. PMLR, 2019. 2

[19] 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*, 2018. 2

[20] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and S Yu Philip. Generalizing to unseen domains: A survey on domain generalization. *IEEE transactions on knowledge and data engineering*, 2022. 1

[21] LeCun Yann, Bottou Léon, Bengio Yoshua, and Haffner Patrick. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 6

[22] Han Yu, Xingxuan Zhang, Renzhe Xu, Jiashuo Liu, Yue He, and Peng Cui. Rethinking the evaluation protocol of domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21897–21908, 2024. 6

[23] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 2, 6, 7

[24] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 2, 6, 7

[25] Yabin Zhang, Minghan Li, Ruihuang Li, Kui Jia, and Lei Zhang. Exact feature distribution matching for arbitrary style transfer and domain generalization. In *CVPR*, 2022. 6, 7

[26] Zheyuan Zhang, Bin Wang, Debesh Jha, Ugur Demir, and Ulas Bagci. Domain generalization with correlated style uncertainty. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024. 1, 2, 6, 7, 8

[27] Xingchen Zhao, Chang Liu, Anthony Sicilia, Seong Jae Hwang, and Yun Fu. Test-time fourier style calibration for domain generalization. *arXiv preprint arXiv:2205.06427*, 2022. 2

[28] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, 2015. 6

[29] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 6, 7, 8

[30] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 4396–4415, 2022. 1

[31] P.C. Álvarez Esteban, E. del Barrio, J.A. Cuesta-Albertos, and C Matrán. A fixed-point approach to barycenters in wasserstein space. *Journal of Mathematical Analysis and Applications*, 2016. 4

ConstStyle: Robust Domain Generalization with Unified Style Transformation

Supplementary Material

A. Details of the Training Process

Details of unified domain determination algorithm, training and inference processes are presented in Algorithms 1, 2 and 3, respectively.

B. Proofs

B.1. Proof of Lemma 1.

Let us start with $L^{\mathcal{S}_k}$, we have:

$$\begin{aligned} L^{\mathcal{S}_k} &= \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} [l(\omega(x), y)] \\ &= \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} l(\zeta(\theta_f(z_x)), y) \\ &= \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} l(\zeta(\theta_f(\sigma_x * \frac{z_x-\mu_x}{\sigma_x} + \mu_x)), y) \\ &= \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} f(\mu_x, \sigma_x, \frac{z_x-\mu_x}{\sigma_x}, y). \end{aligned} \tag{7}$$

Similarly, we have:

$$\begin{aligned} L^{\mathcal{S}_k^T} =& \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} [l(\omega^T(x), y)] \\ =& \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} l(\zeta(\theta_f(z_x^T)), y) \\ =& \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} l(\zeta(\theta_f(\sigma^T * \frac{z_x-\mu_x}{\sigma_x} + \mu^T)), y) \\ =& \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} f(\mu^T, \sigma^T, \frac{z_x-\mu_x}{\sigma_x}, y). \end{aligned} \tag{8}$$

By subtracting 8 from 7, we obtain:

$$\begin{aligned} L^{\mathcal{S}_k^T} - L^{\mathcal{S}_k} = \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} \Big(& f(\mu^T, \sigma^T, \frac{z_x-\mu_x}{\sigma_x}, y) \\ & - f(\mu_x, \sigma_x, \frac{z_x-\mu_x}{\sigma_x}, y) \Big). \end{aligned} \tag{9}$$

Using the Taylor approximation for a function with two variables, we derive:

$$\begin{aligned} & f(\mu^T, \sigma^T, \frac{z_x-\mu_x}{\sigma_x}, y) - f(\mu_x, \sigma_x, \frac{z_x-\mu_x}{\sigma_x}, y) \\ & \approx (\mu^T - \mu_x) \cdot \nabla_{\mu_x} f \quad + (\sigma^T - \sigma_x) \cdot \nabla_{\sigma_x} f. \end{aligned} \tag{10}$$

Let $\mathcal{D}_\mu(\mathcal{T}, \mathcal{S}_k)$ denote the distance between means of the unified instance style $\mathcal{T}$ and seen instance style $\mathcal{S}_k$, while $\mathcal{D}_\sigma(\mathcal{T}, \mathcal{S}_k)$ represents the distance between standard deviations. Let $||v||$ denote the L2-norm of vector v. Assume f is a β-Lipschitz function, we can suppose $\sup_{x\in\mathcal{S}_k} ||\nabla_{\mu_x} f|| = \beta_\mu$, $\sup_{x\in\mathcal{S}_k} ||\nabla_{\sigma_x} f|| = \beta_\sigma$, we have:

$$\begin{aligned} L^{\mathcal{S}_k^T} - L^{\mathcal{S}_k} \leq & \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} (\beta_\mu * ||\mu^T - \mu_x|| + \\ & \beta_\sigma * ||\sigma^T - \sigma_x||) \\ =& \beta_\mu * \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} ||\mu^T - \mu_x|| + \\ & \beta_\sigma * \frac{1}{|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} ||\sigma^T - \sigma_x|| \\ =& \beta_\mu * \mathcal{D}_\mu(\mathcal{T}, \mathcal{S}_k) + \beta_\sigma * \mathcal{D}_\sigma(\mathcal{T}, \mathcal{S}_k) \end{aligned}$$

Let $\beta = \max(\beta_\mu, \beta_\sigma)$, then:

$$L^{\mathcal{S}_k^T} - L^{\mathcal{S}_k} \leq \beta \times (\mathcal{D}_\mu(\mathcal{T}, \mathcal{S}_k) + \mathcal{D}_\sigma(\mathcal{T}, \mathcal{S}_k)) \tag{11}$$

B.2. Proof of Theorem 1

According to B.1, for the seen domains $\{\mathcal{S}_k\}_{k=1}^N$, the total empirical loss across N seen domains is bounded as follows:

$$\sum_{k=1}^N L^{\mathcal{S}_k{}^T} \leq \sum_{k=1}^N L^{\mathcal{S}_k} + \beta * \sum_{k=1}^N (\mathcal{D}_\mu(\mathcal{T}, \mathcal{S}_k) + \mathcal{D}_\sigma(\mathcal{T}, \mathcal{S}_k)) \tag{12}$$

It can be observed that the upper bound of this loss depends on the total distance from the unified domain to N seen domains $\mathcal{S}_k$. Therefore, to minimize the loss over the seen domains, we aim to reduce the distance between the unified domain $\mathcal{T}$ and N seen domains $\mathcal{S}_k$. Consequently, the unified domain style $\mathcal{N}^T = (\mu^T, \Sigma^T)$ is the barycenter of N seen domain styles.

B.3. Proof of Theorem 2

The loss function of the model trained on seen domains, obtained by ConstStyle, and test on unseen domain is given by:

Algorithm 1: ConstStyle Training Process

```
1  Input: Seen data $\mathcal{S} = \{(x, y)\}$, Model $\omega = \zeta(\theta_f(\theta_s(.))$, the update interval $\gamma$, the number of epochs $E$, the learning
      rate $\eta$, and the number of clusters $N'$;
2  Output: Optimal model $\omega^*$, the final unified domain $\mathcal{N}^T$;
3  Algorithm:
4  for epoch $\leq E$ do
5      $\varepsilon \leftarrow \emptyset$;                                       // Set of style features
6      for $x \in \mathcal{S}$ do
7          if epoch $\leq \xi$ then
8              $z_x = \theta_s(x)$;
9              $p(x) = \zeta(\theta_f(z_x))$;
10         else
11             $z_x = \theta_s(x)$;
12             $\epsilon_s \sim \mathcal{N}^T$ ;                          // sample style features
13             $\mu_s, \sigma_s = split(\epsilon_s)$;
14             $z_x^T = \sigma_s * \frac{z_x - \mu_x}{\sigma_x} + \mu_s$ ;           // project to the unified domain
15             $p(x) = \zeta(\theta_f(z_x^T))$;
16         $l = \sum_{c \in \mathcal{C}} y_c . \log(p_c(x))$;
17         $\omega = \omega - \eta . \nabla_\omega l$;                              // Update model
18         if epoch % $\gamma$ == 0 then
19             $\mu_{x_c} = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} z_{x_{c,h,w}}, \sigma_{x_c} = \sqrt{\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} (z_{x_{c,h,w}} - \mu_{x_c})^2}$;
20             $\epsilon_x = concat(\mu_x, \sigma_x)$ ;                  // extract style features
21             $\varepsilon = \varepsilon \cup \epsilon_x$ ;                       // store style features
22     if epoch % $\gamma$ == 0 then
23         $\mathcal{N}(\epsilon^T, \Sigma^T)$ =Unified Domain Determination$(\varepsilon, N')$;
24         $\mathcal{N}^T \leftarrow \mathcal{N}(\epsilon^T, \Sigma^T)$ ;                 // get unified domain style
25     $\omega^* = \omega$;
26 return $\omega^*, \mathcal{N}^T$
```

Algorithm 2: Unified Domain Determination

```
1  Input: Set of all style features $\varepsilon = \{\epsilon_x | x \in S\}$,
      Number of clusters $N'$;
2  Output: Unified Domain Style $\mathcal{N}(\epsilon^T, \Sigma^T)$;
3  Algorithm:
4  $\{\mathcal{C}_k \sim \mathcal{N}(\epsilon_{\mathcal{C}_k}, \Sigma_{\mathcal{C}_k}) | k = 1..N'\} \leftarrow$
      $BayesGMM(\varepsilon, N')$ $\epsilon^T = \frac{1}{N'} \sum_{k=1}^{N'} \epsilon_{\mathcal{C}_k}$;
5  $\Sigma^T = \frac{1}{N'} \sum_{k=1}^{N'} \Sigma_{\mathcal{C}_k}$;
6  $\mathcal{N}^T = \mathcal{N}(\epsilon^T, \Sigma^T)$;
7  return $\mathcal{N}^T$
```

$$
\begin{aligned}
L^{\mathcal{U}^T} &= \frac{1}{|\mathcal{U}|} \sum_{(u,y) \in \mathcal{U}} [l(\omega^T(u), y)] \\
&= \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{U}_c|} \sum_{u \in \mathcal{U}_c} [l(\zeta(\theta_f(z_u^T))), y_c] \\
&= \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{U}_c|} \sum_{u \in \mathcal{U}_c} [l(\zeta(\theta_f(\sigma_u^T * \frac{z_u - \mu_u}{\sigma_u} + \mu_u^T), y_c] \\
&= \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{U}_c|} \sum_{u \in \mathcal{U}_c} [f(\mu_u^T, \sigma_u^T, \frac{z_u - \mu_u}{\sigma_u}, y_c)] \quad (13)
\end{aligned}
$$

Similarly, We have:

$$
\begin{aligned}
L^{\mathcal{S}^T} &= \frac{1}{|\mathcal{S}|} \sum_{(x,y) \in \mathcal{S}} [l(\omega^T(x), y)] \\
&= \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{S}_c|} \sum_{x \in \mathcal{S}_c} [l(\zeta(\theta_f(z_x^T))), y_c] \\
&= \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{S}_c|} \sum_{x \in \mathcal{S}_c} [l(\zeta(\theta_f(\sigma^T * \frac{z_x - \mu_x}{\sigma_x} + \mu^T), y_c] \\
&= \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{S}_c|} \sum_{x \in \mathcal{S}_c} [f(\mu^T, \sigma^T, \frac{z_x - \mu_x}{\sigma_x}, y_c)]. \quad (14)
\end{aligned}
$$

Assume that the cardinality of seen domain S and unseen domain U are the same for all classes, i.e, $|\mathcal{S}_c| = |\mathcal{U}_c| = d = \frac{|\mathcal{U}|}{|\mathcal{C}|} = \frac{|\mathcal{S}|}{|\mathcal{C}|}, \forall c \in \mathcal{C}$. From Equations 13 and 14, we have:

Algorithm 3: ConstStyle Inference Process

```
1  Input: Unseen data $\mathcal{U} = \{u | u \sim \mathcal{U}\}$, Optimal
     model $\omega^*$, Unified domain $\mathcal{N}^T$;
2  Output: Prediction set $L_{\mathcal{U}}$;
3  Algorithm:
4  $L_{\mathcal{U}} = \emptyset$;
5  for $u \in \mathcal{U}$ do
6      $z_u = \theta_s(u)$;
7      $\mu^T, \sigma^T = split(\epsilon^T)$;
8      $z_u^T = (\alpha.\sigma_u + (1-\alpha).\sigma^T).\frac{z_u - \mu_u}{\sigma_u} + (\alpha.\mu_u +$
         $(1-\alpha).\mu^T)$;
9      $p(u) = \zeta(\theta_f(z_u^T))$;
10     $y_u = \arg\max(softmax(p(u))$;
11     $L_{\mathcal{U}} = L_{\mathcal{U}} \cup y_u$;
12 return $L_{\mathcal{U}}$
```

$$L^{\mathcal{U}^T} - L^{\mathcal{S}^T} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}\frac{1}{d}\sum_{u\in U_c, x\in S_c}[f(\mu_u^T, \sigma_u^T, \frac{z_u-\mu_u}{\sigma_u}, y_c) - f(\mu^T, \sigma^T, \frac{z_x-\mu_x}{\sigma_x}, y_c)] \quad (15)$$

By applying the Taylor approximation for three variables, we obtain:

$$\begin{aligned}
&f(\mu_u^T, \sigma_u^T, \frac{z_u-\mu_u}{\sigma_u}, y_c) - f(\mu^T, \sigma^T, \frac{z_x-\mu_x}{\sigma_x}, y_c)\\
&\approx (\mu_u^T - \mu^T)\nabla_{\mu^T} f + (\sigma_u^T - \sigma^T)\nabla_{\sigma^T} f\\
&\quad + (\frac{z_u-\mu_u}{\sigma_u} - \frac{z_x-\mu_x}{\sigma_x})\nabla_{\frac{z_x-\mu_x}{\sigma_x}} f\\
&= (\alpha * \mu_u + (1-\alpha) * \mu^T - \mu^T)\nabla_{\mu^T} f\\
&\quad + (\alpha * \sigma_u + (1-\alpha) * \sigma^T - \sigma^T)\nabla_{\sigma^T} f\\
&\quad + (\frac{z_u-\mu_u}{\sigma_u} - \frac{z_x-\mu_x}{\sigma_x})\nabla_{\frac{z_x-\mu_x}{\sigma}} f\\
&= \alpha * (\mu_u - \mu^T) * \nabla_{\mu^T} f + \alpha * (\sigma_u - \sigma^T) * \nabla_{\sigma^T} f\\
&\quad + (\frac{z_u-\mu_u}{\sigma_u} - \frac{z_x-\mu_x}{\sigma})\nabla_{\frac{z_x-\mu_x}{\sigma}} f \qquad (16)
\end{aligned}$$

Denote $||v||$ as the L2-norm of tensor v. Suppose that $\sup_{x\in\mathcal{S}}(||\nabla_{\mu^T} f||, ||\nabla_{\sigma^T} f||) = \beta$ and $\sup_{x\in S}\nabla_{\frac{z_x-\mu_x}{\sigma}} f = \xi$, then we have:

$$\begin{aligned}
&L^{\mathcal{U}^T} - L^{\mathcal{S}^T}\\
&\le \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}\frac{1}{d}\sum_{u\in\mathcal{U}_c, x\in\mathcal{S}_c}(\alpha\times(\beta\times||\mu^T - \mu_u||\\
&\quad + \beta\times||\sigma^T - \sigma_u||) + \xi\times||\frac{z_u-\mu_u}{\sigma_u} - \frac{z_x-\mu_x}{\sigma}||)\\
&\le \alpha\times\beta\times\frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}(||\mu^T - \mu_u|| + ||\sigma^T - \sigma_u||)\\
&\quad + \xi\times\frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}, x\in\mathcal{S}}||\frac{z_u-\mu_u}{\sigma_u} - \frac{z_x-\mu_x}{\sigma}||. \qquad (17)
\end{aligned}$$

Observed that, $\frac{z_u-\mu_u}{\sigma_u}, \frac{z_x-\mu_x}{\sigma} \sim \mathcal{N}(0, I)$, where I is the identity matrix size $C\times H\times W$, where C, H, W are the channel, height, and width dimensions of z_x. When the cardinality of seen domains $\mathcal{S}$, unseen domain $\mathcal{U}$ is sufficiently large, we can approximate:

$$\frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}, x\in\mathcal{S}}||\frac{z_u-\mu_u}{\sigma_u} - \frac{z_x-\mu_x}{\sigma}|| = \mathbb{E}[||U - X||], \quad (18)$$

where U and X are two random multivariate variables over $\mathbb{R}^{C\times H\times W}$ drawn from standard Gaussian distribution, $U, X \sim \mathcal{N}(0, I)$. We have:

$$\begin{aligned}
&0 \le \mathbb{V}[||U - X||] = \mathbb{E}[||U - X||^2] - (\mathbb{E}[U - X])^2\\
&\to \mathbb{E}[U - X] \le \sqrt{\mathbb{E}[||U - X||^2]} = \sqrt{Tr(2I)}\\
&\to \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}, x\in\mathcal{S}}||\frac{z_u-\mu_u}{\sigma_u} - \frac{z_x-\mu_x}{\sigma}|| \le \sqrt{Tr(2I)}
\end{aligned}$$

Let $\mathcal{D}_\mu(\mathcal{T},\mathcal{U})$ and $\mathcal{D}_\sigma(\mathcal{T},\mathcal{U})$ be the distance between mean and standard deviation of unified domain $\mathcal{T}$ and unseen domain $\mathcal{U}$, respectively. From Equation (17), we obtain:

$$\begin{aligned}
L^{\mathcal{U}^T} - L^{\mathcal{S}^T} &\le \alpha\times\beta\times(\mathcal{D}_\mu(\mathcal{U},\mathcal{T}) + \mathcal{D}_\sigma(\mathcal{U},\mathcal{T}))\\
&\quad + \xi\times\sqrt{2.Tr(I)}
\end{aligned}$$

C. Experiment Setup

Image classification: We train a ResNet18 pretrained on ImageNet for 200 epochs with learning rate of 0.001. Batch size is set to 32 for PACS dataset with 3 integrated ConstStyle layers, and 128 with 1 ConstStyle layer for Digit5 dataset.
Image Corruption: We use WideResNet with a single ConstStyle layer as a backbone, training for 200 epochs with a learning rate of 0.05 and batch size of 512.
Instance Retrieval: We train a model with ResNet50 pretrained on ImageNet as the backbone for 80 epochs with a learning rate of 0.0035. We integrate 3 ConstStyle layers

Method	Venue	M,MM	M,S	M,SY	M,U	MM,S	MM,SY	MM,U	S,SY	S,U	SY,U	Avg
ERM	-	**80.22**	82.77	92.34	97.46	77.60	74.83	71.67	52.49	77.70	87.86	79.49
Crossgrad	ICLR 2018	79.24	82.95	92.01	97.68	76.64	75.01	73.00	50.77	78.77	84.8	77.02
Mixup	ICLR 2018	75.92	84.88	90.81	96.75	75.87	70.71	67.49	44.03	80.51	82.58	76.95
Cutmix	ICCV 2019	74.86	**85.16**	91.61	97.02	77.78	70.04	68.87	45.51	**80.75**	85.59	77.71
EFDMix	CVPR 2022	76.29	82.87	92.53	97.52	77.65	76.14	73.33	52.34	78.57	85.87	78.88
RIDG	ICCV 2023	79.75	84.48	91.97	97.23	77.8	73.77	71.05	50.73	79.74	86.33	79.28
MixStyle	ICLR 2021	77.96	72.69	83.37	86.82	75.09	62.18	68.15	41.53	58.5	71.88	69.81
DSU	ICLR 2022	78.77	83.83	92.1	**97.81**	78.53	74.78	71.89	53.66	78.14	87.62	79.71
CSU	WACV 2024	78.64	84.29	92.72	97.39	77.27	75.61	72.67	57.28	78.56	88.08	80.25
ConstStyle	Ours	**80.22**	84.69	**92.92**	97.33	**78.73**	**76.27**	**74.19**	**57.58**	80.29	**88.24**	**81.04**

Method	Venue	M,MM,S	M,MM,SY	M,MM,U	M,S,SY	M,S,U	M,SY,U	MM,S,SY	MM,S,U	MM,SY,U	S,SY,U	Avg
ERM	-	80.12	76.39	**71.41**	61.89	83.47	91.07	44.49	77.63	76.87	48.47	71.18
Crossgrad	ICLR 2018	79.59	76.32	71.31	60.37	83.21	91.47	36.57	77.71	74.26	46.55	70.34
Mixup	ICLR 2018	78.35	74.19	69.51	57.22	**85.78**	91.16	34.45	77.11	71.34	41.29	68.04
Cutmix	ICCV 2019	79.82	73.12	68.92	58.28	85.64	91.32	32.52	78.3	72.57	39.92	68.04
EFDMix	CVPR 2022	80.38	76.04	70.13	63.48	83.62	91.96	43.46	77.61	73.94	50.43	71.10
RIDG	ICCV 2023	80.51	74.71	70.45	61.76	84.78	91.41	35.02	78.28	75.74	45.53	69.81
MixStyle	ICLR 2021	78.91	74.97	61.48	57.95	71.44	81.43	42.92	71.44	62.3	40.99	64.38
DSU	ICLR 2022	**80.71**	76.25	70.54	62.35	83.25	91.47	42.87	77.84	76.29	48.31	70.98
CSU	WACV 2024	80.63	76.26	69.50	**64.68**	85.09	91.53	47.31	77.64	75.61	52.73	72.09
ConstStyle	Ours	80.32	**77.93**	70.89	**64.68**	84.88	**92.10**	**48.88**	**79.08**	**77.27**	**53.55**	**72.95**

Table 7. Multiple unseen domain generalization (2 and 3 unseen domains) on Digits5 dataset. Abbrevations: (M: MNIST, MM: MNISTM, S: SVHN, SY: SYN, U: USPS). The best result is colored **purple** and the second best result is colored blue.

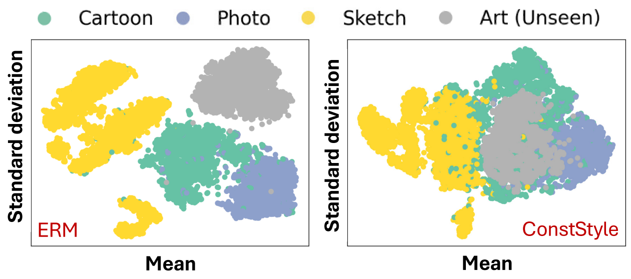


Figure 7. Style statistics of ERM and ConstStyle.

# of clusters	1	2	3	4	5
PACS	86.07	86.51	86.61	**86.77**	86.61
Digit5	85,80	85,60	85,54	**86.88**	85.93

Table 9. Impacts of the number of clusters.

Batchsize	8	16	32	64	128	256
Accuracy	85.43	86.22	**86.77**	86.33	85.91	85.10

Table 10. Impacts of the batch size on accuracy (PACS dataset).

Method	**Dataset**	
	PACS	Digit5
ConstStyle w/ Pretrained features	86.31	76.61
ConstStyle w/ Domain label	86.73	86.37
ConstStyle	**86.77**	**86.88**

Table 8. Different variants of ConstStyle.

into the model.
Across all experiment scenarios, the number of clusters is fixed to 4. All methods are optimized using SGD optimizer. Optimal hyperparameters are selected based on the performance on the validation dataset.

D. Additional Results

D.1. Multiple Unseen Domains on Digit5 dataset

We perform additional experiment with multiple unseen domains on the Digit5 dataset. The results are shown in Table 7. It can be observed that ConstStyle achieves the best performance in most of the scenarios, and obtains the highest average accuracy.

E. Ablation Studies

In this section, we conduct a more in-depth analysis concerning the impacts hyperparameters in ConstStyle's, which is the number of clusters used during the unified domain determination phase, we additionally perform experiments to explore the influence of training batch size and impact of α in the inference process.

E.1. In-depth analysis of ConstStyle

We first conduct additional experiments to further analyze the behaviors of ConstStyle. Figure 7 illustrates the style statistics for both seen and unseen domains, demonstrating that ConstStyle effectively aligns training and test samples within a unified domain, thereby enhancing performance under distribution shift. Additionally, we evaluate Const-

α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
PACS	86.00	85.08	85.76	86.34	86.33	86.34	**86.77**	86.46	86.22	86.03	85.86
Digit5	85.96	85.62	85.77	85.91	85.95	**86.88**	86.01	85.99	85.98	86.01	85.96

Table 11. Impact of α to the model performance on different datasets.

Style with two alternative approaches: **1. Clustering using domain label** and **2. Utilizing pretrained style statistics** with results shown in Table 8. We can observe that while domain labels can produce good performance, they are not always optimal, as some samples have style statistics belonging to other domains; thus, clustering using GMM can form appropriate domain clusters, yielding better performance. Furthermore, using pretrained features for clustering can achieve comparable results if style features are previously learned by the pretrained model, as shown in the PACS dataset in Table 8. However, if the pretrained model has not learned style features, relying on them can significantly degrade ConstStyle's accuracy, as observed in the Digit5 dataset.

E.2. Impacts of the Number of Clusters

We first investigate the impact of the number of clusters during the clustering phase, ranging from one to five. Figure 9 demonstrates that ConstStyle performs consistently across domains, regardless of the number of clusters. This consistency demonstrates ConstStyle's robustness, as the major goal is to construct a single domain by averaging all of the clusters in the visible domains.

E.3. Impacts of the Batch Size

In this section, we investigate the impacts of the batch size on ConstStyle's performance. Experiments are conducted with batch size ranging from 8 to 256, and the results are presented in Table 10. The results suggest that using either very small or very large batch sizes can lead to suboptimal performance, as too few or too many style modifications may disrupt learning stability. The optimal strategy is to use a moderate batch size (about 32), ensuring balanced and steady learning for the model.

E.4. Impacts of Partial Projection

We study the impacts of α on the performance of the proposed method by varying this parameter from 0 to 1, with the results presented in Table 11. It is evident that the impact of α varies significantly across different values, highlighting its important role in achieving optimal performance. When an appropriate value of α is selected, overall performance can improve by up to 0.56% for PACS dataset and up to 0.87% for Digit5 dataset, compared to when no α value is used. This results also highlights the effects of our proposed partial style alignment algorithm (Section 3.5).

Data size	32892	65787	98680	131575
Average training time per epoch (s)	249.2	568.4	857.3	1076.4

Table 12. Scalability of ConstStyle with different number of training data size.

E.5. Scalability against larger datasets

ConstStyle has three components: style statistics distribution estimation, unified style determination, and style aligment. The computational complexity of all three components scales linearly with the training data size. As a result, Constlyle is inherently scalable to large datasets. This scalability is also empirically demonstrated in Table 12, which reports the average training time per epoch when varying the training data size.