Project Report

Maximilian Emmanuel Rittler, Johannes Christian Schuster, Hannes Christian Weber
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
maximilian-emmanuel.rittler@stud.uni-bamberg.de
johannes-christian.schuster@stud.uni-bamberg.de
hannes-christian.weber@stud.uni-bamberg.de

xAI-Proj-M: Domain Generalization
Degree: M.Sc. AI

Abstract

This project addresses the challenge of domain generalization, which involves training models that generalize effectively to unseen target domains without access to target data during training. To this end, we implement and evaluate the MixStyle method, a feature-space data augmentation technique designed to mitigate domain shift by perturbing style statistics within intermediate network layers. Experimental evaluations are conducted on two benchmark datasets of varying scale - PACS (small-scale) and Office-Home (large-scale) - to assess the robustness and effectiveness of MixStyle under different data regimes. The empirical results demonstrate that MixStyle consistently enhances generalization performance across domains, with particularly notable gains in low-data scenarios. These findings highlight the potential of MixStyle as a lightweight yet effective approach for improving model robustness in domain generalization settings.

1 Introduction

2 Methods

This section outlines the methods employed to address the problem of domain generalization during training. These techniques were integrated into several model architectures to enhance their performance and provide deeper insights into their behavior under domain shift conditions.

2.1 MixStyle

The method introduced by Kaiyang Zhou (2023), known as MixStyle, is a novel, lightweight, and effective module aimed at enhancing the generalization capability of neural networks, particularly in scenarios involving domain shifts and previously unseen data. It addresses a fundamental challenge in machine learning and artificial intelligence - namely, the tendency of models to perform poorly when deployed in environments that differ from the training domain.

The *key idea* behind MixStyle is to simulate domain variability by mixing feature statistics - specifically, the mean and standard deviation - across different samples during training. These statistics extracted in shallow layers of the network capture style-related information, which typically varies across domains. By perturbing them in a structured manner, MixStyle implicitly encourages the

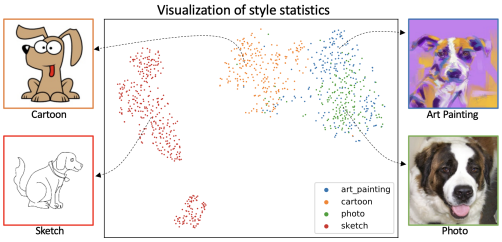


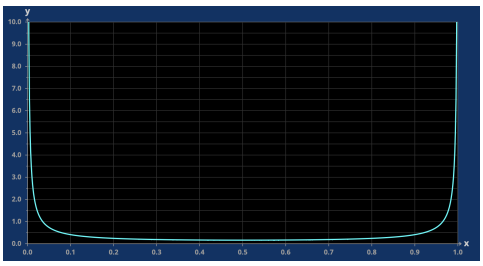
Figure 1: Cluster Overview over different domains of the same class.

network to learn representations that are invariant to style, thereby enhancing its robustness to domain shifts.

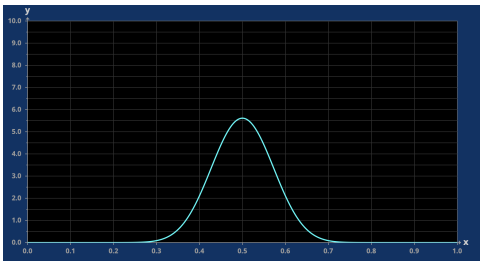
As illustrated in Figure 1, style statistics form distinct clusters for different domains. Domain generalization techniques aim to address this by reducing the separation between these clusters. The objective is to align samples of the same class, regardless of their domain origin, into a single, unified cluster. This consolidation of class-specific representations across domains leads to improved generalization and increased classification accuracy on previously unseen domains.

2.1.1 Image Variablity

To modulate the degree of mixing between feature statistics from different samples, the authors utilize the $Beta$ distribution to stochastically sample a mixing coefficient denoted by λ. This coefficient controls the extent to which the style characteristics - specifically, the mean and standard deviation - of each input sample contribute to the resulting mixed representation. A value of λ close to 0 or 1 results in an asymmetric combination, where the output is dominated by the style statistics of a single sample, with only a minimal influence from the other. Conversely, when λ is near 0.5, the contribution from both samples becomes more balanced, yielding a more homogeneous mixture of their style attributes.



(a) $\alpha = 0.1$



(b) $\alpha = 25$

Figure 2: Probability density functions of the $Beta$ distribution with symmetric parameters ($Beta(\alpha, \alpha)$) for different values of α.

The value of λ is sampled from a symmetric $Beta$ distribution, $Beta(\alpha, \alpha)$, where the hyperparameter α determines the shape of the distribution. For small values of α (e.g., $\alpha = 0.1$) (as shown in Figure 2a), the distribution is U-shaped, assigning higher probability density to values near 0 and 1. This leads to sampled λ values that strongly favor one sample over the other, thus resulting in mixed features dominated by the style of a single image. In contrast, larger values of α (e.g., $\alpha = 25$) (Figure 2b) yield a unimodal distribution centered around 0.5, promoting more balanced combinations where the styles of both samples are equally represented. This parameterization allows fine-grained control over the strength and symmetry of feature mixing.

The entire process is applied over mini-batches during training and supports two distinct strategies for selecting image pairs to be mixed. In the random sampling method, pairs of samples are selected at random from the mini-batch, without considering their domain labels. As a result, samples from the same domain may also be combined. In contrast, the cross-domain (xdomain) strategy enforces domain diversity by ensuring that each selected pair consists of samples originating from different

domains. This targeted mixing encourages the model to generalize across domain boundaries by explicitly learning representations that are invariant to domain-specific style variations.

2.1.2 Mechanism

Algorithm 1: MixStyle Operation Forward Pass

Input: Feature map $\mathbf{f}$, Feature map $\mathbf{f}'$, parameter α

Output: Modified feature map $\hat{\mathbf{f}}$

Compute mean μ and std σ;

Sample $\lambda \sim \text{Beta}(\alpha, \alpha)$;

Compute mixed μ_c and σ_c;

Normalize and re-style;

return $\hat{\mathbf{f}}$

The MixStyle layers are positioned immediately after convolutional layers, which are responsible for extracting task-relevant feature representations from the input images.

Given two feature maps, $\mathbf{f}$ and $\mathbf{f}'$, extracted from two different images (potentially from different domains), the first step involves computing the channel-wise mean and standard deviation of each feature map. These statistics are calculated over the spatial dimensions of each channel as follows:

$$\mu_c = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{c,h,w}, \qquad \sigma_c = \sqrt{\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} (x_{c,h,w} - \mu_c)^2} \tag{1}$$

Here, $x_{c,h,w}$ denotes the activation at channel c and spatial location (h, w), while H and W represent the height and width of the feature map, respectively. The computed μ_c and σ_c describe the style characteristics of the input sample, inspired by style transfer techniques where such statistics are associated with visual appearance.

Following the computation of these style statistics, a mixing coefficient λ is sampled from a Beta distribution, as described in Section 2.1.1. This coefficient determines the extent to which the style from one sample is combined with that of another. Specifically, λ determines whether the resulting feature representation is a strong interpolation of both styles (when $\lambda \approx 0.5$), or largely dominated by one (when λ is close to 0 or 1).

$$\gamma_{\text{mix}} = \lambda\, \sigma(\mathbf{f}) + (1 - \lambda)\, \sigma(\mathbf{f}'); \beta_{\text{mix}} = \lambda\, \mu(\mathbf{f}) + (1 - \lambda)\, \mu(\mathbf{f}') \tag{2}$$

Equation (2) define the computation of the mixed feature statistics used in the MixStyle transformation. Specifically, the mixed standard deviation γ_{mix} and mixed mean β_{mix} are obtained through a random convex combination of the channel-wise statistics from the two feature maps.

$$\hat{\mathbf{f}} = \gamma_{\text{mix}} \odot \frac{\mathbf{f} - \mu(\mathbf{f})}{\sigma(\mathbf{f})} + \beta_{\text{mix}} \tag{3}$$

The final output feature map of each MixStyle layer is computed as shown in Equation (3), where the original feature representation $\mathbf{f}$ is combined with the feature statistics - mean and standard deviation - extracted from a second, randomly selected sample. This mixing process introduces stochastic style variations during training while preserving the semantic content of the features.

The purpose of this stochasticity is to simulate domain shifts in a controlled manner, thereby forcing the model to learn representations that are invariant to domain-specific style variations. By disentangling style from content, MixStyle helps the network generalize more effectively to previously unseen domains.

2.1.3 Benefits

The MixStyle method demonstrates several key advantages described by Kaiyang Zhou (2023) supported by extensive experimental results. First, it significantly improves domain generalization,

consistently outperforming vanilla ResNet-18 and other regularization techniques on benchmark datasets. Specifically, MixStyle achieves approximately a 4% increase in accuracy on the PACS dataset and a 1% improvement on Office-Home compared to the traditional Mixup method. This performance gain is notable given the method's simplicity, which also allows it to surpass more complex approaches such as L2A-OT by nearly 1% on PACS.

MixStyle exhibits particular strength in low-data regimes, delivering over 6% improvement with only 10 labels per class and nearly 7% improvement with 5 labels per class on PACS, demonstrating its effectiveness in scenarios with limited training data. Additionally, the method's versatility is evident as it achieves strong results not only in supervised learning environments but also in unsupervised and semi-supervised domain adaptation settings.

However, while MixStyle excels on smaller datasets, it does not outperform more advanced methods like L2A-OT on larger datasets such as Office-Home when applied alone, indicating opportunities for further refinement or combination with complementary techniques.

2.1.4 Integration

Based on the assumption that domain-related style statistics are more prominent in the shallow layers of convolutional neural networks (CNNs), MixStyle should be integrated into earlier layers of the architecture. For instance, in ResNet architectures, the optimal configuration involves inserting MixStyle behind the first and the second residual blocks. This setup enables the synthesis of "new domains" without disrupting the semantic representations required for the primary task - such as image classification - which are typically learned in deeper layers.

Kaiyang Zhou (2023) recommend this configuration across different task settings. Specifically, for object recognition, they found that placing MixStyle after residual Blocks 1, 2, and 3 yields the best performance. Our setup, as detailed in Section 3, follows a similar approach for image classification tasks. In contrast, introducing MixStyle in later layers leads to a decline in performance, likely due to the mixing of high-level semantic features, which negatively affects classification accuracy.

The implementation, as described in Section 2.1.2, is structured around a Python module that defines the mathematical operations of MixStyle within the forward function as shown in Listing 1. This design ensures that all computations are executed during the forward pass. Additionally, a new module was introduced to encapsulate the various model architectures used during training and evaluation. Within this module, a parent class was defined for each model, extended with parameters that specify the insertion points of the MixStyle layers. This modular and parameterized approach enables seamless integration of MixStyle into different architectures, making it well-suited for use during model training.

```
class MixStyle(nn.Module):
  ...
  def __init__(self, p=0.5, alpha=0.1, eps=1e-6, mix="random", seed
    =42):
  ...

  def forward(x):
  ...
```

Listing 1: MixStyle implementation structure

To support the integration of MixStyle into various model architectures, an additional module was developed to encapsulate and manage different network configurations used for training and evaluation. This module defines a parent class for each model, extended with parameters that allow precise control over the insertion points of MixStyle layers. These configurations are predefined and stored under specific model names that also encode additional details such as the position of MixStyle layers and the setup of the final classification layer.

As illustrated in Listing 2, model instances can be easily loaded by referencing their respective names from the `resnet_ms` module. This approach ensures a flexible and reproducible training pipeline, as each model variant is preconfigured with consistent parameters.

```
  MODEL = resnet_ms.resnet50_fc512_ms12_a0d1
```

Listing 2: Model loading from predefined configurations

Table 1: Final hyperparameters

Epochs	25
Batch size	32
Learning rate	0.001
Weight decay	0.0001
Optimizer	SGD with momentum ($\beta = 0.9$)
LR scheduler	Cosine annealing
Early stopping	10 epochs (delta: 0.00001)

To facilitate experimental reproducibility and ease of use, the most effective architectures were encapsulated as callable model definitions. These predefined models can be conveniently loaded within Jupyter notebooks or scripts, enabling seamless execution of various training and evaluation scenarios across different configurations.

3 Experimental Setup

Building on the provided theoretical foundation, this section offers an overview over the employed datasets, the experimental setup as well as the models and evaluation metrics used.

3.1 Datasets

PACS (Li et al., 2017) is an image dataset specifically designed for domain generalization. It contains seven classes (e.g., "dog", "house", "guitar"), distributed across four domains (photo, art painting, cartoon, sketch). These domain shifts are rather extreme, which is ideal for testing style robustness.

Office-Home (Venkateswara et al., 2017) is another common dataset for domain adaptation and generalization. It has 65 classes containing objects commonly found in office and home environments from four domains: art, clip art, product and real-world. It hence offers a more realistic style variation than PACS and provides us with a different setting in which we can test generalization ability.

3.2 Technical Setup

Both datasets are available on HuggingFace and were accessed via the `datasets` package. For easy access, the datasets were wrapped in a pandas data frame and accessed with a custom `DataLoader`. PACS is a comparably small dataset, so it fits into memory. For Office-Home, images are lazily loaded via indexing. The `DataLoader` API, however, hides this technicality from the user, it enables to iterate through the dataset providing $(\texttt{image}, \texttt{domain}, \texttt{class})$-triples.

Our evaluation was conducted in a Jupyter notebook as it provides a suitable environment for quick changes and good overview. We evaluated in a multi-source domain setting, where we trained the model on three domains, leaving one out for validation. In a later ablation study, we only trained on two domains to have a "real" test set.

Hyperparameters were tuned manually and are listed in table 1. You may notice, that for early stopping, we have a patience of 10, but are training for at most 25 epochs. Due to time constraints, we couldn't find a suitable configuration that employs early stopping and at the same time ensures enough training.

3.3 Models and Metrics

In early experiments, we experimented with different CNN architectures as backbones. These experiments included VGG-16 (Simonyan and Zisserman, 2014) and the two ResNet architectures ResNet-18 and ResNet-50 (He et al., 2016). VGG-16 performed far worse than the ResNet architectures, so we did not consider it further. For the ResNet-18 and ResNet-50, we loaded their weights pretrained on ImageNet. Note though, that this introduces the problem of data leakage. E.g., ImageNet consists for the most part of photos and in our evaluation setup, we would later train on every domain but photo on which we evaluate. Since we are using pretrained weights, however, the model will have seen lots of photos, which makes this domain obsolete for training.

Table 2: Experimental results using three domains of PACS for training and one for validation. Results are given as mean ± standard deviation of the mean accuracy across three seeds. The accuracy is given in percent. The upper two models are our baseline, the models beneath the different configurations from our experiments, all with ResNet-50 as backbone. FC means introducing a 512-dimensional fully connected layer after the feature maps. FRx means freezing layer x. MSx means introducing a Mixstyle layer after layer x.

Model	Art painting	Cartoon	Photo	Sketch	Total
ResNet-18	78.06 ± 0.22	76.51 ± 0.51	97.94 ± 0.28	66.43 ± 1.29	77.89 ± 0.29
ResNet-50	84.80 ± 0.26	74.53 ± 0.22	$\mathbf{98.12 \pm 0.12}$	68.66 ± 1.29	81.53 ± 0.41
+ FC + FR1	85.24 ± 0.58	71.90 ± 1.90	$\mathbf{98.12 \pm 0.16}$	68.84 ± 1.02	81.03 ± 0.81
+ FC + FR1&2	84.13 ± 0.38	72.45 ± 1.15	97.74 ± 0.12	70.83 ± 0.86	81.29 ± 0.15
+ FC + MS1	86.98 ± 0.69	75.87 ± 0.95	$\mathbf{98.00 \pm 0.22}$	74.13 ± 2.24	83.74 ± 0.16
+ FC + MS1&2	88.77 ± 0.63	$\mathbf{76.74 \pm 1.21}$	$\mathbf{98.18 \pm 0.07}$	$\mathbf{75.50 \pm 1.36}$	$\mathbf{84.80 \pm 0.05}$
+ FC + FR1&2 + MS2	87.61 ± 0.53	74.57 ± 2.79	$\mathbf{98.28 \pm 0.03}$	73.79 ± 0.90	83.56 ± 0.67
+ FC + FR1 + MS1&2	$\mathbf{89.11 \pm 0.29}$	$\mathbf{76.70 \pm 0.96}$	$\mathbf{98.10 \pm 0.06}$	75.14 ± 0.75	$\mathbf{84.76 \pm 0.49}$

The most common evaluation metric reported for domain generalization is accuracy. We hence collected the accuracy across domains as well as overall accuracy. The results are given as mean and standard deviation across three seeds.

4 Experiments & Results

This section describes the conducted experiments and highlights the main results, including the most promising configurations as well as their performance. It also presents the results of two small-scale ablation studies.

4.1 Experiments Overview

Firstly, we needed to establish a baseline to compare methods like Mixstyle against. As mentioned previously, the two most promising backbones were ResNet-18 and ResNet-50. It is hence sensible to select their vanilla variants for this purpose. Against that baseline, we conducted three different experiments: layer freezing, Mixstyle and a combination of both. Layer freezing seemed promising as the feature extraction process in the earlier layers should be no different across the domains. Freezing those layers hence could stabilize the feature extraction for the later layers to become more domain invariant. Mixstyle does not need further introduction at this point and a combination of both could yield a more stable perturbation in the feature space. Among the configurations, the most notable handles were which layers to freeze, after which layers to include a Mixstyle layer and what α to use in the Mixstyle layers.

4.2 Effects of Configurations

We employ freezing to enable more stable feature extraction. Freezing later layers has the consequence of reducing the model's ability to generalize to new domains. We noticed that especially freezing the last two layers decreases the performance substantially. Mixstyle is also best applied in earlier layers. Perturbing the output of later layers decreases performance, as semantic information rather than style information would be mixed in doing so.

For configurations with freezing as well as with Mixstyle it has shown that introducing a 512-dimensional fully-connected layer right before the classification layer slightly increases performance. An explanation could be that as both methods make it harder or even impossible for earlier layers to adapt to the data, a fully-connected layer in the end balances the capacity of the model and introduces more learning capabilities in the latent space.

4.3 Experimental Results

The results of our baseline as well as from our most promising configurations are shown in table 2. First of all, note that ResNet-50 is clearly more performant than ResNet18. That is also the reason

Table 3: Results of an ablation study on a three-fold split of PACS into train, validation and test set. Results are given as mean ± standard deviation of percent accuracy. The model used was ResNet-50 + FC + MS1&2 + FR1.

Art painting	73.08 ± 4.52
Cartoon	74.69 ± 0.26
Photo	97.32 ± 0.15
Sketch	70.54 ± 2.95
Total	78.90 ± 1.70

Table 4: Results of one run on Office-Home with the model ResNet-50 + FC + MS1&2. Due to computational and time constraints, only one run was evaluated. Results are given as percent accuracy.

Art	63.29
Clip art	48.74
Product	73.89
Real World	76.29
Total	65.55

why the rest of the experiments are using ResNet-50 as their backbone. The table also shows that freezing in the first two layers doesn't seem to affect the performance. Mixstyle, however, brings a substantial performance increase of about 3%. This is also confirming the results of Kaiyang Zhou (2023). The most notable difference lies in the sketch domain, which is the hardest of the four. Here, Mixstyle even increases performance by about 7%. The first layer is the most important one for introducing Mixstyle, as the boost in performance is most noticeable here. We had the best results with $\alpha = 0.1$, which suggests that more extreme perturbations are more effective in domain generalization. We also tested $\alpha = 0.2$ and $\alpha = 0.3$, for which the performance gradually decreased. Combining Mixstyle with layer freezing does not seem to improve performance further suggesting that Mixstyle is best applied on its own.

The performance across all ResNet-50 models in the photo domain is also consistently high at about 98%. This points out that including this domain when evaluating models pretrained on ImageNet is less informative as the domain gap is almost nonexistent compared to the other domains. A comparison with models trained from scratch would provide more insight here.

To evaluate the performance of Mixstyle in the latent space, we analyzed the output of the models right before the classification layer with tSNE. The corresponding plots of a ResNet-50 model with introduced 512-dimensional fully-connected layer with and without Mixstyle are shown in figure 3. For both models, sketch was used as the validation domain, hence this domain is clearly separated from the rest. Ideally, Mixstyle should make the clusters more uniform, so that one class should fall in the same cluster for all domains. It is barely noticable, but the clusters are a bit more overlapping with Mixstyle. There is also more structure to the sketch cluster which indicates a slightly better separation of classes in this domain.

4.4 Ablation Studies

In a first ablation study, we split the dataset into three parts. One domain was used for validation, one for testing and two for training. We iterated through the domains to select a test domain. The validation domain was chosen among the remaining domains at random. The results are shown in 3. This experiment is again a great illustration that the domain photo brings no insight because of the use of pretrained weights. It is however interesting to see that the other three domains have much more similar performance than on the twofold split.

We additionally evaluated one model on the Office-Home dataset. Due to time constraints, we were only able to run one seed as that takes about 7h on our hardware to complete. Table 4 lists the results. Office-Home has 65 classes, so a worse performance than on PACS was to be expected. Overall, however, the accuracy here is quite high and again confirms the results of Kaiyang Zhou (2023).

5 Discussion

As described in the previous section, freezing early layers doesn't seem to affect the performance at all. This suggests that the feature extractors in the earlier layers are quite stable even without freezing across the domains, which is plausible, as basic shapes stay the same. Mixstyle on the other hand is a very promising technique, which has shown to enable great performance increase.

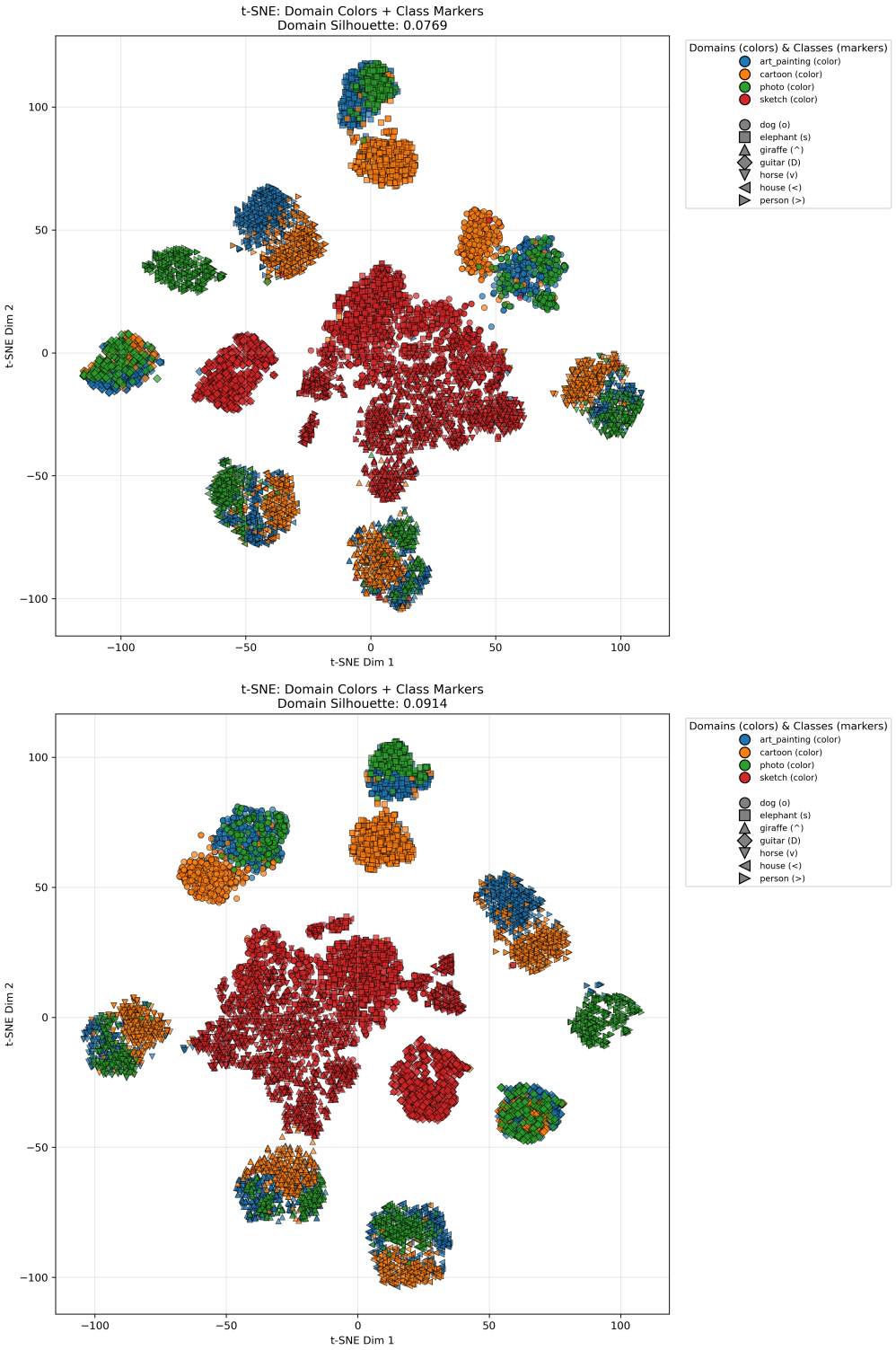


Figure 3: tSNE evaluation from the output of an introduced 512-dimensional fully connected layer right before the classification layer of a ResNet-50 model with (top) and without (bottom) Mixstyle after the first two layers, respectively. For both models, sketch (red) was used as validation domain.

6 Conclusion

7 Notation

Symbols and Definitions

$\mathbf{f}, \mathbf{f}'$	Feature maps extracted from convolutional layers.
$\hat{\mathbf{f}}$	Defining the output of any MixStyle layer.
λ	A scalar sampled from a $Beta$ distribution, controlling the degree of mixing between feature statistics.
α	A scalar controlling the $Beta$ distribution.
μ	Mean.
σ	Standard deviation.

References

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

T. X. Kaiyang Zhou, Yongxin Yang. Mixstyle neural networks for domain generalization and adaptation. 2023.

D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017.

Declaration of Authorship

Ich erkläre hiermit gemäß § 9 Abs. 12 APO, dass ich die vorstehende Seminararbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass die digitale Fassung der gedruckten Ausfertigung der Seminararbeit ausnahmslos in Inhalt und Wortlaut entspricht und zur Kenntnis genommen wurde, dass diese digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Bamberg, July 29, 2025

(Ort, Datum) (Unterschrift)

A Appendix

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material.