Project Report

Maximilian Emanuel Rittler, Johannes Christian Schuster, Hannes Christian Weber
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
maximilian-emanuel.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.

*Sections 1 & 2
†Sections 4, 5 & 6
‡Abstract, Section 3. Code available at: https://github.com/hosthans/domain_generalization

1 Introduction

The rapid advancement of artificial intelligence and machine learning has led to the development of increasingly complex models, which are often trained on large datasets. Performing well when the test data distribution matches closely the training data distribution. When those models are exposed to real-world scenarios, with inputs differing in distribution due changes in style, context, domain semantics or acquisition conditions - so being exposed to out-of-distribution (OOD) data - their performance often degrades. This vulnerability to domain shifts poses a critical challenge for deploying models in real-world applications such as autonomous driving or medical imaging, where OOD is rather the norm than the exception.

Domain Generalization addresses the OOD challence by training models on multiple source domains with the goal of learning representation that generalizes well even to unseen target domains. With their target domain being the OOD data, the goal is to learn domain-invariant features.

While existing approaches relying on meta-learning or domain alignment have shown promise, they often require a complex, resource intensive training pipeline. In contrast, our project explores MixStyle (Zhou et al., 2023) a novel and lightweight approach, yet effective feature-space data augmentation technique, trying to mitigate the effects of OOD data. By perturbing style statistics in the early layers of a convolutional neural network, MixStyle simulates domain variability and encourages the model to learn domain-invariant features.

We will evaluate MixStyle on the PACS and Office-Home datasets using ResNet-18 and ResNet-50 architectures as backbone. Additionally, our experiments also include configurations with and without layer freezing, various data splits and hyperparameter tuning. The results suggest that MixStyle improves the generalization of a model, especially under the tested limited datasets. Thereby, offering a practical and easy to implement solution to strengthen the robustness of a model.

2 Related Work

This section aims to present the theoretical background and related work relevant to this project.

2.1 Domain Generalization

Domain Generalization enables machine learning models to generalize to new, unseen domains, without access to the target domain data during a model's training. Hereby Domain Generalization tries to learn generalized features from the source domain that can perform well even on unseen domains. This is particularly important for computer vision tasks - especially real-world computer vision tasks such as autonomous driving or medical image analysis, where the target domain data is subject to constant changes or is not even available at all, making it impossible to adapt a model continuously (Li et al., 2017; Liu et al., 2023; Blanchard et al., 2011).

Furthermore, Domain Generalization can be broken down into two categories: Multi-Source Domain Generalization and Single-Source Domain Generalization. Multi-Source Domain Generalization assumes that there is more than one relevant domain available, with the goal to use data from multiple sources to learn representations that are independent of different marginal distributions. Single-Source Domain Generalization, on the other hand, assumes that there is only one relevant domain available during training, and the goal is to learn representations that are invariant to the domain shift (Blanchard et al., 2011).

The need to generalize well to new domains arises from the problem of domain shift. Domain shift can significantly worsen the performance of a model - especially CNNs - across different domains (Muandet et al., 2013). These shifts can be cause by a variety of factors. Sources include:

- **Image Style**: Is a major source of domain shifts, such as changes in colours, textures, lighting, etc. (Zhou et al., 2023).
- **Environmental Conditions**: Changes in the environment, such as weather, time of day, or camera settings, can also cause domain shifts (Schwonberg et al., 2023).
- **Visual Abstraction**: Datasets can vary in their level of abstraction, from realistic photos over to abstracts sketches or paintings (Li et al., 2017).

Furthermore, Domain Generalization thus needs to be distinguished from Domain Adaptation, as both paradigms address the challenge of domain shift. While Domain Adaptation aims to adapt a model to perform well on a specific target domain, by reducing the performance gap between the source and target domains. Domain Generalization aims to learn domain-invariant features such that the model can perform well on new unseen target domains, without the need to see the target domain data during training. Conversely, Domain Adaptation methods typically assume the access to the target domain data during training and trying to adapt the model from the source domain to perform well on the target domain (Li et al., 2017; Liu et al., 2023; Wang and Deng, 2018).

To quantify the performance of a model benchmarks and evaluation protocols have been developed. Common benchmarks include:

- **PACS** (Li et al., 2017): A commonly used benchmark including four domains: photo, art painting, cartoon and sketch. It contains 9991 images across seven categories. The domains are designed to be maximally distinct, covering a wide range of visual styles - from realistic photos to abstract sketches.
- **Office-Home** (Venkateswara et al., 2017): This dataset also contains four domains: art, clipart, product and real-world. With 15,500 images across 65 categories it is larger than PACS. The contents are related to objects found in office and home environments.
- **VLCS** (Fang et al., 2013): Consists out of four domains: PASCAL VOC2007 (V), LabelMe (L), Caltech-101 (C), and SUN09 (S). It contains 10729 images across five classes.

Evaluation protocols include:

- **Leave-One-Domain-Out**: Is a common evaluation protocol where one domain is held out as the target domain and the model is evaluated on the remaining domains (Li et al., 2017).
- **Direct Transfer**: Is another approach where the model is trained on one dataset (e.g. MSMT17) and then evaluated on another dataset (e.g. Market1501), without any fine-tuning on the target domain (Chong et al., 2021).

Note all of the above mentioned datasets are also part of the **DomainBed** benchmark suite created by Gulrajani and Lopez-Paz (2020) for Domain Generalization, which includes more datasets, evaluations protocol and algorithms for Domain Generalization.

The most common approaches to Domain Generalization, as outlined by Zhou et al. (2022) and discussed in Gulrajani and Lopez-Paz (2020), include:

- **Domain Alignment**: Most current domain generalization methods fall into the domain alignment category, aiming to reduce disparities between source domains to learn representations that are invariant across them
- **Meta-Learning**: This approach, also known as 'learning to learn', trains a model on a variety of learning tasks to enable it to solve new learning tasks more effectively. In the context of Domain Generalization, it helps the model generalize to new domains by learning from domain shifts in the source data.
- **Data Augmentation**: Involves expanding the original dataset with new transformed data, simulating domain shifts.

In the latter category two techniques also employed in this project are situated: Image Transformation and Feature-Based Augmentation.

Image Transformations are a common technique for data augmentation, where the original image is transformed by methods such as rotation, scaling, cropping, color jittering, etc. On the other hand, Feature-Based Augmentation rely on mixing the CNN feature statistics across different domains - precisely the approach in the focus of this project MixStyle (Zhou et al. (2023)), while others approaches may also combine domains in both feature as well as pixel space.

The following subsection will dive deeper into the Feature Space.

2.2 The Role of the Feature Space

The feature space in CNNs refers to the multidimensional representation of the the data that the network learns during training. This is crucial for the classification tasks performed by CNNs, as it isolates the features extracted from input images, allowing the model to differentiate between various categories, whether it be domains or classes. Hereby, CNNs automatically extract features from images through their convolutional layers, creating a hierachical representation. With each layer capturing different levels of abstraction - starting from low-level features such as edges and textures, then moving to more complex features like shapes and eventually to semantic representations like structures or even objects in deeper layers (Zeiler and Fergus, 2013; Goodfellow et al., 2016).

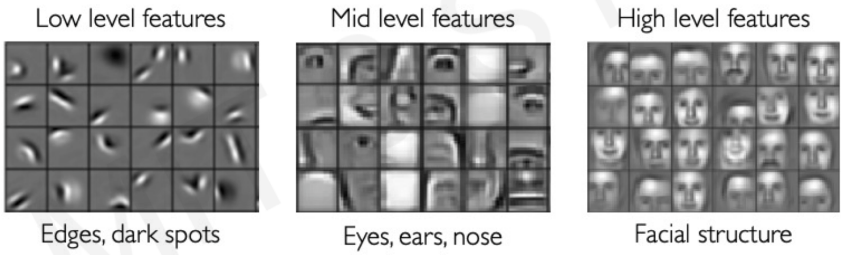


Figure 1: Illustration of feature abstraction hierarchy in CNNs, from low-level (edges) to high-level (semantic object parts). Adapted from Alexander Amini (2025).

This hierarchy show why shallow layers are often more domain-specific, as they are prone to capturing texture or lighting, while deeper layer are more semantic and thus domain-invariant. Feature-space domain generalization methods like MixStyle (Zhou et al., 2023) start their approach there. Feature representations derived from CNNs, often form domain-specific clusters in the latent space. This in particular is problematic for Domain Generalization, where the models needs to learn representations that are invariant to such domain cues (Zhou et al., 2022).

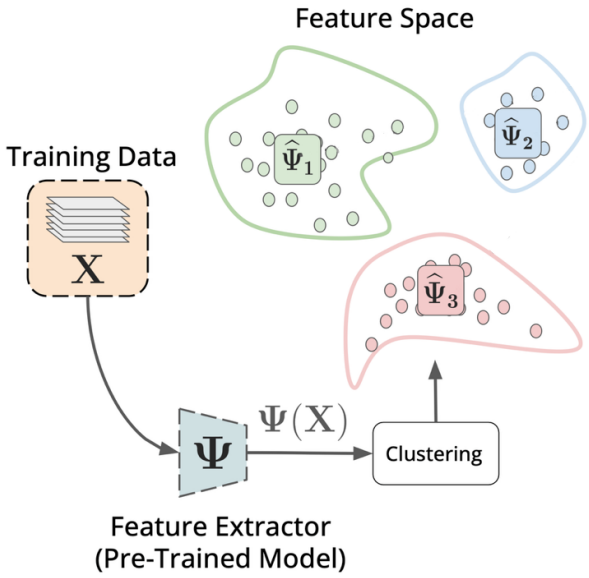


Figure 2: A model pipeline showing input images from different domains processed by a pretrained feature extractor, producing clustered representations in feature space. Image from Thomas and Ghadiyaram (2025), modified.

Clustered representations usually reflect the characteristics of the source domain. Without any intervention from the outside, models trained on these clustered representations might struggle to generalize to new domains.

One effective way to address this issue is by utilizing transfer learning, by reusing pretrained models.

2.3 Transfer Learning

Transfer learning refers to the process of leveraging knowledge obtained from one task or domain to improve performance on a different, yet related, task. In the context of computer vision, it often involves using models pretrained on large-scale datasets — such as ImageNet — and adapting them to new tasks with limited data.

This approach is particularly relevant to Domain Generalization, where the goal is to generalize to an unseen target domain without access to its data during training (Gulrajani and Lopez-Paz, 2020; Li et al., 2017). Transfer learning can support this objective by providing models with general-purpose representations learned from diverse visual categories and conditions.

Convolutional Neural Networks (CNNs) pretrained on large datasets tend to capture a hierarchy of features as seen in figure 1 — from low-level structures like edges and textures to high-level semantic patterns. These pretrained networks are often used as feature extractors enabling downstream models to build upon robust and transferable representations rather than learning from scratch. Furthermore, beyond improved sampling efficiency, transfer learning can also stabilize training and reduce overfitting, especially in cases where the available training data is limited or varies substantially across domains (Thomas and Ghadiyaram, 2025). Thus, transfer learning provides a useful foundation for addressing domain shift in Domain Generalization settings, even though it is not exclusive to them.

However, leveraging pretrained networks also requires careful handling — especially when it comes to controlling which parts of the network are updated during training. This leads to the concept of freezing, which helps maintain stable, domain-invariant representations.

2.4 Freezing

Freezing refers to the practice of preventing specific layers of a neural network from being updated during training. This is commonly applied in transfer learning to retain the general-purpose features learned by pretrained models, especially in the early convolutional layers.

In Domain Generalization, freezing can be used to improve the model's ability to generalize across domains. Specifically, it helps prevent the network from overfitting to the domain-specific features of the source domains — an issue particularly relevant when training on multiple diverse datasets. By keeping the early or mid-level layers fixed, the model is encouraged to rely on more generalizable representations rather than learning domain-specific features. A critical motivation for freezing in Domain Generalization arises from the use of Batch Normalization. Batch Normalization layers compute and update feature statistics (mean and variance) during training, which are sensitive to the distribution of the training data. In a multi-source setting, where training batches may contain samples from several domains, these statistics can become unstable or biased toward dominant domains. By freezing Batch Normalization layers, one can preserve the pretrained statistics and avoid learning domain-specific biases, thus supporting better generalization (Goodfellow et al., 2016; Zhou et al., 2023).

Therefore, freezing is a strategic complement to transfer learning: while transfer learning brings in generalizable knowledge, freezing ensures that this knowledge is not overwritten during training on source domains. It has been used effectively in several recent Domain Generalization approaches such as MixStyle (Zhou et al., 2023).

Summary

In summary, Domain Generalization provides a more demanding but often more practical solution than Domain Adaptation for creating models that are robust to the diverse and unpredictable domain shifts encountered in real-world scenarios.

Transfer Learning and Freezing can be used to supplement the training process, by providing a robust feature extraction backbone and by preventing the model from overfitting to the domain-specific features of the source domains.

3 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.

3.1 MixStyle

The method introduced by Zhou et al. (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.

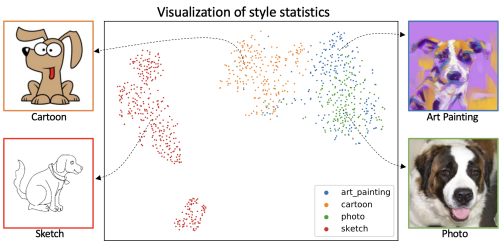


Figure 3: Cluster Overview over different domains of the same class.

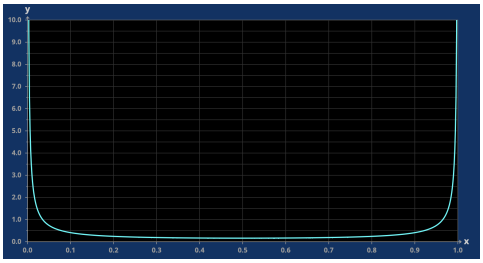
The *key idea* behind MixStyle is to simulate domain variability by mixing feature statistics - specifically, the mean and standard deviation - across different samples during training. These statistics extracted in shallow layers of the network capture style-related information, which typically varies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

As illustrated in Figure 3, 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.

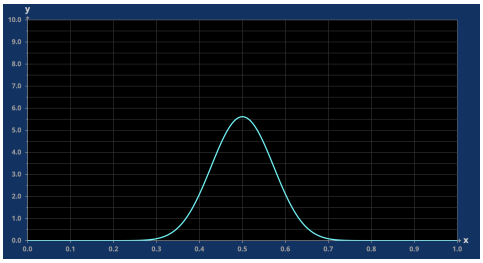
3.1.1 Image Variability

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.

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 4a), 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 4b) 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.



(a) $\alpha = 0.1$



(b) $\alpha = 25$

Figure 4: Probability density functions of the $Beta$ distribution with symmetric parameters ($Beta(\alpha, \alpha)$) for different values of α.

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.

3.1.2 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 3.1.1. 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.

3.1.3 Benefits

The MixStyle method demonstrates several key advantages described by Zhou et al. (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.

3.1.4 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.

Zhou et al. (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 4, 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 3.1.2, 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

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.

4 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.

4.1 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 variation than PACS and provides us with a different setting in which we can test generalization ability.

4.2 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 (`image`, `domain`, `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 computational

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)

resource constraints, we couldn't find a suitable configuration that employs early stopping and at the same time ensures enough training.

4.3 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.

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.

5 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.

5.1 Experiments Overview

First, 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 parameters were which layers to freeze, after which layers to include a MixStyle layer and what α to use in the MixStyle layers.

5.2 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

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 are 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	$\mathbf{84.76 \pm 0.49}$

adapt to the data, a fully-connected layer at the end balances the capacity of the model and introduces more learning capabilities in the latent space.

5.3 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 ResNet-18. That is also the reason why the rest of the experiments use ResNet-50 as their backbone. The table also shows that freezing 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 Zhou et al. (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 t-SNE. The corresponding plots of a ResNet-50 model with an introduced 512-dimensional fully-connected layer with and without MixStyle are shown in figure 5. 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 noticeable, 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.

5.4 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 table 3. This experiment is again a great illustration that the photo domain 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 in the twofold split.

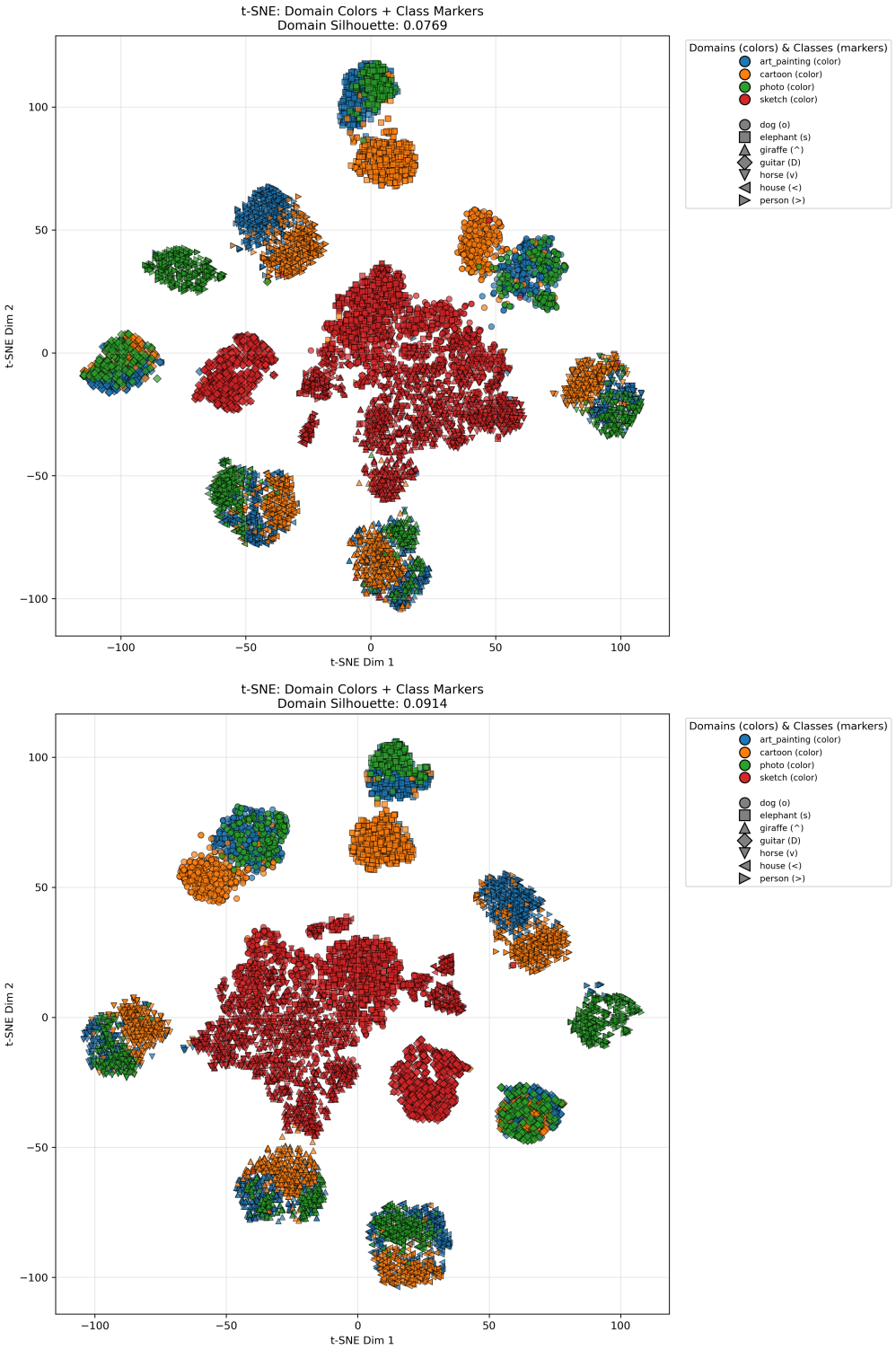


Figure 5: t-SNE evaluation from the output of an introduced 512-dimensional fully connected layer right before the classification layer of a ResNet-50 model with (top) and without (bottom) MixStyle after the first two layers, respectively. For both models, sketch (red) was used as validation domain.

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 $\pm$ 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 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

We additionally evaluated one model on the Office-Home dataset. Due to computational time constraints, we were only able to run one seed as that takes about 7 hours 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, the accuracy here remains high, supporting the results of Zhou et al. (2023).

6 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 and low-level features remain consistent across different domains.

MixStyle on the other hand is a very promising technique, which has shown to enable great performance increases. Our results demonstrate that MixStyle is best applied after the first two layers, with the most important layer being the first one. This makes sense as basic shapes and style information are conveyed at this stage. A perturbation at this level is hence the most effective for domain generalization. Our experiments also show that a smaller α, i.e., a more extreme perturbation, is more effective. By introducing artificial domains with heavier stylistic changes less frequently, the model sees more variety in domains during training while still having information of the original domains. This results in a better ability to generalize across domains.

Our results also reveal a flaw in the experimental setup commonly used in the field. Specifically, using pretrained weights is problematic as certain domains essentially become irrelevant. A prime example is the "photo" domain in PACS as well as the "product" or "real world" domain from Office-Home. In the latter case, even two domains are affected by data leakage. This suggests that results are better gathered by not using pretrained weights when just assessing the performance of domain generalization techniques. In practice, however, pretrained weights are often beneficial, which creates a tension that warrants further investigation.

The datasets used in our study also have limitations that should be acknowledged. PACS and Office-Home represent relatively controlled scenarios with clearly defined domain boundaries. Real-world domain shifts are often more subtle and continuous, and it is unclear whether more gradual style shifts affect the effectiveness of MixStyle. Additionally, while our tSNE analysis shows some improvement in feature clustering, the differences are quite subtle and could benefit from more detailed analysis.

Overall, looking back at the results, MixStyle is a lightweight, parameter-free and highly effective method for domain generalization. The technique shows consistent improvements across different domains, with particularly strong gains in challenging domains like sketch. However, it is important to mention that the datasets used in our study represent controlled scenarios, and future work should investigate how well these findings transfer to more naturalistic domain shifts encountered in practical applications.

References

Alexander Amini. MIT Introduction to Deep Learning - Lecture 3, Jan. 2025. URL https://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L3.pdf.

G. Blanchard, G. Lee, and C. Scott. Generalizing from Several Related Classification Tasks to a New Unlabeled Sample. 2011.

Y. Chong, C. Peng, C. Zhang, Y. Wang, W. Feng, and S. Pan. Learning domain invariant and specific representation for cross-domain person re-identification. *Applied Intelligence*, 51(8): 5219–5232, Aug. 2021. ISSN 0924-669X, 1573-7497. doi: 10.1007/s10489-020-02107-2. URL https://link.springer.com/10.1007/s10489-020-02107-2.

C. Fang, Y. Xu, and D. N. Rockmore. Unbiased Metric Learning: On the Utilization of Multiple Datasets and Web Images for Softening Bias. In *2013 IEEE International Conference on Computer Vision*, pages 1657–1664, Sydney, Australia, Dec. 2013. IEEE. ISBN 978-1-4799-2840-8. doi: 10.1109/ICCV.2013.208. URL http://ieeexplore.ieee.org/document/6751316/.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

I. Gulrajani and D. Lopez-Paz. In Search of Lost Domain Generalization, July 2020. URL http://arxiv.org/abs/2007.01434.

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.

D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, Broader and Artier Domain Generalization, Oct. 2017. URL http://arxiv.org/abs/1710.03077.

C. Liu, L. Wang, L. Lyu, C. Sun, X. Wang, and Q. Zhu. DEJA VU: Continual Model Generalization For Unseen Domains, Mar. 2023. URL http://arxiv.org/abs/2301.10418.

K. Muandet, D. Balduzzi, and B. Schölkopf. Domain Generalization via Invariant Feature Representation, Jan. 2013. URL http://arxiv.org/abs/1301.2115.

M. Schwonberg, F. E. Bouazati, N. M. Schmidt, and H. Gottschalk. Augmentation-based Domain Generalization for Semantic Segmentation, Apr. 2023. URL http://arxiv.org/abs/2304.12122.

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

X. Thomas and D. Ghadiyaram. What's in a Latent? Leveraging Diffusion Latent Space for Domain Generalization, Apr. 2025. URL http://arxiv.org/abs/2503.06698.

H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep Hashing Network for Unsupervised Domain Adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5385–5394, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.572. URL http://ieeexplore.ieee.org/document/8100055/.

M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, Oct. 2018. ISSN 09252312. doi: 10.1016/j.neucom.2018.05.083. URL https://linkinghub.elsevier.com/retrieve/pii/S0925231218306684.

M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks, Nov. 2013. URL http://arxiv.org/abs/1311.2901.

K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy. Domain Generalization: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2022. ISSN 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2022.3195549. URL http://arxiv.org/abs/2103.02503.

K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. MixStyle Neural Networks for Domain Generalization and Adaptation, Sept. 2023. URL http://arxiv.org/abs/2107.02053.

7 Appendix

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.