Preprint of:
Zhao, D., Koh, Y. S., Dobbie, G., Hu, H., Fournier-Viger, P. (2024). Symmetric Self-Paced Learning for Domain Generalization. Proc. 38th AAAI Conference on Artificial Intelligence (AAAI 2024). 16961-16969

Symmetric Self-Paced Learning for Domain Generalization

Di Zhao*[1], Yun Sing Koh[1], Gillian Dobbie[1], Hongsheng Hu[2], Philippe Fournier-Viger[3]

[1] School of Computer Science, University of Auckland
[2] CSIRO's Data61
[3] College of Computer Science and Software Engineering Shenzhen University
dzha866@aucklanduni.ac.nz, {y.koh, g.dobbie}@auckland.ac.nz, Hongsheng.Hu@data61.csiro.au, philfv@szu.edu.cn

Abstract

Deep learning methods often suffer performance degradation due to domain shift, where discrepancies exist between training and testing data distributions. Domain generalization mitigates this problem by leveraging information from multiple source domains to enhance model generalization capabilities for unseen domains. However, existing domain generalization methods typically present examples to the model in a random manner, overlooking the potential benefits of structured data presentation. To bridge this gap, we propose a novel learning strategy, Symmetric Self-Paced Learning (SSPL), for domain generalization. SSPL consists of a Symmetric Self-Paced training scheduler and a Gradient-based Difficulty Measure (GDM). Specifically, the proposed training scheduler initially focuses on easy examples, gradually shifting emphasis to harder examples as training progresses. GDM dynamically evaluates example difficulty through the gradient magnitude with respect to the example itself. Experiments across five popular benchmark datasets demonstrate the effectiveness of the proposed learning strategy.

Introduction

Most machine learning algorithms assume that training (source domains) and testing (target domain) data are independent and identically distributed. Violating this assumption may cause model performance degradation due to a lack of generalization ability in handling domain shifts (Ghifary et al. 2015; Hendrycks and Dietterich 2019). Domain Adaptation (DA) (Pan and Yang 2009; Fernando et al. 2013) is an intuitive solution to deal with domain shifts by utilizing target domain data to align the distribution between the source and target domains (Ganin and Lempitsky 2015) or to fine-tune the model trained on source domains (Long et al. 2015). However, in many scenarios where target domain data is unavailable, large-scale data collection and annotation is prohibitively expensive. For example, in traffic scene semantic segmentation, capturing data that encompasses all traffic scenes under all weather conditions is infeasible (Yue et al. 2019). Domain generalization (DG) (Li et al. 2018) is an emerging solution to relax the constraints inherent in domain adaptation methods. DG aims to learn a universal representation that can effectively generalize to unseen target domains by leveraging labelled data from multiple source domains. Existing DG methods fall into three categories (Zhou et al. 2022; Wang et al. 2022): data augmentation (Zhou et al. 2020), domain invariant representation (Wang et al. 2022), and learning strategy (Arpit et al. 2022; Meng et al. 2022).

However, current DG methods uniformly weigh all training examples, presenting them to the model randomly, overlooking the potential benefits of structured data presentation based on example difficulty (Soviany et al. 2022). From an optimization perspective, training in a structured order can be seen as a continuation method (Bengio et al. 2009), providing a series of optimization objectives, where proceeding objectives serve as a pre-training process that helps to optimize and regularize the succeeding objectives(Wang, Chen, and Zhu 2021). Analogously, in human learning, learning knowledge in a structured order yields notable advantages over a randomized approach (Bengio et al. 2009).

Yet, selecting the optimal structured order in the context of DG poses challenges. Conventional methods typically train models in an easy-to-hard order, progressively expanding the training set from simpler to more complex examples, resulting in a bias towards easy examples (Wang, Chen, and Zhu 2021). In domain generalization, easy examples are often from domains with small domain gaps. Overemphasizing these examples while overlooking hard examples hampers the model's generalizability to domains with large domain gaps. Another problem that arises is how to measure the difficulty of examples. Existing methods either utilize predefined difficulty measures (Curriculum Learning) (Bengio et al. 2009) or dynamically update example difficulty with training loss (Self-Paced Learning) (Kumar, Packer, and Koller 2010). However, predefined difficulty measures do not integrate the model's feedback, while training loss raises an inaccurate difficulty measurement issue when different examples yield identical training losses.

To address the challenges, we propose a novel learning strategy named Symmetric Self-Paced Learning (SSPL) for domain generalization, which is depicted in Figure 1. The contributions of our work are threefold:

- We demonstrate that presenting examples in a structured order can effectively improve the model's generalization ability to unseen domains. The proposed learning strategy effectively improves the model's generalizability, particularly in domains with large domain gaps.

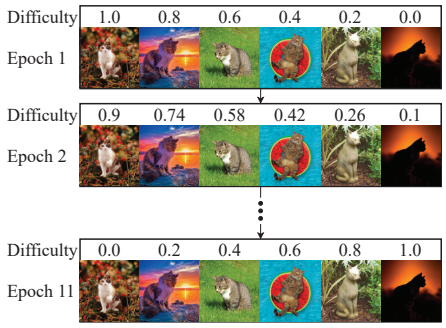


Figure 1: Symmetric Self-Paced Learning (SSPL). In the first epoch, SSPL assigns the highest weight to the easiest example (a cat in autumn) and then decreases weights up to the hardest example (a cat in dim light). As the training proceeds, SSPL gradually reduces the weight of easy examples and increases the weight of hard examples. By the final epoch, the weights between easy and hard examples are reversed: the easiest example receives zero weight, and weights then increase until the hardest example.

- We propose a Symmetric Self-Paced training scheduler that dynamically evaluates example difficulty throughout training and adjusts the data presentation order accordingly. The scheduler initially assigns larger weights to easier examples and lower weights to more challenging ones. These weights gradually change as training progresses, resulting in a reversal of weights by the end of training. This mechanism ensures balanced attention is given to examples with different difficulties.
- We propose a Gradient-based Difficulty Measure that evaluates example difficulty based on gradient magnitude. Unlike the training loss that solely quantifies the difference between predictions and ground truth, GDM also considers input data features. Consequently, GDM effectively addresses the inaccurate difficulty measurement issue. By incorporating GDM, the performance of the proposed training scheduler is further boosted.

The proposed learning strategy is designed to complement existing DG methods, making it applicable alongside any DG method. Experiments conducted on five benchmark datasets, including Digits, PACS, Office-Home, VLCS, and NICO++, demonstrate the effectiveness of the proposed learning strategy. Ablation studies further validate the effectiveness and robustness of the proposed training scheduler and difficulty measure in domain generalization. The code is available in https://github.com/RobustMM/VIGIL.

Related Work

Domain Shift. Many machine learning methods experience a performance drop when discrepancies exist between source (training) and target (testing) domains. The difference in distribution is termed the "domain shift" (Pan and Yang 2009). Domain adaptation (DA) methods have been introduced to mitigate domain shifts by aligning the marginal (Baktashmotlagh et al. 2013; Long et al. 2015) or conditional (Long et al. 2018; Luo et al. 2020) distributions of the source and target domains. DA has received considerable attention across various settings, such as semi-supervised (Saito et al. 2019) and unsupervised (Long et al. 2017) scenarios, which utilize partially labelled or unlabelled target domain data during the training phase. However, collecting target domain data in advance may not always be practical (Yue et al. 2019).

Domain Generalization. The domain generalization (DG) problem was first introduced as a machine learning problem by Blanchard, Lee, and Scott (2011) and later formally named "Domain Generalization" by Muandet, Balduzzi, and Schölkopf (2013). In medical applications (Blanchard, Lee, and Scott 2011), DG is motivated by the fact that the distribution between different patients' data is different, leading to models trained on historical patients' data failing to generalize to new patients. Moreover, acquiring data for new patients in advance is often impractical. In computer vision, Torralba and Efros (2011) proposed a seminal work for cross-domain generalization issues by investigating the cross-data generalization performance of object recognition models with five popular benchmark datasets. Recently, DG problems have gained attention in other computer vision applications (Shi et al. 2020). However, existing DG methods typically train models with data presented in a random order, which overlooks the potential impact of data presentation order on the model's generalization performance.

Curriculum Learning, proposed by Bengio et al. (2009), has demonstrated its effectiveness in improving machine learning models by presenting training examples in a structured order. However, conventional curriculum learning algorithms rely on manually designed difficulty measures to evaluate the difficulty of training data (Wang, Chen, and Zhu 2021). For example, sentence length is commonly utilized as a difficulty measure in Natural Language Processing tasks to express the complexity of a sentence or paragraph (Tay et al. 2019; Platanios et al. 2019). Similarly, measures such as data source (Chen and Gupta 2015) and signal intensity (Choi et al. 2019) have been designed for image data. But these predefined difficulty measures remain fixed during training and do not integrate the model's feedback (Wang, Chen, and Zhu 2021). To address this limitation, Kumar, Packer, and Koller (2010) introduced Self-Paced Learning (SPL), which dynamically updates the difficulty measure by using the example-wise training loss of the current model as a criterion. SPL has been successfully applied to various areas, including Multi-Task Learning (Li et al. 2017a) , Active Learning (Tang and Huang 2019), Object Detection (Sangineto et al. 2018), and Domain Adaptation (Soviany et al. 2021). Nonetheless, the effectiveness of SPL in DG has yet to be explored (Dogan et al. 2020).

Preliminaries

Notation. Let $\mathcal{X}$ denote an input feature space with dimension d and $\mathcal{Y}$ a target label space. A domain is composed of data sampled from a distribution $\mathcal{D}$, where $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^n \sim P_{(X,Y)}$, $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$, $y \in \mathcal{Y} \subset \mathbb{R}$ and n is the number of data in the domain. Here, $P_{(X,Y)}$ denotes the joint distribution of the input sample and output label, where

X and Y denote the corresponding random variables (Zhou et al. 2022; Wang et al. 2022).

Domain Generalization. For the task of domain generalization, the input is N source domains (training set), $\mathcal{S} = \left\{\mathcal{D}^i \mid i = 1, \cdots, N\right\}$, where $\mathcal{D}^i = \left\{\left(\boldsymbol{x}_j^i, y_j^i\right)\right\}_{j=1}^{n_i}$ denotes the i^{th} domain. The joint distributions between each pair of domains are different: $P_{(X,Y)}^{(i)} \neq P_{(X,Y)}^{(j)}, i \neq j$. The goal of domain generalization is to learn a robust and generalizable predictive function $f : \mathcal{X} \to \mathcal{Y}$ from the N source domains to achieve a minimum prediction error on an unseen target domain $\mathcal{T}$, where $\mathcal{T}$ cannot be accessed during training and $P_{(X,Y)}^{(\mathcal{T})} \neq P_{(X,Y)}^{(i)}$ for $i \in \{1, \cdots, N\}$.

Methodology

We begin by describing the learning objective of SSPL, followed by the proposed Symmetric Self-Paced training scheduler. Then we outline the proposed Gradient-based Difficulty Measure. The structure of SSPL is illustrated in Figure 2, and the algorithm is summarized in Algorithm 1.

Learning Objective

The learning objective of Symmetric Self-Paced Learning for domain generalization is formulated as follows:

$$\min_{w} \mathbb{E}(w) \sum_{i=1}^{N} v_i(\ell\left(f_w(T(\boldsymbol{x}_i)), y_i\right)). \tag{1}$$

Here, $v_i \in [0, 1]$ denotes the weight assigned to example $\boldsymbol{x}_i$ and dynamically changes as the training proceeds; $\ell(\cdot, \cdot)$ denotes the loss function; $f_w(\cdot)$ denotes the predictive function, parameterized by w; and $T(\cdot)$ denotes the domain generalization methods, which can be any existing DG method.

Algorithm 1: Symmetric Self-Paced Learning for Domain Generalization

Input: $\mathcal{D}$: training set; $f_w(\cdot)$: the learning model parameterized by w; $T(\cdot)$: domain generalization method; $\ell(\cdot, \cdot)$: loss function; n_e: maximum number of epochs.
Output: w^*: the optimal parameters for $f_w(\cdot)$
Compute γ_e by Eq. 2 ▷ Compute epoch step size
for $i = 1$ to n_e **do**
    Compute v_e^i by Eq. 3 ▷ Compute the weight assigned to the easiest example in epoch i
    Compute v_h^i by Eq. 4 ▷ Compute the weight assigned to the hardest example in epoch i
    Compute γ_d^i by Eq. 5 ▷ Compute element step size for epoch i
    Compute ξ_x by Eq. 8 ▷ Compute difficulty for each example
    Sort($\mathcal{D}, \xi_x$) ▷ Sort examples according to their difficulty rank.
    Compute v_x^i by Eq. 6 ▷ Compute weight for each example according to their difficulty
    $\ell_x = v_x^i \cdot \ell(f_w(T(x)), y)$ ▷ Compute weighted loss
    Update w^*
end for

Symmetric Self-Paced Training Scheduler

Conventional Curriculum Learning and Self-Paced Learning training schedulers progressively increase the training set from easier to harder examples until the entire training set is included (Wang, Chen, and Zhu 2021; Soviany et al. 2022). However, this approach biases the training towards easy examples, which are trained more frequently than hard ones. Although frequently trained easy examples accelerate initial convergence, hard examples are more informative for learning in the later stages (Shrivastava, Gupta, and Girshick 2016). Overlooking these hard examples compromises data sample diversity, resulting in a suboptimal training process and guiding the model towards a suboptimal solution. In domain generalization, easy examples are often from domains with small domain gaps. Overemphasizing these examples while overlooking hard examples hampers the model's generalizability to domains with large domain gaps.

To address this challenge, we propose a Symmetric Self-Paced training scheduler that ensures balanced attention is given to examples with different difficulties. The proposed training scheduler dynamically adjusts weights assigned to easy and hard examples, resulting in a reversal of weights between easy and hard examples at the end of training. As depicted in Figure 1, in the first epoch, the easiest examples receive a weight of one while the hardest ones are assigned zero weight. Throughout the training, the weights of easy examples gradually decrease while those of hard examples increase. By the last epoch, the weights assigned to the easiest and hardest examples have been reversed.

As previously mentioned, throughout the training process, from the first epoch to the final one, the weight assigned to the easiest example decreases from one to zero, while the weight assigned to the hardest example increases from zero to one. Consequently, the epoch step size γ_e, which signifies the magnitude of the weight modifications for the easiest and hardest examples, is computed with the following equation,

$$\gamma_e = \frac{v_e^1 - v_e^{n_e}}{n_e} = \frac{1}{n_e}. \tag{2}$$

In this context, n_e is the number of training epochs, while v_e^1 and $v_e^{n_e}$ designate the weights assigned to the easiest example in the first and final epochs, respectively, with $v_e^1 = 1$ and $v_e^{n_e} = 0$. Therefore, the weights assigned to the easiest and hardest examples in epoch i, denoted as v_e^i and v_h^i, are computed with Equations 3 and 4.

$$v_e^i = v_e^1 - \gamma_e \cdot (i - 1) = 1 - \frac{i - 1}{n_e} \tag{3}$$

$$v_h^i = v_h^1 + \gamma_e \cdot (i - 1) = 0 + \frac{i - 1}{n_e} \tag{4}$$

Once v_e^i and v_h^i are calculated, the element step size, γ_d^i, can be determined, which represents the magnitude of weight changes for each example within the i^{th} epoch. The computation of γ_d^i is depicted in Equation 5, where n_d is the total number of examples.

$$\gamma_d^i = \frac{v_e^i - v_h^i}{n_d} \tag{5}$$

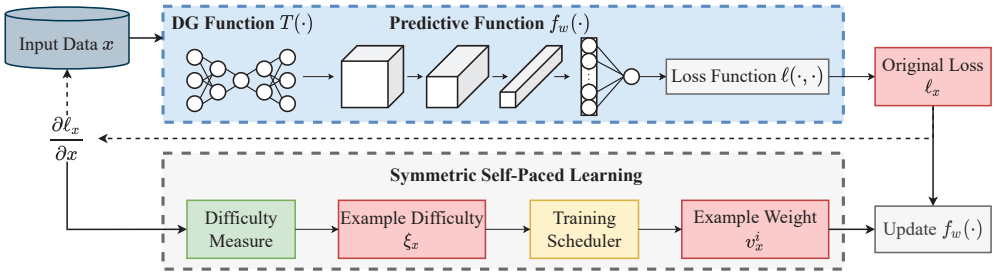


Figure 2: Overview of the Symmetric-Self Paced Learning (SSPL) for domain generalization within one epoch. Given training data x and domain generalization function $T(\cdot)$, the target predictive function $f_w(\cdot)$ yields the original loss ℓ_x. Subsequently, SSPL calculates the difficulty ξ_x for x with the provided difficulty measure, and the training scheduler determines its weight, v_x^i, based on the difficulty rank. Then, $f_w(\cdot)$ is updated using the weighted loss.

Finally, the proposed training scheduler calculates the weight, denoted as v_j^i, for each example within the i^{th} epoch, where j signifies the example ranked j^{th} in terms of difficulty. The weight v_j^i is calculated by Equation 6.

$$v_j^i = v_e^1 - \gamma_e \cdot (i-1) - \gamma_d^i \cdot j \tag{6}$$

Note that γ_d^i can be computed either globally or locally. When computed globally, γ_d^i is computed over the entire dataset at once for each epoch. This approach assigns unique weights to examples based on their difficulty rank across the entire dataset, thereby providing precise weight calculation. However, computing γ_d^i globally can be computationally expensive as it necessitates storing difficulty information for each example until the end of an epoch, limiting its scalability for large datasets. To address this limitation, γ_d^i can be computed locally with Equation 7,

$$\gamma_d^i = \frac{v_e^i - v_h^i}{n_b}, \tag{7}$$

where n_b denotes the batch size used for model training. The local γ_d^i is computed batch-wise and weights are assigned to examples based on their difficulty rank within a batch. Since each batch is sampled independently, the local γ_d^i serves as an estimate of the global γ_d^i. As the batch size increases, the local γ_d^i approximates the global γ_d^i more closely. When the batch size equals the dataset size, the local γ_d^i will be equivalent to the global γ_d^i. Although the local γ_d^i is less precise than the global γ_d^i, it is fast to compute and devoid of the need to store the difficulty of examples. The trade-off between accuracy and computational efficiency makes the local γ_d^i a practical alternative in scenarios where the computational overhead or memory constraints are a concern.

Gradient-based Difficulty Measure

Current methods evaluate example difficulty through predefined metrics, such as sentence length, or dynamically update it based on training loss, such as cross-entropy loss. However, predefined difficulty measures do not integrate the model's feedback, and training loss, focusing only on the difference between predictions and ground truth, raises an inaccurate difficulty measurement issue when different examples yield identical training losses.

To address these limitations, we propose the Gradient-based Difficulty Measure (GDM), which evaluates example difficulty through dynamic measurement of the gradient magnitude with respect to the example itself. Unlike training loss, the gradient avoids inaccurate difficulty assessment by taking input features into account. As a result, even if examples yield the same loss, their gradients can differ. Additionally, loss landscapes can encompass plateaus or saddle points, where training loss remains relatively stable even with substantial shifts in model parameters. In these scenarios, utilizing training loss for evaluating example difficulty can be misleading. Conversely, the gradient provides finer-grained insights into changes in model parameters, making the gradient magnitude a more informative approach for evaluating difficulty. The GDM is computed with Equation 8, where $\xi_{\boldsymbol{x}}$ denotes the difficulty of the example $\boldsymbol{x}$.

$$\xi_{\boldsymbol{x}} = \left\| \frac{\partial \ell(f(\boldsymbol{x}), y)}{\partial \boldsymbol{x}} \right\|_2 \tag{8}$$

Experiments

Experiment Setting

Datasets. The proposed approach is evaluated on five popular domain generalization benchmark datasets, which cover a variety of image classification problems. (1) **Digits** (Zhou et al. 2020) consists of four digit recognition tasks, namely MNIST (LeCun et al. 1998), MNIST-M (Ganin and Lempitsky 2015), SVHN (Netzer et al. 2011), and SYN (Ganin and Lempitsky 2015). (2) **PACS** (Li et al. 2017b) consists of four domains, namely Photo, Art Painting, Cartoon and Sketch. (3) **Office-Home** (Venkateswara et al. 2017) was initially introduced for domain adaptation and is becoming popular in the DG community (Carlucci et al. 2019). It contains four domains: Artistic, Clipart, Product, and Real World, where each domain has 65 classes related to office and home objects. (4) **VLCS** (Fang, Xu, and Rockmore 2013) consists of four domains of data collected from Caltech101 (Fei-Fei, Fergus, and Perona 2004), PASCAL (Everingham et al. 2010), LabelMe (Russell et al. 2008), and SUN (Choi et al. 2010), where five common categories are collected: bird, car, chair, dog and person. (5) **NICO++** (Zhang et al. 2023) is the latest domain generalization dataset that was

constructed in 2023 for OOD (Out-of-Distribution) image classification. Compared with the previous four datasets, NICO++ is much larger in scale, with 88,866 images in total. Figure 3 depicts example images showcasing domain gaps across benchmark datasets. Due to page constraints, a comprehensive illustration is provided in the supplementary[1].

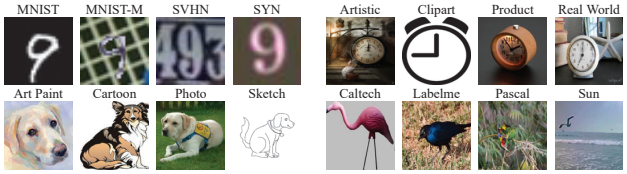


Figure 3: Example images from Digits (1st row 1-4 columns), PACS (2nd row 1-4 columns), Office-Home (1st row 5-8 columns), and VLCS (2nd row 5-8 columns) datasets demonstrate the presence of domain gaps, posing significant challenges for domain generalization.

Baselines. We assess the efficacy and mode-agnostic characteristic of our learning strategy by incorporating several state-of-the-art domain generalization methods from various categories. These methods include CrossGrad (Shankar et al. 2018), MixStyle (Zhou et al. 2021), DomainMix (Sun et al. 2022), and EFDMix (Zhang et al. 2022). Additionally, we apply our strategy alongside classic data augmentation techniques such as RandomErasing, RandomRotation, Flip, and ColorJitter. An Empirical Risk Minimization (ERM) baseline is also included, which merges data from all source domains without utilizing domain generalization techniques.

Evaluation Metrics. We adopt the leave-one-out-test evaluation strategy as the evaluation metric following the prior works (Li et al. 2017b; Carlucci et al. 2019; Li et al. 2019; Zhou et al. 2022). Specifically, we select one domain as the test domain at a time and use the remaining domains as the source domains for training. We report the accuracy for each separate domain. Performance measures are reported as top-1 classification accuracy (%) averaged over ten runs, along with their corresponding 95% confidence intervals.

Network Structure. The network structure is chosen by following the previous work (Carlucci et al. 2019; Li et al. 2019; Zhou et al. 2020). In the Digits dataset, images are resized to 32×32 and converted to RGB by replicating channels. The backbone of the neural network is constructed by 3×3 Conv layers (64 kernels), each followed by a ReLU activation function and a 2×2 max-pooling layer. For the PACS, Office-Home, VLCS and NICO++ datasets, images are resized to 224×224, and the ImageNet pretrained ResNet18 (He et al. 2016) was chosen as the backbone.

Training. Our methodology is implemented using the PyTorch libraries. The optimizer utilized for training is Stochastic Gradient Descent (SGD), with a momentum of 0.9 and a weight decay of 5e-4. For the Digits dataset, we train the networks with an initial learning rate of 0.05 and a batch size of 64 for 50 epochs. The learning rate is decayed by a factor of 0.1 every 20 epochs. For the PACS, Office-Home, and VLCS datasets, the networks are trained with a learning rate of 0.01 and a batch size of 32 for 50 epochs. For the NICO++ dataset, the networks are trained with a learning rate of 0.005 and a batch size of 64 for 50 epochs. All experiments are conducted on NVIDIA Tesla A100 GPUs.

[1]Please refer to the version with Appendix in arXiv.

	MNIST	MNIST-M	SVHN	SYN	Avg.
ERM	96.4 ± .2	62.6 ± .1	66.7 ± .1	83.8 ± .2	77.4 ± .2
ERM_{SS}	96.9 ± .1	65.5 ± .4	67.7 ± .4	84.4 ± .3	78.6 ± .3
Improv.	↑**0.54%**	↑**4.61%**	↑**1.42%**	↑**0.67%**	↑**1.55%**
Erasing	95.0 ± .2	55.4 ± .2	68.5 ± .4	83.7 ± .4	75.7 ± .3
$\text{Erasing}_{\text{SS}}$	95.9 ± .1	57.9 ± .4	71.6 ± .4	84.9 ± .5	77.6 ± .4
Improv.	↑**0.93%**	↑**4.49%**	↑**4.57%**	↑**1.41%**	↑**2.51%**
Rotation	95.0 ± .5	55.4 ± .6	68.4 ± .4	83.7 ± .4	75.6 ± .5
$\text{Rotation}_{\text{SS}}$	97.3 ± .2	64.5 ± .3	69.9 ± .4	88.6 ± .7	80.1 ± .4
Improv.	↑**2.42%**	↑**16.31%**	↑**2.12%**	↑**5.84%**	↑**5.95%**
ColorJitter	96.6 ± .3	65.7 ± .3	68.8 ± .3	83.8 ± .3	78.7 ± .3
$\text{ColorJitter}_{\text{SS}}$	96.9 ± .3	68.2 ± .3	70.5 ± .4	85.1 ± .6	80.2 ± .4
Improv.	↑0.17%	↑**3.71%**	↑**2.34%**	↑**1.53%**	↑**1.91%**
CrossGrad	96.1 ± .3	62.2 ± .3	65.6 ± .3	83.3 ± .3	76.8 ± .3
$\text{CrossGrad}_{\text{SS}}$	97.1 ± .4	63.5 ± .6	68.9 ± .4	84.2 ± .2	78.4 ± .4
Improv.	↑**0.96%**	↑**2.07%**	↑**4.92%**	↑**0.97%**	↑**2.08%**
MixStyle	96.7 ± .2	64.4 ± .4	71.1 ± .2	85.3 ± .3	79.4 ± .3
$\text{MixStyle}_{\text{SS}}$	97.3 ± .4	65.9 ± .4	72.7 ± .4	86.7 ± .3	80.7 ± .4
Improv.	↑0.55%	↑**2.20%**	↑**2.26%**	↑**1.59%**	↑**1.64%**
DomainMix	95.1 ± .4	57.4 ± .3	66.3 ± .5	77.6 ± .4	74.1 ± .4
$\text{DomainMix}_{\text{SS}}$	96.3 ± .6	61.0 ± .3	69.4 ± .6	77.8 ± .4	76.1 ± .5
Improv.	↑**1.23%**	↑**6.09%**	↑**4.57%**	↑0.23%	↑**2.70%**
EFDMix	96.4 ± .2	65.1 ± .6	73.1 ± .4	85.7 ± .4	80.1 ± .4
$\text{EFDMix}_{\text{SS}}$	96.6 ± .2	67.0 ± .4	74.2 ± .6	87.2 ± .5	81.3 ± .4
Improv.	↑0.16%	↑**2.84%**	↑**1.42%**	↑**1.73%**	↑**1.50%**

Table 1: Leave-one-domain-out results on Digit dataset (with 95% confidence intervals).

Experimental Results

In the presented tables, significant improvements are highlighted in bold, while minor improvements and declines remain in regular text. The subscripts S and SS denote the results of the baseline integrated with classic SPL and SSPL, respectively. Due to page constraints, we only show evaluation results of the Digits, PACS, Office-Home, and VLCS datasets. Please refer to the supplementary material for the results of the Flip baseline and NICO++ dataset.

Evaluation on Digits. Table 1 presents the enhanced performance achieved through our SSPL strategy across all domains, in combination with various baselines. Notable improvements of up to 2.42%, 16.31%, 4.92%, and 5.84% are observed in the MNIST, MNIST-M, SVHN, and SYN domains, respectively. Intriguingly, SSPL amplifies the effectiveness of Random Rotation, achieving state-of-the-art performance in the MNIST and SYN domains, surpassing most existing Domain Generalization techniques. Although Random Rotation alone yields a modest 55.41% accuracy in the MNIST-M domain, integrating it with SSPL boosts performance to 64.45%, aligning with most baselines. As reflected in Table 1, SSPL significantly enhances the model's generalization performance to target domains with substantial domain gaps, such as MNIST-M and SVHN. On the other hand, SSPL also achieves modest improvements in domains

with a smaller domain gap, like MNIST.

Evaluation on PACS. The key findings from Table 2 are summarized as follows. (1) Our SSPL strategy gains accuracy improvement of up to 4.58%, 6.12%, 1.11%, and 10.41% in the Art Painting, Cartoon, Photo, and Sketch domains, respectively. (2) These improvements are not linked to specific domain generalization methods but are more closely correlated to the magnitude of the domain gap. As earlier noted, regardless of baseline methods, SSPL consistently yields significant improvements in domains with large domain gaps, such as Cartoon and Sketch, and moderate improvements in the domains with smaller gaps, like Photo (see Figure 3). This observation underscores the model-agnostic attributes of the proposed learning strategy.

	Art	Cartoon	Photo	Sketch	Avg.
ERM	75.8 ± .4	72.7 ± .2	94.9 ± .2	62.9 ± .1	76.6 ± .2
ERM_{SS}	79.3 ± .4	75.3 ± .2	95.9 ± .5	69.5 ± .4	80.0 ± .4
Improv.	↑**4.58%**	↑**3.66%**	↑**1.11%**	↑**10.41%**	↑**4.44%**
Erasing	77.2 ± .4	71.5 ± .3	95.6 ± .3	63.0 ± .4	76.8 ± .4
$Erasing_{SS}$	80.1 ± .4	73.4 ± .3	95.9 ± .5	66.8 ± .5	79.1 ± .4
Improv.	↑**3.82%**	↑**2.67%**	↑0.38%	↑**5.99%**	↑**2.99%**
Rotation	76.8 ± .4	69.7 ± .5	95.6 ± .4	67.9 ± .4	77.5 ± .4
$Rotation_{SS}$	79.5 ± .5	71.6 ± .5	95.7 ± .4	69.9 ± .3	79.2 ± .4
Improv.	↑**3.53%**	↑**2.74%**	↑0.14%	↑**2.90%**	↑**2.19%**
ColorJitter	76.3 ± .1	67.3 ± .3	94.7 ± .3	68.6 ± .5	76.7 ± .3
$ColorJitter_{SS}$	79.0 ± .3	71.5 ± .6	95.4 ± .4	72.1 ± .4	79.5 ± .4
Improv.	↑**3.47%**	↑**6.12%**	↑0.73%	↑**5.03%**	↑**3.65%**
CrossGrad	76.5 ± .3	72.3 ± .3	94.9 ± .5	61.8 ± .7	76.4 ± .5
$CrossGrad_{SS}$	79.1 ± .2	74.7 ± .6	95.5 ± .4	65.8 ± .3	78.8 ± .4
Improv.	↑**3.44%**	↑**3.36%**	↑0.63%	↑**6.41%**	↑**3.14%**
MixStyle	77.8 ± .5	73.8 ± .3	95.6 ± .6	64.6 ± .8	78.0 ± .5
$MixStyle_{SS}$	80.8 ± .6	75.5 ± .6	96.0 ± .5	69.3 ± .4	80.4 ± .5
Improv.	↑**3.83%**	↑**2.28%**	↑0.41%	↑**7.26%**	↑**3.08%**
DomainMix	77.9 ± .5	64.1 ± .3	94.1 ± .7	58.6 ± .3	73.7 ± .5
$DomainMix_{SS}$	79.2 ± .5	67.2 ± .5	94.3 ± .3	62.4 ± .6	75.8 ± .5
Improv.	↑**1.63%**	↑**4.95%**	↑0.19%	↑**6.38%**	↑**2.85%**
EFDMix	82.3 ± .3	75.4 ± .4	95.7 ± .5	71.6 ± .3	81.5 ± .4
$EFDMix_{SS}$	83.6 ± .3	77.3 ± .5	96.0 ± .3	74.2 ± .7	82.8 ± .5
Improv.	↑**1.65%**	↑**2.44%**	↑0.29%	↑**3.66%**	↑**1.60%**

Table 2: Leave-one-domain-out results on PACS dataset (with 95% confidence intervals).

Evaluation on Office-Home and VLCS. From Tables 3 and 4, we observe similar results as in Tables 1 and 2. In the Office-Home dataset, SSPL gains improvements of up to 2.27%, 5.33%, 2.07%, and 1.42% in the Artistic, Clipart, Product, and RealWorld domains. Similarly, in the VLCS dataset, SSPL gains improvements of up to 2.11%, 5.37%, 3.26%, and 7.49% in the Caltech, Labelme, Pascal, and Sun domains. Once more, SSPL consistently achieves notable improvements in domains with large domain gaps and modest improvements in domains with smaller domain gaps. These results further underscore the effectiveness of SSPL in addressing domain generalization challenges, particularly in contexts with substantial domain shifts.

Ablation Study

Comparison with Classic Self-Paced Learning (SPL) Algorithms. To further validate the effectiveness of the proposed learning strategy, we compare it with the classic Self-Paced Learning (SPL) strategy. The comparison results are shown in Table 5. Due to space constraints, Table 5 only includes comparison results using Empirical Risk Minimization (ERM) as the baseline. A complete comparison across all baselines is provided in the supplementary material.

	Artistic	Clipart	Product	RealWorld	Avg.
ERM	58.6 ± .3	47.9 ± .4	73.7 ± .4	75.8 ± .4	64.0 ± .3
ERM_{SS}	59.4 ± .4	49.2 ± .2	74.3 ± .3	76.0 ± .3	64.7 ± .3
Improv.	↑**1.28%**	↑**2.67%**	↑0.73%	↑0.21%	↑**1.09%**
Erasing	59.5 ± .6	47.1 ± .4	73.6 ± .3	75.2 ± .3	63.9 ± .4
$Erasing_{SS}$	59.7 ± .3	49.1 ± .4	75.1 ± .6	76.3 ± .3	65.1 ± .4
Improv.	↑0.49%	↑**4.31%**	↑**2.07%**	↑**1.42%**	↑**1.88%**
Rotation	57.3 ± .5	45.5 ± .5	73.7 ± .4	74.4 ± .3	62.7 ± .4
$Rotation_{SS}$	58.2 ± .4	46.3 ± .3	74.5 ± .4	74.6 ± .4	63.4 ± .4
Improv.	↑**1.59%**	↑**1.78%**	↑**1.13%**	↑0.14%	↑1.12%
ColorJitter	56.8 ± .3	48.0 ± .4	70.9 ± .3	73.2 ± .4	62.2 ± .4
$ColorJitter_{SS}$	58.1 ± .5	50.6 ± .5	71.3 ± .4	73.7 ± .4	63.4 ± .5
Improv.	↑**2.27%**	↑**5.33%**	↑0.49%	↑0.59%	↑**1.93%**
CrossGrad	58.4 ± .6	47.9 ± .5	73.7 ± .2	75.2 ± .3	63.8 ± .4
$CrossGrad_{SS}$	59.4 ± .3	48.2 ± .3	74.5 ± .3	75.6 ± .5	64.4 ± .4
Improv.	↑**1.69%**	↑0.69%	↑**1.00%**	↑0.48%	↑0.94%
MixStyle	59.2 ± .3	48.6 ± .3	73.9 ± .4	75.4 ± .2	64.3 ± .3
$MixStyle_{SS}$	60.2 ± .3	49.9 ± .6	74.5 ± .5	75.5 ± .3	65.0 ± .3
Improv.	↑**1.69%**	↑**2.51%**	↑0.76%	↑0.17%	↑**1.09%**
DomainMix	57.4 ± .4	45.8 ± .5	73.1 ± .4	75.2 ± .4	62.9 ± .4
$DomainMix_{SS}$	58.4 ± .3	47.2 ± .4	73.9 ± .3	75.3 ± .2	63.7 ± .3
Improv.	↑**1.83%**	↑**2.86%**	↑**1.05%**	↑0.20%	↑**1.27%**
EFDMix	60.1 ± .3	52.0 ± .4	73.8 ± .4	75.2 ± .3	65.3 ± .4
$EFDMix_{SS}$	60.4 ± .3	53.5 ± .3	74.5 ± .4	75.3 ± .2	65.9 ± .3
Improv.	↑0.58%	↑**2.87%**	↑0.92%	↑0.06%	↑0.92%

Table 3: Leave-one-domain-out results on Office-Home dataset (with 95% confidence intervals).

Comparing the results presented in Tables 1 - 5, we can see there is a notable decline in generalization performance when integrating classic Self-Paced Learning strategy with ERM, particularly in domains with large domain gaps, such as SVHN, Sketch and Pascal. Additionally, there is an approximate 20% performance drop across all domains in the Office-Home dataset. In contrast, SSPL demonstrates a significant improvement in these domains. These observations further demonstrate the effectiveness of the proposed learning strategy and highlight the importance of hard examples in enhancing model generalizability. Overlooking these examples hampers the model's generalization performance in unseen domains.

Effectiveness of GDM. To validate the effectiveness of our proposed difficulty measure, we conduct a comparative analysis with the conventional difficulty measure, training loss, as depicted in Table 6. Given that the benchmark datasets are designed for image classification tasks, the training loss criterion is based on cross-entropy. For this analysis, we also use ERM as the baseline. As illustrated in Table 6, the Gradient-based Difficulty Measure (GDM) consistently outperforms cross-entropy loss, particularly in domains with substantial domain gaps, such as MNIST-M, Cartoon, Clipart, and Sun. These results provide evidence of the effectiveness of GDM, highlighting gradient magnitude as a more

	Caltech	Labelme	Pascal	Sun	Avg.
ERM	96.0 ± .3	63.2 ± .4	74.1 ± .2	70.7 ± .6	76.0 ± .4
ERM_{SS}	96.9 ± .3	65.0 ± .2	75.7 ± .4	73.7 ± .4	77.8 ± .4
Improv.	↑**0.91%**	↑**2.78%**	↑**2.06%**	↑**4.26%**	↑**2.37%**
Erasing	96.0 ± .3	62.5 ± .4	75.3 ± .5	69.8 ± .3	75.9 ± .4
$Erasing_{SS}$	96.9 ± .4	64.0 ± .4	75.6 ± .6	71.6 ± .4	77.0 ± .5
Improv.	↑**0.90%**	↑**2.34%**	↑0.38%	↑**2.52%**	↑**1.45%**
Rotation	95.0 ± .5	61.4 ± .5	70.3 ± .3	67.2 ± .2	73.5 ± .4
$Rotation_{SS}$	96.2 ± .2	62.8 ± .5	70.7 ± .4	70.4 ± .6	75.0 ± .4
Improv.	↑**1.28%**	↑**2.21%**	↑0.48%	↑**4.76%**	↑**2.04%**
ColorJitter	92.9 ± .6	64.0 ± .4	67.9 ± .3	62.9 ± .1	71.9 ± .4
$ColorJitter_{SS}$	94.0 ± .5	66.4 ± .5	69.4 ± .5	67.6 ± .4	74.4 ± .5
Improv.	↑**1.15%**	↑**3.72%**	↑**2.21%**	↑**7.49%**	↑**3.48%**
CrossGrad	96.2 ± .4	63.2 ± .2	74.2 ± .5	70.7 ± .5	76.1 ± .4
$CrossGrad_{SS}$	97.2 ± .6	66.5 ± .3	75.0 ± .3	73.4 ± .4	78.0 ± .4
Improv.	↑**1.03%**	↑**5.37%**	↑1.07%	↑**3.95%**	↑**2.50%**
MixStyle	95.9 ± .3	63.3 ± .4	74.1 ± .4	70.3 ± .5	75.9 ± .4
$MixStyle_{SS}$	96.5 ± .3	64.5 ± .4	74.7 ± .4	73.0 ± .4	77.2 ± .4
Improv.	↑0.58%	↑**1.90%**	↑0.81%	↑**3.88%**	↑**1.71%**
DomainMix	93.9 ± .4	63.2 ± .3	70.0 ± .3	69.1 ± .3	74.1 ± .3
$DomainMix_{SS}$	95.9 ± .4	63.7 ± .4	72.3 ± .3	71.6 ± .4	75.9 ± .4
Improv.	↑**2.11%**	↑0.86%	↑**3.26%**	↑**3.60%**	↑**2.43%**
EFDMix	96.9 ± .4	62.9 ± .3	74.7 ± .3	70.3 ± .3	76.2 ± .3
$EFDMix_{SS}$	98.0 ± .5	63.8 ± .3	75.6 ± .3	73.3 ± .4	77.7 ± .4
Improv.	↑**1.17%**	↑**1.56%**	↑**1.18%**	↑**4.22%**	↑**1.97%**

Table 4: Leave-one-domain-out results on VLCS dataset (with 95% confidence intervals).

informative approach for evaluating example difficulty in domain generalization. Notably, even when solely utilizing cross-entropy loss as the difficulty measure, our Symmetric Self-Paced Learning strategy still yields significant performance improvement across most domains, highlighting its robustness and effectiveness. To further validate the effectiveness of GDM, we also compare GDM and cross-entropy loss within the classic SPL framework. Please refer to the supplementary material for detailed results.

Further Analysis

Class-specific clusters are seen in the visualization of SSPL's feature embeddings in Figure 4a, demonstrating SSPL's efficacy at classifying classes. Figure 4b shows the average difficulty per iteration, confirming that SSPL allows the model to learn effective representations, as evidenced by the convergence of the average example difficulty. This also suggests that SSPL does not lead to forgetting the initial easy examples since this would increase these examples' gradients and prevent difficulty convergence.

Conclusion

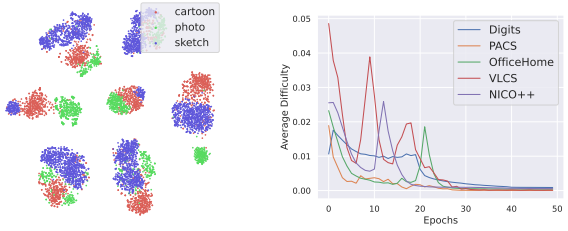
This paper proposes a novel learning strategy, Symmetric Self-Paced Learning (SSPL), for domain generalization. It effectively improves the model's generalization ability to unseen domains, particularly in domains with large domain gaps. SSPL consists of a Symmetric Self-Paced training scheduler and a Gradient-based Difficulty Measure (GDM). The proposed Symmetric Self-Paced training scheduler ensures balanced attention is given to examples with different

	MNIST	MNIST-M	SVHN	SYN	Avg.
ERM_S	90.5 ± .5	55.4 ± .4	48.7 ± .4	64.9 ± .8	64.9 ± .5
Improv.	↓6.07%	↓11.59%	↓26.99%	↓22.56%	↓16.15%
ERM_{SS}	96.9 ± .1	65.5 ± .4	67.7 ± .4	84.4 ± .3	78.6 ± .3
Improv.	↑**0.54%**	↑**4.61%**	↑**1.42%**	↑**0.67%**	↑**1.55%**
	Art	Cartoon	Photo	Sketch	Avg.
ERM_S	66.4 ± .9	71.0 ± .4	91.0 ± .2	55.3 ± .8	70.9 ± .6
Improv.	↓12.43%	↓2.27%	↓4.07%	↓12.05%	↓7.44%
ERM_{SS}	79.3 ± .4	75.3 ± .2	95.9 ± .5	69.5 ± .4	80.0 ± .4
Improv.	↑**4.58%**	↑**3.66%**	↑**1.11%**	↑**10.49%**	↑**4.44%**
	Artistic	Clipart	Product	RealWorld	Avg.
ERM_S	48.0 ± .5	38.5 ± .4	60.0 ± .3	60.1 ± .7	51.7 ± .5
Improv.	↓18.18%	↓19.56%	↓18.67%	↓20.66%	↓19.22%
ERM_{SS}	59.4 ± .4	49.2 ± .2	74.3 ± .3	76.0 ± .3	64.7 ± .3
Improv.	↑**1.28%**	↑**2.67%**	↑0.73%	↑0.21%	↑**1.09%**
	Caltech	Labelme	Pascal	Sun	Avg.
ERM_S	96.5 ± .4	64.0 ± .3	64.5 ± .5	54.0 ± .5	69.8 ± .4
Improv.	↑0.49%	↑1.17%	↓13.06%	↓23.06%	↓8.16%
ERM_{SS}	96.9 ± .3	65.0 ± .2	75.7 ± .4	73.7 ± .4	77.8 ± .4
Improv.	↑**0.91%**	↑**2.78%**	↑**2.06%**	↑**4.26%**	↑**2.37%**

Table 5: Comparison between SSPL and SPL.

	MNIST	MNIST-M	SVHN	SYN	Avg.
Loss	96.6 ± .3	64.7 ± .3	67.0 ± .2	84.1 ± .2	78.1 ± .3
GDM	96.9 ± .1	65.5 ± .4	67.7 ± .4	84.4 ± .3	78.6 ± .3
	Art	Cartoon	Photo	Sketch	Avg.
Loss	77.8 ± .2	73.6 ± .3	95.3 ± .3	67.3 ± .2	78.5 ± .3
GDM	79.3 ± .4	75.3 ± .2	95.9 ± .5	69.5 ± .4	80.0 ± .4
	Artistic	Clipart	Product	RealWorld	Avg.
Loss	59.2 ± .2	48.5 ± .3	74.1 ± .2	75.5 ± .1	64.3 ± .2
GDM	59.4 ± .4	49.2 ± .2	74.3 ± .3	76.0 ± .3	64.7 ± .3
	Caltech	Labelme	Pascal	Sun	Avg.
Loss	96.6 ± .2	64.3 ± .3	75.0 ± .3	72.9 ± .2	77.2 ± .3
GDM	96.9 ± .3	65.0 ± .2	75.7 ± .4	73.7 ± .4	77.8 ± .4

Table 6: Comparison between GDM and Loss.



(a) T-SNE (b) Difficulty Change

Figure 4: Visualization of Further Analysis.

difficulties by gradually shifting emphasis from easy to hard examples as training progresses. The proposed difficulty measure dynamically evaluates example difficulty through the gradient magnitude with respect to the example itself. GDM avoids the inaccurate example difficulty measurement issue and provides a more informative difficulty evaluation. In future research, we aim to extend SSPL to other challenging tasks, such as person re-identification, semantic segmentation, and machine translation.

Acknowledgements

This research was supported by New Zealand MBIE Strategic Science Investment Fund (SSIF) Data Science platform - Time-Evolving Data Science / Artificial Intelligence for Advanced Open Environmental Science (UOWX1910).

References

Arpit, D.; Wang, H.; Zhou, Y.; and Xiong, C. 2022. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *NeurIPS*, 35: 8265–8277.

Baktashmotlagh, M.; Harandi, M. T.; Lovell, B. C.; and Salzmann, M. 2013. Unsupervised domain adaptation by domain invariant projection. In *ICCV*, 769–776.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *ICML*, 41–48.

Blanchard, G.; Lee, G.; and Scott, C. 2011. Generalizing from several related classification tasks to a new unlabeled sample. *NeurIPS*, 24.

Carlucci, F. M.; D'Innocente, A.; Bucci, S.; Caputo, B.; and Tommasi, T. 2019. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2229–2238.

Chen, X.; and Gupta, A. 2015. Webly supervised learning of convolutional networks. In *ICCV*, 1431–1439.

Choi, J.; Jeong, M.; Kim, T.; and Kim, C. 2019. Pseudo-labeling curriculum for unsupervised domain adaptation. *arXiv preprint*.

Choi, M. J.; Lim, J. J.; Torralba, A.; and Willsky, A. S. 2010. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 129–136.

Dogan, Ü.; Deshmukh, A. A.; Machura, M. B.; and Igel, C. 2020. Label-similarity curriculum learning. In *ECCV*, 174–190. Springer.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *IJCV*, 303–338.

Fang, C.; Xu, Y.; and Rockmore, D. N. 2013. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *ICCV*, 1657–1664.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPRw*, 178–178.

Fernando, B.; Habrard, A.; Sebban, M.; and Tuytelaars, T. 2013. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, 2960–2967.

Ganin, Y.; and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*, 1180–1189.

Ghifary, M.; Kleijn, W. B.; Zhang, M.; and Balduzzi, D. 2015. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, 2551–2559.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Hendrycks, D.; and Dietterich, T. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *ICLR*.

Kumar, M.; Packer, B.; and Koller, D. 2010. Self-paced learning for latent variable models. *NeurIPS*, 23.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *IEEE*, 2278–2324.

Li, C.; Yan, J.; Wei, F.; Dong, W.; Liu, Q.; and Zha, H. 2017a. Self-Paced Multi-Task Learning. *AAAI*.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. 2018. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, volume 32.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. M. 2017b. Deeper, broader and artier domain generalization. In *ICCV*, 5542–5550.

Li, D.; Zhang, J.; Yang, Y.; Liu, C.; Song, Y.-Z.; and Hospedales, T. M. 2019. Episodic training for domain generalization. In *ICCV*, 1446–1455.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. 2015. Learning transferable features with deep adaptation networks. In *ICML*, 97–105.

Long, M.; Cao, Z.; Wang, J.; and Jordan, M. I. 2018. Conditional adversarial domain adaptation. *NeurIPS*, 31.

Long, M.; Zhu, H.; Wang, J.; and Jordan, M. I. 2017. Deep transfer learning with joint adaptation networks. In *ICML*, 2208–2217. PMLR.

Luo, Y.; Wang, Z.; Huang, Z.; and Baktashmotlagh, M. 2020. Progressive graph learning for open-set domain adaptation. In *ICML*, 6468–6478. PMLR.

Meng, R.; Li, X.; Chen, W.; Yang, S.; Song, J.; Wang, X.; Zhang, L.; Song, M.; Xie, D.; and Pu, S. 2022. Attention diversification for domain generalization. In *ECCV*, 322–340. Springer.

Muandet, K.; Balduzzi, D.; and Schölkopf, B. 2013. Domain generalization via invariant feature representation. In *ICML*, 10–18.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. *NeurIPS*.

Pan, S. J.; and Yang, Q. 2009. A survey on transfer learning. *TKDE*, 1345–1359.

Platanios, E. A.; Stretcu, O.; Neubig, G.; Poczos, B.; and Mitchell, T. 2019. Competence-based Curriculum Learning for Neural Machine Translation. In *NAACL*, 1162–1172.

Russell, B. C.; Torralba, A.; Murphy, K. P.; and Freeman, W. T. 2008. LabelMe: a database and web-based tool for image annotation. *IJCV*, 157–173.

Saito, K.; Kim, D.; Sclaroff, S.; Darrell, T.; and Saenko, K. 2019. Semi-supervised domain adaptation via minimax entropy. In *ICCV*, 8050–8058.

Sangineto, E.; Nabi, M.; Culibrk, D.; and Sebe, N. 2018. Self paced deep learning for weakly supervised object detection. *TPAMI*, 712–725.

Shankar, S.; Piratla, V.; Chakrabarti, S.; Chaudhuri, S.; Jyothi, P.; and Sarawagi, S. 2018. Generalizing Across Domains via Cross-Gradient Training. In *ICLR*.

Shi, Y.; Yu, X.; Sohn, K.; Chandraker, M.; and Jain, A. K. 2020. Towards universal representation learning for deep face recognition. In *CVPR*, 6817–6826.

Shrivastava, A.; Gupta, A.; and Girshick, R. 2016. Training region-based object detectors with online hard example mining. In *CVPR*, 761–769.

Soviany, P.; Ionescu, R. T.; Rota, P.; and Sebe, N. 2021. Curriculum self-paced learning for cross-domain object detection. *CVIU*, 103166.

Soviany, P.; Ionescu, R. T.; Rota, P.; and Sebe, N. 2022. Curriculum learning: A survey. *IJCV*, 1–40.

Sun, Z.; Shen, Z.; Lin, L.; Yu, Y.; Yang, Z.; Yang, S.; and Chen, W. 2022. Dynamic Domain Generalization. In *IJCAI*, 1342–1348.

Tang, Y.-P.; and Huang, S.-J. 2019. Self-paced active learning: Query the right thing at the right time. In *AAAI*, 5117–5124.

Tay, Y.; Wang, S.; Luu, A. T.; Fu, J.; Phan, M. C.; Yuan, X.; Rao, J.; Hui, S. C.; and Zhang, A. 2019. Simple and Effective Curriculum Pointer-Generator Networks for Reading Comprehension over Long Narratives. In *ACL*, 4922–4931.

Torralba, A.; and Efros, A. A. 2011. Unbiased look at dataset bias. In *CVPR*, 1521–1528.

Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 5018–5027.

Wang, J.; Lan, C.; Liu, C.; Ouyang, Y.; Qin, T.; Lu, W.; Chen, Y.; Zeng, W.; and Yu, P. 2022. Generalizing to unseen domains: A survey on domain generalization. *TKDE*.

Wang, X.; Chen, Y.; and Zhu, W. 2021. A survey on curriculum learning. *TPAMI*.

Yue, X.; Zhang, Y.; Zhao, S.; Sangiovanni-Vincentelli, A.; Keutzer, K.; and Gong, B. 2019. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *ICCV*, 2100–2110.

Zhang, X.; He, Y.; Xu, R.; Yu, H.; Shen, Z.; and Cui, P. 2023. Nico++: Towards better benchmarking for domain generalization. In *CVPR*, 16036–16047.

Zhang, Y.; Li, M.; Li, R.; Jia, K.; and Zhang, L. 2022. Exact feature distribution matching for arbitrary style transfer and domain generalization. In *CVPR*, 8035–8045.

Zhou, K.; Liu, Z.; Qiao, Y.; Xiang, T.; and Loy, C. C. 2022. Domain generalization: A survey. *TPAMI*.

Zhou, K.; Yang, Y.; Hospedales, T.; and Xiang, T. 2020. Deep domain-adversarial image generation for domain generalisation. In *AAAI*, 13025–13032.

Zhou, K.; Yang, Y.; Qiao, Y.; and Xiang, T. 2021. Domain Generalization with MixStyle. In *ICLR*.