Improving Out-of-distribution Robustness by Adversarial Training with Structured Priors

Qixun Wang
School of Electronic Engineering and Computer Science
Peking University
1900012925@pku.edu.cn

Yifei Wang
School of Mathematical Sciences
Peking University
yifei_wang@pku.edu.cn

Hong Zhu
Huawei Technologies Ltd.

Yisen Wang
School of Electronic Engineering and Computer Science
Peking University
yisenwang@pku.edu.cn

Abstract

Deep models often fail to generalize well in test domains when the data distribution differs from that in the training domain. Among numerous approaches to address this Out-of-Distribution (OOD) generalization problem, there has been a growing surge of interest in exploiting the input-robustness obtained by Adversarial Training (AT) to improve OOD performances. Recent works have revealed that the robust model obtained by conducting sample-wise AT also retains transferability to biased test domains. In this paper, we empirically show that sample-wise AT has limited improvement on OOD performance. Specifically, we find that AT can only maintain performance at smaller scales of perturbation while Universal AT (UAT) are more robust to larger-scale perturbations. This provides us with clues that the adversarial perturbations with universal (low dimensional) structures can enhance the robustness to large data distribution shifts which are common in OOD scenarios. Inspired by this, we propose two AT variants with low-rank structures to train OOD-robust models. Extensive experiments on DomainBed benchmark show that our proposed approaches outperform Empirical Risk Minimization (ERM) and sample-wise AT.

1 Introduction

Existing deep learning methods have achieved good performance on visual classification tasks under the same distribution of training sets and test sets. However, when the data distribution of the test set is different from that of the training set, the classification performance of the deep neural networks (DNNs) may decrease sharply [1]. This is mainly because the network may capture spurious features such as the background and style information to assist the fast fitting during the training process [2]. However, in real-world scenarios, test data may differ from training data in the background and style information, thus DNNs that use unstable spurious features to make predictions will fail. Solving the above problem is known as the out-of-distribution (OOD) generalization.

Another scenario where DNNs may fail is that deep models are often vulnerable to adversarial examples [3]. Adversarial training (AT) is originally proposed as an effective way to defend against

Preprint. Under review.

adversarial attacks [4]. Moreover, there is work showing that adversarial training helps to solve the OOD generalization problem because OOD data can be seen as stronger perturbations to some extent [5]. The reason why AT can defend adversarial attacks meanwhile benefits OOD generalization is that it can make the network robust to the interference of spurious features, such as randomly injected noise (in adversarial examples) or the spurious correlation between labels and background information (in OOD generalization). In other words, AT enables the network to make predictions using intrinsic features rather than spurious features.

A potential problem, however, is that existing AT methods for solving OOD generalization ignore the specific design of perturbations. They usually simply conduct sample-wise AT [6], which only brings limited performance improvement to OOD generalization. The essential reason for the failure of this type of approach is that the perturbations it uses cannot distinguish between invariant and spurious features. As a result, it improves the robustness at the expense of the decreasing standard accuracy [7]. Moreover, we empirically find that when adapting universal AT [8] to OOD problems, i.e., conducting AT with domain-wise perturbations, it will show stronger input-robustness when facing larger-scale perturbations comparing to sample-wise AT (see Section 3.2). Since the sample injected with large-scale perturbations can be regarded as OOD samples [5], we draw inspiration from this phenomenon that AT with universal (low-dimensional) structures can be the key to solving OOD generalization. To address the above problems, we propose to use structured low-rank perturbations related to domain information in AT, which can help the model to filter out background and style information, thus benefiting OOD generalization. We make the following contributions in our work:

- We identify the limitations of sample-wise AT on OOD generalization through a series of experiments. To alleviate this problem, we further propose two simple but effective AT variants with structured priors to improve OOD performances.
- We theoretically prove that our proposed structured AT approach can accelerate the convergence of reliance on spurious features to 0 when using finite-time-stopped gradient descent, thus enhancing the robustness of the model against spurious correlations.
- By conducting experiments on the DomainBed benchmark [9], we demonstrate that our algorithms outperform ERM and sample-wise AT on various OOD datasets.

2 Related Work

Solving OOD Generalization with AT. According to [3], the performance of deep models is susceptible to small-scale perturbations injected in the input images, even if these perturbations are imperceptible to humans. Adversarial training (AT) [4] is an effective approach to improve the robustness to input perturbations. However, many of the recent works have begun to focus on the connection between AT and OOD due to the fact that OOD data can be regard as one kind of large-scale perturbations. These works seek to exploit the robustness provided by AT to improve the OOD generalization. For instance, [6] applies sample-wise AT to OOD generalization. They theoretically find that if a model is robust to input perturbation on training samples, it also generalizes well on OOD data. [5] theoretically establishes a link between the objective of AT and the OOD robustness. They reveal that the AT procedure can be regarded as a heuristic solution to the worst-case problem around the training domain distribution. Nevertheless, the discussion of [6] and [5] is restricted to the framework of using Wasserstein distance to measure the distribution shift, which is less practical for the real-world OOD setting where domain shifts are diverse. Additionally, they only study the case of sample-wise AT and did not further investigate the effect of different forms of AT (not sample-wise) on the robustness performance of OOD. Other works such as [10] focus on the structure design of the perturbations. In [10] they use multi-scale perturbations within one picture. But they do not exploit the universal information within one training domain. In our work, we focus on real-world OOD scenarios where there are additional clues lying in the distribution shifts, i.e, the low-rank structures in the spurious features (such as background and style information) across one domain. We further design a low-rank structure in the perturbations to specifically eliminate such low-rank spurious correlations.

OOD Evaluation Benchmark. The DomainBed benchmark [9] provides a fair way of evaluating different state-of-the-art OOD methods, which has been widely accepted by the community. By conducting rigorous experiments in a consistent setting, [9] reveals that many algorithms that claim to outperform previous methods cannot even outperform ERM. Unlike previous works using AT to

Table 1: Test accuracy (%) on four OOD datasets on DomainBed benchmark with a fixed set of hyperparameters. The improvement of AT is marginal.

	Datasets				
Algorithm	PACS	OfficeHome	VLCS	NICO	avg
ERM	79.7 ± 0.0	59.6 ± 0.0	74.4 ± 1.0	$\mathbf{70.7 \pm 1.0}$	71.1
AT	$\mathbf{81.5 \pm 0.4}$	$\mathbf{59.9 \pm 0.4}$	$\mathbf{75.3 \pm 0.7}$	68.2 ± 2.2	**71.2**

address OOD generalization, such as [6] and [5], in this work we adopt the Domainbed benchmark for a fair comparison of our approach with existing state-of-the-art methods.

3 Weakness of Sample-wise AT for OOD Generalization

3.1 Preliminaries

Out-of-distribution (OOD) Generalization. Let $x \in \mathcal{X}$ denote the random data in the input space $\mathcal{X}$ and let $y \in \mathcal{Y}$ denote the target random data in label space $\mathcal{Y}$. $\phi : \mathcal{X} \to \mathcal{Z}$ denotes the feature extractor. We have the predictor $f = w \circ \phi(x)$ where $z = \phi(x)$ is the feature variable in the feature space $\mathcal{Z}$. $w : \mathcal{Z} \to \mathcal{Y}$ denotes the classifier.

Now we give the formal definition of the OOD generalization problem. We have a set of m training domains $\mathcal{E} = \{E_1, E_2, ..., E_m\}$, where each domain E_e is characterized by a input dataset $E_e := \{(x_i^e, y_i^e)\}_{i=1}^{n_e}$ containing n_e i.i.d input samples drawn from the distribution of $\mathcal{P}_e$, and a test domain E_{m+1} with data following the distribution of $\mathcal{P}_{te}$, where $\mathcal{P}_{te} \neq \mathcal{P}_i,\ i = 1, 2, ..., m$. $\mathcal{L} : \mathcal{X} \to \mathbb{R}^+$ denotes the loss function. The ultimate goal of OOD generalization is to find an optimal predictor f that minimizes the risk on the unseen test domain:

$$\min_f \mathbb{E}_{(x,y)\sim\mathcal{P}_{te}(x,y)}[\mathcal{L}(f(x), y)]. \tag{1}$$

Adversarial Training (AT)[1]. According to [4], AT can be expressed as the following optimization problem:

$$\min_f \mathbb{E}_{(x,y)\sim\mathcal{P}(x,y)}[\max_{\delta\in\mathcal{S}} \mathcal{L}(f(x+\delta), y)] \text{ s.t. } \|\delta\|_p \leq \epsilon, \tag{2}$$

where $\delta \in \mathcal{S}$ is the random injected perturbation with l_p norm bounded by $\epsilon > 0$. The inner maximization problem can be optimized by fast gradient sign method (FGSM [11]), a simple one-step scheme:

$$x = x + \epsilon \text{sgn}(\nabla_x \mathcal{L}(f(x), y)), \tag{3}$$

where sgn$(\cdot)$ is the sign function, or by projected gradient descent (PGD [4]), a more powerful multi-step variant:

$$x^{t+1} = \prod_{\mathcal{S}}(x^t + \gamma \text{sgn}(\nabla_x \mathcal{L}(f(x), y))), \tag{4}$$

where $\prod_{\mathcal{S}}$ is the projection operator onto the set $\mathcal{S}$, γ is the step size and t denotes the iteration.

3.2 Weakness of AT for OOD Generalization

We now highlight some weaknesses of sample-wise AT for OOD generalization based on series of empirical evidence. We first conduct a toy experiment on the DomainBed benchmark [9] to evaluate the OOD performance of AT. We run ERM and AT on four OOD datasets: PACS [12], OfficeHome [13], VLCS [14], and NICO [15] with a fixed set of hyperparamters (Detailed experimental setting can be found in Appendix B.1). The results are shown in Table 1. We can see that the improvement of OOD performance by AT is limited with an average improvement of only 0.1%.

We further investigate the reason behind the limitations of performance improvements on OOD datasets of AT. Although previous works have revealed that the robust features obtained by AT can

[1]For simplicity, we denote 'AT' for sample-wise AT by default in the rest of the paper.

improve OOD generalization ([6] [5] [16]), we find that sample-wise AT only tolerates small-scale perturbations. Thus, we design an experiment on NICO dataset with multiple scales of perturbations. The scale is calculated with l_2 norm of the perturbation matrix. The experiment details are shown in Appendix B.1.

We show that AT will suffer severe performance degradation when using large perturbations (shown in Figure 1). This provides clues to understanding the failure of AT in OOD scenarios. The distribution shifts in OOD data usually have much larger scales than the invisible perturbations commonly used in AT [5]. Hence, AT methods designed for small perturbations cannot handle these large-scale domain shifts that often appear in OOD data. However, our experiment shows that we can alleviate this problem by adapting universal AT (UAT, [8]) to the OOD setting, i.e., using a perturbation for each domain.

Figure 1 shows that UAT will keep its generalization performance when the perturbation scale is large. There are two empirical explanations for this: first, the background and style information usually have a low-rank structure, such as the grassland and snowfield that have recurring parts. Second, similar spurious features often appear within one specific domain, such as PACS [12] and VLCS [14] datasets. As stated in [8], the universal perturbation lies in a low dimensional space. Hence using universal (domain-wise) perturbations will help to resist such low-rank shifts and improve the robustness of the model.

Inspired by this, we proposed two new AT algorithms with more sophisticated low-rank structures on different dimensions to improve OOD generalization in the next section.

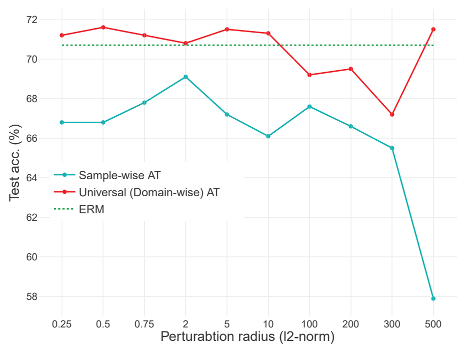

Figure 1: Test accuracy of AT, UAT (l_2 norm) and ERM on NICO.

4 The Proposed Structured AT Method

In order to construct low-rank structured perturbations, we start by analyzing the structure of sample-wise perturbations. Assume that each input data x has a shape of $N \times N \times C$. N is the size of the input image and C is the number of the channels. For simplicity, we assume $C = 3$. We reparameterize the sample-wise perturbations as a series of 2-D matrices $\{D_1^1, D_2^1, ..., D_m^1\}$, $\{D_1^2, D_2^2, ..., D_m^2\}$, $\{D_1^3, D_2^3, ..., D_m^3\}$ where $D_e^c \in R^{n_e \times N^2}$ denotes the perturbations in the e-th domain for the input channel c, n_e is the number of the samples in domain E_e, and m is the number of domains. The i-th row of D_e^c represents the c-th channel of the i-th sample in domain E_e (see the first column in Figure 2 for illustration). By such reparameterization, it is natural to find that there are two orientations to reduce the rank of the perturbations:

1. **Along the dimension of the number of samples** (along the red arrow in the upper left corner of Figure 2). This corresponds to reducing the number of the perturbations used within one domain.
2. **Along the dimension of the input scale** (along the blue arrow in the upper left corner of Figure 2). This corresponds to reducing the rank of the perturbation used for a specific input sample.

In the following sections, we will propose two AT variants with structured priors that reduce the rank in these two directions.

4.1 MAT: Adversarial Training with Combinations of Multiple Perturbations

In this section, we propose domain-wise Multiple-perturbation Adversarial Training (MAT). It aims to conduct rank minimization along the dimension of the number of samples. Instead of using sample-wise perturbations, MAT constructs a combination of multiple perturbations and shares this mixed perturbation within a domain. Specifically, we choose to train the linear combination of k

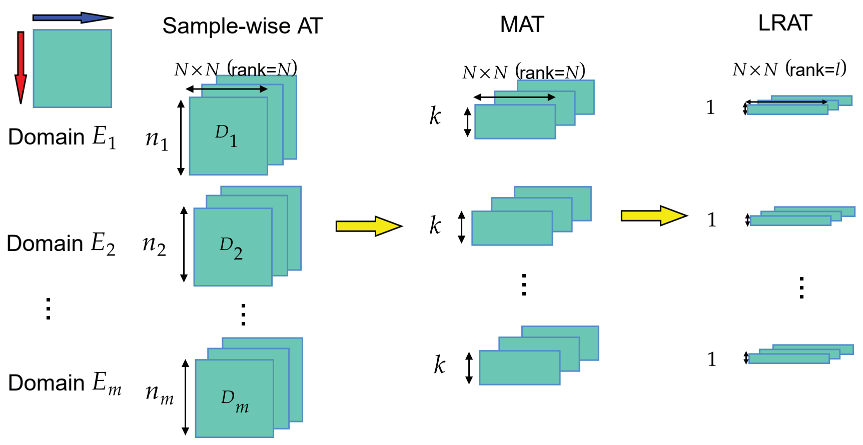

Figure 2: Illustration of how our proposed structured AT reduces the rank of the perturbations comparing to sample-wise AT. The left column shows how we reparameterize sample-wise AT. The perturbations are segmented by domains. A block represents the n_e perturbations injected in a channel of the samples in domain E_e. This figure shows the case where the input image has three channels (RGB). The red and the blue arrow in the upper left shows the two orientations to reduce the rank of the perturbations, i.e, along the dimension of the total number of the perturbations and along the dimension of the rank of a single perturbation, respectively. The mid column illustrates that MAT reduces the number of the perturbations used for domain E_e from n_e to k ($k \ll n_e$). The right column shows that LDAT further reduces the number of the perturbations from k to 1. Moreover, it reduces the rank of a specific perturbation from N to l ($l \ll N$).

perturbations for each domain E_e to conduct AT. Here k is a hyperparameter and k is far less than the number of samples in domain E_e. The optimization problem can be reformulated as:

$$\min_f \sum_e \mathbb{E}_{(x,y)\sim\mathcal{P}_e(x,y)}[\mathcal{L}(f(x+\delta^e),y)], \tag{5}$$

$$\text{s.t. } \delta^e = \sum_{i=1}^{k} \alpha_i^{e*}\delta_i^{e*},\ \|\delta_i^{e*}\|_p \le \epsilon,\ \sum_{i=1}^{k}\alpha_i^{e*} = 1,\ \alpha_i^{e*} \ge 0 \text{ for } i = 1,2,...,k, \tag{6}$$

where

$$\alpha_i^{e*},\ \delta_i^{e*} = \underset{\alpha_i^e,\ \delta_i^e}{\text{argmax}} \mathbb{E}_{(x,y)\sim\mathcal{P}_e(x,y)}[\mathcal{L}(f(x+\sum_{i=1}^{k}\alpha_i^e\delta_i^e),y)]. \tag{7}$$

Here e denotes the subscript of a training domain and α_i^e is the weight that can be learned for each perturbation δ_i^e. Detailed training procedure of MAT is in Appendix C.

MAT works as a low-rank version of sample-wise AT. In sample-wise AT, we maintain n_e perturbations for each domain E_e, where n_e is the number of training samples in domain E_e. As for MAT, it reduces the number of perturbations available to samples from n_e to k and to obtain low-rank structures (see the second column in Figure 2 for illustration). Therefore, it fulfills rank reduction along the sample-number dimension.

4.2 LDAT: Adversarial Training with Low-rank Decomposed Perturbations

Based on MAT, we further propose Adversarial Training with Low-rank Decomposed perturbations (LDAT). Analogous to MAT, LDAT still shares one perturbation in a specific domain. Moreover, LDAT imposes a low-rank constraint on the perturbation itself, which corresponds to the dimension of the input scale. Technically, we obtain the domain-wise low-rank perturbation matrix $\delta \in \mathcal{R}^{N\times N\times C}$ by multiplying two matrices: $\delta = AB$. Here $A \in \mathcal{R}^{N\times l\times C}$ and $B \in \mathcal{R}^{l\times N\times C}$, l is a hyperparameter and $l \ll N$. Since rank$(AB) \le$ rank(A) and rank$(AB) \le$ rank(B) hold for arbitrary matrices A, B, we have rank$(\delta) \le l$. Therefore LDAT reduces the rank of the perturbation from a large value N

to a relatively small value l (see the third column in Figure 2 for illustration). The formal definition of the LDAT objective is:

$$\min_{f} \sum_{e} \mathbb{E}_{(x,y)\sim\mathcal{P}_e(x,y)}[\mathcal{L}(f(x+\delta^e), y)], \text{ s.t. } \delta^e = A^{e*}B^{e*}, \|\delta^e\|_p \leq \epsilon, \tag{8}$$

where

$$A^{e*},\ B^{e*} = \underset{A^e, B^e}{\text{argmax}} \mathbb{E}_{(x,y)\sim\mathcal{P}_e(x,y)}[\mathcal{L}(f(x+A^eB^e), y)],\ A^e \in \mathcal{R}^{N\times l\times C},\ B^e \in \mathcal{R}^{l\times N\times C}. \tag{9}$$

Detailed training procedure of LDAT is in Appendix C. Comparison to MAT, LDAT reduces the number of the perturbations available to the samples in a domain from k to 1. In addition, it reduces the rank of the perturbation for a single channel of a sample from N to l (see the third column in Figure 2).

4.3 Theoretical Analysis

In this part, we theoretically explain why the domain-wise perturbation proposed in MAT and LDAT can help to improve the robustness of the model against spurious correlations following [2] and [17]. In general, we prove that MAT and LDAT can prevent the model from relying more on the spurious features to make predictions as the spurious correlations in the training data increase. Consequently, the model trained with MAT or LDAT will generalize better on OOD data.

Notations. Let $x \in \mathcal{X}$ denote the random data in the input space $\mathcal{X}$ and let $y \in \mathcal{Y}$ denote the target random data in label space $\mathcal{Y}$. For simplicity, let $\mathcal{Y} \in \{1, -1\}$ in this section. Let $\mathbb{D}$ denote an underlying class of distributions over $\mathcal{X} \times \mathcal{Y}$. Let x_{inv} and x_{sp} denote the invariant features and the spurious features respectively. Also for simplicity, assume that there exists an identity mapping $\Phi : \mathcal{X}_{inv} \times \mathcal{X}_{sp} \to \mathcal{X}$ such that each $\mathcal{D} \in \mathbb{D}$ is induced by a distribution over $\mathcal{X}_{inv} \times \mathcal{X}_{sp}$ (so x can be denoted as $x = (x_{inv}, x_{sp})$). Let x_{sp} take values in $\{+\beta, -\beta\}$ for some $\beta > 0$.

A Simple OOD Task. Consider a simple OOD task where we have two training domains representing the grass and desert background respectively. Both domains have two classes: the cow class and the camel class. In the grass/desert domain, the cow/camel class predominates. During test-time, the correlation between the labels and the background flips. We can abstract this cow-camel dataset into the following model: a training dataset $\mathcal{S}$ with four groups of data points drawn from the four quadrants of the feature space $\{-1, +1\} \times \{-\beta, +\beta\}$ respectively. We set the invariant features $x_{inv} = y$ and the spurious features x_{sp} to be $y\beta$ with probability $p \in [0.5, 1)$ and $-y\beta$ with probability $1-p$. For example, the data points of the cow class in the grass and the desert domain can be expressed as $x = (1, \beta)$ and $x = (1, -\beta)$ respectively. Note that p measures the intensity of spurious correlations in a certain environment. Specifically, in the desert domain, the ratio of the camel samples is p and as for the grass domain, the ratio of the cow samples is p. It is easy to know that when $p = 0.5$, there are no spurious correlations between the labels and the background in this dataset.

Consider a linear classifier $h(x) = w_{inv}x_{inv} + w_{sp}x_{sp}$. Following [17], let us consider MAT/LDAT trained with gradient descent algorithm stopped in finite time t. In order to characterize the dependence of the model on spurious features during the training process, we investigate the convergence rate of $\frac{w_{sp}\beta}{|w_{inv}(t)x_{inv}|}$ to 0 on the above toy dataset. The value $\frac{w_{sp}\beta}{|w_{inv}(t)x_{inv}|}$ denotes the ratio between the output of the spurious component to that of the invariant component. We will prove that after adding the domain-wise perturbations in finite-time-stopped gradient descent, the lower bound of the convergence rate of this ratio does not increase monotonically with p. Hence, the model will not learn a large prediction weight based on spurious features even if the spurious correlation is strong (p is large).

In the following theorem, we denote the domain-wise perturbation in MAT/LDAT as δ. Theorem 4.1 applies to both MAT and LDAT since they both use domain-wise perturbations. See Appendix A for a formal statement and full proof of Theorem 4.1.

Theorem 4.1. ***(informal)*** *Let $\mathcal{H}$ be the set of linear classifiers $h(x) = w_{inv}(t)x_{inv} + w_{sp}(t)x_{sp}$. Consider the above 2-D OOD dataset $\mathcal{S}$. Assume that the empirical distribution of x_{inv} given $x_{sp} \cdot y > 0$ is identical to the empirical distribution of x_{inv} given $x_{sp} \cdot y < 0$. δ is the optimal perturbation obtained by optimizing the object in Eq.* (7) *or Eq.* (9). *Let $w_{inv}(t)x_{inv} + w_{sp}(t)x_{sp}$*

be initialized to the origin, and trained with MAT/LDAT to minimize the exponential loss on $\mathcal{S}$. Then, for any $(x, y) \in \mathcal{S}$, we have:

$$\Omega(\frac{\frac{1}{\beta-\delta y}\ln[\frac{c_1+p}{c_2+p^{\frac{1}{2}+\epsilon}(1-p)^{\frac{1}{2}-\epsilon}}]}{M\ln(t+1)}) \leq \frac{w_{sp}(t)\beta}{|w_{inv}(t)x_{inv}|}, \tag{10}$$

where $\epsilon := \frac{\delta y}{2\beta}$ is a real number close to 0, $c := \frac{2(2M-1)}{\beta^2}$, $c_1 := \frac{2(2M-1)}{\beta(\beta-\delta y)^2 e^{w_{inv}\delta y}}$, $c_2 := \frac{2(2M-1)}{\beta(\beta^2-\delta^2)e^{w_{inv}\delta y}(\frac{\beta-\delta y}{\beta+\delta y})^{\frac{\beta+\delta y}{2\beta}}}$. $M = \max_{x\in S} \hat{w}\cdot x$ denotes the maximum value of the margin of the max-margin classifier $\hat{w}$ on $\mathcal{S}$. $\Omega(\cdot)$ is the lower bound of a given function within a constant factor. Therefore, the lower bound of the convergence rate does not increase monotonically with p under the condition that $c_2 + (\frac{c_1}{1-c_1})^{\frac{1}{2}+\epsilon}(c_1 - 2\epsilon) < 0$.

To sum up, since we can prevent this lower bound from growing monotonically with p, we accelerate the convergence rate of $\frac{w_{\text{sp}}(t)\beta}{|w_{\text{inv}}(t)x_{\text{inv}}|}$ to 0 when there is stronger spurious correlation (larger p). Recall that the ratio $\frac{w_{\text{sp}}(t)\beta}{|w_{\text{inv}}(t)x_{\text{inv}}|}$ reflects the degree of reliance on spurious features. Therefore, a faster convergence of this ratio to 0 (smaller lower bound) means that the model will end up relying less on the spurious correlations within finite training time. In other words, the OOD robustness can be enhanced by using domain-wise perturbations.

Remark. Here, we demonstrate that MAT and LDAT show stronger OOD robustness comparing to ERM. We compare the result in Theorem 4.1 to that in the Theorem 2 in [2]. According to the Theorem 2 in [2], even if the max-margin classifier does not rely on the x_{sp} for any level of spurious correlation $p \in [0.5, 1)$, ERM trained by gradient descent stopped in finite time still fails to avoid using spurious features. Moreover, when conducting ERM with finite-time-stopped gradient descent, the lower bound of the convergence rate of $\frac{w_{sp}\beta}{|w_{inv}x_{inv}|}$ to 0 is

$$\Omega(\frac{\ln\frac{c+p}{c+\sqrt{p(1-p)}}}{M\ln t}) \leq \frac{w_{sp}(t)\beta}{|w_{inv}(t)x_{inv}|}, \tag{11}$$

where M and c follow the definition in Theorem 4.1. This lower bound grows monotonically with p, thus ERM will have slower convergence for larger spurious correlations. However, with domain-wise perturbations, we can modify the lower bound so that it does not increase monotonically with the spurious correlation p.

5 Experiments

5.1 Experimental Setup

We conduct experiments on the DomainBed benchmark [9], a testbed for OOD generalization that implements consistent experimental protocols across various approaches to ensure fair comparisons. We evaluate on PACS [12], OfficeHome [13], VLCS [14], NICO [15], and Colored MNIST [1]. There are several changes in our experimentation setting comparing to DomainBed:

1. **Backbone Network.** We use ResNet-18 [18] as our backbone network for datasets excluding Colored MNIST instead of ResNet-50 used in [9] for efficiency.
2. **Hyperparameter Search Space.** We use a smaller hyperparameter search space than [9]. We conduct a random search of 8 trials for PACS, OfficeHome, and VLCS while 6 trials for NICO and Colored MNIST in the hyperparameter search space, instead of 20 trials adopted in [9] for feasibility. See Appendix B.2 for more details.

Model Selection Strategy. Since hyperparameter choice has a significant impact on the OOD performance, it is critical to use appropriate model selection method. For PACS, OfficeHome, and VLCS datasets, we use training-domain validation proposed in [9] since it is more in line with the OOD scenario. For NICO, we adopt OOD validation following [19]. For Colored MNIST, we use test-domain validation [9] since it can enlarge the gaps in OOD performance among the algorithms while the gap induced by training-domain validation on Colored MNIST is marginal.

Table 2: Test accuracy (%) on OOD datasets within DomainBed benchmark using ResNet-18. Here "avg^1" denotes the average accuracy on PACS, OfficeHome, NICO, Colored MNIST datasets and “avg^2” denotes the average accuracy on all five datasets. The best results are in **bold**.

	Datasets						
Algorithm	PACS	OfficeHome	VLCS	NICO	Colored MNIST	avg^1	avg^2
ERM (Our runs)	81.7 ± 0.3	62.1 ± 0.1	74.4 ± 1.0	73.2 ± 1.9	28.1 ± 1.5	61.3	63.9
AT (Our runs)	82.6 ± 0.4	62.1 ± 0.3	**76.2 ± 0.3**	69.7 ± 1.6	29.1 ± 1.5	60.9	64.3
ERM[19]	81.5 ± 0.0	63.3 ± 0.2	-	71.4 ± 1.3	29.9 ± 0.1	61.5	-
RSC[22]	**82.8** ± 0.4	62.9 ± 0.4	-	69.7 ± 0.3	28.6 ± 1.5	61.0	-
MMD[23]	81.7 ± 0.2	63.8 ± 0.1	-	68.3 ± 1.8	50.7 ± 0.1	66.1	-
SagNet[24]	81.6 ± 0.4	62.7 ± 0.4	-	69.3 ± 1.0	30.5 ± 0.7	61.0	-
CORAL[25]	81.6 ± 0.6	63.8 ± 0.3	-	68.3 ± 1.4	30.0 ± 0.5	61.0	-
IRM[1]	81.1 ± 0.3	63.0 ± 0.2	-	67.6 ± 1.4	60.2 ± 2.4	68.0	-
VREx[21]	81.8 ± 0.1	63.5 ± 0.1	-	71.0 ± 1.3	56.3 ± 1.9	68.2	-
GroupDRO[26]	80.4 ± 0.3	63.2 ± 0.2	-	71.8 ± 0.8	32.5 ± 0.2	62.0	-
DANN[27]	81.1 ± 0.4	62.9 ± 0.6	-	68.6 ± 1.1	24.5 ± 0.8	59.3	-
MTL[28]	81.2 ± 0.4	62.9 ± 0.2	-	70.2 ± 0.6	29.3 ± 0.1	60.9	-
Mixup[29]	79.8 ± 0.6	63.3 ± 0.5	-	66.6 ± 0.9	27.6 ± 1.8	59.3	-
ANDMask[30]	79.5 ± 0.0	62.0 ± 0.3	-	72.2 ± 1.2	27.2 ± 1.4	60.2	-
MLDG[31]	73.0 ± 0.4	52.4 ± 0.2	-	51.6 ± 6.1	32.7 ± 1.1	52.4	-
MAT (Our work)	82.3 ± 0.5	**64.5 ± 2.1**	74.6 ± 0.8	74.2 ± 1.5	**65.4 ± 8.1**	**71.6**	**72.2**
LDAT (Our work)	82.6 ± 0.5	61.0 ± 0.9	75.3 ± 0.3	**74.4 ± 1.6**	52.5 ± 5.4	67.6	69.1

Hyperparameters for MAT and LDAT. To retain low-rank structures in perturbations, we set the upper bound of the search space of the perturbation number k in MAT to be 20. Similarly, the upper bound of the rank of the perturbation used in LDAT l is 20. The complete setup of the hyperparameters for MAT and LDAT is provided in Appendix B.2.

5.2 OOD Performance on Benchmark datasets.

Table 2 summarizes the results on the five OOD datasets. The results of other approaches for PACS, OfficeHome, NICO, and Colored MNIST datasets are adopted from [19]. The results on VLCS of other algorithms are missing (denoted as "-") because [19] does not experiment on this dataset.

Comparison with ERM and Sample-wise AT. From Table 2, we observe that both MAT and LDAT outperform ERM (on both our runs and the results in [19]) and AT on average. In particular, MAT achieves consistently better results than ERM on all five datasets. Additionally, the average performance of AT is worse than ERM, which is consistent with our observations in Section 3.2.

Comparison with Existing State-of-the-Art Approaches. Although the results from [19] use a different training protocol from ours: they use a larger search space and 20 random search for the hyperparameter combinations, the comparison between ERM ([19]) and ERM (our runs) indicates that their corresponding performances are close. A similar comparison has been made in [20]. We find that MAT outperforms all previous algorithms and LDAT ranked fourth among all methods, merely after VREx [21] and IRM [1].

Comparison between MAT and LDAT. From Table 2 we can see that MAT outperforms LDAT on average. Since LDAT reduces the number of the perturbations used in a domain from k to 1 (shown in Figure 2), LDAT can be regarded as a low-rank version of MAT. This indicates that the oversimplified perturbations may be less effective than the ones maintaining some flexibility. On the other hand, it is usually hard to verify which factors of an algorithm play a key role in its performance due to the complexity of the factors it contains. Since both MAT and LDAT outperform most existing State-of-the-Art methods and they both exploit low-rank structures, these two methods mutually corroborate the effectiveness of low-rank structure for OOD generalization.

5.3 Empirical Understanding

Visualization. To empirically show that MAT and LDAT can reduce the reliance on spurious features, we visualize the pixel attention heatmap of ERM, AT, MAT, and LDAT on NICO dataset

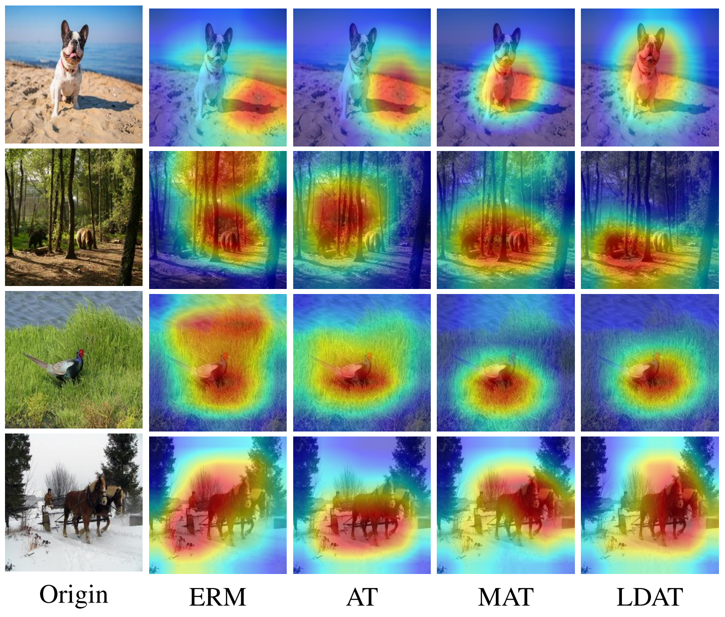

Figure 3: The pixel attention heatmap of ERM, AT, MAT and LDAT on NICO dataset. The redder part indicates that the model relies more on this part to make predictions.

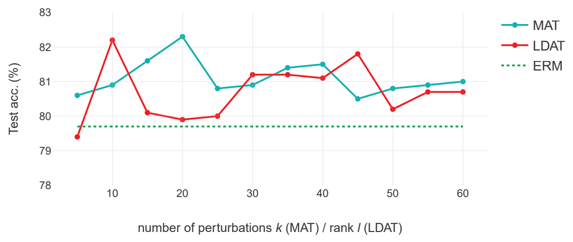

Figure 4: The performance at different values of the number of perturbations to combined within one domain k (in MAT) and the rank of the perturbation in a domain l (in LDAT).

using GradCam [32]. The heatmap is calculated with the second-order gradients of the last layer of the network w.r.t the output for the corresponding class. It reflects the contribution of different components of the feature map to the prediction results. We pick the model with the best performance for each method. The results in Figure 3 indicate that the model trained by MAT and LDAT focuses more on the object itself, while ERM and AT adopt the background information that spuriously correlates to the class to make predictions.

Parameter Analysis. The number of the perturbations used in a domain k in MAT and the rank of the perturbation l in LDAT are two key hyperparameters. We conduct further experiments to analyze the impact on the performances of these two parameters. We adopt a fixed set of parameters except for k and l (see Appendix B.1), and evaluate on PACS dataset. The results in Figure 4 show that MAT and LDAT are able to keep their performances over ERM as long as k and l are far less than the number of the samples N. Additionally, we can observe from the trend that when k and l are too small $(= 5)$, the performances degenerate. This implies the oversimplified structures of the perturbations can be less effective for generalization. Additional analysis on the impact of the learning rate for the perturbations is in Appendix B.2.

6 Conclusion

In this work, we empirically reveal the limitations of AT on OOD tasks. Due to the lack of constraints on the perturbation, AT fails to generalize well when facing large-scale perturbations which is close to the real-world OOD scenarios. Based on the observation that UAT shows a stronger robustness to large perturbations, we further proposed MAT and LDAT to exploit the low-rank structure to alleviate this issue. We validate the effectiveness of the proposed method on OOD tasks both theoretically and empirically.

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. In *ICML*, 2020.

[2] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. *arXiv preprint arXiv:2010.15775*, 2020.

[3] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[4] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[5] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *arXiv preprint arXiv:1805.12018*, 2018.

[6] Mingyang Yi, Lu Hou, Jiacheng Sun, Lifeng Shang, Xin Jiang, Qun Liu, and Zhiming Ma. Improved ood generalization via adversarial training and pretraing. In *ICML*, 2021.

[7] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *NeurIPS*, 2018.

[8] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *CVPR*, 2017.

[9] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[10] Charles Herrmann, Kyle Sargent, Lu Jiang, Ramin Zabih, Huiwen Chang, Ce Liu, Dilip Krishnan, and Deqing Sun. Pyramid adversarial training improves vit performance. *arXiv preprint arXiv:2111.15121*, 2021.

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *Computer Science*, 2014.

[12] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

[13] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017.

[14] Chen Fang, Ye Xu, and Daniel N Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *ICCV*, 2013.

[15] Yue He, Zheyan Shen, and Peng Cui. Towards non-iid image classification: A dataset and baselines. *Pattern Recognition*, 110:107383, 2021.

[16] Klim Kireev, Maksym Andriushchenko, and Nicolas Flammarion. On the effectiveness of adversarial training against common corruptions. *arXiv preprint arXiv:2103.02325*, 2021.

[17] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 2018.

[18] Shaoqing Ren, Jian Sun, K He, and X Zhang. Deep residual learning for image recognition. In *CVPR*, 2016.

[19] Nanyang Ye, Kaican Li, Lanqing Hong, Haoyue Bai, Yiting Chen, Fengwei Zhou, and Zhenguo Li. Ood-bench: Benchmarking and understanding out-of-distribution generalization datasets and algorithms. *arXiv preprint arXiv:2106.03721*, 2021.

[20] Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *arXiv preprint arXiv:2110.10832*, 2021.

[21] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *ICML*, pages 5815–5826. PMLR, 2021.

[22] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *ECCV*. Springer, 2020.

[23] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *CVPR*, pages 5400–5409, 2018.

[24] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap via style-agnostic networks. *arXiv preprint arXiv:1910.11645*, 2(7):8, 2019.

[25] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450. Springer, 2016.

[26] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *ICLR*, 2019.

[27] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[28] Gilles Blanchard, Aniket Anand Deshmukh, Urun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *arXiv preprint arXiv:1711.07910*, 2017.

[29] Shen Yan, Huan Song, Nanxiang Li, Lincan Zou, and Liu Ren. Improve unsupervised domain adaptation with mixup training. *arXiv preprint arXiv:2001.00677*, 2020.

[30] Giambattista Parascandolo, Alexander Neitz, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf. Learning explanations that are hard to vary. *arXiv preprint arXiv:2009.00329*, 2020.

[31] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[32] Jacob Gildenblat and contributors. Pytorch library for cam methods. `https://github.com/jacobgil/pytorch-grad-cam`, 2021.

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section .
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.2. We mention that the oversimplified perturbation (LDAT) will be less effective than the ones with more flexibility (MAT).
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Full assumptions are in Appendix A.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] It will release upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section B.1 and B.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 2 for an example.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use DomainBed benchmark and GradCam and cite their creators.
   (b) Did you mention the license of the assets? [No] All assets we use are open source.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] All datasets we use are open source.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [No]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [No]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No]

A Proof of Theorem 4.1

Theorem 4.1 **(formal)** Let $\mathcal{H}$ be the set of linear classifiers $h(x) = w_{inv}(t)x_{inv} + w_{sp}(t)x_{sp}$. Consider any task that satisfies all the constraints in Section 3.1. in [2]. Consider a dataset $\mathcal{S}$ drawn from $\mathcal{D}$ such that the empirical distribution of x_{inv} given $x_{sp} \cdot y > 0$ is identical to the empirical distribution of x_{inv} given $x_{sp} \cdot y < 0$. δ is the optimal perturbation obtained by optimizing object (7) or (9). δ can be seen as a random variable.

Let $w_{inv}(t)x_{inv} + w_{sp}(t)x_{sp}$ be initialized to the origin, and trained with MAT/LDAT with an infinitesimal learning rate to minimize the exponential loss on $\mathcal{S}$. Then, for any $(x, y) \in \mathcal{S}$, we have:

$$\Omega(\frac{\frac{1}{\beta-\delta y}\ln[\frac{c_1+p}{c_2+p^{\frac{1}{2}+\epsilon}(1-p)^{\frac{1}{2}-\epsilon}}]}{M\ln(t+1)}) \leq \frac{w_{sp}(t)\beta}{|w_{inv}(t)x_{inv}|}$$

where $\epsilon := \frac{\delta y}{2\beta}$ is a real number close to 0, $c_1 := \frac{2(2M-1)}{\beta(\beta-\delta y)^2 e^{w_{inv}\delta y}}$, $c_2 := \frac{2(2M-1)}{\beta(\beta^2-\delta^2)e^{w_{inv}\delta y}(\frac{\beta-\delta y}{\beta+\delta y})^{\frac{\beta+\delta y}{2\beta}}}$. $M = \underset{x\in S}{max}\ \hat{w}\cdot x$ denotes the maximum value of the margin of the margin of the max-margin classifier $\hat{w}$ on $\mathcal{S}$. $c := \frac{2(2M-1)}{\beta^2}$. Therefore, the lower bound of the convergence rate does not increase monotonically with p under the condition that $c_2 + (\frac{c_1}{1-c_1})^{\frac{1}{2}+\epsilon}(c_1 - 2\epsilon) < 0$.

Proof. For brevity, we use w_c and w_s to represent w_{inv} and w_{sp} respectively. Also, we use x_c to represent x_{inv}. We use x_e to represent x_{sp} since the spurious feature is correlated to the environment E_e. Let $\mathcal{S}_{min}$ and $\mathcal{S}_{maj}$ denote the subset of datapoints in $\mathcal{S}$ where $x_e \cdot y < 0$ and $x_e \cdot y > 0$ respectively. Let $\mathcal{D}_{inv}$ denote the uniform distribution over x_c induced by drawing x uniformly from $\mathcal{S}_{min}$. By the assumption of the theorem, this distribution would be the same if x was drawn uniformly from $\mathcal{S}_{maj}$. Then, the loss function that is being minimized in this setting corresponds to:

$$\begin{aligned}\mathcal{L}[f(x), y] &= E_{x_c\sim\mathcal{D}_{inv}}[e^{-f(x)y}] \\ &= E_{x_c\sim\mathcal{D}_{inv}}[e^{-[w_c x_c + w_s x_e + (w_c+w_s)\delta]y}] \\ &= E_{x_c\sim\mathcal{D}_{inv}}[pe^{-[w_c x_c + w_s\beta y + (w_c+w_s)\delta]y} + (1-p)e^{-[w_c x_c - w_s\beta y + (w_c+w_s)\delta]y}] \\ &= E_{x_c\sim\mathcal{D}_{inv}}(e^{-w_c + w_c\delta y})e^{\delta y w_s}[pe^{-\beta w_s} + (1-p)e^{\beta w_s}]\end{aligned}$$

The update on w_s can be written as:

$$\begin{aligned}\Delta w_s &= -\frac{\partial\mathcal{L}[f(x), y]}{\partial w_s} \\ &= E_{x_c\sim\mathcal{D}_{inv}}(e^{-w_c+w_c\delta y})[p(\beta-\delta y)e^{(\delta y-\beta)w_s} - (1-p)(\beta+\delta y)e^{(\delta y+\beta)w_s}]\end{aligned}$$

Proof of bounds on $w_c(t)x_c$. Using the result of [17] and [2], we get

$$|w_c(t)x_c| \in [0.5\ln(1+t), 2M\ln(1+t)]$$

for a sufficiently large t and for all $x \in \mathcal{S}$.

Proof of the upper bound on w_s. To calculate the lower bound of w_s, we prove the upper bound as auxiliary first. Note that Δw_s decreases monotonically with w_s. Let $\Delta w_s = 0$, we get

$$w_s = \frac{1}{2\beta}\ln[\frac{p}{1-p}(\frac{\beta-\delta y}{\beta+\delta y})] =: w_0.$$

When $w_s < w_0$, $\Delta w_s > 0$ and when $w_s > w_0$, $\Delta w_s < 0$. As a result, for any system that is initialized at 0, w_s can never cross the point w_0. Thus, we get the upper bound of $w_s(t)$:

$$w_s(t) \leq w_0 = \frac{1}{2\beta}\ln[\frac{p}{1-p}(\frac{\beta-\delta y}{\beta+\delta y})].$$

Proof of the lower bound on w_s. We lower bound w_s via the upper bound on w_s as:

$$\begin{aligned}\Delta w_s &\geq E_{x_c\sim\mathcal{D}_{inv}}(e^{-w_c+w_c\delta y})[p(\beta-\delta y)e^{(\delta y-\beta)w_s} - (1-p)(\beta+\delta y)e^{(\delta y+\beta)w_0}] \\ &= E_{x_c\sim\mathcal{D}_{inv}}(e^{-w_c+w_c\delta y})[p(\beta-\delta y)e^{(\delta y-\beta)w_s} - (\beta+\delta y)[p(\frac{\beta-\delta y}{\beta+\delta y})]^{\frac{\beta+\delta y}{2\beta}}(1-p)^{1-\frac{\beta+\delta y}{2\beta}}]\end{aligned}$$

.

Next, using the upper bound on $w_c(t)x_c$, we get:

$$\begin{aligned}\Delta w_s &\geq E_{x_c\sim\mathcal{D}_{inv}}(e^{-w_c+w_c\delta y})[p(\beta-\delta y)e^{(\delta y-\beta)w_s}-(\beta+\delta y)[p(\frac{\beta-\delta y}{\beta+\delta y})]^{\frac{\beta+\delta y}{2\beta}}(1-p)^{1-\frac{\beta+\delta y}{2\beta}}] \\ &\geq \frac{1}{(t+1)^{2M}}E_{x_c\sim\mathcal{D}_{inv}}(e^{w_c\delta y})[p(\beta-\delta y)e^{(\delta y-\beta)w_s}-(\beta+\delta y)[p(\frac{\beta-\delta y}{\beta+\delta y})]^{\frac{\beta+\delta y}{2\beta}}(1-p)^{1-\frac{\beta+\delta y}{2\beta}}] \\ &.\end{aligned}$$

For brevity, we denote the term $(\beta+\delta y)[p(\frac{\beta-\delta y}{\beta+\delta y})]^{\frac{\beta+\delta y}{2\beta}}(1-p)^{1-\frac{\beta+\delta y}{2\beta}}$ as L. Rearranging this and integrating, we get:

$$\int_0^{w_s}\frac{1}{p(\beta-\delta y)e^{(-\beta+\delta y)w_s}-L}dw_s \geq \int_0^t \frac{e^{w_c\delta y}}{(t+1)^2M}dt,$$
$$\frac{\ln(p(\beta-\delta y)-L)-\ln(p(\beta-\delta y)-e^{(\beta-\delta y)w_s}L)}{(\beta-\delta y)L}\geq\frac{\beta}{2M-1}[1-\frac{1}{(1+t)^{2M-1}}]e^{w_c\delta y}.$$

Since for a sufficiently large t, $1-\frac{1}{(1+t)^{2M-1}}>\frac{1}{2}$, we have:

$$\ln[\frac{p(\beta-\delta y)-L}{p(\beta-\delta y)-e^{(\beta-\delta y)w_s}L}]\geq\frac{\beta(\beta-\delta y)Le^{w_c\delta y}}{2(2M-1)},$$

we can further lower bound the right hand side by applying the inequality $x\geq ln(x+1)$ for positive x:

$$\ln[\frac{p(\beta-\delta y)-L}{p(\beta-\delta y)-e^{(\beta-\delta y)w_s}L}]\geq\ln[1+\frac{\beta(\beta-\delta y)Le^{w_c\delta y}}{2(2M-1)}].$$

Thus,

$$\frac{p(\beta-\delta y)-L}{p(\beta-\delta y)-e^{(\beta-\delta y)w_s}L}\geq 1+\frac{\beta(\beta-\delta y)Le^{w_c\delta y}}{2(2M-1)}.$$

Rearraging this inequality:

$$\begin{aligned}e^{w_s(\beta-\delta y)} &\geq \frac{p(\beta-\delta y)}{L}-\frac{p(\beta-\delta y)/L-1}{1+\frac{\beta(\beta-\delta y)Le^{w_c\delta y}}{2(2M+1)}} \\ &=[\frac{p(\beta-\delta y)}{L}+\frac{\beta(\beta-\delta y)e^{w_c\delta y}p(\beta-\delta y)}{2(2M-1)}-\frac{p(\beta-\delta y)}{L}+1]/[1+\frac{\beta(\beta-\delta y)e^{w_c\delta y}L}{2(2M-1)}]. \\ &=\frac{1+\frac{\beta(\beta-\delta y)^2e^{w_c\delta y}p}{2(2M-1)}}{1+\frac{\beta(\beta-\delta y)e^{w_c\delta y}L}{2(2M-1)}}\end{aligned}$$

Putting $L=(\beta+\delta y)[p(\frac{\beta-\delta y}{\beta+\delta y})]^{\frac{\beta+\delta y}{2\beta}}(1-p)^{1-\frac{\beta+\delta y}{2\beta}}$ back into the inequality,

$$\begin{aligned}e^{w_s(\beta-\delta y)} &\geq \frac{1+\frac{\beta(\beta-\delta y)^2e^{w_c\delta y}p}{2(2M-1)}}{1+\frac{\beta(\beta^2-\delta^2)e^{w_c\delta y}[p(\frac{\beta-\delta y}{\beta+\delta y})]^{\frac{\beta+\delta y}{2\beta}}(1-p)^{1-\frac{\beta+\delta y}{2\beta}}}{2(2M-1)}} \\ &=\frac{\frac{2(2M-1)}{\beta(\beta-\delta y)^2e^{w_c\delta y}}+p}{\frac{2(2M-1)}{\beta(\beta^2-\delta^2)e^{w_c\delta y}(\frac{\beta-\delta y}{\beta+\delta y})^{\frac{\beta+\delta y}{2\beta}}}+p^{\frac{\beta+\delta y}{2\beta}}(1-p)^{1-\frac{\beta+\delta y}{2\beta}}}\end{aligned}.$$

Let $c_1:=\frac{2(2M-1)}{\beta(\beta-\delta y)^2e^{w_c\delta y}}$, $c_2:=\frac{2(2M-1)}{\beta(\beta^2-\delta^2)e^{w_c\delta y}(\frac{\beta-\delta y}{\beta+\delta y})^{\frac{\beta+\delta y}{2\beta}}}$, $\epsilon:=\frac{\delta y}{2\beta}$. Finally, we get the lower bound on w_s:

$$w_s\geq\frac{1}{\beta-\delta y}\ln\left[\frac{c_1+p}{c_2+p^{\frac{1}{2}+\epsilon}(1-p)^{\frac{1}{2}-\epsilon}}\right]$$

To show that the lower bound on the dependency on spurious correlations induced by MAT and LDAT does not increase monotonically with p under some conditions, we take the derivative of the obtained

Table 3: Hyperparameter setting of the experiment of Table 1, Figure 1, Figure 4, Table 5 and Table 6.

Parameter	Value
learning rate r	0.00005
batch size b	64
weight decay	0.001
drop out	0.1
AT perturbation radius ϵ (excluding Figure 1)	0.1
FGSM step size γ	0.1
perturbation weight α learning rate η (MAT)	0.001
factor matrix A (B) learning rate ρ (LDAT)	0.01

lower bound $g(p)$ in Theorem 4.1 with respect to p:

$$\frac{\partial g(p)}{\partial p} := \frac{\partial\left(\frac{c_1+p}{c_2+p^{\frac{1}{2}+\epsilon}(1-p)^{\frac{1}{2}-\epsilon}}\right)}{\partial p}$$

Since the denominator of $\frac{\partial g(p)}{\partial p}$ is positive, we pick out the numerator: $c_2 + (\frac{p}{1-p})^{\frac{1}{2}+\epsilon}[\frac{1}{2} - \epsilon + c_1((\frac{1}{2}-\epsilon) - (\frac{1}{2}+\epsilon)\frac{1-p}{p})]$. In order to study the positive and negative change of the numerator, we continue to derive it with respect to p and obtain

$$(\frac{p}{1-p})^{-\frac{1}{2}+\epsilon}\frac{(\frac{1}{4}-\epsilon^2)}{1-p}\frac{p+c_1}{(1-p)p}.$$

Thus, when $p < c_1$, $\frac{\partial g(p)}{\partial p}$ decrease with p monotonically, and when $p > c_1$, $\frac{\partial g(p)}{\partial p}$ increase with p monotonically. $\frac{\partial g(p)}{\partial p}$ reach its minimum at $p = c_1 = \frac{2(2M-1)}{\beta(\beta-\delta y)^2 e^{w_c \delta y}}$. When $\frac{\partial g(c_1)}{\partial p} = c_2 + (\frac{c_1}{1-c_1})^{\frac{1}{2}+\epsilon}(c_1 - 2\epsilon) < 0$, the lower bound does not increase with p monotonically.

□

B Experiment Details

B.1 Setting of the Tiny Experiments

For the experiment in Table 1, Figure 1, Figure 4, Table 5 and Table 6, we use a fixed set of hyperparameters (see Table 3) instead of conducting a random search of 20 trials over the hyperparameter distribution (the setting in [9]) for efficiency. We report the average of the across three independent runs. For model selection method, training-domain validation [9] is used for PACS, OfficeHome and VLCS. For NICO, the OOD validation method is adopted using an OOD validation set following [19].

B.2 Experiment Setting and Additional Results of Table 2

Overall setup. We conduct a random search of 8 trials for PACS, OfficeHome, VLCS and 6 random trials for NICO and Colored MNIST in the hyperparameter search space, instead of 20 trials adopted in [9] for feasibility. We then average the best results for each hyperparameter combination and dataset (according to each model selection criterion) across test domains (except for Colored MNIST where we test on one biased domain only). Finally, we report the average of this number across three independent runs, and its corresponding standard error. We run all datasets for 8000 epochs during the training process.

Hyperparameter Search Space. We use a smaller hyperparameter search space than that in [9]. The search space for PACS, OfficeHome, VLCS, NICO and Colored MNIST is shown in Table 4.

Model Selection Stategy. For PACS, OfficeHome and VLCS datasets, we use training-domain validation proposed in [9]. This model selection method first randomly collect 20% of each training domain to form a validation set. Then, it chooses the hyperparameter maximizing the accuracy on the

Table 4: Hyperparameter setting of the experiment on PACS, OfficeHome, VLCS, NICO and Colored MNIST of Table 2

Dataset	Parameter	Value
PACS, OfficeHome, VLCS	learning rate r	0.00005
	batch size b	64
	weight decay (ERM, AT)	$10^{\text{Uniform}(-4,-3)}$
	weight decay (MAT, LDAT)	0.001
	drop out (ERM, AT)	RandomChoice([0,0.1,0.5])
	drop out (MAT, LDAT)	0.1
	perturbation number k (MAT)	RandomChoice([5,10,15,20])
	perturbation weight α learning rate η (MAT)	RandomChoice([0.01,0.001])
	perturbation rank l (LDAT)	RandomChoice([5,10,15,20])
	factor matrix A (B) learning rate ρ (LDAT)	RandomChoice([0.1,0.01])
NICO	learning rate r	0.00005
	batch size b	64
	weight decay	$10^{\text{Uniform}(-4,-3)}$
	drop out	RandomChoice([0,0.1,0.5])
	perturbation number k (MAT)	Uniform$(10, 20)$
	perturbation weight α learning rate η (MAT)	0.001
	perturbation rank l (LDAT)	Uniform$(10, 20)$
	factor matrix A (B) learning rate ρ (LDAT)	0.01
Colored MNIST	learning rate r	$10^{\text{Uniform}(-4.5,-3.5)}$
	batch size b	$2^{\text{Uniform}(3,9)}$
	weight decay	0
	drop out	RandomChoice([0,0.1,0.5])
	perturbation number k (MAT)	Uniform$(5, 20)$
	perturbation weight α learning rate η (MAT)	$10^{\text{Uniform}(-3,-2)}$
	perturbation rank l (LDAT)	Uniform$(10, 20)$
	factor matrix A (B) learning rate ρ (LDAT)	0.01
	AT perturbation radius ϵ (MAT, LDAT)	$10^{\text{Uniform}(-1,2)}$
	FGSM step size γ (MAT)	$10^{\text{Uniform}(-2,1)}$
	FGSM step size γ (AT)	0.1
All except Colored MNIST	AT perturbation radius ϵ	0.1
	FGSM step size γ (AT, MAT)	0.1

validation set. For NICO, we adopt the OOD validation proposed in [19]. This method chooses the model maximizing the accuracy on a validation set that follows neither the distribution of the training domain or the distribution of the test domain. For Colored MNIST, we use test-domain validation, i.e., using a validation set that follows the distribution of the test domain. This is because it can enlarge the gaps in OOD performance among the algorithms while the gap induced by training-domain validation on Colored MNIST is marginal.

Backbone Network. We use ResNet-18 [18] pretrained on ImageNet [33] for PACS, OfficeHome and VLCS. We use unpretrained ResNet-18 for NICO since it contains images largely overlapped with ImageNet classes. As for Colored MNIST, We use a small CNN-architecture following [9].

Additional Parameter Analysis. We further investigate the impact on OOD performances of the learning rate for the perturbation weights in MAT and the learning rate for the decomposed factors in LDAT. We use the experimental setting introduced in Appendix B.1. The results of MAT and LDAT are shown in Table 5 and 6 respectively. In Table 5 and 6 we observe that the learning rate for the perturbations has a marginal effect on the OOD accuracy. Both MAT and LDAT outperform ERM and AT on PACS when using different values of learning rate.

Table 5: The test accuracy (%) on PACS of MAT when the learning rate (η) for the perturbation weights takes different values. We set the number of perturbations $k = 20$. The other hyperparameters take value in Table 3.

		MAT		
ERM	AT	$\eta = 0.1$	$\eta = 0.01$	$\eta = 0.001$
79.7 $\pm$ 0.0	81.5 $\pm$ 0.4	81.6 $\pm$ 0.2	82.2 $\pm$ 0.4	82.3 $\pm$ 0.5

Table 6: The test accuracy (%) on PACS of LDAT when the learning rate (ρ) for the decomposed factors takes different values. We set the rank of perturbations $l = 10$. The other hyperparameters take value in Table 3.

		LDAT	
ERM	AT	ρ=0.1	$\rho = 0.01$
79.7 $\pm$ 0.0	81.5 $\pm$ 0.4	82.2 $\pm$ 0.6	82.6 $\pm$ 0.2

C Detailed Description of MAT and LDAT

In this section, we describe the detailed training procedure of MAT (see Algorithm 1) and LDAT (see Algorithm 2). We conduct a single-step gradient ascent for the inner maximization for the perturbations in MAT and LDAT. We adopt l_2 norm for the perturbations.

Algorithm 1 Detailed Training Procedure of MAT

```
Input:
    Labeled training data of $m$ domains $E_1, ..., E_m$, where $E_e := \{(x_i^e, y_i^e)\}_{i=1}^{n_e}$,
    number of the perturbations to be combined $k$, perturbation weight $\alpha$ learning rate $\eta$,
    FGSM step size $\gamma$, perturbation radius $\epsilon$,
    number of training epochs $T$, learning rate for model parameters $r$, batch size $b$.
Output:
    Updated model $f_\theta$ with parameter $\theta$.
 1: Randomly initiate $\theta$, perturbation $\delta_i^e$, weight $\alpha_i^e$ such that $\sum_{i=1}^k \alpha_i^e = 1$, $\alpha_i^e \geq 0$, $||\delta_i^e||_2 \leq \epsilon$,
    $\forall i \in \{1, ..., k\}$ and $\forall e \in \{1, ..., m\}$.
 2: for iterations in $1, 2, ..., T$ do
 3:     for $e$ in $1, 2, ..., m$ do
 4:         Randomly select batch $\mathcal{B}^e = \{(x_u^e, y_u^e)\}_{u=1}^b$ from domain $E_e$.
 5:         Compute the adversarial sample: $x_u^{e'} = x_u^e + \sum_{j=1}^k \alpha_j^e \delta_j^e$, $\forall u \in \{1, ..., b\}$
 6:         Update $\delta^e$ by $\delta_i^e \leftarrow \delta_i^e + \gamma \frac{1}{b} \sum_{u=1}^b \nabla_{\delta_i^e} \mathcal{L}(f_\theta(x_u^{e'}), y_u^e)$, $\forall i \in \{1, ..., k\}$, $\forall u \in \{1, ..., b\}$.
 7:         Update $\alpha_i^e$ by $\alpha_i^e \leftarrow \alpha_i^e + \eta \frac{1}{b} \sum_{u=1}^b \nabla_{\alpha_i^e} \mathcal{L}(f_\theta(x_u^{e'}), y_u^e)$, $\forall i \in \{1, ..., k\}$, $\forall u \in \{1, ..., b\}$.
 8:         Project $\delta_i^e$ to the $l_2$ ball of radius $\epsilon$.
 9:         Compute the adversarial sample: $x_u^{e'} = x_u^e + \sum_{j=1}^k \alpha_j^e \delta_j^e$, $\forall u \in \{1, ..., b\}$
10:         Update model parameter: $\theta \leftarrow \theta - r \frac{1}{b} \sum_{u=1}^b \nabla_\theta \mathcal{L}(f_\theta(x_u^{e'}), y_u^e)$, $\forall u \in \{1, ..., b\}$.
11:     end for
12: end for
```

Algorithm 2 Detailed Training Procedure of LDAT

```
Require:
    Labeled training data of $m$ domains $E_1, ..., E_m$, where$E_e := \{(x_i^e, y_i^e)\}_{i=1}^{n_e}$,
    rank of the perturbations $l$, factor $A$, $B$ learning rate $\rho$,
    FGSM step size $\gamma$, perturbation radius $\epsilon$,
    number of training epochs $T$, learning rate for model parameters $r$, batch size $b$.
Ensure:
    Updated model $f_\theta$ with parameter $\theta$.
 1: Randomly initiate $\theta$, perturbation $\delta^e$, factor $A^e$, $B^e$ such that $||\delta^e||_2 \leq \epsilon, \forall e \in \{1, ..., m\}$.
 2: for iterations in $1, 2, ..., T$ do
 3:     for $e$ in $1, 2, ..., m$ do
 4:         Randomly select batch $\mathcal{B}^e = \{(x_u^e, y_u^e)\}_{u=1}^b$ from domain $E_e$.
 5:         Compute the adversarial sample: $x_u^{e'} = x_u^e + A^e B^e, \forall u \in \{1, ..., b\}$
 6:         Update $A^e$ by $A^e \leftarrow A^e + \rho \frac{1}{b} \sum_{u=1}^b \nabla_{A^e} \mathcal{L}(f_\theta(x_u^{e'}), y_u^e), \forall u \in \{1, ..., b\}$.
 7:         Update $B^e$ by $B^e \leftarrow B^e + \rho \frac{1}{b} \sum_{u=1}^b \nabla_{B^e} \mathcal{L}(f_\theta(x_u^{e'}), y_u^e), \forall u \in \{1, ..., b\}$.
 8:         Project $\delta_i^e$ to the $l_2$ ball of radius $\epsilon$.
 9:         Compute the adversarial sample: $x_u^{e'} = x_u^e + A^e B^e, \forall u \in \{1, ..., b\}$
10:         Update model parameter: $\theta \leftarrow \theta - r \frac{1}{b} \sum_{u=1}^b \nabla_\theta \mathcal{L}(f_\theta(x_u^{e'}), y_u^e), \forall u \in \{1, ..., b\}$.
11:     end for
12: end for
```