Turkish Journal of Electrical Engineering and Computer Sciences

Volume 32 | Number 2 Article 7

3-14-2024

Uncovering and mitigating spurious features in domain generalization

SAEED KARIMI
saeed.karimi@bilkent.edu.tr

HAMDİ DİBEKLİOĞLU
dibeklioglu@cs.bilkent.edu.tr

Recommended Citation

KARIMI, SAEED and DİBEKLİOĞLU, HAMDİ (2024) "Uncovering and mitigating spurious features in domain generalization," *Turkish Journal of Electrical Engineering and Computer Sciences*: Vol. 32: No. 2, Article 7. https://doi.org/10.55730/1300-0632.4071
Available at: https://journals.tubitak.gov.tr/elektrik/vol32/iss2/7

TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2024) 32: 320 – 337
© TÜBİTAK
doi:10.55730/1300-0632.4071

Uncovering and mitigating spurious features in domain generalization

Saeed KARIMI[*], **Hamdi DİBEKLIOĞLU**
Department of Computer Engineering, Faculty of Engineering, Bilkent University, Ankara, Turkiye

Received: 22.11.2023 • **Accepted/Published Online:** 08.02.2024 • **Final Version:** 14.03.2024

Abstract: Domain generalization (DG) techniques strive to attain the ability to generalize to an unfamiliar target domain solely based on training data originating from the source domains. Despite the increasing attention given to learning from multiple training domains through the application of various forms of invariance across those domains, the enhancements observed in comparison to ERM are nearly insignificant under specified evaluation rules. In this paper, we demonstrate that the disentanglement of spurious and invariant features is a challenging task in conventional training since ERM simply minimizes the loss and does not exploit invariance among domains. To address this issue, we introduce an effective method called specific domain training (SDT), which detects the spurious features and makes them more discernible. By exploiting a masking strategy and weight averaging, it decreases their harmful effects. We provide theoretical and experimental evidence to show the effectiveness of SDT for out-of-distribution generalization. Notably, SDT achieves comparable results to SWAD, the state of the art in DomainBed benchmarks.

Key words: Specific domain training, domain generalization, image processing, deep neural networks, computer vision

1. Introduction

Machine learning algorithms typically assume that the training data and the test data originate from a shared underlying distribution. This assumption poses a fundamental problem when data comes from different distributions and domains during the training phase, as well as when encountering new, unseen distributions at test time, i.e., out-of-distribution (OoD) data. Empirical risk minimization (ERM) [2] forces the model to exploit all features, whether they are invariant or spurious, in order to reduce the training error. This can lead to errors in new environments when spurious features vary notably.

Learning invariances across domains has been the primary approach in out-of-domain generalization. Most of these works focus on detecting invariant features, such as domain adversarial neural networks [3], correlation alignment (CORAL) [4], and maximum mean discrepancy (MMD) for domain generalization [5]. Invariant risk minimization is another line of research that involves learning intermediate features in such a way that there is an invariant predictor across domains. Recently, newer approaches have been explored to identify invariance across domain-level gradients rather than features. Fish [6] increases the interdomain gradient products, Fishr aligns the covariances of gradients at the domain level, and AND mask [7] updates weights when domain-level gradients have the same direction.

Domain generalization methods primarily focus on identifying invariant features, but they overlook a crucial aspect: neural networks tend to learn simple features over complex ones, even if the predictive power is lower [8, 9]. This phenomenon poses a challenge in detecting complex invariant features [10, 11]. In the domain

[*]Correspondence: saeed.karimi@bilkent.edu.tr

generalization (DG) setting, it is assumed that each training domain contains both spurious and invariant features. Spurious features exhibit high predictive power for a specific domain, but this predictivity diminishes significantly when applied to samples from other domains. The prevalence of these spurious features in training domains is a key factor contributing to the limited generalization to new test domains in existing DG methods.

To address this issue, specific domain training (SDT) pursues two objectives: firstly, it aids the model in discerning spurious features. Unlike classical empirical risk minimization (ERM) and other DG methods that sample from all training domains at each iteration, SDT employs specific domain sampling for some iterations. This enables the model to identify predictive features unique to a specific domain, with corresponding weights receiving increased emphasis. The dynamics of these weights during specific domain training help distinguish between spurious weights (high gradients, significant changes when the training domain switches) and invariant weights (low gradients, relatively constant through domain changes).

To achieve the second objective of suppressing spurious features, SDT employs two strategies: 1) a masking strategy and 2) variance-aware stochastic weight averaging (SWA). While masking strategies have been used in previous DG works [7, 12], SDT introduces a more advanced approach. It adjusts corresponding weights based on agreement across domain-level gradients, preventing catastrophic failures observed in previous methods for some out-of-distribution (OOD) tasks. The second strategy involves leveraging an ensemble method, specifically SWA [13], in the DG setting to harness knowledge from different domains. While ensemble methods are known to be beneficial in DG tasks, SDT's variance-aware SWA uses information from training domains to mitigate the impact of spurious features. It updates weights based on the agreement of domain-level gradients, reducing the contribution of spurious weights with high domain-level gradient variances in the weight averaging process.

We summarize our contributions as follows:

- We present a novel method named SDT designed specifically for distinguishing between model weights related to spurious and invariant features.
- Our approach employs a masking strategy and variance-aware weight averaging technique to effectively identify and prevent excessive updates to spurious weights.
- We provide both theoretical principles and empirical findings to substantiate the effectiveness of our hypothesis.
- Our empirical results, obtained from the DomainBed benchmark, validate our claims using real-world datasets. Specifically, our method improves accuracy in PACS, VLCS, and TerraIncognita by 0.3pp, 0.6pp, and 1.8pp, respectively, when compared to the previous state-of-the-art method [1].

2. Related work

Methods for deep domain generalization can typically be categorized into three groups: domain alignment, metalearning, and data augmentation. Ensemble methods have also been shown to be effective in domain generalization. Our approach is closely related to domain alignment and ensemble methods.

2.1. Domain alignment

Domain alignment stands out as the most intuitive method, extensively explored in the context of domain adaptation (DA) problems. The objective is to learn latent representations that exhibit similar distributions across various domains [4–6, 14]. The work in [3] introduces domain adversarial neural networks (DANN), a domain adaptation (DA) technique that leverages generative adversarial networks (GANs) [15]. DANN strives to learn a feature representation that aligns across diverse training domains. The work in [5] utilizes GANs and maximum mean discrepancy (MMD) [16] to align feature distributions across domains. The work in [17] employs clustering methods to acquire domain-invariant features, even in scenarios where environments are not explicitly specified. Both [4] and [18] align the feature covariance, focusing on matching second-order statistics across various training domains within a certain representation layer. The work in [19] introduces invariant risk minimization (IRM), a method that aims to learn an intermediary representation. This representation ensures that the optimal classifiers for all domains, built upon this representation, are identical. The work in [20] suggests an approximation to the invariant risk minimization (IRM) problem. This approximation involves minimizing the variance of risk averages among different domains.

2.2. Gradient alignment

Gradient alignment is a recent line of work that focuses on aligning the gradients of the loss function concerning the model's weights θ rather than aligning features among domains. This approach emphasizes two key aspects. First, ensuring similarity in the distributions of gradients at the domain level is crucial for fostering shared properties across domains in deep neural networks (DNNs). Second, gradients are considered to be more expressive and richer compared to features, motivating their use in domain alignment strategies. In particular, it has been demonstrated that gradients exhibit a better ability to cluster inputs that are semantically close [21]. Several studies, including [22–25], have shown that neural networks (NNs) often learn superficial features and prioritize low-level statistical patterns over capturing meaningful features and sophisticated abstractions. Therefore, gradient alignment is considered a potential approach to partially mitigate the challenge of poor out-of-domain performance in neural networks. The work in [26] proposes a method where representations associated with higher gradients at each epoch are discarded, and the model is compelled to make predictions using the remaining information. Notably, this approach does not leverage any knowledge about the partitions of source domains. On the contrary, SDT encourages the model to advertently intensify the spurious gradients and then avoid the weight updates corresponding to those spurious gradients. The authors in [6, 7, 27, 28] try to find a shared minima among domains by tackling the domain-level gradients. Specifically, when we have domain set $E = \{A, B\}$, IGA [27] minimizes $\| g_A - g_B \|_2^2$; Fishr matches gradient covariances in a domain-level manner. AND-mask [7] updates the weights only when g_A and g_B have the same direction. Our gradient alignment is similar to this work with a difference that we compare the gradients of current training domain with the average of gradients of the rest of training domains, which is a relaxed version of AND-mask. Fish [6] increases $g_A . g_B$. In Section 4.3, we show that SDT also increases the gradient inner products.

2.3. Ensemble methods

Ensemble methods in deep learning have gained prominence for their effectiveness in improving model performance, robustness, and generalization. Approaches such as model averaging, stacking, and snapshot ensembles have demonstrated success in various applications. Notable studies include snapshot ensembles [43, 44], showcasing the benefits of ensembling models at different training stages. Dropout ensembles [45, 46], leverage dropout regularization for uncertainty estimation, contributing to enhanced model reliability. Bayesian ensemble methods [47–49] provide a probabilistic framework for combining predictions while considering model uncertainties. Ensemble distillation methods [50–52] address the performance gap between large and small models through ensemble-based techniques. Temporal ensembles [53] consider predictions over different time windows or segments, particularly useful for time-series data. These techniques have been applied across domains such as computer vision, natural language processing, and speech recognition, with ensemble methods consistently proving valuable in addressing challenges and improving overall model performance. A significant finding is outlined in [13], illustrating that the practice of averaging weights periodically throughout training, whether with a fixed or varying learning rate, results in enhanced generalization compared to standard training methods. Additionally, the study reveals that this procedure, known as stochastic weight averaging (SWA), leads to the discovery of much flatter solutions than those obtained with SGD. In this paper, we modify the SWA method such that the model weight averaging considers interdomain gradient variances for weight components, so weights with higher interdomain gradient variances contribute less in SWA model at each iteration.

3. Methodology

Consider a training dataset D_{tr} containing k domains $D_{tr} = \{D_1, D_2, ..., D_K\}$, where each domain k is defined by a dataset $D_k = \{(x_i, y_i)\}_{i=1}^{n_k}$ containing data drawn i.i.d from some probability distribution. Additionally, there is a test dataset D_{te} with T domains $\{D_{K+1}, ..., D_{K+T}\}$, where $D_{tr} \cap D_{te} = \emptyset$. The overarching goal of domain generalization is to train a model with parameters θ in such a way that it generalizes effectively to previously unseen domains in the test dataset D_{te}. The objective is to achieve the following:

$$\underset{\theta}{\operatorname{argmin}} \mathbb{E}_{D\sim D_{te}} \mathbb{E}_{(x,y)\sim D}[l((x,y);\theta)], \tag{1}$$

where $l(x, y : \theta)$ is the loss of model θ on (x, y).

ERM minimizes the average loss over D_{tr}, not considering the differences across the training domains:

$$L_{ERM}(D_{tr};\theta) = \mathbb{E}_{D\sim D_{tr}} \mathbb{E}_{(x,y)\sim D} l((x,y);\theta), \tag{2}$$

The ERM objective solely focuses on minimizing the loss of the training dataset without taking into account the invariances among different domains. To illustrate this, we present a simple linear example, adapted from [6], to highlight ERM's limited consideration of invariant features. Subsequently, we examine how weight gradients corresponding to invariant and spurious features behave during separate training runs of ERM and SDT.

3.1. ERM reluctance to invariant features: a linear case

Consider a binary classification task, where data $(x, y) \in \mathbb{B}^4 \times \mathbb{B}$ and a data instance is $x = [f_1, f_2, f_3, f_4]$ with label y. Training and test datasets are sampled form $\{D_1, D_2\}$ and D_3 distributions, respectively. A linear model, $Wx + b = y$, where $W \in \mathbb{R}^4$, $b \in \mathbb{R}$, is trained on the training dataset and tested on the test data. Data setup for each domain is illustrated in Figure 1.

According to Figure 1, f_1 is invariant feature, since correlation between f_1 and y is the same for all domains. However, for features f_2, f_3 and f_4, the correlation changes for each domain, so they are called spurious features. For each domain, there exists a high predictive spurious feature, which has higher correlation with y than invariant feature f_1. For example, using only f_2 attains 97% on $D1$. However, using only f_1 gains 93% on D_1.

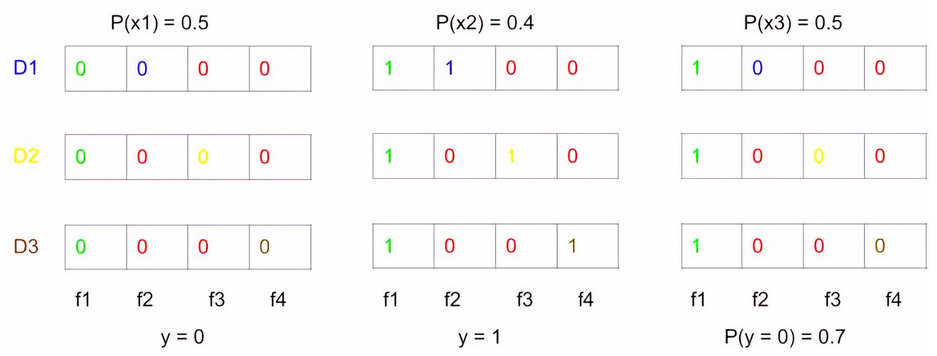


Figure 1. There are 3 kinds of data as x_1, x_2, and x_3, each shown in one column. The first column contains data $x_1 = [0, 0, 0, 0]$ and $y = 0$ for all domains. In the second column, x_2 changes for each domain, y is always 1. In the third column, $x_3 = [1, 0, 0, 0]$ and $y = 1$ for 30% of data of type 3 and $y = 0$ for 70% of this data type. Type 1, 2, and 3 contain 50%, 40%, and 10% of data for each domain.

The poor performance of ERM on test domain has been shown in Table 1. As illustrated in the table for W parameters, ERM assigns higher weight for spurious features f_2 and f_3, while comparatively assigning less weight to the invariant feature.

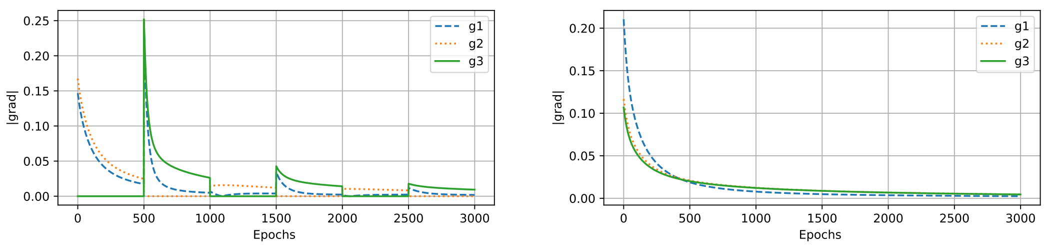
Table 1. Performance comparison on the linear example.

Method	Train acc.	Test acc.	W
ERM	97%	57%	$[3.9, 4.3, 4.3, 0.0]$
SDT	93%	93%	$[3.4, 3.0, 3.0, -0.1]$

3.2. Domain-specific training for disentangling spurious and invariant features

To address the pitfall of ERM in learning spurious features, we introduce a straightforward yet highly effective method for distinguishing between spurious and invariant features. We then apply mechanisms to reinforce the weights corresponding to invariant features while weakening the weights associated with spurious features during each training iteration. In this approach, we train one specific domain intensively for several iterations and then switch to the next domain, continuing this process until the end of training. Intuitively, when we train on one domain for a number of iterations, the weights corresponding to the spurious features of that domain become stronger (with an increase in absolute value). However, when we switch to the next domain, the absolute value of these weights stops increasing and may even start to decrease.

To validate our approach, we apply specific domain training (SDT) to the previous linear example and track the gradients of the corresponding weights as indicators of change for each weight at each training step. As depicted in Figure 2, during SDT training, spurious features exhibit the largest gradients at each training interval. For instance, during the interval $[0, 500]$, when training on D_1, the gradient with respect to feature f_2, which is highly predictive for D_1, exhibits the largest gradient value. During the second interval, when training on D_2, feature f_3, which is spurious for D_2, has the largest gradient value. In contrast, in ERM training, gradients for all weights change smoothly and are challenging to discriminate easily.



Gradient weights in SDT

Gradient weights in ERM

Figure 2. Absolute gradient of loss w.r.t $W = [w_1, w_2, w_3]$ shown as g_1, g_2, g_3.

3.3. The spurious and invariant features model

In this context, a set of definitions is introduced to formally establish the concepts of ”invariant” and ”spurious” features. These definitions serve as the foundation for presenting theoretical results.

Setup. The study focuses on binary classification, where $(x, y) \in X \times \{\pm 1\}$ are sampled from $D_{tr} = \{D_1, D_2, ..., D_S\}$. The objective is to train a classifier $C : X \to \{\pm 1\}$, where X represents the input space, to predict a label y given an input x.

A feature is formally defined as a function that maps from the input space X to the real numbers. The set of all features is denoted as $F = \{f : X \to \mathbb{R}\}$. The features within the set F are treated such that they are shifted and scaled to have a zero mean and a unit variance, to ensure scale invariance in the upcoming definitions.

Invariant features: We call a feature f an invariant feature across all $D \in D_{tr}$, if it is correlated with the true label such that

$$\forall D_i \in D_{tr}, \mathbb{E}_{(x,y)\sim D_i}[y.f(x)] = \rho. \tag{3}$$

We consider positive correlations here ($\rho > 0$).

Spurious features: For a given distribution D_s, we designate a feature f a spurious feature of D_s, if it is correlated with the true label such that

$$\mathbb{E}_{(x,y)\sim D_s}[y.f(x)] = \rho \ \wedge \ \forall D_i \in D_{tr}, \mathbb{E}_{(x,y)\sim D_i}[y.f(x)] < \rho. \tag{4}$$

Standard training. Consider a classifier $C = (F, \omega, b)$ which includes a set of features F, weight vector ω and an scalar bias b. Training of the classifier involves minimizing a loss function through ERM. The goal is to minimize the loss by modifying the correlation between the weighted combination of features and the corresponding labels. We employ a straightforward loss function in our equation; however, it can be easily extended to more practical loss functions, such as hinge or logistic loss.

$$L_\Theta(x, y) = -\mathbb{E}_{D\sim D_{tr}}\mathbb{E}_{(x,y)\sim D}[y.(b + \sum_{f\in F} \omega_f.f(x))]. \tag{5}$$

Theorem 1 *Consider $D_{tr} = \{D_1, D_2\}$ and feature f is a spurious feature for D_1 i.e. $\mathbb{E}_{(x,y)\sim D_1}[y.f(x)] = \rho_1$ and $\mathbb{E}_{(x,y)\sim D_2}[y.f(x)] = \rho_2$, where $\rho_1 > \rho_2$ and ω_f is corresponding weight for feature f in classifier c. Then*

$$|\mathbb{E}_{(x,y)\sim D_1\cup D_2}[\frac{\partial L}{\partial \omega_f}]^t - \mathbb{E}_{(x,y)\sim D_1\cup D_2}[\frac{\partial L}{\partial \omega_f}]^{t+1}| = 0, \tag{6}$$

$$|\mathbb{E}_{(x,y)\sim D_1}[\frac{\partial L}{\partial \omega_f}]^t - \mathbb{E}_{(x,y)\sim D_2}[\frac{\partial L}{\partial \omega_f}]^{t+1}| = \rho_1 - \rho_2. \tag{7}$$

Theorem 1 states that when we train the model using ERM, the gradient $\frac{\partial L}{\partial \omega_f}$ remains constant at each step, with a value of $-\frac{\rho_1+\rho_2}{2}$. In contrast, when applying SDT with one-step intervals, at each step, we switch between domains D_1, D_2. Consequently, the gradient value oscillates between ρ_1 and ρ_2. As a result, the gap for spurious gradients in SDT is equal to $|\rho_1 - \rho_2|$, while it remains zero for ERM at each training step.

3.4. SDT components to reduce the effects of spurious features

By exclusively training each domain for a certain number of steps, we allow the model to differentiate between spurious and invariant weights. In this section, we introduce two methods: the masking strategy and stochastic weight averaging (SWA), designed to prevent the excessive growth of weights associated with detected spurious features.

Masking strategy: In the process of disentangling spurious and invariant weights through SDT, we utilize the masking strategy to mitigate the influence of spurious features. Let us consider the current training domain in SDT as s, and denote the rest of the domains as $\overline{s}$. At each training step, we calculate the mean loss gradient with respect to the corresponding weight component j (note that this method is applied only to the final linear classifier) for both the current domain s and the rest of the domains $\overline{s}$. These gradients are denoted as $[\nabla L_s]j$ and $\frac{1}{|\overline{s}|}\sum e \in \overline{s}[\nabla L_e]_j$, respectively. A weight component j is considered invariant at iteration t if the gradients $[\nabla L_s]j$ and $\frac{1}{|\overline{s}|}\sum e \in \overline{s}[\nabla L_e]_j$ have the same sign. Otherwise, it is categorized as a spurious weight. For invariant weights, we update them with the gradient calculated for domain s, which is $[\nabla L_s]_j$. In the case of spurious weights, some previous methods, like [7], do not update them at all, or the method in [12] zeroes out the weight. However, these methods can be restrictive and may lead to weights receiving no further gradients, causing the network to become stuck and perform poorly.

In our approach, we update the weight associated with the gradient $[\nabla L_s]_j$ in a direction that brings the weight closer to zero, i.e. $|\Theta_j^{t+1}| \leq |\Theta_j^t|$. This reduces the impact of spurious weight Θ_j on the classification. Notably, although masking has shown limited effectiveness in previous works like [7] and [32] in real-world datasets, it notably improves the performance of SDT in our empirical experiments. This improvement could be attributed to the fact that SDT excels at distinguishing between invariant and spurious weights compared to ERM; therefore, applying masking helps further prevent the growth of these spurious weights.

Variance aware stochastic weight averaging: The SWA method, as introduced in [13], is based on averaging model weights θ at some checkpoints throughout training. Empirical results presented in [1] demonstrate that SWA can find flatter minima, leading to better generalization by approximating ensembles of model weights along the SGD trajectory. It is well-established that finding flatter minima can guarantee improved generalization performance, as noted in [33]. Consequently, SWA has proven to be beneficial in domain generalization tasks.

In our model training, we also incorporate SWA to address the instability of training when each domain is trained individually and to enhance generalization. Within the model weight space $\Omega = \omega_0, \omega_1, \ldots, \omega_N$, where N represents the number of training steps, we initiate the sampling of weights at some initial step m

and continue this process for subsequent steps. During each step t, if the loss exceeds a predefined threshold τ, the weight ω_t is not included in the SWA ensemble ω_{swa}.

To align SWA with SDT and reduce the impact of spurious weights, we calculate the inter-domain gradient variances for the current batch within each domain d, denoted as $v = Var_{d \in D}(\nabla L_d)$. In the model averaging process, we leverage these normalized variances to compute the SWA model weights as follows:

$$\theta_{swa} = \frac{\frac{1}{v}\theta + n\theta_{swa}}{n+1}, \tag{8}$$

where n is the number of models used for averaging so far. By doing normalized variances as SWA coefficients, we encourage the SWA model to give more credit for the weight components, which have less interdomain gradient variances and hence are more invariant across domains. Algorithm 1 illustrates SDT with all of its components.

Algorithm 1 Domain-specific training

```
**Inputs:** Specific domain interval $I$, Model weight $\theta$, SWA Model $\theta_{swa}$, SWA acceptance ratio $\tau$, Normalized inter-domain gradient variance $v$.
$n = 0$
$D_c = D_1$    \ \ $D_c$ is current domain
**while** not end of training **do**
    **for** $t = 1$ to $I$ **do**
        $sample\ batch\ d_c \sim D_c$
        $g_c = E_{d_c}[\frac{\partial l(x,y);\theta}{\partial \theta}]$
        $sample\ batch\ d_{\overline{c}} \sim D \setminus D_c$
        $g_{\overline{c}} = E_{\overline{c}}[\frac{\partial l(x,y);\theta}{\partial \theta}]$
        **for** $\omega \in \theta$ **do**
            **if** $Sign([g_c]_\omega) == Sign([g_{\overline{c}}]_\omega)$ **then**
                Update $\omega = \omega - \alpha [g_c]_\omega$
            **else**
                Update $\omega = \omega - \alpha Sign(\omega) |[g_c]_\omega|$
            **end if**
        **end for**
        **if** $E_{c \cup \overline{c}} l(x,y) < \tau$ **then**
            $\theta_{swa} = \frac{\frac{1}{v}\theta + n\theta_{swa}}{n+1}$
            $n = n + 1$
        **end if**
    **end for**
    $D_c = next\_domain()$
**end while**
```

4. Experiments

We divide our experiments into three parts. First, we show the effectiveness of SDT in learning complex invariant features by exploiting the synthetic dataset presented in [11]. Then, we compare SDT with other DG methods on DomainBed benchmark. Finally, an ablation study is conducted to evaluate the effectiveness of various components of SDT.

4.1. Effectiveness of SDT in learning invariant complex features

The work in [11] proposes that neural networks have tendency to learn simple features rather than complex features, although their predictive power may be weaker than complex ones. This phenomenon, which is named simplicity bias, causes neural networks to learn simple spurious correlations and be reluctant to learn complex features, which may be invariant among environments and learning them are necessary to have good out of distribution generalization. In this section, we empirically show the effectiveness of SDT in learning complex invariant features.

Toy dataset buliding blocks: We borrow the synthetic dataset introduced in [11] to build our train and test domains. The dataset consists of three one-dimensional data blocks, each containing different patterns: linear, noisy linear, and k-slabs, as illustrated in Figure 3. Within the linear block, positive examples are uniformly distributed in the range $[0.1, 1]$, while negative examples are uniformly distributed in the range $[-1, -0.1]$. In the noisy linear block, characterized by a noise parameter $p \in [0, 1]$, a fraction $1 - p$ of points follows the distribution described in the linear block mentioned earlier. Additionally, a fraction p of the points is uniformly distributed in the range $[-0.1, 0.1]$. In the k-slab blocks, positive and negative examples are dispersed among k distinct regions, alternating between positive and negative instances. For linear block, a single linear classifier can attain accuracy 1 and for noisy linear block, single linear classifier will get $1 - p/2$ accuracy. Finally, for k-slab data block, $k-1$ linear classifier is needed in order to attain accuracy 1. Hence, linear blocks are the simplest features and k-slab blocks are more complex and as k increases, the complexity of the data block increases as well.

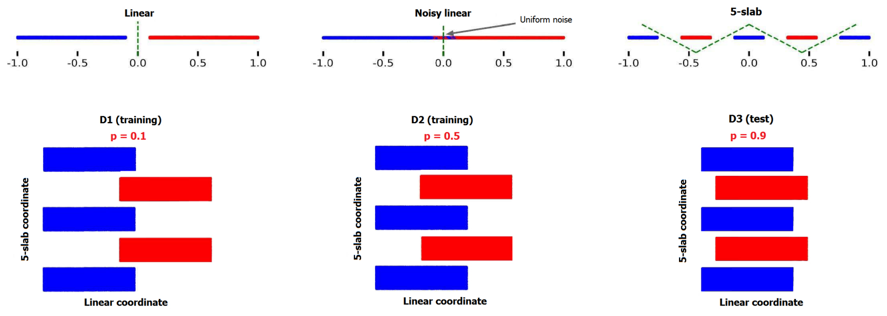


Figure 3. **Top row:** One-dimensional building blocks. **Bottom row:** Synthetic two dimensional domains: D_1 and D_2 are training domains and D_3 is test domain. Noise level for noisy linear coordinate is 0.1 for D_1, 0.5 for D_2 and 0.9 for D_3.

Synthetic domains: Here, we have generated toy domains using synthetic data blocks, as illustrated above. As depicted in Figure 3, we have designated D_1 and D_2 as training domains, while D_3 serves as the test domain. Each domain consists of two-dimensional data blocks. The first dimension within each domain represents a simpler noisy linear feature, with varying noise levels across domains (spurious feature). The second dimension encompasses a more complex 5-slab feature, which remains invariant across all domains. With increasing noise levels in the linear feature, the predictive power of that feature diminishes. In such cases, the model searches for other predictive features, likely more complex ones, to minimize the loss.

Results: We conducted a binary classification task on the synthetic domains described earlier. In an ideal domain generalization scenario, we expect the model to exclusively learn the invariant feature (5-slab feature). However, due to the simplicity bias, the ERM approach predominantly learns the simpler spurious feature (linear feature). This bias is further exacerbated by the noise level differences between D_1 and D_2

(0.1 vs. 0.5) and the inherent simplicity of the linear feature compared to the 5-slab feature. As a result, the contribution of D_1 is more pronounced in ERM training, leading to an overreliance on domain D_1 instead of D_2.

In an ideal domain generalization task, all domains should contribute equally, as they share common invariant features, and classification should be based on these shared features. In contrast, SDT, which involves exclusive training of D_2 for some iterations, provides the model with more opportunities to extract the 5-slab feature. The high noise in the linear feature of D_2 makes it less predictive, driving the model to rely more on the 5-slab feature. Table 2 validates this hypothesis by presenting the validation accuracy for each domain and the test accuracy for the test domain when using both ERM and SDT with a fully connected neural network (FCN) as our model.

Table 2. Performance comparison on the synthetic dataset.

Method	D1 validation acc.	D2 validation acc.	D3 test acc.
ERM	88%	73%	48%
SDT	77%	74%	63%

We hypothesize that specific domain training exploits more invariant features among domains, as demonstrated by the experiments on the toy dataset. To empirically support this hypothesis, we compare the reliance of SDT and ERM on input features using the same domain settings as in our toy dataset experiment. We calculate the absolute loss gradient with respect to input features for both ERM and SDT. A higher absolute loss gradient on any feature indicates a stronger correlation of the loss with that feature, signifying that the model relies more on that feature. As depicted in Figure 4, ERM predominantly relies on the first spurious feature. In contrast, SDT elicits the second invariant and complex feature. This difference arises because exclusive training of domain D_2 compels the model to extract more predictive features. Furthermore, since the first feature of D_2 is not sufficiently predictive, the model relies more on the second feature, which is more informative for classification.

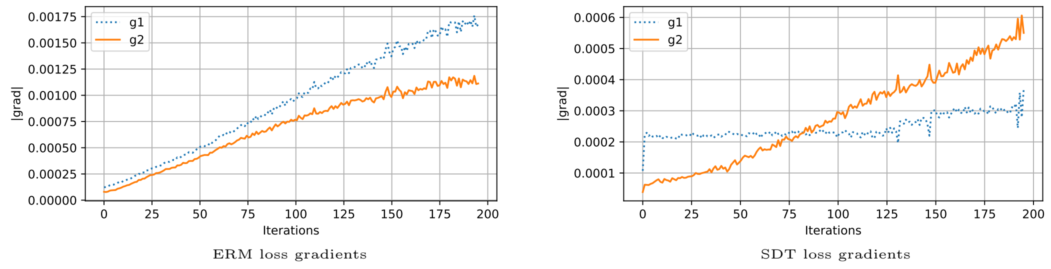


Figure 4. Absolute gradient of loss w.r.t each of the input features of toy dataset. x_1 is the linear, noisy, and spurious feature while x_2 is 5-slab invariant feature. g_1 and g_2 are the gradient of loss w.r.t x_1 and x_2, respectively.

4.2. Comparison to other methods

In this section, we initially introduce the DomainBed benchmark, its datasets, and evaluation protocols. Subsequently, we conduct a comparative analysis of our method with other DG approaches.

Benchmark datasets: We evaluate our method on five famous benchmarks on domain generalization task and compare its results with other state-of-the-art methods in DG. PACS [34] comprises 9991 examples and 7 classes collected from 4 domains: art, cartoons, photos, and sketches. VLCS [35] includes four domains: Caltech101, LabelMe, SUN09, and VOC2007, containing 10,729 images and 5 classes. OfficeHome [36] consists of 4 domains: art, clipart, product, and real, encompassing 15,588 images and 65 classes. TerraIncognita [37] includes 24,788 images and 10 classes with 4 domains: L100, L38, L43, and L46. DomainNet [38] encompasses six domains: clipart, infograph, painting, quickdraw, real, and sketch with 586,575 images and 345 classes.

Implementation details: For an equitable comparison among domain generalization (DG) methods, we adhere to the training and evaluation guidelines outlined in [39]. This includes incorporating data augmentation, conducting hyperparameter searches, and employing dataset splits. However, the evaluation protocol in [39] is computationally intensive. It performs a random search comprising 20 trials across the hyperparameter distribution for each algorithm and test environment. To optimize computational efficiency, we simplify the search space of our approach, by setting some hyperparameters such as weight decay and dropout to a default value 0. All performance metrics are evaluated through leave-one-domain-out cross-validation. Specifically, we designate one domain as the target (test) domain, with the remaining domains serving as training domains. In the DomainBed [39] framework, which standardizes experimental settings for DG algorithms, a hold-out validation strategy is utilized for consistency in comparing with other DG methods. Nevertheless, in our preliminary experiments, we have also employed 3- and 5-fold cross validation. Nonetheless, the outcomes did not indicate a notable difference. Of the training domains, 80% are used for training, while the remaining 20% are employed for validation and model selection.

In the context of weight initialization, we utilize the pretrained ResNet-50 [31] trained on ImageNet [40] as the initial weight. Additionally, batch normalization statistics are frozen throughout the training process. For optimization, we utilize the Adam optimizer with a learning rate of $5e-5$. Dropout probability and weight decay is set to 0. For each training iteration, we build up mini-batches of size 32 of specific domain. Total number of iterations differ for each dataset: It is set to 8000, 2000, 8000, 15000, 25000 for PACS, VLCS, OfficeHome, TerraIncognita, and DomainNet respectively. Averaging start iteration is selected based on the convergence iterations of each dataset. Therefore, it is set to 1000 for PACS, VLCS, and OfficeHome and 10000 for TerraIncognita and DomainNet. SWA acceptance threshold is searched in $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ in PACS dataset and the searched value used in all other datasets. Finally, domain training interval is set to 100 for first 1000 iterations, then increased to 200 until 4000 iterations, and 400 for the end of iterations.

Infrastructures: Each experiment is carried out using a standalone NVIDIA GeForce GTX 1080, along with Python 3.10.9, PyTorch 1.12.1, Torchvision 0.8.2, and CUDA 12.1.

Results: For each domain, we conduct three model training sessions with random data splits and report the average test results. The results presented in Table 3 are borrowed from SWAD [1] and [14]. In these tables, the results for all other methods are obtained by training with a ResNet-50 backbone using the same training and validation protocols as described above. DST demonstrates superior performance compared to SWAD in PACS, VLCS, and TerraIncognita by 0.5%, 0.5%, and 1.8%, respectively. For the remaining datasets, its performance is slightly lower than SWAD by small margins.

Table 3. Out-of-domain accuracies of domain-specific training (ours) with other DG methods on five benchmarks.

Dataset	ERM [2]	CORAL [4]	SagNet [41]	RSC [26]	AND-mask [7]	Sand-mask [32]	Fishr [14]	SWAD [1]	SDT
PACS	85.5	86.2	86.3	85.2	84.4	84.6	85.5	88.1	**88.6** ±0.3
VLCS	77.5	78.8	77.8	77.1	78.1	77.4	77.8	79.1	**79.6** ±0.4
HomeOffice	66.5	68.7	68.1	65.5	65.6	65.8	67.8	**70.6**	70.2 ±0.2
TerraInc	46.1	47.7	48.6	46.6	44.6	42.9	47.4	50.0	**51.8** ±0.6
DomainNet	40.9	41.5	40.3	38.9	37.2	32.1	41.7	**46.5**	45.4 ±0.6
Avg.	63.3	64.5	64.2	62.7	62.0	60.6	64.0	66.9	**67.1**
PACS - Art	84.7	88.3	87.4	85.4	85.3	85.8	88.4	**89.3**	**89.3** ±0.4
PACS - Cartoon	80.8	80.0	80.7	79.7	79.2	79.2	78.7	**83.4**	83.2 ±0.3
PACS - Photo	97.2	97.5	97.1	97.6	96.9	96.3	97.0	**97.3**	97.2 ±0.2
PACS - Sketch	79.3	78.8	80.0	78.2	76.2	76.9	77.8	82.5	**84.6** ±0.4
VLCS - Caltech	97.7	98.3	97.9	97.9	97.8	98.5	**98.9**	98.8	97.6 ±0.1
VLCS - LabelMe	64.3	**66.1**	64.5	62.5	64.3	63.6	64.0	63.3	63.3 ±0.7
VLCS - Sun09	73.4	73.4	71.4	72.3	73.5	70.4	71.5	75.3	**78.7** ±0.5
VLCS - Voc2007	74.6	77.5	77.5	75.6	76.8	77.1	76.8	**79.2**	78.8 ±0.6
OfficeHome - Art	61.3	65.3	63.4	60.7	59.5	60.3	62.4	**66.1**	65.2 ±0.4
OfficeHome - Clipart	52.4	54.4	54.8	51.4	51.7	53.3	54.4	57.7	**58.5** ±0.5
OfficeHome - Product	75.8	76.5	75.8	74.8	73.9	73.5	76.2	**78.4**	77.6 ±0.1
OfficeHome - Photo	76.6	78.4	78.3	75.1	77.1	76.2	78.3	**80.2**	79.5 ±0.3
TerraInc - L100	54.3	51.6	53.0	50.2	50.0	45.7	50.2	55.4	**60.8** ±0.2
TerraInc - L38	42.5	42.2	43.0	39.2	40.2	31.6	43.9	44.9	**46.1** ±0.7
TerraInc - L43	55.6	57.0	57.9	56.3	53.3	55.1	55.7	**59.7**	58.5 ±0.3
TerraInc - L46	38.8	39.8	40.4	40.8	34.8	39.0	39.8	39.9	**41.8** ±0.3
DomainNet - Clip	63.0	59.2	57.7	55.0	52.3	43.8	58.2	**66.0**	64.5 ±0.2
DomainNet - Info	21.2	19.7	19.0	18.3	16.6	14.8	20.2	**22.4**	22.3 ±0.4
DomainNet - Paint	50.1	46.6	45.3	44.4	41.6	38.2	47.7	**53.5**	52.2 ±0.1
DomainNet - Quick	13.9	13.4	12.7	12.2	11.3	9.0	12.7	**16.1**	14.2 ±0.3
DomainNet - Real	63.7	59.8	58.1	55.7	55.8	47.0	60.3	**65.8**	63.9 ±0.4
DomainNet - Sketch	52.0	50.1	48.8	47.8	45.4	39.9	50.8	**55.5**	55.3 ±0.2

As presented in Table 3, our method outperforms SWAD in some domains by a magnificent margin. In PACS dataset, for sketch domain, we get 2.1pp performance gain, in VLCS dataset, for Sun09 domain, we get 3.3pp increase in accuracy, and in TerraIncognita, for L100 domain, we achieve 5.4pp performance gain. We conjecture that for these test domains, our method avoids higher reliance on one training domain with spurious features and elicit more invariant features among all training domains.

4.3. Ablation study

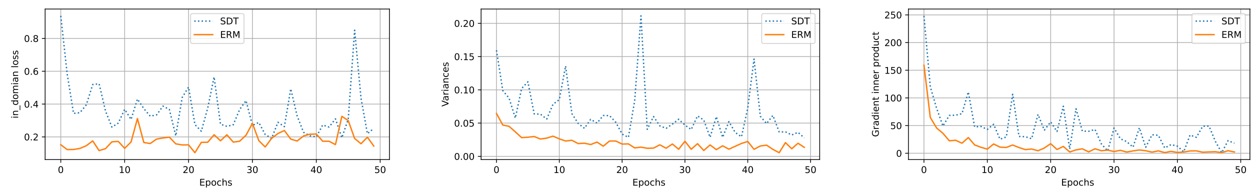
In this section, we do an ablation study on PACS dataset with the same protocols and implementation details as DomainBed. Specifically, we investigate in-domain losses, domain level gradient variances and gradient product. Then, we evaluate effectiveness of SDT components individually. Finally, we analyze SDT by varying its hyperparameters.

In-domain losses: SDT does not decrease in-domain losses; however, it leads to an improvement in out-of-domain loss. In Figure 5a, we present the validation losses obtained from training on the PACS dataset using both ERM and SDT. In this dataset, Sketch serves as the test domain, while Art, Cartoon, and Photo are the training domains. As depicted in Figure 5a, the average in-domain losses of SDT are higher than those of ERM for all the domains. However, our method shows a drop in test loss. This suggests that SDT is better at eliciting invariant features across domains. The underlying reason is that through specific domain training, the model avoids overfitting to the training domains. Additionally, with the aid of the masking strategy and variance-aware weight averaging, it attempts to learn more invariant features that generalize well across domains.

Variances among domain gradients: As demonstrated in the linear example and the theoretical evidence presented above, SDT outperforms standard training in the disentanglement of invariant and spurious features. We further investigate this phenomenon using the real-world PACS dataset. As shown in Figure 5b, inter-domain gradient variances in SDT are notably higher compared to ERM. We hypothesize that when the domain gradient variance for a weight component $[\theta]_j$ is larger, there is a higher likelihood that it represents a spurious weight. This is because a higher variance suggests that the gradients across domains are dissimilar either in direction or magnitude. In contrast, in ERM, domain gradient variances for weight components are generally too small for all weights, making it challenging to distinguish between spurious and invariant weights.

Gradients inner product: In [6], the authors introduce an algorithm called Fish and provide both theoretical and experimental evidence demonstrating that Fish aligns with domain-level gradients. To summarize, let us consider θ as the current model weight and $\theta^{'}$ as a copy of it. In each iteration, Fish samples mini-batches from each training domain and sequentially updates $\theta^{'}$ for each training domain. After completing one pass by sampling from all training domains, it updates the original model weight θ using the rule $\theta = \theta + \epsilon(\theta - \theta^{'})$. When both the training domain interval and the parameter ϵ are set to 1, SDT behaves similarly to Fish.

We compare the interdomain gradient inner products (GIP) for both ERM and SDT. The GIP is calculated as the sum of inner products between gradients from any two different training domains in the classification layer, denoted as $\sum_{i,j \in S}^{i \neq j} G_i \cdot G_j$, where i and j represent distinct domains, and G_i represents the mean gradient for a specific domain i. As illustrated in Figure 5c, SDT consistently exhibits higher GIP values compared to ERM throughout the training process. This observation provides further evidence that SDT, like Fish, aligns with domain-level gradients, indicating its proficiency in extracting more invariant features compared to ERM.



(a) (b) (c)

Figure 5. Comparison of SDT and ERM in PACS dataset for: (a) in-domain losses, (b) domain gradient variances, and (c) inner gradient products. Art, Cartoon, and Photo are the training domains and Sketch is the test domain. In (a), in-domain loss is the average validation loss of training domains. In (b), variances are calculated for the classification layer of the network. In (c), gradient inner product (GIP) is calculated by sum of inner products for any two training domains in classification layer.

ERM vs SDT and components: In Table 4, we compare ERM and SDT with the variants by adding the components discussed in previous sections, i.e. SWA and masking strategy for PACS dataset. We can observe that SWA has a great impact in increasing the out-of-domain accuracy. Specifically, it increases the accruacy abut 4.2%. Masking also increases the accuracy about 1.6% compared to pure SDT. However, applying both masking and SWA has a negligible impact on the accuracy. One reason could be that both of the masking and SWA try to decrease the spurious features effect, and in this case, SWA dominates masking effect as the results in Table 4 validate this claim. Comparing ERM and pure SDT in Table 4 shows that pure SDT has 1.2% less accuracy than ERM. This drop in accuracy is expected since by exclusively training one domain, model learns the spurious features of that domain intensively. However, it helps the model detect the weights corresponding to spurious features and exploiting masking and weight averaging avoids these weights to highly affect the accuracy. As shown in Table 4, masking has negative impact on accuracy of ERM method. The reason could be that since in ERM interdomain gradient variances are small, masking is not effective in detecting spurious weights. SWA is also highly effective in increasing the accuracy of ERM (2.2%), although its impact is still less pronounced compared to its affect on SDT (4.2%).

Table 4. Comparison of out-of-domain accuracies of SDT and ERM with the component variants on PACS.

Model components	Art	Cartoon	Photo	Sketch	Avg.
SDT + SWA + masking	89.3±0.3	83.2±0.4	97.2±0.1	84.5±0.7	88.6
SDT + SWA	89.4±0.4	83.3±0.6	97.2±0.1	84.1±0.6	88.5
SDT + masking	84.8±1.1	81.3±1.3	97.1±0.2	80.4±0.8	85.9
SDT (pure)	82.6±1.2	79.4±1.1	96.2±0.3	79.1±1.4	84.3
ERM + SWA	89.2±0.4	83.1±0.3	97.3±0.1	81.4±0.8	87.7
ERM + masking	84.1±0.3	80.3±0.5	97.1±0.2	79.4±0.8	85.2
ERM	84.7±0.4	80.8±0.6	97.2±0.3	79.3±1.0	85.5

SDT intervals: First, we investigate the impact of domain training intervals on the out-of-domain accuracies of the PACS dataset. As depicted in Figure 6, we report the accuracies for intervals of $50, 100, 200, 300, 400$. Additionally, we conduct an experiment involving a mixture of intervals, where we initiate training with 100 iterations per domain at the outset and progressively extend the intervals as training proceeds. Our experiments indicate that the mixture of intervals yields the highest accuracy. We hypothesize that, early in the training process, as suggested in [42], the model tends to learn easier features before gradually tackling more complex ones. Consequently, we commence with shorter intervals for each domain and incrementally extend them to encourage the network to acquire more intricate features over time.

SWA acceptance threshold: We examine the influence of the SWA acceptance threshold on domain generalization performance. In all our experiments, we assume that if at least half of the training domains meet the threshold condition, the current model weights will be included in the final SWA model. We conduct training with thresholds ranging from $0, 0.05, 0.1, 0.15, 0.2$, and the results are displayed in Figure 6. A threshold of 0 means that the current model will be averaged into the SWA model without any threshold requirement for mean loss.

5. Discussion and limitations

SDT slightly degrades performance for some domains. Compared to SWAD [1], some domain accuracies exhibit a slight degradation in SDT, as shown in Table 3. In these cases, we hypothesize that the features in the test domain are more correlated with a dominant training domain (a domain that contains more easily learnable predictive features). Therefore, when SDT is applied, the dominant domain contributes less to the training compared to ERM, resulting in a drop in test accuracy. However, we believe that this decrease in accuracy is not due to SDT's inability to learn invariant features. On the contrary, since SDT focuses on learning more invariant features and because the spurious features of the test and dominant domains are highly correlated, such degradation in performance can occur.

SDT needs more iterations to converge. Training each domain exclusively at each interval can cause the model to diverge from the optimal minima in the loss landscape for some intervals, thereby requiring more iterations for convergence. Conversely, SDT aims to explore more expansive regions in the loss landscape, and by using the SWA method, it may discover flatter minima among these regions. However, the issue of late convergence becomes more pronounced when dealing with larger datasets and a higher number of domains.

For instance, in the DomainNet dataset, despite increasing the number of iterations from 15,000 to 25,000, the performance, while higher than ERM, still falls short of SWAD [1], as shown in Table 3.

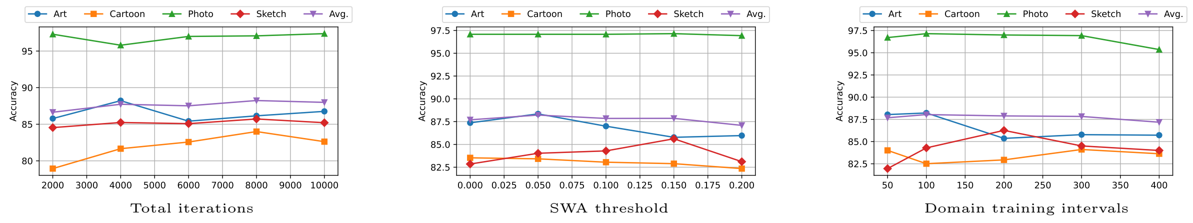


Figure 6. Out-of-domain accuracy on PACS dataset with varying model hyperparameters.

6. Future scope and practical applications

SDT and other gradient-based techniques discussed in this paper fall under a category of methods aiming to ensure consistency across domains concerning gradients with respect to θ. One limitation of these gradient-based methods is the per-domain batch averaging of gradients, which leads to the removal of more detailed statistics. Notably, this averaging eliminates information related to pairwise interactions between gradients from samples within the same domain. Coral [4], which implements covariance matching across domain feature representations, demonstrates superior performance across various out-of-distribution (OOD) generalization tasks. A potential strategy for future research involves applying covariance matching across domain gradients instead of features. This approach will be explored in subsequent studies. As highlighted in [39], domain generalization (DG) methods do not significantly enhance classical empirical risk minimization (ERM) since they lack access to test data. Another avenue for research could involve implementing SDT in test time adaptation (TTA) methods, where an online batch of test data is available, enabling adaptation before the inference stage.

SDT and DG methods can be applied to medical imaging tasks, where models need to generalize across images from different hospitals, medical devices, or populations to ensure robust diagnostic performance. In the context of autonomous vehicles, DG methods help models adapt to diverse driving conditions, such as different weather, lighting, and road scenarios, ensuring safe and reliable performance across various environments. In finance, DG models can generalize across different markets, economic conditions, or financial instruments are valuable. This is particularly important for tasks like stock price prediction and risk assessment. Models used for environmental monitoring tasks, such as climate prediction or pollution detection, can benefit from domain generalization to adapt to different geographical regions, seasons, and data sources.

7. Conclusion

In this paper, we delve into the disentanglement of spurious and invariant features when dealing with domain generalization tasks involving multiple domains. We argue that standard training methods fall short in properly extracting invariant features as they primarily focus on minimizing the loss and disregard the invariance among domains. To address this issue, we introduce specific domain training (SDT), a novel approach that involves training individual domains exclusively for specific intervals. During the initial stages of each training interval, the model tends to learn simple spurious features associated with the current training domain. SDT effectively identifies these spurious features, and through the application of masking strategies and variance-aware weight averaging, helps the model in avoiding the learning of these unwanted features. We provide both theoretical and empirical evidence to highlight the effectiveness of SDT in detecting and suppressing spurious features within the model. Notably, SDT achieves competitive results when compared to SWAD, the current state-of-the-art method, on DomainBed benchmarks.

References

[1] Cha J, Chun S, Lee K, Cho HC, Park S et al. SWAD: Domain generalization by seeking flat minima. Advances in Neural Information Processing Systems. 2021 Dec 6;34:22405-18.

[2] Vapnik VN. An overview of statistical learning theory. IEEE Transactions on Neural Networks. 1999 Sep;10(5):988-99.

[3] Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H et al. Domain-adversarial training of neural networks. The journal of machine learning research. 2016 ;17 (1):2096-30.

[4] Sun B, Saenko K. Deep coral: Correlation alignment for deep domain adaptation. In: Computer Vision–ECCV 2016 Workshops: Amsterdam, the Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14 2016 (pp. 443-450). Springer International Publishing.

[5] Li H, Pan SJ, Wang S, Kot AC. Domain generalization with adversarial feature learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2018 (pp. 5400-5409).

[6] Shi Y, Seely J, Torr PH, Siddharth N, Hannun A et al. Gradient matching for domain generalization. arXiv preprint arXiv:2104.09937. 2021 Apr 20.

[7] Parascandolo G, Neitz A, Orvieto A, Gresele L, Schölkopf B. Learning explanations that are hard to vary. arXiv preprint arXiv:2009.00329. 2020 Sep 1.

[8] Kalimeris D, Kaplun G, Nakkiran P, Edelman B, Yang T et al. SGD on neural networks learns functions of increasing complexity. Advances in neural information processing systems. 2019;32.

[9] Valle-Perez G, Camargo CQ, Louis AA. Deep learning generalizes because the parameter-function map is biased towards simple functions. arXiv preprint arXiv:1805.08522. 2018

[10] Lake BM, Ullman TD, Tenenbaum JB, Gershman SJ. Building machines that learn and think like people. Behavioral and brain sciences. 2017;40:e253.

[11] Shah H, Tamuly K, Raghunathan A, Jain P, Netrapalli P. The pitfalls of simplicity bias in neural networks. Advances in Neural Information Processing Systems. 2020;33:9573-85.

[12] Mansilla L, Echeveste R, Milone DH, Ferrante E. Domain generalization via gradient surgery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision 2021 (pp. 6630-6638).

[13] Izmailov P, Podoprikhin D, Garipov T, Vetrov D, Wilson AG. Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407. 2018 Mar 14.

[14] Rame A, Dancette C, Cord M. Fishr: Invariant gradient variances for out-of-distribution generalization. In: International Conference on Machine Learning 2022 Jun 28 (pp. 18347-18377). PMLR.

[15] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D et al. Generative adversarial nets. Advances in neural information processing systems. 2014;27.

[16] Gretton A, Borgwardt KM, Rasch MJ, Schölkopf B, Smola A. A kernel two-sample test. The Journal of Machine Learning Research. 2012;13 (1):723-73.

[17] Matsuura T, Harada T. Domain generalization using a mixture of multiple latent domains. In: Proceedings of the AAAI Conference on Artificial Intelligence 2020;34 (7): 11749-11756

[18] Rahman MM, Fookes C, Baktashmotlagh M, Sridharan S. Correlation-aware adversarial domain adaptation and generalization. Pattern Recognition. 2020 ;100:107124.

[19] Arjovsky M, Bottou L, Gulrajani I, Lopez-Paz D. Invariant risk minimization. arXiv preprint arXiv:1907.02893. 2019 Jul 5.

[20] Krueger D, Caballero E, Jacobsen JH, Zhang A, Binas J et al. Out-of-distribution generalization via risk extrapolation (rex). In: International Conference on Machine Learning 2021 Jul 1 (pp. 5815-5826). PMLR.

[21] Fort S, Nowak PK, Jastrzebski S, Narayanan S. Stiffness: a new perspective on generalization in neural networks. arXiv preprint arXiv:1901.09491. 2019 Jan 28.

[22] Jo J, Bengio Y. Measuring the tendency of CNNs to learn surface statistical regularities. arXiv preprint arXiv:1711.11561. 2017 Nov 30.

[23] Geirhos R, Rubisch P, Michaelis C, Bethge M, Wichmann FA et al. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231. 2018 Nov 29.

[24] McCoy RT, Pavlick E, Linzen T. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. arXiv preprint arXiv:1902.01007. 2019.

[25] Oakden-Rayner L, Dunnmon J, Carneiro G, Ré C. Hidden stratification causes clinically meaningful failures in machine learning for medical imaging. In: Proceedings of the ACM conference on health, inference, and learning 2020: 151-159

[26] Huang Z, Wang H, Xing EP, Huang D. Self-challenging improves cross-domain generalization. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16 2020: 124-140. Springer International Publishing.

[27] Koyama M, Yamaguchi S. Out-of-distribution generalization with maximal invariant predictor.

[28] Rame A, Dancette C, Cord M. Fishr: Invariant gradient variances for out-of-distribution generalization. In: International Conference on Machine Learning 2022: 18347-18377. PMLR.

[29] Zhou ZH. Ensemble methods: foundations and algorithms. CRC press; 2012 Jun 6.

[30] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems. 2012;25.

[31] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2016 :770-778.

[32] Shahtalebi S, Gagnon-Audet JC, Laleh T, Faramarzi M, Ahuja K et al. Sand-mask: An enhanced gradient masking strategy for the discovery of invariances in domain generalization. arXiv preprint arXiv:2106.02266. 2021 Jun 4.

[33] He H, Huang G, Yuan Y. Asymmetric valleys: Beyond sharp and flat local minima. Advances in neural information processing systems. 2019;32.

[34] Li D, Yang Y, Song YZ, Hospedales TM. Deeper, broader and artier domain generalization. In: Proceedings of the IEEE International Conference on Computer Vision 2017: 5542-5550.

[35] Fang C, Xu Y, Rockmore DN. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In: Proceedings of the IEEE International Conference on Computer Vision 2013: 1657-1664.

[36] Venkateswara H, Eusebio J, Chakraborty S, Panchanathan S. Deep hashing network for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2017: 5018-5027.

[37] Beery S, Van Horn G, Perona P. Recognition in terra incognita. In: Proceedings of the European Conference on Computer Vision (ECCV) 2018 (pp. 456-473).

[38] Peng X, Bai Q, Xia X, Huang Z, Saenko K et al. Moment matching for multi-source domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision 2019: 1406-1415.

[39] Gulrajani I, Lopez-Paz D. In search of lost domain generalization. arXiv preprint arXiv:2007.01434. 2020 Jul 2.

[40] Russakovsky O, Deng J, Su H, Krause J, Satheesh S et al. Imagenet large scale visual recognition challenge. International journal of computer vision. 2015 ;115:211-52.

[41] Nam H, Lee H, Park J, Yoon W, Yoo D. Reducing domain gap by reducing style bias. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021: 8690-8699.

[42] Nam J, Cha H, Ahn S, Lee J, Shin J. Learning from failure: De-biasing classifier from biased classifier. Advances in Neural Information Processing Systems. 2020;33:20673-84.

[43] Huang G, Li Y, Pleiss G, Liu Z, Hopcroft JE et al. Snapshot ensembles: Train 1, get m for free. arXiv preprint arXiv:1704.00109. 2017 Apr 1.

[44] Zhang W, Jiang J, Shao Y, Cui B. Snapshot boosting: a fast ensemble framework for deep neural networks. Science China Information Sciences. 2020 Jan;63:1-2.

[45] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 2014 ;15 (1):1929-58.

[46] Gal Y, Hron J, Kendall A. Concrete dropout. Advances in neural information processing systems. 2017;30.

[47] Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D. Weight uncertainty in neural network. In: International conference on machine learning 2015 :1613-1622. PMLR.

[48] Ren X, Mi Z, Cai T, Nolte CG, Georgopoulos PG. Flexible Bayesian ensemble machine learning framework for predicting local ozone concentrations. Environmental Science & Technology. 2022;56 (7):3871-83.

[49] He B, Lakshminarayanan B, Teh YW. Bayesian deep ensembles via the neural tangent kernel. Advances in neural information processing systems. 2020;33:1010-22.

[50] Allen-Zhu Z, Li Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. arXiv preprint arXiv:2012.09816. 2020 Dec 17.

[51] Du S, You S, Li X, Wu J, Wang F et al. Agree to disagree: Adaptive ensemble knowledge distillation in gradient space. advances in neural information processing systems. 2020; 33:12345-55.

[52] Chen Y, Wang S, Liu J, Xu X, de Hoog F et al. Improved feature distillation via projector ensemble. Advances in Neural Information Processing Systems. 2022 ;35:12084-95.

[53] Middlehurst M, Large J, Cawley G, Bagnall A. 2021. The temporal dictionary ensemble (TDE) classifier for time series classification. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I (pp. 660-676). Springer International Publishing.