SDT: SPECIFIC DOMAIN TRAINING IN DOMAIN GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Domain generalization (DG) methods aim to achieve generalizability to an unseen target domain by using only training data from the source domains. Although there has been a growing interest to learn from multiple training domains by applying different types of invariance across those domains, the improvements compared to empirical risk minimization (ERM) are almost negligible under controlled evaluation protocols. In this paper, we demonstrate that the disentanglement of spurious and invariant features is a tough task in standard training, since ERM simply minimize the loss and does not exploit invariance among domains. To address the issue, we introduce a simple yet effective method called specific domain training (SDT), which intensifies the trace of spurious features and make them more discernible and exploit masking strategy to decrease their effect. We provide a theoretical and experimental evidence to show the effectiveness of SDT for out-of-distribution generalization. Notably, SDT outperforms previous state of the art Cha et al. (2021) in DomainNet benchmarks 0.2pp in average. Furthermore, SDT improves accuracy of some domains such as Sketch in PACS, SUN09 in VLCS and L100 in TerraIncognita by clear margins 2.5pp, 3.4pp, and 5.4pp respectively.

1 INTRODUCTION

Machine learning algorithms mostly rely on the assumption that training data and test data are sampled from the same distribution. This assumption causes a fundamental problem when data comes from different distributions and domains at training phase as well as new unseen distribution at test time i.e. out of distribution (OoD) data. In a standard machine learning task, we gather a large dataset and shuffle at random train and test data, bringing the test and train distributions closer to each other. By shuffling, we discard the information about different domains and how the distribution of data changes among the environments. We also lose information about stable features and spurious correlations. Empirical risk minimization (ERM) Vapnik (1999) forces the model to exploit all features whether invariant or spurious to reduce the training error and this causes the error in new environments when spurious features vary notably. As an example, consider a classification task of images containing class label camel in which we have two training domains as D1 and D2 with 90$\%$ and 80$\%$ of there's data are camel photos with desert background that means there is a strong but varying correlation between camel class and desert background in train domains. However, in test domain D3, this spurious correlation diminishes and becomes 10$\%$ of the data since the photos are mostly taken in jungle, which results in poor generalization and accuracy on test data. However, if the model could recognize that while the landscapes change with climate (spurious features), the biological characteristics of the animals like humps, neck, legs, and etc remain invariant and use those features to determine the species, we have a much better chance at generalizing to unseen domains.

Neural networks have tendency to learn simple features rather than complex ones even though their predictive power may be less Kalimeris et al. (2019), Valle-Perez et al. (2018). In case of distribution shift, this simplicity bias degrades performance and more complex invariant features are needed to be extracted Lake et al. (2017),Shah et al. (2020). In a standard training, we sample batches from each training domain and by feeding them to the network and calculating the loss, we update model weights with mean of domain-level gradients. In this case, the disentanglement of spurious and invariant features becomes harder, since the variance among the domain-level gradients become small as training proceeds. In order to recognize the spurious features and to let them show off themselves, we exploit a simple yet effective method, which we call specific domain training (SDT). In SDT, we train each domain exclusively for some iterations. By this type of training, weights correspond to spurious features of the specific domain are updated intensely, specially at the beginning of the training interval. However, invariant weights have almost identical update patterns among different domain training intervals. In order to avoid weights to be updated by the spurious gradients, we devise a masking strategy in which update the weights if direction of current training domain gradient is the same as the direction of mean gradients of other training domains. Our masking strategy is more flexible compared to previous works like Parascandolo et al. (2020) that apply AND mask among all training domains or Mansilla et al. (2021), which zeros out the weight component if the gradients have different directions.

Contribution: We present a method named SDT to discriminate the model weights related to spurious and invariant features and use a masking strategy to avoid updating spurious weights. We provide a theoretical and empirical evidence of our hypothesis. Notably, our empirical results on DomainBed benchmark, validates our claim on real world dataset. In particular, we increase the accuracy of PACS, VLCS and TerrraIncognita by 0.3pp and 0.6pp and 1.8pp respectively compared to the previous SOTA Cha et al. (2021). We also improve per domain accuracy of some domains such as Sketch in PACS, SUN09 in VLCS and L100 in TerraIncognita by 2.50pp, 3.4pp, and 5.4pp respectively.

2 Related work

Deep domain generalization methods can be divided generally into three groups: (i) domain alignment (ii) meta learning and (iii) data augmentation.
Domain alignment. is the most intuitive method, which has extensively studied for domain adaption (DA) problems. It aims to learn latent representations that have similar distribution across different domains Sun and Saenko (2016),Li et al. (2018b),Shi et al. (2021),Rame et al. (2021). Ganin et al. (2016) propose Domain Adversarial Neural Networks (DANN), a DA technique which uses generative adversarial networks GANsGoodfellow et al. (2014), to learn a feature representation that matches across training domains. Li et al. (2018b) employ GANs and the maximum mean discrepancy Gretton et al. (2012) to align feature distributions across domains. Matsuura and Harada (2020) exploits clustering techniques to learn domain invariant features even when environments are not explicitly stated. Sun and Saenko (2016) and Rahman et al. (2020) match the feature covariance (second order statistics) across training domains at some level of representation. Arjovsky et al. (2019) propose IRM that learns an intermediate representation such that the optimal classifiers (on top of this representation) of all domains are the same. The motivation is to exploit invariant causal effects between domains while reducing the effect of domain-specific spurious correlations. Krueger et al. (2021) propose an approximation to the IRM problem consisting in reducing the variance of error averages across domains.

Meta learning: In the meta learning category, source domains are separated into two disjoint sets, namely meta-train and meta-validation and a model is optimised on meta-train so as to boost the performance on meta-validation. Li et al. (2018a) employ Model-Agnostic Meta-Learning, or MAML Finn et al. (2017), to build a predictor that learns how to adapt fast between training domains. Dou et al. (2019) use a similar MAML strategy, together with two regularizers that encourage features from different domains to respect inter-class relationships, and be compactly clustered by class labels.

Data augmentation is a common practice to train deep neural networks, e.g. flipping and rotation. However, conventional data augmentation methods can not address the issue of distributional shift and learning-based augmentation strategies are required. Wang et al. (2020) use mixup to blend examples from the different training distributions. Carlucci et al. (2019) constructs an auxiliary classification task aimed at solving jigsaw puzzles of image patches. Wang et al. (2019) remove textural information from images to focus on shape based information, so improve domain generalization.

A recent line of work aligns the gradients of loss with respect to model weight θ rather than aligning features among domains. First, having similar domain-level gradient distributions is critical so that the DNN has shared properties across domains. Second, gradients are more expressive and richer than features. Specifically, gradients were shown to better cluster semantically close inputs Fort et al. (2019). Several works such as Jo and Bengio (2017), Geirhos et al. (2018), McCoy et al. (2019) and

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%	$[3.4, 3.0, 3.0, -0.1]$

Oakden-Rayner et al. (2020) demonstrate that NNs tend to learn spurious features and low-level statistical patterns rather than semantic features and high-level abstractions. So gradient alignment could partially address the issue of poor out of domain performance. Huang et al. (2020) discards the representations associated with the higher gradients at each epoch, and forces the model to predict with remaining information. However, it does not utilize any knowledge of partitions of source domains. On the contrary, SDT encourages the model to advertently intensify the spurious gradients and then avoid the weight updates correspond to those spurious gradients. Koyama and Yamaguchi (2020), Parascandolo et al. (2020), Shi et al. (2021), Rame et al. (2022) try to find a shared minima among domains by tackling the domain-level gradients. Specifically, when we have domain set $E = \{A, B\}$, IGA Koyama and Yamaguchi (2020) minimizes $\| g_A - g_B \|_2^2$; Fishr matches the domain-level gradient covariances, i.e., the second moment of the gradient distributions. AND mask Parascandolo et al. (2020) update weights only when g_A and g_B point to 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 Shi et al. (2021) increase $g_A.g_B$. In section 4.3 we show that SDT also increase the gradient inner products.

3 Methodology

Consider training dataset D_{tr} containing K domains $\{D_1, D_2, ..., D_K\}$ with each domain contains samples $\{(x_i, y_i)\}_{i=1}^{n_k}$ and test dataset D_{te} with T domains $\{D_{K+1}, ..., D_{K+T}\}$ all drawn i.i.d from some probability distribution where $D_{tr} \cap D_{te} = \emptyset$. The aim of domain generalization is to train weights θ such that generalize well on unseen test dataset D_{te}. ERM approach trains θ on D_{tr} such that:

$$L_{ERM}(D_{tr}; \theta) = E_{D \sim D_{tr}} E_{(x,y) \sim D} l((x, y); \theta), \tag{1}$$

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

The ERM objective does not consider the invariances among different domains and only minimizes the loss of training dataset. In the following, we give a simple linear example adopted from Shi et al. (2021) to show the reluctance of ERM to invariant features. Then, we observe the behavior of weight gradients corresponding to invariant and spurious features while training ERM and SDT distinctly.

3.1 ERM reluctance to invariant features: a linear example

Consider a binary classification where data $(x, y) \in B^4 \times B$ and a data instance is $x = [f_1, f_2, f_3, f_4]$, y. Training datasets are D_1 and D_2 and test domain is D_3. A linear model is $Wx + b = y, W \in R^4, b \in R$ is trained on the train dataset and tested on test domain. 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 the 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. For example, using only f_2 attains 97% on $D1$. However, using only f_1 gains 93% on D_1.

The poor performance of ERM on test domain has been shown in Table 1. As illustrated in the table for W parameter, ERM puts higher weight for spurious features f_2 and f_3, while invarient feature f_1 receives less weight.

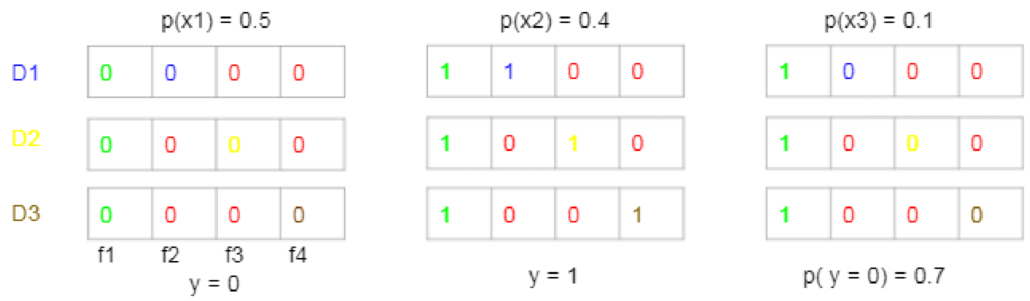


Figure 1: There are 3 kinds of data as x_1, x_2 and x_3, each shown in one column. First column contains data $x_1 = [0, 0, 0, 0]$ and $y = 0$ for all domains and . In second column x_2 changes for each domain, y is always 1. In third column, $x_3 = [1, 0, 0, 0]$ and $y = 1$ for 30% of data of type 3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

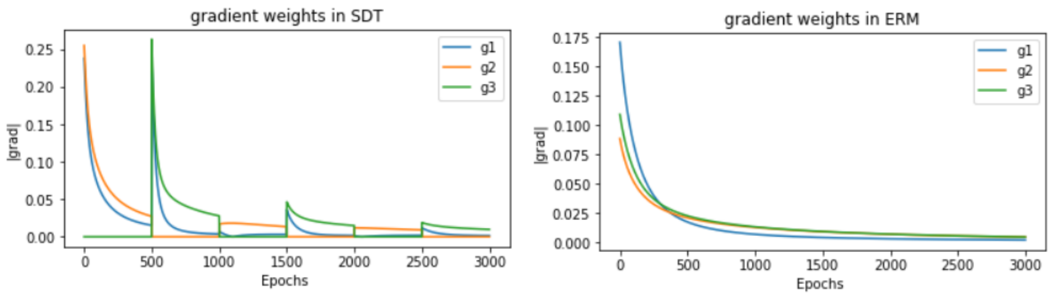


Figure 2: Absolute gradient of loss w.r.t $W = [w1, w2, w3]$. In left plot, we apply SDT training with 500 epochs for each training domain $\{D_1, D_2\}$ and repeat it till the end of training.

3.2 Domain specific training for disentangling spurious and invariant features

In order to mitigate ERM pitfall in learning spurious features, we introduce a simple yet effective method to discriminate spurious and invariant features and then we apply mechanisms to strengthen the weights correspond to invariant features and weaken the weights correspond to spurious features. In this regard, we train one domain specifically for some iterations, then switch to the next domain and continue this process till the end of training. Intuitively, when we train one domain for some iterations, weights correspond to spurious features of that domain become stronger (absolute value of the weight increase), however, when we switch to the next domain, the absolute value of the aforementioned weights stop increasing and may start to decrease as well. We validate our claim by applying SDT (specific domain training) on the previous linear example and trace the gradients of corresponding weights as an indicator of change for each weight at each step of training. As presented in Figure 2, in SDT training, spurious features have the largest gradients at each training interval. For example at interval $[0, 500]$ we train D_1, so the gradient w.r.t feature f_2, which is a highly predictive spurious feature for D_1 has the largest gradient value. for the second interval, f_3, spurious for D_2 is the highest value. On the contrary, in ERM training, gradients for all weights change smoothly and can not be discriminated.

3.3 The spurious and invariant features Model

We develop a framework, that helps us to rigorously refer to "invariant" and "spurious" features. In particular, we present a set of definitions which enable us to formally describe our setup and theoretical results.

Setup. We consider binary classification, where $(x, y) \in X \times \{\pm 1\}$ are sampled from $D_{tr} = \{D_1, D_2, ..., D_S\}$. The goal is to learn a classifier $C : X \to \{\pm 1\}$ which predicts a label y given an input x.

Feature is defined as a function mapping from the input space X to the real numbers, with the set of all features thus being $F = \{f : X \to R\}$. Features in F are considered to be shifted/scaled to be mean-zero and unit-variance (i.e., so that $E_{D \sim D_{tr}} E_{(x,y) \sim D}[f(x)] = 0$ and $E_{D \sim D_{tr}} E_{(x,y) \sim D}[f(x)^2] = 1$), in order to make the following definitions scale-invariant.

ρ-spurious features: For a given distribution D_s, we call a feature f ρ-spurious, if it is correlated with the true label such that

$$E_{(x,y) \sim D_s}[y.f(x)] = \rho \ \wedge \ \forall D_i \in \overline{\mathrm{D_s}}, E_{(x,y) \sim D_i}[y.f(x)] < \rho. \tag{2}$$

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

Standard Training. We consider a linear classifier which includes a set of features F, weight vector W and an scalar bias b. Training a linear classifier is performed by minimizing a loss function via ERM that decreases with the correlation between the weighted combination of the features and the label. We use a simple loss function in our equation, however, more practical losses such as logistical or hinge can be used instead.

$$L_\Theta(x, y) = -E_{D \sim D_{tr}} E_{(x,y) \sim D}[y.(b + \sum_{f \in F} w_f.f(x))]. \tag{3}$$

Theorem 1 *Consider $D_{tr} = \{D_1, D_2\}$ and feature f is ρ_1 spurious feature for D_1 i.e. $E_{(x,y) \sim D_1}[y.f(x)] = \rho_1$ and assume $E_{(x,y) \sim D_2}[y.f(x)] = \rho_2$, where $\rho_1 > \rho_2$ and w_f is corresponding weight for feature f in our linear classifier. Then $E_{(x,y) \sim D_1 \cup D_2}[\frac{\partial L}{\partial w_f}] = -y.f(x) = -\frac{\rho_1 + \rho_2}{2}$.*

As a result of the Theorem 1, when we train the model by ERM, at each step $\frac{\partial L}{\partial w_f} = -\frac{\rho_1 + \rho_2}{2}$, which is constant. However, By applying SDT with one step interval, at each step we switch between $\{D_1, D_2\}$. So the corresponding gradient value swings between ρ_1 and ρ_2, hence the gap for spurious gradients in SDT is $|\rho_1 - \rho_2|$ and 0 for ERM at each training step. However, experiments indicate that using more practical loss functions such as cross entropy with longer interval steps increase the gradient gap corresponding to spurious features, notably, at the initial steps of domain switches.

Masking strategy for spurious weights: By disentangling spurious and invariant weights applying SDT, We exploit methods to strengthen invariant weights and diminish spurious weights. Consider our current training domain in SDT is s, then we show the rest of domains with $\overline{s}$. At each training step, we calculate mean loss gradient w.r.t corresponding weight component j (we apply this method only on final linear classifier) for s and $\overline{s}$ i.e. $[\nabla L_s]_j$ and $\frac{1}{|\overline{s}|} \sum_{e \in \overline{s}} [\nabla L_e]_j$. Weight component j is called invariant, if the two aforementioned gradients have the same sign, otherwise, it is a spurious weight. We take two strategies to deal with spurious and invariant weights. For invariant weights, we update them by the gradient calculated for domain s that is $[\nabla L_s]_j$. For spurious weights, some previous works like Parascandolo et al. (2020) don't update them or Mansilla et al. (2021) zero out the weight, which is a restrictive method and may cause the weight not to receive any gradients in further updates, hence leads the network to stuck and poor performance. To address this problem, we update the weight with the gradient $[\nabla L_s]_j$ in a direction which makes the weight to be closer to zero, that is $|\Theta_j^{t+1}| <= |\Theta_j^t|$, which reduces the effect of spurious weight Θ_j in the classification.

Stochastic weight averaging: The SWA method Izmailov et al. (2018) is based on averaging model weights θ along the trajectory of SGD with cyclical or constant learning rates. Empirical results prove that SWA finds better flat minima and hence better generalization by approximating ensembles of model weights in SGD trajectory. It is known that finding a flatter minima guarantees better generalization performance He et al. (2019), So it has been useful in domain generalization tasks. We apply SWA in our model training in order to avoid instability of training when each domain is trained solely and to get better generalization as well. Given model weight space $\Omega = \{\omega_0, \omega_1, ..., \omega_N\}$, where N is the number of training steps. We start sampling weights at some initial step m and proceed it for each future steps. In any step t, if the loss is higher than some threshold τ, then ω_t is not included in ω_{swa}.

4 Experiments

4.1 Datasets and Settings

Benchmark dataset: We evaluate our method on five famous benchmarks on domain generalization task and compare it's results with other state of the art in DG. The datasets are PACS Li et al. (2017) (9,991 images, 7 classes, and domains), VLCS Fang et al. (2013) (10,729 images, 5 classes, and 4 domains), OfficeHome Venkateswara et al. (2017) (15,588 images, 65 classes, and 4 domains), TerraIncognita Beery et al. (2018) (24,788 images, 10 classes, and 4 domains), and DomainNet Peng et al. (2019) (586,575 images, 345 classes, and 6 domains). For all datasets, as in Gulrajani and Lopez-Paz Gulrajani and Lopez-Paz (2020), we train using the following data augmentations: crops of random size and aspect ratio, resizing to 224 × 224 pixels, random horizontal flips, random color jitter, grayscaling the image with 10% probability, and normalization using the ImageNet channel means and standard deviations.

Evaluation protocols: For a fair comparison among DG methods, we follow the training and evaluation rules demonstrated in Gulrajani and Lopez-Paz Gulrajani and Lopez-Paz (2020), including data augmentation, hyperparameter search and dataset splits. However, it's evaluation protocol is computationally too expensive. Hence, we reduce the search space of our method for computational efficiency following the protocols illustrated in SWAD Cha et al. (2021). All performance scores are evaluated by leave-one-out cross-validation. We choose one domain as the target(test) domain and other remaining domains as training domains where 80% of it used for training and 20% used for validation and model selection.

Implementation details: For weight initialization, ImageNet Russakovsky et al. (2015) trained ResNet-50 He et al. (2016) is employed as the initial weight and batch normalization statistics are frozen during training. For optimization, we exploit Adam Kingma and Ba (2014) with learning rate 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.

4.2 Main results

We report out-of-domain accuracies for each domain i.e., a model is trained and validated on training domains and evaluated on the unseen target domain. For each domain, We train the model three times with random data splits and report the average test results. We borrow the results of the Table 2 from SWAD Cha et al. (2021), in which the results for all other methods acquired by training with resnet-50 backbone with the same training and validation protocols as described above. Domain-specific training outperforms SWAD in PACS, VLCS and TerraIncognita by 0.5pp and 0.5pp and 1.8pp respectively and for the remaining datasets it's lower than SWAD by small margins. In the following, we compare our method with DG methods in each domain of the datasets.

Per domain comparison with SWAD: As presented in Tables 3, 4 and 5, Our method outperforms SWAD in some domains by a magnificent margin. In PACS dataset, for sketch domain, we get 3pp performance gain, In VLCS dataset, for Sun09 domain, we get 3.5pp increase in accuracy and In TerraIncognita, L100 domain we achieve 5.4pp performance gain. We conjecture that for these test domains our method avoid higher reliance on one train domain with spurious features and elicit more invariant features among all training domains.

Table 2: Out-of-domain accuracies of domain specific training (ours) with other DG methods on five benchmarks.

Algorithm	PACS	VLCS	HomeOffice	TerraInc	DomainNet	Avg.
ERM	85.5	77.5	66.5	46.1	40.9	63.3
CORAL	86.2	78.8	68.7	47.7	41.5	64.5
SagNet	86.3	77.8	68.1	48.6	40.3	64.2
SWAD	88.1	79.1	**70.6**	50.0	**46.5**	66.9
Ours	**88.6**	**79.6**	70.2	**51.8**	45.4	**67.1**

Table 3: Out-of-domain accuracies of SDT and DG methods for PACS (left) and VLCS (right).

Algorithm	Art	Cartoon	Photo	Sketch	Avg.
ERM	84.7	80.8	97.2	79.3	85.5
CORAL	88.3	80.0	97.5	78.8	86.2
SagNet	87.4	80.7	97.1	80.0	86.3
SWAD	**89.3**	**83.4**	97.3	82.5	88.1
SDT	**89.3**±0.4	83.2±0.3	97.2±0.2	**84.6**±0.4	**88.6**

Algorithm	Caltech	LabelMe	Sun09	Voc2007	Avg.
ERM	97.7	64.3	73.4	74.6	77.5
CORAL	98.3	**66.1**	73.4	77.5	78.8
SagNet	97.9	64.5	71.4	77.5	77.8
SWAD	**98.8**	63.3	75.3	**79.2**	79.1
SDT	97.6±0.1	63.3±0.7	**78.7**±0.5	78.8±0.6	**79.6**

4.3 Analysis

In-domain losses: SDT does not decrease in-domain losses, however, its out-of-domain loss improves. In Figure 3, we show the validation losses of training on PACS dataset with both ERM and SDT. Sketch is test domain while Art, Cartoon and Photo are training domains. As shown in Figure 3 in-domain losses of SDT are higher than ERM on all the domains, however, the test loss has been dropped in our method. This means that SDT elicit more invariant features across domains. The reason is that by specific domain training, the model avoids overfitting in training domains and equipped with masking strategy it tries to learn more invariant features across domains.

Variances among domain gradients: As demonstrated in linear example and theoretical evidence above, SDT is better than standard training in disentangling invariant and spurious features. We investigate this in real world PACS dataset. As presented in Figure 3, domain gradients variances in SDT are higher than ERM. We conjecture that the more domain gradient varinace for a weight component $[\theta]_j$, the higher probability that it is a spurious weight. Because higher variance indicates that domain gradients are not similar to each other either in direction or magnitude. In ERM, domain gradient variances of weight components are too small for all weights which makes the discrimination of spurious and invariant weights hard.

Gradients inner product: In Shi et al. (2021), they proposed an algorithm called Fish in which they provide a theoretical and experimental evidence that Fish matches the domain-level gradients. In a nutshell, Consider θ as current model weight and $\theta^{'}$ as a copy of it. In each pass, Fish samples mini-batches from each training domains and updates $\theta^{'}$ one by one for each training domain. After one pass and sampling from all training domains, It updates the original model weight $\theta = \theta + \epsilon(\theta - \theta^{'})$. SDT is similar to Fish when training domain interval and ϵ both are set to 1. We compare the inter-domain gradient inner products for both ERM and SDT. Gradient inner product (GIP) is calculated by sum of inner products for any two training domains in classifier layer i.e. $\sum_{i,j \in S}^{i \neq j} G_i . G_j$. As shown in Figure 3, SDT has higher GIP compared to ERM through the training epochs. It demonstrates that SDT as Fish matches the domain-level gradients, as another clue that SDT extracting more invariant features compared to ERM.

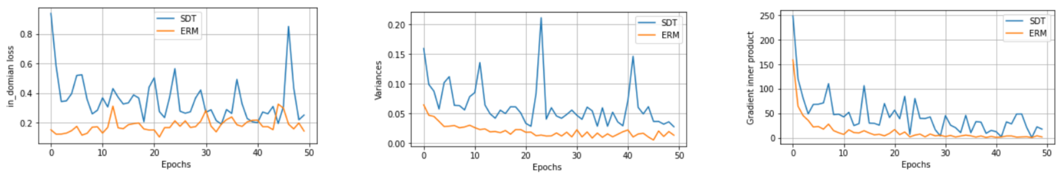
Table 4: Out-of-domain accuracies of SDT and DG methods for OfficeHome (left) and TerraInc (right).

Algorithm	Art	Clipart	Product	Photo	Avg.
ERM	61.3	52.4	75.8	76.6	66.5
CORAL	65.3	54.4	76.5	78.4	68.7
SagNet	63.4	54.8	75.8	78.3	68.1
SWAD	**66.1**	57.7	**78.4**	**80.2**	**70.6**
SDT	65.2±0.4	**58.5** ±0.5	77.6±0.1	79.5 ±0.3	70.2

Algorithm	L100	L38	L43	L46	Avg.
ERM	54.3	42.5	55.6	38.8	47.8
CORAL	51.6	42.2	57.0	39.8	47.7
SagNet	53.0	43.0	57.9	40.4	48.6
SWAD	55.4	44.9	**59.7**	39.9	50.0
SDT	**60.8**±0.2	**46.1**±0.7	58.5±0.3	**41.8**±0.3	**51.8**

Table 5: Out-of-domain accuracies of SDT and DG methods for DomainNet.

Algorithm	Clip	Info	Paint	Quick	Real	Sketch	Avg.
ERM	63.0	21.2	50.1	13.9	63.7	52.0	44.0
CORAL	59.2	19.7	46.6	13.4	59.8	50.1	41.5
SagNet	57.7	19.0	45.3	12.7	58.1	48.8	40.3
SWAD	**66.0**	**22.4**	**53.5**	**16.1**	**65.8**	**55.5**	**46.5**
SDT	64.5±0.2	22.3±0.4	52.2±0.1	14.2±0.3	63.9±0.4	55.3±0.2	45.4



(a) In-domain loss (b) Domain gradient variances (c) Inner gradient products

Figure 3: Comparison of SDT and ERM in PACS dataset for illustrated statistics in 4.3. Here (Art, Cartoon, Photo) are train domains and sketch is test domain. In-domain loss is the average validation loss of training domains. Gradient variances and gradient inner products are calculated for the weight components in the classifier layer.

4.4 Ablation Study

In Table 6, we compare ERM and variants of SDT 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 rising the out of domain accuracy. Specifically, it increase the accruacy abut 3%. Masking also increase the accuracy about 0.3% 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 outperform masking as the results in Table 6 validate this claim.

SDT intervals: First, we study the effect of domain training intervals on out-of-domain accuracies of PACS dataset. As demonstrated in Figure 4 we present the accuracies for intervals $\{50,100,200,300,400\}$. We also do an experiment with the mixture of intervals where we start with 100 iterations per domain at the beginning of training and increase the intervals as training proceeds. Mixture of intervals results in the highest accuracy in our experiments. We postulate that at the beginning of training as stated in Nam et al. (2020), the model learns easy features then it learns hard features as far as possible. So, at the beginning we train with little iterations for each domain and increase the interval in order to force the network learn harder features.

Averaging start iteration: we analyze the effect of starting iteration of weight averaging in domain accuracies. In this regard, we set the start averaging iteration to $\{0,1000,2000,3000,4000\}$. In this experiment on PACS dataset, our total iterations is 6000. In Figure 4 (left), the accuracies has been shown.

SWA acceptance threshold: We investigate how domain generalization performance is impacted by the choice of swa acceptance threshold. In all of our experiments, we assume that if half of the training domains satisfy the threshold, then current model weight will be averaged by the final swa model. We train the model with thresholds $\{0, 0.05, 0.1, 0.15, 0.2\}$ and show the results in Figure

Table 6: Out-of-domain accuracies of SDT with the component variants for 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±0.5	88.6
SDT + SWA	89.4±0.4	83.3±0.3	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	84.6±1.2	81.1±1.1	96.8±0.3	80.1±1.4	85.6

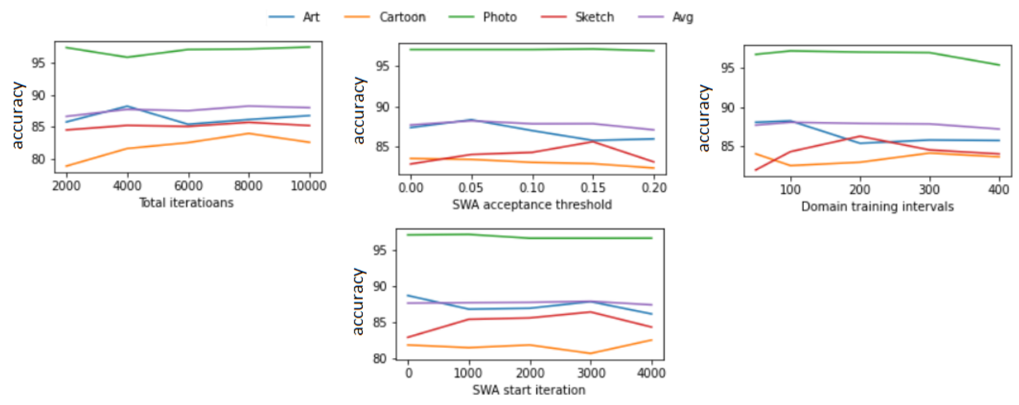


Figure 4: Out-of-Domain accuracy for PACS dataset with varying model hyperparameters.

4. 0 threshold means that current model will be averaged into swa model without any constraint on validation losses.

5 Discussion and Limitations

SDT slightly degrades performance for some domains. Compared to SWAD Cha et al. (2021), some domain accuracies slightly degrade in SDT as shown in Tables 3, 4 and 5. In such cases, we postulate that test domain features are more correlated to a dominant train domain (a domain which has more predictive easy features to be learnt). Hence when SDT applied in these cases, the dominant domain has less contribution in training compared to ERM, and consequently the test accuracy drops. However, we think that this drop in accuracy is not because SDT does not learn invariant features. On the contrary, since SDT is learning more invariant features and becuase spurious features of test and dominant domain are highly correlated, such degradation in performance occurs.

SDT needs more iterations to converge. Training each domain exclusively at each interval causes the model to diverge from optimal minima in loss landscape for some intervals hence convergence needs more iterations. On the other hand, by SDT we seek more expansive areas in loss landscape and by using swa method we may find more flatter minima among those areas. However, the issue of late convergence become more extreme when the dataset is bigger and also the number of domains increase. As an example, in DomainNet dataset, we increase the number of iterations form 15000 to 25000, however, the performance although higher than ERM but still lower than SWAD Cha et al. (2021) in all domains as presented in Table 5.

Pure SDT sometimes diverges: When SDT is applied without masking mechanism or weight averaging, it causes the model to diverge for some training domains. The reason is that these domains have high easy and predictive spurious features and by learning them, the loss for other domains soar and cause the model to diverge. However, by exploiting masking strategy we update the invariant weights only and hence the loss of other domains does not change sharply. SWA also helps the issue by averaging the weights for different domain. Specifically, by SWA, invariant weights are updated for all training intervals. However, spurious weights are updated only for a specific domain, so by averaging all the weights in different intervals, spurious weights become less effective.

6 Conclusion

In this paper, we investigate the spurious and invariant features disentanglement in the presence of multiple domains. We claim that standard training doesn't extract invariant features properly since it focuses on only minimizing the loss and neglects the invariance among domains. To alleviate the issue, we propose SDT in which trains the domains exclusively for some interval. We theoretically and empirically observe that SDT discerns the spurious features better and using masking strategy further avoids the model from learning these spurious features. Notably, SDT achieves comparable results to SWAD Cha et al. (2021), current SOTA, in DomainBed benchmarks.

REFERENCES

M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018.

F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

J. Cha, S. Chun, K. Lee, H.-C. Cho, S. Park, Y. Lee, and S. Park. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34, 2021.

Q. Dou, D. Coelho de Castro, K. Kamnitsas, and B. Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in Neural Information Processing Systems*, 32, 2019.

C. Fang, Y. Xu, and D. N. Rockmore. 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*, pages 1657–1664, 2013.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

S. Fort, P. K. Nowak, S. Jastrzebski, and S. Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

I. Gulrajani and D. Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

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

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

Z. Huang, H. Wang, E. P. Xing, and D. Huang. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision*, pages 124–140. Springer, 2020.

P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

J. Jo and Y. Bengio. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017.

D. Kalimeris, G. Kaplun, P. Nakkiran, B. Edelman, T. Yang, B. Barak, and H. Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

M. Koyama and S. Yamaguchi. Out-of-distribution generalization with maximal invariant predictor. 2020.

D. Krueger, E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. Le Priol, and A. Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

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

D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.

H. Li, S. J. Pan, S. Wang, and A. C. Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5400–5409, 2018b.

L. Mansilla, R. Echeveste, D. H. Milone, and E. Ferrante. Domain generalization via gradient surgery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6630–6638, 2021.

T. Matsuura and T. Harada. Domain generalization using a mixture of multiple latent domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11749–11756, 2020.

R. T. McCoy, E. Pavlick, and T. Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*, 2019.

J. Nam, H. Cha, S. Ahn, J. Lee, and J. Shin. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33:20673–20684, 2020.

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

G. Parascandolo, A. Neitz, A. Orvieto, L. Gresele, and B. Schölkopf. Learning explanations that are hard to vary. *arXiv preprint arXiv:2009.00329*, 2020.

X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019.

M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan. Correlation-aware adversarial domain adaptation and generalization. *Pattern Recognition*, 100:107124, 2020.

A. Rame, C. Dancette, and M. Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. *arXiv preprint arXiv:2109.02934*, 2021.

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

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

Y. Shi, J. Seely, P. H. Torr, N. Siddharth, A. Hannun, N. Usunier, and G. Synnaeve. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937*, 2021.

B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

G. Valle-Perez, C. Q. Camargo, and A. A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018.

V. N. Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10 (5):988–999, 1999.

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

H. Wang, Z. He, Z. C. Lipton, and E. P. Xing. Learning robust representations by projecting superficial statistics out. *arXiv preprint arXiv:1903.06256*, 2019.

Y. Wang, H. Li, and A. C. Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3622–3626. IEEE, 2020.

7 APPENDIX

You may include other additional sections here.