Gradient-Guided Annealing for Domain Generalization

Aristotelis Ballas
Dpt of Informatics and Telematics
Harokopio University of Athens
Omirou 9, Tavros, Athens, Greece
aballas@hua.gr

Christos Diou
Dpt of Informatics and Telematics
Harokopio Univesity of Athens
Omirou 9, Tavros, Athens, Greece
cdiou@hua.gr

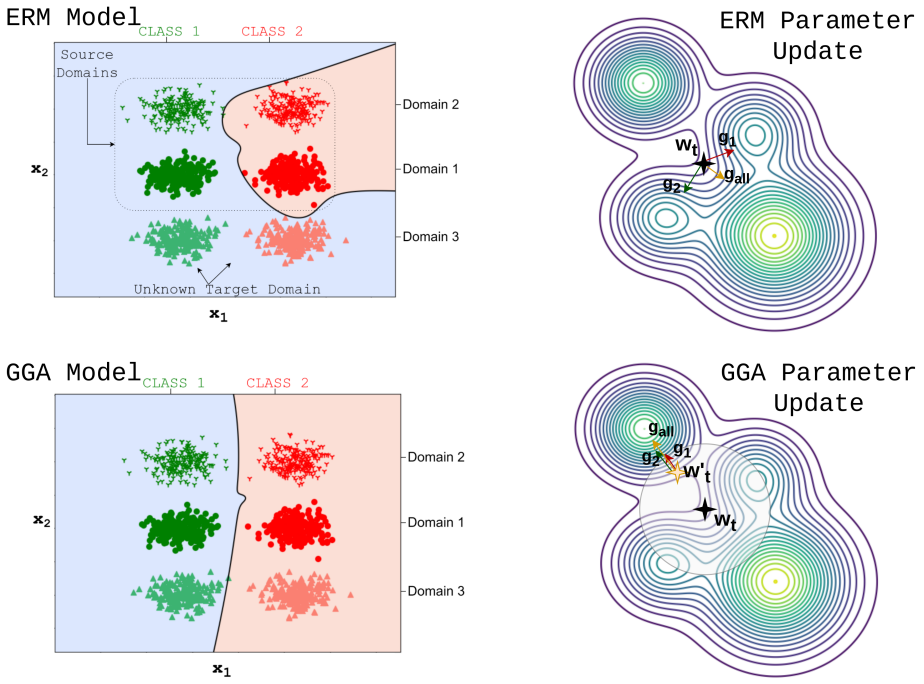

Figure 1. *(left) Decision boundaries of a 4^{th}-degree polynomial logistic regression model with 2D input. In this example, feature x_1 is class-specific and x_2 is domain-specific, while color represents classes and shapes represent domains. The samples with solid red and green colors are included in the training data, whereas the fainted samples are part of the hidden held-out test set. As a result, domain shift is represented by a change in x_2. Although the classifier should only infer based on x_1, traditional gradient descent leads to overfitting (top-left). The proposed method, GGA (bottom-left), introduces an annealing process that depends on gradient agreement, leading to models that generalize well to new, unobserved target domains. (right) Schematics of the parameter updates of ERM (top-right) and GGA (bottom-right). Parameters updated via ERM are driven by gradient conflict, whereas GGA searches for a point where gradients align before continuing descending towards a minima.*

Abstract

Domain Generalization (DG) research has gained considerable traction as of late, since the ability to generalize to unseen data distributions is a requirement that eludes even state-of-the-art training algorithms. In this paper we observe that the initial iterations of model training play a key role in domain generalization effectiveness, since the loss landscape may be significantly different across the training and test distributions, contrary to the case of i.i.d. data. Conflicts between gradients of the loss components of each domain lead the optimization procedure to undesirable local minima that do not capture the domain-invariant features of the target classes. We propose alleviating domain conflicts in model optimization, by iteratively annealing the parameters of a model in the early stages of training and searching for points where gradients align between domains. By discovering a set of parameter values where gradients are updated towards the same direction for each data distribution present in the training set, the proposed

Gradient-Guided Annealing (GGA) algorithm encourages models to seek out minima that exhibit improved robustness against domain shifts. The efficacy of GGA is evaluated on five widely accepted and challenging image classification domain generalization benchmarks, where its use alone is able to establish highly competitive or even state-of-the-art performance. Moreover, when combined with previously proposed domain-generalization algorithms it is able to consistently improve their effectiveness by significant margins[1].

1. Introduction

The vast majority of neural networks today are trained via stochastic gradient descent methods, such as SGD [7] or ADAM [25], where the gradient direction guides the optimization through the loss landscape, aiming to converge to a minimum. What is more, it has been empirically shown that the astounding performance and generalization capabilities of these over-parameterized models stems from the fact that loss surfaces of large neural networks have multitudinous local minima [2, 42], most of which yield similar performance upon model convergence [12, 37].

The above optimization process assumes that all available data samples are independent and identically distributed (i.i.d.). When this assumption holds, the parameter values reached during training is likely to generalize to the test distribution. However, in several real-world scenarios the i.i.d. assumption is not met, and models are evaluated on similar but distinct out-of-distribution (OOD) data resulting from domain shifts to the training distribution, leading to a domain generalization (DG) [60] problem. In this case, the training loss minimum may lead to a poor parameter configuration for the test data.

The disagreement of gradients across data from different domains during training, can provide an indication that the described problem occurs. This observation about gradient conflicts was initially made in the multi-task learning paradigm [58], where gradients of different tasks pointed to conflicting directions on the loss surface. To mitigate the effects of gradient conflicts, methods to balance the relative gradient magnitudes were proposed [11, 23], along with algorithms that remove disagreeing gradient components among tasks [45, 55]. Similarly, gradient conflicts were also addressed in the context of DG in [34], where gradients with different signs among domains were either muted or set to random values; as they are assumed to contain domain-specific information. With that being said, it has been empirically shown that even though the above issues arise, the correct selection and tuning of hyperparameters can be enough for vanilla network training to surpass even state-of-the-art algorithms [19]. This evidence leads us to believe that there exist local minima on the vanilla loss surface that lead to more robust and generalizable models. Depending on the starting parameter configuration, even a model optimized by traditional Empirical Risk Minimization can reach a solution that is locally optimal for all distinct domains. These hypotheses, along with problems poised by gradient disagreement, are demonstrated in Section 1.1 (illustrated in Fig. 1) and are further explored in Section 3.2.

Starting from these ideas, we propose an alternative strategy for updating the parameters of a neural network during the optimization process, in an attempt to "*set the model up for success*". Specifically, inspired by Simulated Annealing Optimization [26], during the early stages of model training we iteratively anneal, i.e. randomly perturb the parameters of the model, and search for a set of parameter values where the gradients between all training domains agree, before minimizing the total loss across all training domains. We call this simple, yet effective, strategy *Gradient-Guided Annealing* or *GGA*. When evaluated on extensive and challenging DG benchmarks, GGA is able to boost the performance of the baseline by significant margins, even yielding state-of-the-art results without the application of additional data processing or augmentation techniques. Additionally, since GGA can be considered a general strategy for handling multi-domain datasets it can also be combined with previously proposed algorithms. Experimental results of the combined methods, demonstrate the efficacy of GGA, as it is able to enhance their generalization capacity and overall accuracy, boosting their performance over the baseline benchmarks.

Our primary contributions are as follows:

- We present Gradient-Guided Annealing (GGA), a DG method for training neural networks such that gradients align across domains.
- We validate the effectiveness of GGA on multiple, challenging Domain Generalization benchmarks both as a standalone algorithm and by combining it with previous state-of-the-art methods.
- We offer further evidence on the effectiveness of the proposed method by investigating the domain gradients during training and its sensitivity to the choice of hyperparameters.

1.1. Gradient Disagreement and Domain Overfitting

To demonstrate how the domain generalization problem manifests during optimization in the source domains during training, we consider the following simple synthetic binary classification problem. For training, data is sampled from a mixture of two domains, while testing takes place on a different, previously unseen, target domain. Each sample

[1]Code available at: https://github.com/aristotelisballas/GGA

has two features, a class-specific feature x_1 and a domain-specific feature x_2. Our goal is to learn a model on the source domains, that can discriminate between classes effectively on the unseen target domain.

Concretely, for the source domains, we draw 200 points from a 2-D Gaussian distribution with an isotropic covariance matrix for each domain and class, i.e.,

$$\boldsymbol{x}_y^{(d)} \sim \mathcal{N}(\boldsymbol{\mu}_y^{(d)}, \sigma \boldsymbol{I}_2) \tag{1}$$

Where the subscript y indicates the class and the exponent d the domain, while $\boldsymbol{\mu}_1^{(1)} = [-2.5, -2.5]$, $\boldsymbol{\mu}_1^{(2)} = [-2.5, 2.5]$, $\boldsymbol{\mu}_2^{(1)} = [2.5, -2.5]$, $\boldsymbol{\mu}_2^{(2)} = [2.5, 2.5]$, $\sigma = 0.5$ and $\boldsymbol{I}_2$ is the 2×2 identity matrix. We also draw an additional 400 samples from a held-out test domain with means at $\boldsymbol{\mu}_1^{(3)} = [-2.5, -7.5]$ and $\boldsymbol{\mu}_2^{(3)} = [2.5, -7.5]$. The drawn samples are shown in the left column of Fig. 1. In this example, the classes can be distinguished solely on x_1 (the "class-specific" feature), while domains differ in terms of feature x_2 (the "domain-specific" feature).

We train a 4$^{\text{th}}$ degree polynomial logistic regression model trained with Empirical Risk Minimization (ERM, [49]) and binary cross-entropy loss using the SGD optimizer. In the top-left example of Fig. 1 the initial parameter conditions were such that training converged to a local minimum of the loss that leads the model to consider both x_1 and x_2 for its decisions. In this case, the model has clearly overfit its source domains and will fail when presented with out-of-distribution data from the held-out domain. An indicator of this was the fact that gradients of the loss for samples of different domains were dissimilar during training. This is in contrast to the bottom-left GGA model in Fig. 1 which mostly relies on x_1 to discriminate between classes. This model was trained by using the proposed gradient agreement strategy and although the resulting training loss is slightly higher, the model successfully generalizes to new domains.

In the rest of the paper, we first discuss the most relevant works in the domain generalization literature (Section 2) and then present the proposed methodology (Section 3). Followingly, we present the experimental setup and results (Section 4), and finally conclude the paper with a discussion on limitations of our method and directions for future research (Section 5).

2. Related Work

Domain Generalization (DG) methods focus on learning a model from one or multiple *source* data sets, or domains, which can generalize to previously unseen, out-of-distribution *target* domains. Existing DG methods in the literature can be categorized into two major groups; single-source and multi-source. In addition to not having any knowledge about the unseen data, single-source algorithms do not leverage information regarding the presence of distinct domains in the training set. On the other hand, multi-source methods utilize domain labels and often take advantage of the statistical differences in the sample distributions. Specifically, most popular algorithms include data augmentation [9, 59] which proves beneficial for regularizing over-parameterized neural networks and improving generalization, meta-learning [3, 14, 30, 57], which exposes models to domain shifts during training, and disentangled representation learning [4, 39, 51, 56], where models most commonly include modules or architectures that focus on decomposing learned representations into domain-specific and domain-invariant parts. Additionally, domain alignment [17, 35, 53] and causal representation learning algorithms [32, 33] have also been proposed in the literature towards producing robust models that retain their generalization capabilities on unseen data. Finally, ensemble learning methods [61] have also been proposed for DG, where techniques such as weight averaging [22] lead to improved generalization [10].

Gradient operations for DG. Lately, there has been a surging interest in addressing the DG problem from a gradient-aware perspective. The most relevant works to ours, leverage gradient information to learn generalized representations from the source datasets. For example, [21] proposes a self-challenging learning scheme, by muting the feature representations associated with the highest gradients and forcing the network to learn via alternative routes. In another work, the authors of [47] propose Fish, a first-order algorithm that approximates the optimization of the gradient inner product between domains. Inspired by gradient surgery in multi-task learning [55], [34] proposes aligning the gradient vectors among source domains by retaining the individual same-sign gradients, and either setting the rest to zero or random values. Finally, there has also been great interest into researching the properties of Sharpness-Aware Minimization (SAM) [16, 28, 52, 62], as it has been hypothesized that flatter minima lead to smaller DG gaps and improved generalization.

Simulated Annealing for Deep Learning. Although explored in past literature, simulated annealing (SA) has not been explicitly proposed for DG. To avoid being "trapped" in local minima during model optimization, [8] proposes SA-GD, a simulated annealing method for gradient descent. Similarly, [40] shows that by sacrificing computation time, simulated annealing optimization can also improve the results of standard CNN architectures. In a more recent work, the authors of [44] propose SEAL and apply simulated annealing in the early layers of networks to prohibit them from learning overly specific representations and improve model generalization. Furthermore, there has also been research regarding the combination of SA with reinforcement learning (RL) algorithms, where RL is used to optimize specific

hyperparameters of the SA process [13].

When compared to previously proposed methods which take into consideration gradient behaviour, GGA has some key differences. In contrast to gradient surgery algorithms [34, 55] that either mute, aggregate or set gradients to random values, and Fish [47] that approximately optimizes the inner-product between domain gradients, GGA searches for existing parameter space points where gradients of different domains have pairwise small angles. Furthermore, as GGA is applied in the early stages of training and for a limited number of training iterations, the high computational burden of traditional simulated annealing methods is avoided.

3. Methods

3.1. Preliminaries

Consider a classification problem with K classes. During model training we have access to a data set composed of distinct source data distributions (or *domains*), $\mathcal{S} = \left\{D_1, D_2, \ldots, D_{|\mathcal{S}|}\right\}$. From each domain D_i, we observe n_i labeled data points, such that $(\boldsymbol{x}_j^{(i)}, y_j^{(i)}) \sim D_i$, for $j = 1, \ldots, n_i$. Similarly, the test dataset consists of $\mathcal{T} = \left\{D_1^T, D_2^T, \ldots, D_{|\mathcal{T}|}^T\right\}$ *unseen* target data distributions, from which the model cannot retrieve any information during training. The goal is to learn a single labeling function $h(\boldsymbol{x}; \boldsymbol{\theta})$, parameterized by $\boldsymbol{\theta}$, which correctly maps input observations $\boldsymbol{x}_j^{(i)}$ to their labels $y_j^{(i)}$ for both the seen source and the unseen target domains.

If $\mathcal{L}_i(\boldsymbol{\theta}) = \frac{1}{n_i}\sum_{j=1}^{n_i} \ell(h(\boldsymbol{x}_j{}^{(i)}; \boldsymbol{\theta}), y_j^{(i)})$ represents the loss associated to the i-th source data domain in the training set, we define the overall cost function $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \mathcal{L}_i(\boldsymbol{\theta})$, as the average loss over all available source domains. The function $\ell(\cdot, \cdot)$ is a classification loss, in our case cross-entropy, that measures the error between the predicted label $\hat{y}$ of an input observation and its true label y. The standard way of model training is *Empirical Risk Minimization* or *ERM* which uses the following objective on the training domain data

$$\boldsymbol{\theta}^* = \arg_{\boldsymbol{\theta}} \min \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \mathcal{L}_i(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta}) \tag{2}$$

where $R(\cdot)$ is a regularization term and λ a hyperparameter responsible for controlling the contribution of regularization to the loss, leading to a parameter vector $\boldsymbol{\theta}^*$. In practice, networks trained via ERM have been shown to overfit the data distributions present in the training set. Previous works (such as [47]) have observed that the directions of gradients for different domains during training play a significant role in model generalization. Given source domain losses $\mathcal{L}_i(\boldsymbol{\theta})$ and $\mathcal{L}_j(\boldsymbol{\theta})$ and their corresponding gradients $\boldsymbol{g}_i = \nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta})$, $\boldsymbol{g}_j = \nabla_{\boldsymbol{\theta}}\mathcal{L}_j(\boldsymbol{\theta})$, gradient **conflicts** arise when the angle of the gradients grows e.g., above $|\pi/2|$, or, equivalently, their cosine similarity $\frac{\boldsymbol{g}_i^T \cdot \boldsymbol{g}_j}{\|\boldsymbol{g}_i\|\|\boldsymbol{g}_j\|}$ becomes negative. In the following section, we further explore the role of gradient similarity as an indicator for domain overfitting.

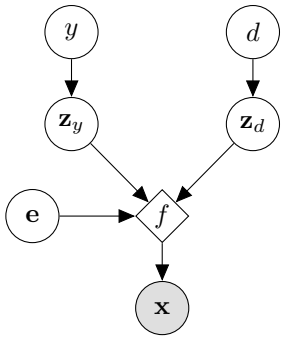

Figure 2. Simplified generative model for multi-domain data.

3.2. A Generative Model for Domain Generalization

To assist with the development and understanding of the proposed method consider the generative model of Figure 2, summarized by the following process:

$$\begin{aligned} y &\sim Multinoulli(p_1, \ldots, p_K) \\ d &\sim Multinoulli(q_1, \ldots, q_L) \\ \boldsymbol{z}_y &\sim p_y \\ \boldsymbol{z}_d &\sim p_d \\ \boldsymbol{e} &\sim p_e \\ \boldsymbol{x} &= f(\boldsymbol{z}_y, \boldsymbol{z}_d, \boldsymbol{e}) \end{aligned}$$

where y is a variable corresponding to the class, d is a variable corresponding to the domain (with $L = |\mathcal{S}| + |\mathcal{T}|$), $\boldsymbol{z}_y$ is a latent multivariate class-specific representation of class y, drawn from an unknown distribution p_y, and $\boldsymbol{z}_d$ is a latent multivariate domain-specific representation of the domain d, drawn from an unknown distribution p_d. Notice that these representations are disentangled, i.e., $\boldsymbol{z}_y$ does not depend on d. The observed sample $\boldsymbol{x}$, on the other hand, is derived using the function $f(\boldsymbol{z}_y, \boldsymbol{z}_d, \boldsymbol{e})$ that depends on both the class and domain representations, as well as independent nuisance variables, $\boldsymbol{e}$.

Without loss of generality, let's assume that we have two source domains ($|\mathcal{S}| = 2$) and two classes ($K = 2$). Then, during model training we sample only from the source domains and use the loss $\mathcal{L} = \alpha_1\mathcal{L}_1 + \alpha_2\mathcal{L}_2$,

$$\mathcal{L}_i = -\mathbb{E}_{d=i}\left[\log h(y|\boldsymbol{x}; \boldsymbol{\theta})\right] \tag{3}$$

where h is the probability estimated by our model for class y, α_i is the percentage of samples of domain i, and $i = 1, 2$. The expectation is over samples that have been generated

using the above process when $d = i$. Each gradient update step depends on the loss gradient

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L} &= \alpha_1\nabla_{\boldsymbol{\theta}}\mathcal{L}_1 + \alpha_2\nabla_{\boldsymbol{\theta}}\mathcal{L}_2 \\ &= -\alpha_1\mathbb{E}_{d=1}\left[\nabla_{\boldsymbol{\theta}}\log h(y|\boldsymbol{x};\boldsymbol{\theta})\right] \\ &- \alpha_2\mathbb{E}_{d=2}\left[\nabla_{\boldsymbol{\theta}}\log h(y|\boldsymbol{x};\boldsymbol{\theta})\right]\end{aligned} \tag{4}$$

where we have used the linearity of the expectation to obtain the expectation of the derivatives. Notice that the gradient update step depends on $\nabla_{\boldsymbol{\theta}}\log h(y|\boldsymbol{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\log h(y|f(\boldsymbol{z}_y, \boldsymbol{z}_d, \boldsymbol{e};\boldsymbol{\theta}))$.

For models that use an internal domain-invariant representation (i.e., a representation that does not depend on $\boldsymbol{z}_d$) it will hold that $h(y|\boldsymbol{x}) = h(y|f(\boldsymbol{z}_y, \boldsymbol{z}_1, \boldsymbol{e})) = h(y|f(\boldsymbol{z}_y, \boldsymbol{z}_2, \boldsymbol{e}))$, and therefore $\mathbb{E}_{d=1}\left[u(h(y|\boldsymbol{x}))\right] = \mathbb{E}_{d=2}\left[u(h(y|\boldsymbol{x}))\right]$ for any function u. Thus, both the losses, $\mathcal{L}_i$ and their gradients, $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i$, should be equal, in expectation, for different domains i. This observation provides a *necessary* condition for domain-invariance, which in turn provisions the main motivation for the development of our method, presented in the following section.

3.3. Gradient-Guided Annealing for DG

The analysis presented in the previous section shows that we can use the agreement of gradients $\nabla\mathcal{L}_i$ as an indicator of domain invariance, since the less the model $h(\cdot|\boldsymbol{\theta})$ depends on the domain d, the more similar the expected value of the gradient, $\mathbb{E}[\nabla\mathcal{L}_d]$, will be (see Eq. (4)). Inspired by Simulated Annealing [26], we achieve this by adding random noise to the parameters of the model in search for a point with high domain gradient agreement, as measured by the increase of the minimum gradient similarity across any pair of domains,

$$\text{grad\_sim} = \min\Big(\sum_{\substack{i=1,j=1,\\ i\neq j}}^{i\leq|\mathcal{S}|,j\leq|\mathcal{S}|} \frac{\boldsymbol{g}_i^T\cdot\boldsymbol{g}_j}{\|\boldsymbol{g}_i\|\|\boldsymbol{g}_j\|}\Big) \tag{5}$$

In more detail, as it is common in DG literature, we start with a pre-trained model $f(\cdot, \boldsymbol{\theta}_0)$ and perform a small number of warmup training steps. This ensures that the model approaches a region of the parameter space that achieves a low loss for the target problem. We then calculate the minimum pairwise gradient similarity among source domains and begin searching the neighborhood of the parameter space for points where both: (a) the domain gradients agree and (b) the loss are reduced[2]. This is implemented through iterative random perturbations $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \mathcal{U}(-\rho, \rho)$, where $\mathcal{U}$ is the multivariate uniform distribution. Each new point is selected only if it simultaneously achieves higher gradient agreement and lower loss (thus improving the Pareto front). After a fixed number of iterations, the point that has achieved the highest gradient agreement and the lowest loss is selected. This process is repeated after every training iteration step. After a number of optimization steps, we allow the model to be trained without parameter perturbations, following a standard SGD-based procedure, as usual. The GGA algorithm is presented in **Algorithm** 1.

[2]In our experiments, we relax the loss constraint and accept the new parameter set when the gradient similarity has increased and training loss has been reduced within a specific range (i.e $\mathcal{L} - \mathcal{L}' < 0.1$), where $\mathcal{L}'$ is the loss value at the updated set of parameters.

Algorithm 1 Implementing GGA in training

Require: Pretrained DNN h with parameters $\boldsymbol{\theta}_0$, training dataset with domains $\mathcal{S} = \{D_i\}_1^{|\mathcal{S}|}$. Loss gradients $\boldsymbol{g}$. Learning rate η, perturbation parameter ρ, optimization step to start the annealing process A_s, optimization step to end the annealing process A_e, number of annealing iterations per optimization step n_a. Total number of training iterations n.

for $t \leftarrow 1$ **to** n **do**
  Sample a mini-batch: $\mathcal{B} \leftarrow \mathcal{B}_{\mathcal{D}_1} + ... + \mathcal{B}_{\mathcal{D}_{|\mathcal{S}|}}$
  if $A_s \leq t \leq A_e$ **then**
    `#Begin Gradient-Guided Annealing:`
    Compute the mini-batch loss at starting params:
      $\mathcal{L}_B \leftarrow \mathcal{L}(\mathcal{B};\boldsymbol{\theta}_t)$
    Calculate minimum domain grad pair sim:
    $\text{sim} \leftarrow \min\Big(\sum_{\substack{i=1,j=1,\\ i\neq j}}^{i\leq|\mathcal{S}|,j\leq|\mathcal{S}|} \frac{\boldsymbol{g}_i^T\cdot\boldsymbol{g}_j}{\|\boldsymbol{g}_i\|\|\boldsymbol{g}_j\|}\Big)$
    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1}$
    for step $a \leftarrow 1$ **to** n_a **do**
      $\boldsymbol{\theta}_a \leftarrow \boldsymbol{\theta}_t + \mathcal{U}(-\rho, \rho)$
      Calculate minimum domain grad pair sim:
      $\text{sim}_a \leftarrow \min\Big(\sum_{\substack{i=1,j=1,\\ i\neq j}}^{i\leq|\mathcal{S}|,j\leq|\mathcal{S}|} \frac{\boldsymbol{g}_i^T\cdot\boldsymbol{g}_j}{\|\boldsymbol{g}_i\|\|\boldsymbol{g}_j\|}\Big)$
      Compute the mini-batch loss at new params:
      $\mathcal{L}_a \leftarrow \mathcal{L}(\mathcal{B};\boldsymbol{\theta}_a)$
      if $\big(\text{sim}_a > \text{sim}\big) \wedge \big(\mathcal{L}_a - \mathcal{L}_\mathcal{B} < 0.1\big)$ **then**
        $\mathcal{L}_\mathcal{B} \leftarrow \mathcal{L}_a$, $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_a$, $\text{sim} \leftarrow \text{sim}_a$
      end if
    end for
  end if
  `#Update weights:`
  Compute mini-batch loss: $\mathcal{L}_\mathcal{B} = \mathcal{L}(\mathcal{B};\boldsymbol{\theta}_t)$
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\cdot\nabla\mathcal{L}_\mathcal{B}(\mathcal{B};\boldsymbol{\theta}_t)$
  $t = t + 1$
end for

Table 1. **Comparison of GGA with the ERM baseline**. The top out-of-domain accuracies on five domain generalization benchmarks averaged over three trials, are presented.

Algorithm	PACS	VLCS	OfficeHome	TerraInc	DomainNet	Avg.
ERM	85.5±0.2	77.3±0.4	66.5±0.3	46.1±1.8	43.8±0.3	63.9
GGA (ours)	**87.3**±0.4	**79.9**±0.4	**68.5**±0.2	**50.6**±0.1	**45.2**±0.2	**66.3**

Table 2. **Comparison with state-of-the-art domain generalization methods.** Out-of-domain accuracies on five domain generalization benchmarks are shown. Top performing methods are highlighted in **bold** while second-best are *underlined*. The results marked by †, ‡ are copied from Gulrajani and Lopez-Paz [19] and Wang et al. [52], respectively. For fair comparison, the training of each algorithm combined with **GGA**, were run on the respective codebases. Average accuracies and standard errors are calculated from three trials for the combination of GGA with past algorithms and from 5 trials for GGA. In green and red, we highlight the performance boost and decrease of applying GGA on top of each algorithm respectively, averaged over three trials. Due to computational resources, for DomainNet we do not combine GGA with previous methods.

Algorithm	PACS	VLCS	OfficeHome	TerraInc	Avg.	DomainNet	Total
Mixstyle‡ [59]	85.2±0.3 (+0.3)	77.9±0.5 (+0.6)	60.4±0.3 (+0.5)	44.0±0.7 (+1.1)	66.9 (+0.6)	34.0±0.1	60.3
GroupDRO‡ [43]	84.4±0.8 (+1.4)	76.7±0.6 (+0.6)	66.0±0.7 (+2.2)	43.2±1.1 (+1.5)	67.6 (+1.4)	33.3±0.2	60.7
MMD‡ [31]	84.7±0.5 (+0.8)	77.5±0.9 (+1.3)	66.3±0.1 (+1.6)	42.2±1.6 (+6.3)	67.7 (+2.5)	23.4±9.5	58.8
AND-mask [46]	84.4±0.9 (+0.1)	78.1±0.9 (+0.3)	65.6±0.4 (+1.2)	44.6±0.3 (−0.4)	68.2 (+0.3)	37.2±0.6	62.0
ARM‡ [57]	85.1±0.4 (+0.5)	77.6±0.3 (+0.9)	64.8±0.3 (+2.0)	45.5±0.3 (+0.8)	68.3 (+1.1)	35.5±0.2	61.7
IRM† [1]	83.5±0.8 (−1.5)	78.5±0.5 (−0.9)	64.3±2.2 (−2.1)	47.6±0.8 (−3.9)	68.5 (−2.1)	33.9±2.8	61.6
MTL‡ [6]	84.6±0.5 (+0.9)	77.2±0.4 (+2.0)	66.4±0.5 (+0.4)	45.6±1.2 (+0.6)	68.5 (+0.9)	40.6±0.1	62.9
VREx‡ [27]	84.9±0.6 (+0.6)	78.3±0.2 (+0.1)	66.4±0.6 (+1.3)	46.4±0.6 (+2.0)	69.0 (+1.0)	33.6±2.9	61.9
MLDG† [30]	84.9±1.0 (+0.7)	77.2±0.4 (+1.3)	66.8±0.6 (+1.2)	47.7±0.2 (+1.1)	69.2 (+1.2)	41.2±0.1	63.6
Mixup† [54]	84.6±0.6 (+1.2)	77.4±0.6 (+1.8)	68.1±0.3 (+1.0)	47.9±0.8 (+2.0)	69.5 (+1.5)	39.2±0.1	63.4
SagNet† [36]	86.3±0.2 (−1.0)	77.8±0.5 (+0.9)	68.1±0.1 (+0.3)	48.6±1.0 (+0.7)	70.2 (+0.4)	40.3±0.1	64.2
CORAL† [48]	86.2±0.3 (+0.7)	78.8±0.6 (−0.4)	68.7±0.3 (+0.2)	47.6±1.0 (+0.3)	70.3 (+0.2)	41.5±0.1	64.5
RSC† [21]	85.2±0.9 (+0.1)	77.1±0.5 (+0.2)	65.5±0.9 (+0.0)	46.6±1.0 (+0.2)	68.6 (+0.1)	38.9±0.5	62.7
Fish ‡ [47]	85.5±0.3 (+0.1)	77.8±0.3 (+0.9)	68.6±0.4 (−0.6)	45.1±1.3 (+3.8)	69.3 (+1.0)	42.7±0.2	63.9
SAM ‡ [16]	85.8±0.2 (+0.6)	79.4±0.1 (+0.7)	69.6±0.1 (+0.4)	43.3±0.7 (+2.6)	69.5 (+1.1)	44.3±0.0	64.5
GSAM ‡ [62]	85.9±0.1 (+0.4)	79.1±0.2 (+1.0)	69.3±0.0 (+0.3)	47.0±0.8 (+0.6)	70.3 (+0.2)	44.6±0.2	65.1
SAGM ‡ [52]	86.6±0.2 (+0.2)	**80.0**±0.3 (−0.3)	**70.1**±0.2 (−0.6)	48.8±0.9 (−0.1)	71.4 (−0.2)	45.0±0.2	66.1
GGA (ours)	**87.3**±0.4	79.9±0.4	68.5±0.2	**50.6**±0.1	**71.7**	**45.2**±0.2	**66.3**

4. Experiments

4.1. Experimental setup and implementation details

In our experiments, we follow the protocol of DomainBed [19] and exhaustively evaluate our algorithm against state-of-the-art algorithms on five DG benchmarks, using the same dataset splits and model selection. For the hyperparameter search space, we follow [10] in order to avoid the high computational burden of DomainBed. The datasets included in the benchmarks are, PACS [29] (9,991 images, 7 classes, and 4 domains), VLCS [15] (10,729 images, 5 classes, and 4 domains), OfficeHome [50] (15,588 images, 65 classes, and 4 domains), TerraIncognita [5] (24,788 images, 10 classes, and 4 domains) and DomainNet [38] (586,575 images, 345 classes, and 6 domains).

In all experiments, the *leave-one-domain-out* cross-validation protocol is followed. Specifically, in each run a single domain is left out as the target (test) domain, while the rest of the domains are used for training. The final performance of each algorithm is calculated by averaging the top-1 accuracy on the target domain, with different train-validation splits. For training, we utilize a ResNet-50 [20] pretrained on ImageNet [41] for the backbone feature extractor and ADAM for the optimizer. Regarding GGA, during training we let each algorithm run for several training iterations depending on the dataset and then begin the parameter space search, as described in Section 3. For the neighborhood size in the search process, we set ρ to 0.0005[3]

[3]The selection of ρ was based on previous algorithms that implement weight perturbations in ResNet-50 networks [16, 28, 52]. The sensitivity

and search for a total of $A = 250$ steps before moving to the next mini-batch of 48 samples from each domain, in each dataset. In all experiments, we perform search iterations for 100 different mini-batches during early training stages. The rest of the hyperparameters, such as learning rate, weight decay and dropout rate, are tuned according to [10] and are presented in Table 3. To account for the variability introduced in the random search of GGA, we repeat the experiments with 5 different seeds for each dataset. All models were trained on a cluster containing 4×40GB NVIDIA A100 GPU cards, split into 8 20GB virtual MIG devices and 1×24GB NVIDIA RTX A5000 GPU card.

Hyperparameter	PACS	VLCS	OH	TI	DN
Learning rate	3e-5	1e-5	1e-5	1e-5	3e-5
Dropout	0.5	0.5	0.5	0.5	0.5
Weight decay	1e-4	1e-4	1e-4	1e-4	1e-4
Training Steps	5000	5000	5000	5000	15000
ρ	5e-5	5e-5	5e-5	5e-5	5e-5
GGA Start-End	100-150, 1500-1550	100-200	100-200	500-600	100-200

Table 3. Hyperparameters for DG experiments. OH, TI and DN stand for OfficeHome, TerraIncognita and DomainNet respectively. ρ is the parameter space search used during the annealing steps in GGA, while the last row indicates the training iterations during which GGA occurs.

4.2. Comparative Evaluation

The average OOD performances of the baseline vanilla ERM [49] and state-of-the-art DG methods on a total of 5 DG benchmarks, are reported in Tables 1 and 2 respectively. The results for each separate domain in each benchmark are reported in the Appendix. In the experiments, we apply GGA on-top of the rest of the DG algorithms and show that its properties generalize to other methods as well, boosting their overall performance. It should also be noted that GGA can be adapted to training scenarios where distributions shifts are present in training data.

Initially, to validate whether the application of GGA in the early stages of model training leads to models with increased generalizabilty, we compare the results with the vanilla ERM baseline. As presented in Table 1, GGA is able to boost the performance of the baseline model by an average of 2.4% on all benchmarks, demonstrating the efficacy of the proposed algorithm. For further evaluation, we also compare GGA with state-of-the-art DG methods [1, 6, 16, 18, 21, 27, 30, 31, 36, 43, 46–49, 52, 54, 57, 59, 62], before applying GGA to them as well. We note that we only include previous works that have implemented a ResNet-50 as the backbone encoder and do not use additional components or ensembles. In Table 2 we differentiate between the methods that operate on model gradients [16, 21, 47, 52, 62] and the rest of the algorithms. Even without its application to other methods, GGA is able to surpass most of the previously proposed algorithms, while also setting the state-of-the-art on PACS ($+0.7\%$) and on the challenging TerraIncognita ($+1.8$) dataset, while remaining competitive in the other two. What's more important is that the application of GGA in conjunction with the rest of the DG methods, proves beneficial and ultimately boosts their overall performance in almost each case by an average of around 1% and in some cases up to even 6.3%. With regards to IRM, which seems to be significantly impacted negatively by the application of GGA, its learning objective emphasizes on simultaneously minimizing the training loss of each source domain and not necessarily on the pairwise agreement of gradients among domains. It is therefore not able to converge to a good solution after the initial model's weights have been perturbed during training.

From the experimental results, it is evident that the application of GGA and its search for parameter values where gradients align between domains is beneficial to model training. By introducing the proposed annealing step before the final stages of training, the majority of the models exceed their previous performance and exhibit improved generalization capabilities.

4.3. Evaluating the impact of GGA on gradient disagreement

As discussed in Section 3, when a training dataset is composed as a mixture of multiple domains, conflicting gradients between mini-batches drawn from each domain lead to models that do not infer based on domain-invariant features and which generalize to previously unseen data samples, but are hindered by domain-specific, spurious correlations. This is evident in the case of vanilla models trained via ERM where the average gradient similarity among domains continues to remain low upon reaching a local minima. Our hypothesis is that this behavior can be avoided by searching for a parameter set of common agreement between domains before optimizing via gradient descent.

To demonstrate the operation of the proposed algorithm in practice against ERM, we calculate the average gradient cosine similarity between mini-batches from source domains during training for the VLCS dataset, along with the training loss in each iteration. As a result, each sub-figure in Figure 3 illustrates the progression of the training gradient alignment between domains, against the total training loss.

As expected, in the very initial iterations the gradients of the pretrained model parameters point towards a common direction. However, in the case of ERM as training progresses and the loss is minimized, the domain gradients begin to disagree leading the model to converge to undesirable minima that do not generalize across domains. On the other hand, when GGA is applied the model searches for parameters such that gradients are aligned before continuing

analysis presented in subsection 4.4 also reveals that $\rho = 0.0005$ yields the best performance.

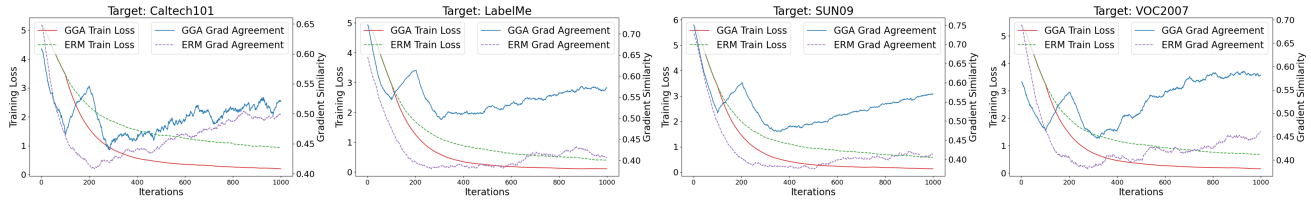

Figure 3. Impact of GGA on gradient alignment during model training on the VLCS dataset. As illustrated, in the case of vanilla ERM, even though the training loss is minimized, the average cosine similarity among domain gradients remains low. In the case of GGA however, after the algorithm searches for points in the parameter space with increased gradient similarity, the gradients continue to agree during training, while the total training loss is also minimized.

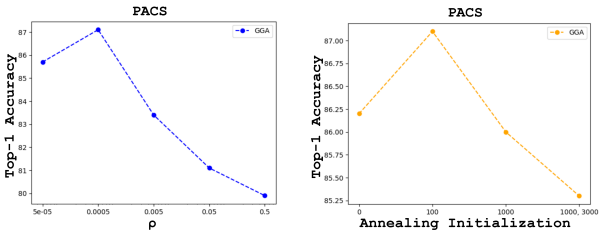

Figure 4. Sensitivity analysis of the parameter space search magnitude ρ and the training stage application of GGA. The analysis on PACS reveals smaller weight perturbations and gradient annealing during early training iterations lead to increased model performance.

training. This is illustrated by the spike in gradient similarity, during iterations 100 up to 200. After GGA concludes, we observe that the model continues training by descending into minima where gradients agree among domains.

4.4. Sensitivity Analysis

The two core parameters of GGA, are the size of the parameter search ρ and the moment of our methods implementation during training, i.e., during the early, mid or late training stages. To justify their selection, we conduct a sensitivity analysis (Fig. 4) by varying one of the above parameters while fixing the other at its optimal value.

Regarding the magnitude of weight perturbations during the application of GGA, we found that the optimal value was $\rho = 5e - 5$. As illustrated in Figure 4, a larger magnitude of perturbation led to decreased model performance. Intuitively, the application of larger noise to the model parameters leads to sets that are not close to the solution, making it increasingly difficult for the model to converge. On the other hand, smaller perturbations seem to have little to no effect on training, as the search is limited to spaces near the current parameters, which is why the model performance falls back close to that of ERM. With regards to the stage of training during which GGA will be applied, we found that the models yielded better performance when the search was initialized in earlier stages.We hypothesize that applying perturbations near the end of training displaces the model from a local optimum, requiring additional iterations to converge.

5. Conclusions

In this work, we investigated gradient conflicts in the context of domain generalization, leading to the development of GGA; a simple yet effective algorithm that identifies points in the parameter space where domain gradients align early in training before continuing optimization, ultimately enhancing model generalization. Through a comprehensive comparative analysis we have demonstrated the efficacy of GGA, which is able to achieve superior generalization capabilities in standard DG benchmarks and outperform SoTA baselines. What's more, as GGA is both model and method agnostic it can also be utilized by other algorithms. Interestingly, its combination with previous methods proves beneficial, as their majority yields improved performance over all benchmarks. Finally, we validate the impact of our algorithm on gradient alignment through additional experiments, showing that, unlike the baseline, domain gradients align during training.

However, our method does not come without some key limitations. First of all, the annealing steps and gradient similarity computation add computational overhead. Furthermore, GGA relies on domain labels, making it inapplicable to single-source DG, where domain labels are not available during training. Its effectiveness is also batch-size dependent, as larger batches provide further information regarding gradient alignment and can lead to improved results. Future work will further explore gradient alignment in multi-domain settings and refine GGA to make annealing steps adaptive rather than random.

Acknowledgement. This work was supported by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 965231, project REBECCA (REsearch on BrEast Cancer induced chronic conditions supported by Causal Analysis of multi-source data).

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 6, 7

[2] Peter Auer, Mark Herbster, and Manfred KK Warmuth. Exponentially many local minima for single neurons. *Advances in neural information processing systems*, 8, 1995. 2

[3] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems*, 31, 2018. 3

[4] Aristotelis Ballas and Christos Diou. Multi-scale and multi-layer contrastive learning for domain generalization. *IEEE Transactions on Artificial Intelligence*, pages 1–14, 2024. 3

[5] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018. 6

[6] Gilles Blanchard, Aniket Anand Deshmukh, Urun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *Journal of Machine Learning Research*, 22(2):1–55, 2021. 6, 7

[7] Léon Bottou. Online algorithms and stochastic approximations. *Online learning in neural networks*, 1998. 2

[8] Zhicheng Cai. Sa-gd: Improved gradient descent learning strategy with simulated annealing. *arXiv preprint arXiv:2107.07558*, 2021. 3

[9] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2229–2238, 2019. 3

[10] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34: 22405–22418, 2021. 3, 6, 7

[11] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018. 2

[12] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR, 2015. 2

[13] Alvaro H.C. Correia, Daniel E. Worrall, and Roberto Bondesan. Neural simulated annealing. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 4946–4962. PMLR, 2023. 4

[14] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in neural information processing systems*, 32, 2019. 3

[15] Chen Fang, Ye Xu, and Daniel 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. 6

[16] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. 3, 6, 7

[17] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016. 3

[18] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(1):2096–2030, 2016. 7

[19] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021. 2, 6

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[21] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. *European Conference on Computer Vision*, 2020. 3, 6, 7

[22] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 3

[23] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 2

[24] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *International Conference on Computer Vision*, 2021.

[25] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[26] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598): 671–680, 1983. 2, 5

[27] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International conference on machine learning*, pages 5815–5826. PMLR, 2021. 6, 7

[28] Binh M Le and Simon S Woo. Gradient alignment for cross-domain face anti-spoofing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 188–199, 2024. 3, 6

[29] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017. 6

[30] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 3, 6, 7

[31] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Computer Vision and Pattern Recognition*, 2018. 6, 7

[32] Fangrui Lv, Jian Liang, Shuang Li, Bin Zang, Chi Harold Liu, Ziteng Wang, and Di Liu. Causality inspired representation learning for domain generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8046–8056, 2022. 3

[33] Divyat Mahajan, Shruti Tople, and Amit Sharma. Domain generalization using causal matching. In *International conference on machine learning*, pages 7313–7324. PMLR, 2021. 3

[34] Lucas Mansilla, Rodrigo Echeveste, Diego H Milone, and Enzo Ferrante. Domain generalization via gradient surgery. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6630–6638, 2021. 2, 3, 4

[35] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International conference on machine learning*, pages 10–18. PMLR, 2013. 3

[36] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *Computer Vision and Pattern Recognition*, 2021. 6, 7

[37] Quynh Nguyen and Matthias Hein. Optimization landscape and expressivity of deep cnns. In *International conference on machine learning*, pages 3730–3739. PMLR, 2018. 2

[38] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019. 6

[39] Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. Domain agnostic learning with disentangled representations. In *International conference on machine learning*, pages 5102–5112. PMLR, 2019. 3

[40] L.M. Rasdi Rere, Mohamad Ivan Fanany, and Aniati Murni Arymurthy. Simulated annealing algorithm for deep learning. *Procedia Computer Science*, 72:137–144, 2015. The Third Information Systems International Conference 2015. 3

[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 6

[42] Itay Safran and Ohad Shamir. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, pages 774–782. PMLR, 2016. 2

[43] Shiori Sagawa*, Pang Wei Koh*, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020. 6, 7

[44] Amir M Sarfi, Zahra Karimpour, Muawiz Chaudhary, Nasir M Khalid, Mirco Ravanelli, Sudhir Mudur, and Eugene Belilovsky. Simulated annealing in early layers leads to better generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20205–20214, 2023. 3

[45] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018. 2

[46] Soroosh Shahtalebi, Jean-Christophe Gagnon-Audet, Touraj Laleh, Mojtaba Faramarzi, Kartik Ahuja, and Irina Rish. Sand-mask: An enhanced gradient masking strategy for the discovery of invariances in domain generalization. *arXiv preprint arXiv:2106.02266*, 2021. 6, 7

[47] Yuge Shi, Jeffrey Seely, Philip Torr, Siddharth N, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. In *International Conference on Learning Representations*, 2022. 3, 4, 6, 7

[48] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, 2016. 6

[49] V Vapnik. Statistical learning theory. *NY: Wiley*, 1998. 3, 7

[50] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017. 6

[51] Guoqing Wang, Hu Han, Shiguang Shan, and Xilin Chen. Cross-domain face presentation attack detection via multi-domain disentangled representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6678–6687, 2020. 3

[52] Pengfei Wang, Zhaoxiang Zhang, Zhen Lei, and Lei Zhang. Sharpness-aware gradient matching for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3769–3778, 2023. 3, 6, 7

[53] Ziqi Wang, Marco Loog, and Jan Van Gemert. Respecting domain relations: Hypothesis invariance for domain generalization. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9756–9763. IEEE, 2021. 3

[54] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *AAAI Conference on Artificial Intelligence*, 2020. 6, 7

[55] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020. 2, 3, 4

[56] Hanlin Zhang, Yi-Fan Zhang, Weiyang Liu, Adrian Weller, Bernhard Schölkopf, and Eric P Xing. Towards principled disentanglement for domain generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8024–8034, 2022. 3

[57] Marvin Zhang, Henrik Marklund, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: A meta-learning approach for tackling group shift. *arXiv preprint arXiv:2007.02931*, 2020. 3, 6, 7

[58] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34 (12):5586–5609, 2022. 2

[59] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *International Conference on Learning Representations*, 2021. 3, 6, 7

[60] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain Generalization: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4396–4415, 2023. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. 2

[61] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012. 3

[62] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha C Dvornek, sekhar tatikonda, James s Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. In *International Conference on Learning Representations*, 2022. 3, 6, 7