SWAD: Domain Generalization by Seeking Flat Minima

Junbum Cha[1†] **Sanghyuk Chun**[2*] **Kyungjae Lee**[3*]
Han-Cheol Cho[4] **Seunghyun Park**[4] **Yunsung Lee**[5] **Sungrae Park**[6†]

[1] Kakao Brain [2] NAVER AI Lab [3] Chung-Ang University
[4] NAVER Clova [5] Korea University [6] Upstage AI Research

Abstract

Domain generalization (DG) methods aim to achieve generalizability to an unseen target domain by using only training data from the source domains. Although a variety of DG methods have been proposed, a recent study shows that under a fair evaluation protocol, called DomainBed, the simple empirical risk minimization (ERM) approach works comparable to or even outperforms previous methods. Unfortunately, simply solving ERM on a complex, non-convex loss function can easily lead to sub-optimal generalizability by seeking sharp minima. In this paper, we theoretically show that finding flat minima results in a smaller domain generalization gap. We also propose a simple yet effective method, named Stochastic Weight Averaging Densely (SWAD), to find flat minima. SWAD finds flatter minima and suffers less from overfitting than does the vanilla SWA by a dense and overfit-aware stochastic weight sampling strategy. SWAD shows state-of-the-art performances on five DG benchmarks, namely `PACS`, `VLCS`, `OfficeHome`, `TerraIncognita`, and `DomainNet`, with consistent and large margins of +1.6% averagely on out-of-domain accuracy. We also compare SWAD with conventional generalization methods, such as data augmentation and consistency regularization methods, to verify that the remarkable performance improvements are originated from by seeking flat minima, not from better in-domain generalizability. Last but not least, SWAD is readily adaptable to existing DG methods without modification; the combination of SWAD and an existing DG method further improves DG performances. Source code is available at `https://github.com/khanrc/swad`.

1 Introduction

Independent and identically distributed (i.i.d.) condition is the underlying assumption of machine learning experiments. However, this assumption may not hold in real-world scenarios, *i.e.*, the training and the test data distribution may differ significantly by *distribution shifts*. For example, a self-driving car should adapt to adverse weather or day-to-night shifts [1, 2]. Even in a simple image recognition scenario, systems rely on wrong cues for their prediction, *e.g.*, geographic distribution [3], demographic statistics [4], texture [5], or backgrounds [6]. Consequently, a practical system should require generalizability to distribution shift, which is yet often failed by traditional approaches.

Domain generalization (DG) aims to address *domain shift* simulated by training and evaluating on different domains. DG tasks assume that both task labels and domain labels are accessible. For example, `PACS` dataset [7] has seven task labels (*e.g.*, “dog”, “horse”) and four domain labels (*e.g.*, “photo”, “sketch”). Previous approaches explicitly reduced domain gaps in the latent space [8–

[*] Equal contribution [†] Part of work done while at NAVER Clova
Correspondence to: Junbum Cha <junbum.cha@kakaobrain.com>, Sungrae Park <sungrae.park@upstage.ai>

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Table 1: **Comparisons with SOTA.** The proposed SWAD outperforms other state-of-the-art DG methods on five different DG benchmarks with significant gaps (+1.6pp in the average).

	PACS	VLCS	OfficeHome	TerraInc	DomainNet	**Avg.**
ERM [29]	85.5	77.5	66.5	46.1	40.9	63.3
Best SOTA competitor	86.6 [30]	78.8 [31]	68.7 [31]	48.6 [32]	43.6 [15, 33]	65.3
SWAD (proposed)	**88.1**	**79.1**	**70.6**	**50.0**	**46.5**	**66.9**
Previous SOTA [31] + SWAD	88.3	78.9	71.3	51.0	46.8	67.3

12], obtained well-transferable model parameters by the meta-learning framework [13–16], data augmentation [17–19], or capturing causal relation [20, 21]. Despite numerous previous attempts for a decade, Gulrajani and Lopez-Paz [22] showed that a simple empirical risk minimization (ERM) approach works comparably or even outperforms the previous attempts on diverse DG benchmarks under a fair evaluation protocol, called "DomainBed".

Unfortunately, although ERM showed surprising empirical success on DomainBed, simply minimizing the empirical loss on a complex and non-convex loss landscape is typically not sufficient to arrive at a good generalization [23–26]. In particular, the connection between the generalization gap and the flatness of loss landscapes has been actively discussed under the i.i.d. condition [23–28]. Izmailov et al. [25] argued that seeking flat minima will lead to robustness against the loss landscape shift between training and test datasets, while a simple ERM converges to the boundary of a wide flat minimum and achieves insufficient generalization. In the DG scenario, because training and test loss landscapes differ more drastically due to the domain shift, we conjecture that the generalization gap between flat and sharp minima is larger than expected in the i.i.d. scenario.

To show that flatter minima generalize better to unseen domains, we formulate a robust risk minimization (RRM) problem defined by the worst-case empirical risks within neighborhoods in parameter space [26, 34]. We theoretically show that the generalization gap of DG, *i.e.*, the error on the target domain, is upper bounded by RRM, *i.e.*, a flat optimal solution. Based on our theoretical observation, we modify stochastic weight averaging (SWA) [25], one of the popular existing flatness-aware solvers, by introducing a dense and overfit-aware stochastic weight sampling strategy. First, we suggest to sample weights ***densely***, *i.e.*, for every iteration. Also, we search the start and end iterations for averaging by considering the validation loss to ***avoid overfitting***. We empirically show that the proposed Stochastic Weight Averaging Densely (SWAD) finds flatter minima than the vanilla SWA does, resulting in better generalization to unseen domains.

Contribution. Our main contribution is introducing flatness into DG, and showing remarkably outperforming performances against existing DG methods. As shown in Table 1, our SWAD improves the average DG performances by 3.6pp against the ERM baseline and 1.6pp against the existing best methods. Furthermore, by combining SWAD and previous SOTA [31], we even achieve 0.4pp improvements against the vanilla SWAD results. We also empirically show that while popular in-domain generalization methods without considering flatness, *e.g.*, Mixup [35] or CutMix [36], are not effective to out-of-domain generalization (Table 3), flatness-aware methods, *e.g.*, SWA [25] or SAM [26], are only effective methods to both in-domain and out-of-domain generalization.

2 A Theoretical Relationship between Flatness and Domain Generalization

Let $\mathcal{D} := \{\mathcal{D}_i\}_i^I$ be a set of training domains, where $\mathcal{D}_i$ is a distribution over input space $\mathcal{X}$, and I is the total number of domains. From each domain, we observe n training data points which consist of input x and target label y, $(x_j^i, y_j^i)_{j=1}^n \sim \mathcal{D}_i$. We also define a set of target domain $\mathcal{T} := \{\mathcal{T}_i\}_i^T$ similarly, where the number of target domains T is usually set to one. For the sake of simplicity, unlike Ben-David et al. [37], we assume that there exists a global labeling function $h(x)$ that generates target label for multiple domains, *i.e.*, $y_j^i = h(x_j^i)$ for all i and j. Domain generalization (DG) aims to find a model parameter $\theta \in \Theta$ which generalizes well over both multiple training domains $\mathcal{D}$ and unseen target domain $\mathcal{T}$. More specifically, let us consider a bounded instance loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, c]$, such that $\ell(y_1, y_2) = 0$ holds if and only if $y_1 = y_2$ where $\mathcal{Y}$ is a set of labels. For simplicity, we set c to one in our proofs, but we note that $\ell(\cdot, \cdot)$ can be generalized for any bounded loss function. Then, we can define a population loss over multiple domains by $\mathcal{E}_{\mathcal{D}}(\theta) = \frac{1}{I} \sum_{i=1}^I \mathbb{E}_{x^i \sim \mathcal{D}_i}[\ell(f(x^i; \theta), y^i))]$, where $f(\cdot; \theta)$ is a model parameterized by θ. Formally, the goal of DG is to find a model which minimizes both $\mathcal{E}_{\mathcal{D}}(\theta)$ and $\mathcal{E}_{\mathcal{T}}(\theta)$ by only minimizing an empirical risk $\hat{\mathcal{E}}_{\mathcal{D}}(\theta) := \frac{1}{In}\sum_{i=1}^{I}\sum_{j=1}^{n}\ell(f(x^i;\theta),y^i))$ over training domains $\mathcal{D}$.

In practice, ERM, *i.e.*, $\arg\min_\theta \hat{\mathcal{E}}_{\mathcal{D}}(\theta)$, can have multiple solutions that provide similar values of the training losses but significantly different generalizability on $\mathcal{E}_{\mathcal{D}}(\theta)$ and $\mathcal{E}_{\mathcal{T}}(\theta)$. Unfortunately, the typical optimization methods, such as SGD and Adam [38], often lead sub-optimal generalizability as finding sharp and narrow minima even under the i.i.d. assumption [23–28]. In the DG scenario, the generalization gap between empirical loss and target domain loss becomes even worse due to domain shift. Here, we provide a theoretical interpretation of the relationship between finding a flat minimum and minimizing the domain generalization gap, inspired by previous studies [23–28].

We consider a robust empirical loss function defined by the worst-case loss within neighborhoods in the parameter space as $\hat{\mathcal{E}}^{\gamma}_{\mathcal{D}}(\theta) := \max_{\|\Delta\|\leq\gamma}\hat{\mathcal{E}}_{\mathcal{D}}(\theta+\Delta)$, where $\|\cdot\|$ denotes the L2 norm and γ is a radius which defines neighborhoods of θ. Intuitively, if γ is sufficiently larger than the "radius" of a sharp optimum θ_s of $\hat{\mathcal{E}}_{\mathcal{D}}(\theta)$, θ_s is no longer an optimum of $\hat{\mathcal{E}}^{\gamma}_{\mathcal{D}}(\theta)$ as well as its neighborhoods within the γ-ball. On the other hand, if an optimum θ_f has larger "radius" than γ, there exists a local optimum within γ-ball – See Figure 1. Hence, solving the robust risk minimization (RRM), *i.e.*, $\arg\min_\theta \hat{\mathcal{E}}^{\gamma}_{\mathcal{D}}(\theta)$, will find a near solution of a flat optimum showing better generalizability [26, 34]. However, as domain shift worsen the generalization gap by breaking the i.i.d. assumption, it is not trivial that RRM will find an optimum with better DG performance. To answer the question, we first show the generalization bound between $\hat{\mathcal{E}}^{\gamma}_{\mathcal{D}}$ and $\mathcal{E}_{\mathcal{T}}$ as follows:

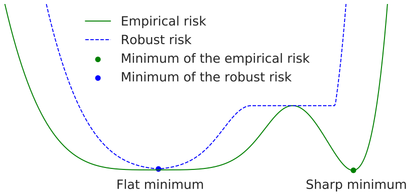


Figure 1: **Robust risk minimization (RRM) and flat minima.** With proper γ, RRM will find flat minima.

Theorem 1. *Consider a set of N covers $\{\Theta_k\}_{k=1}^{N}$ such that the parameter space $\Theta \subset \cup_k^N \Theta_k$ where $diam(\Theta) := \sup_{\theta,\theta'\in\Theta}\|\theta-\theta'\|_2$, $N := \left\lceil (diam(\Theta)/\gamma)^d \right\rceil$ and d is dimension of Θ. Let v_k be a VC dimension of each Θ_k. Then, for any $\theta \in \Theta$, the following bound holds with probability at least $1-\delta$,*

$$\mathcal{E}_{\mathcal{T}}(\theta) < \hat{\mathcal{E}}^{\gamma}_{\mathcal{D}}(\theta) + \frac{1}{2I}\sum_{i=1}^{I}\mathbf{Div}(\mathcal{D}_i,\mathcal{T}) + \max_{k\in[1,N]}\sqrt{\frac{v_k\ln(m/v_k)+\ln(N/\delta)}{m}}, \tag{1}$$

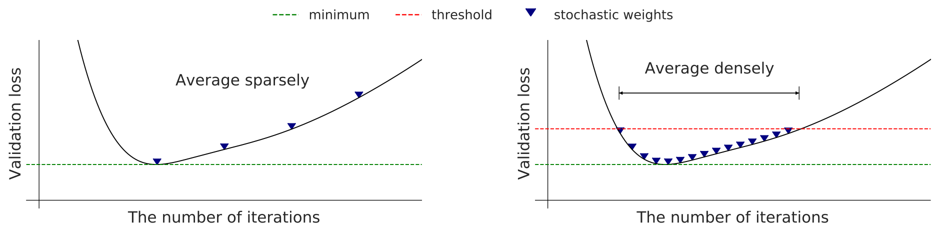
where $m = nI$ is the number of the training samples and $\mathbf{Div}(\mathcal{D}_i,\mathcal{T}) := 2\sup_A|\mathbb{P}_{\mathcal{D}_i}(A)-\mathbb{P}_{\mathcal{T}}(A)|$ is a divergence between two distributions.

Proof can be done similarly as [37] and [34]. In Theorem 1, the test loss $\mathcal{E}_{\mathcal{T}}(\theta)$ is bounded by three terms: (1) the robust empirical loss $\hat{\mathcal{E}}^{\gamma}_{\mathcal{D}}(\theta)$, (2) the discrepancy between training distribution and test distribution, *i.e.*, the quantity of domain shift, and (3) a confidence bound related to the radius γ and the number of the training samples m. Our theorem is similar to Ben-David et al. [37], while our theorem does not have the term related to the difference in labeling functions across the domains. It is because we simply assume there is no difference between labeling functions for each domain for simplicity. If one assumes a different labeling function, the dissimilarity term can be derived easily because it is independent and compatible with our main proof. More details of Theorem 1, including proof and discussions on the confidence bound, are in Appendix C.1 and C.2.

From Theorem 1, one can conjure that minimizing the robust empirical loss is directly related to the generalization performances on the target distribution. We show that the domain generalization gap on the target domain $\mathcal{T}$ by the optimal solution of RRM, $\hat{\theta}^{\gamma}$, is upper bounded as follows:

Theorem 2. *Let $\hat{\theta}^{\gamma}$ denote the optimal solution of the RRM, i.e., $\hat{\theta}^{\gamma} := \arg\min_\theta \hat{\mathcal{E}}^{\gamma}_{\mathcal{D}}(\theta)$, and let v be a VC dimension of the parameter space Θ. Then, the gap between the optimal test loss, $\min_{\theta'}\mathcal{E}_{\mathcal{T}}(\theta')$, and the test loss of $\hat{\theta}^{\gamma}$, $\mathcal{E}_{\mathcal{T}}(\hat{\theta}^{\gamma})$, has the following bound with probability at least $1-\delta$.*

$$\begin{aligned}\mathcal{E}_{\mathcal{T}}(\hat{\theta}^{\gamma}) - \min_{\theta'}\mathcal{E}_{\mathcal{T}}(\theta') \quad &\leq \quad \hat{\mathcal{E}}^{\gamma}_{\mathcal{D}}(\hat{\theta}^{\gamma}) - \min_{\theta''}\hat{\mathcal{E}}_{\mathcal{D}}(\theta'') + \frac{1}{I}\sum_{i=1}^{I}\mathbf{Div}(\mathcal{D}_i,\mathcal{T}) \\ &\quad + \max_{k\in[1,N]}\sqrt{\frac{v_k\ln(m/v_k)+\ln(2N/\delta)}{m}} + \sqrt{\frac{v\ln(m/v)+\ln(2/\delta)}{m}}\end{aligned} \tag{2}$$



(a) SWA

(b) SWAD (proposed)

Figure 2: **Comparison between SWA and SWAD.** (a) SWA collects stochastic weights for every K epochs from the pre-defined K_0 epochs to the final epoch. (b) Our SWAD collects stochastic weights ***densely***, *i.e.*, for every iteration, to obtain sufficiently many weights. SWAD collects the weights from the start iteration t_s to the end iteration t_e, where t_s and t_e are obtained by monitoring the validation loss (***overfit-aware*** scheduling).

Proof is in Appendix C.3. It implies that if we find the optimal solution of the RRM (*i.e.*, $\hat{\theta}^\gamma$), then the generalization gap in the test domain (*i.e.*, $\mathcal{E}_\mathcal{T}(\hat{\theta}^\gamma) - \min_{\theta'} \mathcal{E}_\mathcal{T}(\theta')$) is upper bounded by the gap between the RRM and ERM (*i.e.*, $\hat{\mathcal{E}}^\gamma_\mathcal{D}(\hat{\theta}^\gamma) - \min_{\theta''} \hat{\mathcal{E}}_\mathcal{D}(\theta'')$). Other terms in Theorem 2 are the discrepancy between the train domains $\mathcal{D}$ and the target domain $\mathcal{T}$, and the confidence bounds caused by sample means. We remark that if we choose a proper γ, the optimal solution of the RRM will find a point near a flat optimum of ERM as shown in Figure 1. Hence, Theorem 2 and the intuition from Figure 1 imply that seeking a flat minimum of ERM will lead to a better domain generalization gap.

3 SWAD: Domain Generalization by Seeking Flat Minima

We have shown that flat minima will bring a better domain generalization. In this section, we propose Stochastic Weight Averaging Densely (SWAD) algorithm, and provide empirical quantitative and qualitative analyses on SWAD and flatness to understand why SWAD works better than ERM.

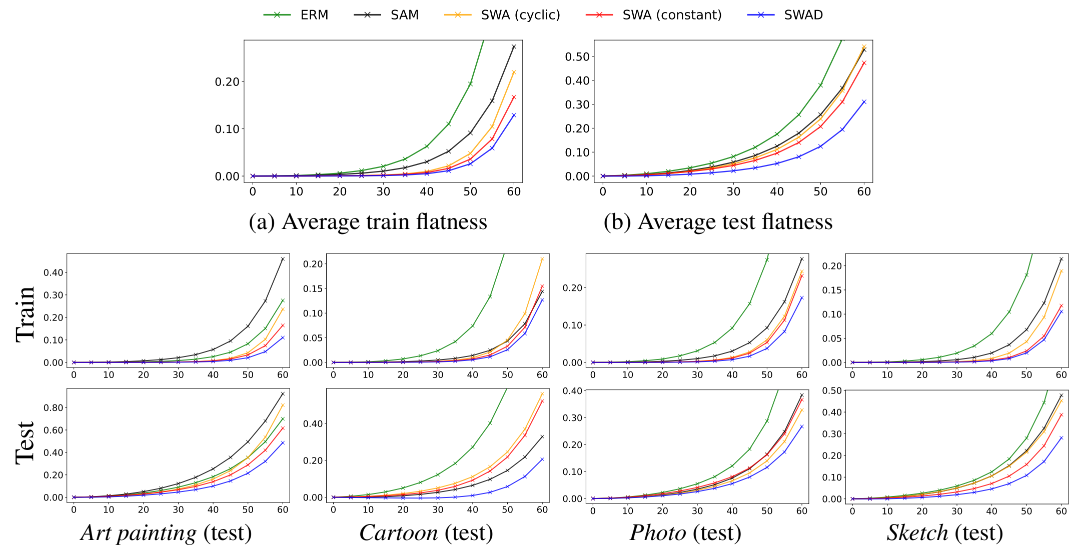
3.1 A baseline method: stochastic weight averaging

Since the importance of flatness in loss landscapes has emerged [23–28], several methods have been proposed to find flat minima [25, 26, 39]. We select stochastic weight averaging (SWA) [25] as a baseline, which finds flat minima by a weight ensemble approach. More specifically, SWA updates a pretrained model (namely, a model trained with sufficiently enough training epochs, K_0) with a cyclical [40] or high constant learning rate scheduling. SWA gathers model parameters for every K epochs during the update and averages them for the model ensemble. SWA finds an ensembled solution of different local optima found by a sufficiently large learning rate to escape a local minimum. Izmailov et al. [25] empirically showed that SWA finds flatter minima than ERM. We also considered sharpness-aware minimization (SAM) [26], which is another popular flatness-aware solver, but SWA finds flatter minima than SAM (See Figure 3). We illustrate an overview of SWA in Figure 2a.

3.2 Dense and overfit-aware stochastic weight sampling strategy

Despite its advantages, directly applying SWA to DG task has two problems. First, SWA averages a few weights (usually less than ten) by sampling weights for every K epochs, results in an inaccurate approximation of flat minima on a high-dimensional parameter space (*e.g.*, 23M for ResNet-50 [41]). Furthermore, a common DG benchmark protocol uses relatively small training epochs (*e.g.*, Gulrajani and Lopez-Paz [22] trained with less than two epochs for `DomainNet` benchmark), resulting in insufficient stochastic weights for SWA. From this motivation, we propose a ***"dense"*** sampling strategy for gathering sufficiently enough stochastic weights.

In addition, widely used DG datasets, such as `PACS` ($\approx$ 10K images, 7 classes) and `VLCS` ($\approx$ 11K images, 5 classes), are relatively smaller than large-scale datasets, such as ImageNet [42] ($\approx$ 1.2M images, 1K classes). In this case, we observe that a simple ERM approach is rapidly reached to a local optimum only within a few epochs, and easily suffers from the overfitting issue, *i.e.*, the validation loss is increased after a few training epochs. It implies that directly applying the vanilla SWA will suffer from the overfitting issue by averaging sub-optimal solutions (*i.e.*, overfitted parameters). Hence, we need an ***"overfit-aware"*** sampling scheduling to omit the sub-optimal solutions for SWA.



(a) Average train flatness

(b) Average test flatness

Art painting (test)

Cartoon (test)

Photo (test)

Sketch (test)

(c) Flatness for each target domain

Figure 3: **Local flatness comparisons.** We plot the local flatness via loss gap, *i.e.*, $\mathcal{F}_\gamma(\theta) = \mathbb{E}_{\|\theta'\|=\|\theta\|+\gamma}[\mathcal{E}(\theta') - \mathcal{E}(\theta)]$, of ERM, SAM, SWA, and SWAD by varying radius γ on different domains of PACS dataset. For each figure, Y-axis indicates the flatness $\mathcal{F}_\gamma(\theta)$ and X-axis indicates the radius γ. We measure the train flatness $\mathcal{F}_\gamma^{\mathcal{D}}(\theta)$ on seen domains and the test flatness $\mathcal{F}_\gamma^{\mathcal{T}}(\theta)$ on unseen domain. Each point is computed by Monte-Carlo approximation with 100 random samples. This comparisons show SWAD finds flatter minima than not only ERM but also SAM and SWA.

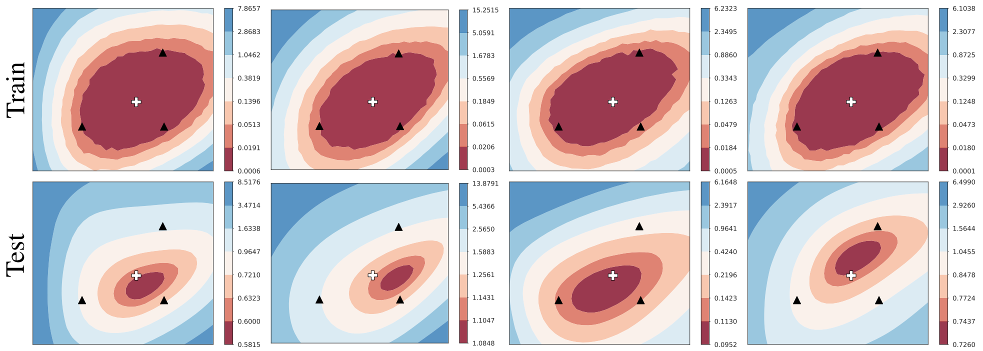
The main idea of Stochastic Weight Averaging Densely (SWAD) is a dense and overfit-aware stochastic weight gathering strategy. First, instead of collecting weights for every K epochs, SWAD collects weights for every iteration. This dense sampling strategy easily collects sufficiently many weights than the sparse one. We also employ overfit-aware sampling scheduling by considering traces of the validation loss. Instead of sampling weights from K_0 pretraining epochs to the final epoch, we search the start iteration (when the validation loss achieves a local optimum for the first time) and the end iteration (when the validation loss is no longer decreased, but keep increasing). More specifically, we introduce three parameters: an optimum patient parameter N_s, an overfitting patient parameter N_e, and the tolerance rate r for searching the start iteration t_s and the end iteration t_e. First, we search t_s which satisfies $\min_{i\in[0,\dots,N_s-1]} \mathcal{E}_{\text{val}}^{(t_s+i)} = \mathcal{E}_{\text{val}}^{(t_s)}$, where $\mathcal{E}_{\text{val}}^{(i)}$ denotes the validation loss at iteration i. Simply, t_s is the first iteration where the loss value is no longer decreased during N_s iterations. Then, we find t_e satisfying $\min_{i\in[0,1,\dots,N_e-1]} \mathcal{E}_{\text{val}}^{(t_e+i)} > r\mathcal{E}_{\text{val}}^{(t_s)}$. In other words, t_e is the first iteration where the validation loss values exceed the tolerance r during N_e iterations.

We illustrate the overview of SWAD and the comparison of SWAD to SWA in Figure 2. Detailed pseudo code is provided in Appendix B.4. We compare SWAD with other possible SWA strategies in §4.3 and show that our design choice works better for DG tasks.

3.3 Empirical analysis of SWAD and flatness

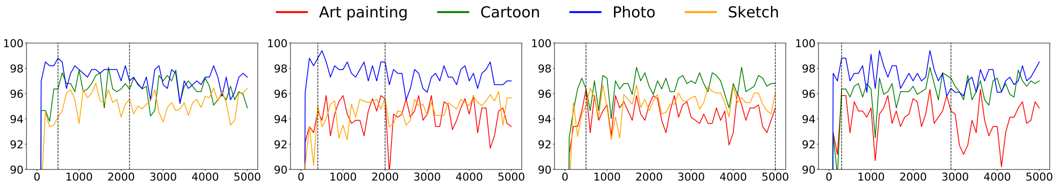
Here, we analyze solutions found by SWAD in terms of flatness. We first verify that the SWAD solution is flatter than those of ERM, SWA, and SAM. Our loss surface visualization shows that the SWAD solution is located on the center of the flat region, while ERM finds a boundary solution. Finally, we show that the sharp boundary solutions by ERM are not generalized well, resulting in sensitivity to the model selection. All following empirical analyses are conducted on PACS dataset, validating by all four domains (art painting, cartoon, photo, and sketch).

Local flatness anaylsis. To begin with, we quantify the local flatness of a model parameter θ by assuming that flat minima will have smaller changes of loss value within its neighborhoods than sharp minima. For the given model parameter θ, we compute the expected loss value changes between θ and parameters on the sphere surrounding θ with radius γ, *i.e.*, $\mathcal{F}_\gamma(\theta) = \mathbb{E}_{\|\theta'\|=\|\theta\|+\gamma}[\mathcal{E}(\theta') - \mathcal{E}(\theta)]$. In



Art painting (test) *Cartoon* (test) *Photo* (test) *Sketch* (test)

Figure 4: **Loss surfaces on model parameters in** `PACS` **dataset for each target domain.** The three triangles indicate model weights chosen at the end of training phase with equal intervals. Each plane is defined by the three weights and losses upon the plane are visualized with contours. The center cross mark is averaged point of the three weights. The first and second rows show the averaged training loss and the test loss surfaces, respectively.



(a) *Art painting* (test) (b) *Cartoon* (test) (c) *Photo* (test) (d) *Sketch* (test)

Figure 5: **Validation accuracies for in-domains.** The X- and Y-axis indicate the training iterations and accuracy, respectively, about the validation domains (legend) and the test domain (caption). The vertical dot lines represent start and end iterations, t_s and t_e, identified by the overfit-aware sampling strategy of SWAD.

practice, $\mathcal{F}_\gamma(\theta)$ is approximated by Monte-Carlo sampling with 100 samples. Note that the proposed local flatness $\mathcal{F}_\gamma(\theta)$ is computationally efficient than measuring curvature using the Hessian-based quantities. Also, $\mathcal{F}_\gamma(\theta)$ has an unbiased finite sample estimator, while the worst-case loss value, *i.e.*, $\max_{\|\theta'\|=\|\theta\|+\gamma}[\mathcal{E}(\theta') - \mathcal{E}(\theta)]$ has no unbiased finite sample estimator.

In Figure 3, we compare $\mathcal{F}_\gamma(\theta)$ of ERM, SAM, SWA with cyclic learning rate, SWA with constant learning rate, and SWAD by varying radius γ. SAM and SWA find the solutions with lower local flatness than ERM on average. SWAD finds the most flat minimum in every experiment.

Loss surface visualization. We visualize the loss landscapes by choosing three model weights on the optimization trajectory $(\theta_1, \theta_2, \theta_3)$[2], and computing the loss values by linear combinations of $\theta_1, \theta_2, \theta_3$[3] as [25]. More details are in Appendix B.5. In Figure 4, we observe that for all cases, ERM solutions are located at the boundary of a flat minimum of training loss, resulting in poor generalizability in test domains, that is aligned with our theoretical analysis and empirical flatness analysis. Since ERM solutions are located on the boundary of a flat loss surface, we observe that ERM solutions are very sensitive to model selection. In Figure 5, we illustrate the validation accuracies for each train-test domain combination of `PACS` by ERM, over training iterations (one epoch is equivalent to 83 iterations). We first observe that ERM rapidly reaches the best accuracy within only a few training epochs, namely less than 6 epochs. Furthermore, the ERM validation accuracies fluctuate a lot, and the final performance is very sensitive to the model selection criterion.

On the other hand, we observe that SWA solutions are located on the center of the training loss surfaces as well as of the test loss surfaces (Figure 4). Also, our overfit-aware stochastic weight gathering strategy (denoted as the vertical dot lines in Figure 5) prevents the ensembled weight from overfitting and makes SWAD model selection-free.

[2] We choose weights at iteration 2500, 3500, 4500 during the training.

[3] Each point is defined by two axes u and v computed by $u = \theta_2 - \theta_1$ and $v = \frac{(\theta_3-\theta_1)-\langle\theta_3-\theta_1,\theta_2-\theta_1\rangle}{\|\theta_2-\theta_1\|^2\cdot(\theta_2-\theta_1)}$.

Table 2: **Comparison with domain generalization methods and SWAD.** Out-of-domain accuracies on five domain generalization benchmarks are shown. We highlight the **best results** and the second best results. Note that ERM (reproduced), Mixstyle are reproduced numbers, and other numbers are from the original literature and Gulrajani and Lopez-Paz [22] (denoted with †). Our experiments are repeated three times.

Algorithm	`PACS`	`VLCS`	`OfficeHome`	`TerraInc`	`DomainNet`	**Avg.**
MASF [14]	82.7	-	-	-	-	-
DMG [33]	83.4	-	-	-	43.6	-
MetaReg [15]	83.6	-	-	-	43.6	-
ER [12]	85.3	-	-	-	-	-
pAdaIN [47]	85.4	-	-	-	-	-
EISNet [48]	85.8	-	-	-	-	-
DSON [30]	86.6	-	-	-	-	-
ERM† [29]	85.5	77.5	66.5	46.1	40.9	63.3
ERM (reproduced)	84.2	77.3	67.6	47.8	44.0	64.2
IRM† [20]	83.5	78.6	64.3	47.6	33.9	61.6
GroupDRO† [49]	84.4	76.7	66.0	43.2	33.3	60.7
I-Mixup† [50–52]	84.6	77.4	68.1	47.9	39.2	63.4
MLDG† [13]	84.9	77.2	66.8	47.8	41.2	63.6
CORAL† [31]	86.2	78.8	68.7	47.7	41.5	64.5
MMD† [53]	84.7	77.5	66.4	42.2	23.4	58.8
DANN† [9]	83.7	78.6	65.9	46.7	38.3	62.6
CDANN† [10]	82.6	77.5	65.7	45.8	38.3	62.0
MTL† [54]	84.6	77.2	66.4	45.6	40.6	62.9
SagNet† [32]	86.3	77.8	68.1	48.6	40.3	64.2
ARM† [16]	85.1	77.6	64.8	45.5	35.5	61.7
VREx† [21]	84.9	78.3	66.4	46.4	33.6	61.9
RSC† [55]	85.2	77.1	65.5	46.6	38.9	62.7
Mixstyle [17]	85.2	77.9	60.4	44.0	34.0	60.3
SWAD (ours)	**88.1** (±0.1)	**79.1** (±0.1)	**70.6** (±0.2)	**50.0** (±0.3)	**46.5** (±0.1)	**66.9**

4 Experiments

4.1 Evaluation protocols

Dataset and optimization protocol. Following Gulrajani and Lopez-Paz [22], we exhaustively evaluate our method and comparison methods on various benchmarks: `PACS` [7] (9,991 images, 7 classes, and 4 domains), `VLCS` [43] (10,729 images, 5 classes, and 4 domains), `OfficeHome` [44] (15,588 images, 65 classes, and 4 domains), `TerraIncognita` [45] (24,788 images, 10 classes, and 4 domains), and `DomainNet` [46] (586,575 images, 345 classes, and 6 domains).

For a fair comparison, we follow training and evaluation protocol by Gulrajani and Lopez-Paz [22], including the dataset splits, hyperparameter (HP) search and model selection (while SWAD does not need it) on the validation set, and optimizer HP, except the HP search space and the number of iterations for `DomainNet`. We use a reduced HP search space to reduce the computational costs. We also tripled the number of iterations for `DomainNet` from 5,000 to 15,000 because we observe that 5,000 is not sufficient to convergence. We re-evaluate ERM with 15,000 iterations, and observe 3.1pp average performance improvement ($40.9\% \rightarrow 44.0\%$) in `DomainNet`. For training, we choose a domain as the target domain and use the remaining domains as the training domain where 20% samples are used for validation and model selection. ImageNet [42] trained ResNet-50 [41] is employed as the initial weight, and optimized by Adam [38] optimizer with a learning rate of 5e-5. We construct a mini-batch containing all domains where each domain has 32 images. We set SWAD HPs N_s to 3, N_e to 6, and r to 1.2 for `VLCS` and 1.3 for the others by HP search on the validation sets. Additional implementation details, such as other HPs, are given in Appendix B.

Evaluation metrics. We report out-of-domain accuracies for each domain and their average, *i.e.*, a model is trained and validated on training domains and evaluated on the unseen target domain. Each out-of-domain performance is an average of three different runs with different train-validation splits.

4.2 Main results

Comparison with domain generalization methods. We report the full out-of-domain performances on five DG benchmarks in Table 2. The full tables including out-of-domain accuracies for each domain are in Appendix E. In all experiments, our SWAD achieves significant performance gain against ERM as well as the previous best results: +2.6pp in `PACS`, +0.3pp in `VLCS`, +1.4pp in `TerraIncognita`, +1.9pp in `OfficeHome`, and +2.9pp in `DomainNet` comparing to the previous best results. We observe that SWAD provides two practical advantages comparing to previous methods. First, SWAD does not need any modification on training objectives or model architecture, *i.e.*, it is universally applicable to any other methods. As an example, we show that SWAD actually improves the performances of other DG methods, such as CORAL [31] in Table 4. Moreover, as we discussed before, SWAD is free to the model selection, resulting in stable performances (*i.e.*, small standard errors) on various benchmarks. Note that we only compare results with ResNet-50 backbone for a fair comparison. We describe the implementation details of each comparison method and the hyperparameter search protocol in Appendix B.

Table 3: **Comparison between generalization methods on `PACS`.** The scores are averaged over all settings using different target domains. (↑) and (↓) indicate statistically significant improvement and degradation from ERM.

	Out-of-domain	In-domain
ERM	85.3±0.4	96.6±0.0
EMA	85.5±0.4(-)	97.0±0.1(↑)
SAM	85.5±0.1(-)	97.4±0.1(↑)
Mixup	84.8±0.3(-)	97.3±0.1(↑)
CutMix	83.8±0.4(↓)	97.6±0.1(↑)
VAT	85.4±0.6(-)	96.9±0.2(↑)
Π-model	83.5±0.5(↓)	96.8±0.2(↑)
SWA	85.9±0.1(↑)	97.1±0.1(↑)
SWAD	**87.1**±0.2(↑)	**97.7**±0.1(↑)

Comparison with conventional generalization methods. We also compare SWAD with other conventional generalization methods to show that the remarkable domain generalization gaps by SWAD is not achieved by better generalization, but by seeking flat minima. The comparison methods include flatness-aware optimization methods, such as SAM [26], ensemble methods, such as EMA [56], data augmentation methods, such as Mixup [35] and CutMix [36], and consistency regularization methods, such as VAT [57] and Π-model [58]. We also split in-domain datasets into training (60%), validation (20%), and test (20%) splits, while no in-domain test set used for Table 2. Every experiment is repeated three times.

The results are shown in Table 3. We observe that all conventional methods helps in-domain generalization, *i.e.*, performing better than ERM on in-domain test set. However, their out-of-domain performances are similar to or even worse than ERM. For example, CutMix and Π-model improve in-domain performances by 1.0pp and 0.2pp but degrade out-of-domain performances by 1.5pp and 1.8pp. SAM, another method for seeking flat minima, slightly increases both in-domain and out-of-domain performances but the out-of-domain performance is not statistically significant. We will discuss performances of SAM in other benchmarks later. In contrast, the vanilla SWA and our SWAD significantly improve both in-domain and out-of-domain performances. SWAD improves the performances by SWA with statistically significantly gaps: 1.2pp on the out-of-domain and 0.6pp on the in-domain. Further comparison between SWA and SWAD is provided in §4.3.

Table 4: **Combination of SWAD and other methods.** The scores are averaged over every target domain case. The performances of ERM, CORAL, and SAM are optimized by HP searches of DomainBed. In contrast, for the SWAD combination cases, CORAL and SAM use default HPs without additional HP search. We additionally compare SWAD to $\text{SWA}_{\text{w/ const}}$. Note that ERM + SWAD is same as "SWAD" in Table 2.

	`PACS`	`VLCS`	`OfficeHome`	`TerraInc`	`DomainNet`	Avg. (Δ)
ERM	85.5 ±0.2	77.5 ±0.4	66.5 ±0.3	46.1 ±1.8	40.9 ±0.1	63.3
ERM + $\text{SWA}_{\text{w/ const}}$	86.9 ±0.2	76.6 ±0.1	69.3 ±0.3	49.2 ±1.2	45.9 ±0.0	65.6 (+2.3)
ERM + SWAD	88.1 ±0.1	79.1 ±0.1	70.6 ±0.2	50.0 ±0.3	46.5 ±0.1	66.9 (+3.6)
CORAL	86.2 ±0.3	78.8 ±0.6	68.7 ±0.3	47.6 ±1.0	41.5 ±0.1	64.5
CORAL + SWAD	88.3 ±0.1	78.9 ±0.1	71.3 ±0.1	51.0 ±0.1	46.8 ±0.0	67.3 (+2.8)
SAM	85.8 ±0.2	79.4 ±0.1	69.6 ±0.1	43.3 ±0.7	44.3 ±0.0	64.5
SAM + SWAD	87.1 ±0.2	78.5 ±0.2	69.9 ±0.1	45.3 ±0.9	46.5 ±0.1	65.5 (+1.0)

Combinations with other methods. Since SWAD does not require any modification on training procedures and model architectures, SWAD is universally applicable to any other methods. Here, we combine SWAD with ERM, CORAL [31], and SAM [26]. Results are shown in Table 4. Both CORAL and SAM solely show better performances than ERM with +1.2pp average out-of-domain accuracy gap. Note that SAM is not a DG method but a sharpness-aware optimization method to find flat minima. It supports our theoretical motivation: DG can be achieved by seeking flat minima.

By applying SWAD on the baselines, the performances are consistently improved by 3.6pp on ERM, 2.8pp on CORAL, and 1.0pp on SAM. Interestingly, CORAL + SWAD show the best performances with both incorporating different advantages of utilizing domain labels and seeking flat minima. We also observe that SAM + SWAD shows worse performance than ERM + SWAD, while SAM performs better than ERM. We conjecture that it is because the objective control by SAM restricts the model parameter diversity durinig training, reducing the diversity for SWA ensemble. However, applying SWAD on SAM still leads to better performances than the sole SAM. The results demonstrate that the application of SWAD on other baselines is a simple yet effective method for DG.

4.3 Ablation study

Table 5: **Ablation studies of the stochastic weights selection strategies on `PACS` and `VLCS`.** In the configuration, "t_s", "t_e", "lr", and "interval" indicate start and end iterations of sampling, a learning rate schedule, and a stochastic weight sampling interval, respectively. "Opt" and "Overfit" indicate the start and end iterations identified by our overfit-aware sampling strategy, and "Val" means the start and end iterations whose averaging shows the best accuracy on the validation set. "Cyclic" and "Const" represent cyclic and constant learning rate schedules. All experiments are repeated three times.

	Configuration				Out-of-domain			In-domain		
	t_s	t_e	lr	interval	`PACS`	`VLCS`	Avg.	`PACS`	`VLCS`	Avg.
SWA$_{\text{w/ cyclic}}$	4000	5000	Cyclic	100	85.9 ±0.1	76.6 ±0.1	81.2	97.1 ±0.1	85.0 ±0.2	91.0
SWA$_{\text{w/ const}}$	4000	5000	Const	100	86.5 ±0.3	76.7 ±0.2	81.6	97.3 ±0.1	85.0 ±0.2	91.1
SWAD$_{\text{w/o Dense}}$	Opt	Overfit	Const	100	86.5 ±0.4	78.0 ±0.7	82.2	97.6 ±0.1	85.8 ±0.4	91.7
SWAD$_{\text{w/o Opt-Overfit}}$	4000	5000	Const	1	86.6 ±0.6	76.9 ±0.3	81.7	97.5 ±0.1	85.2 ±0.1	91.3
SWAD$_{\text{w/o Overfit}}$	Opt	5000	Const	1	**87.1** ±0.3	77.6 ±0.1	82.4	**97.7** ±0.1	85.8 ±0.3	91.8
SWAD$_{\text{fit-on-val}}$	Val	Val	Const	1	86.2 ±0.2	78.6 ±0.1	82.4	97.5 ±0.2	85.8 ±0.3	91.7
SWAD (proposed)	Opt	Overfit	Const	1	**87.1** ±0.2	**78.9** ±0.2	**83.0**	**97.7** ±0.1	**86.1** ±0.5	**91.9**

Table 5 provides ablative studies on the starting and ending iterations for averaging, the learning rate schedule, and the sampling interval. SWA$_{\text{w/ cyclic}}$ (SWA in Table 3) and SWA$_{\text{w/ constant}}$ are vanilla SWAs with fixed sampling positions. We also report SWAD by eliminating three factors: the dense sampling strategy, and searching the start iteration, searching the end iteration. The dense sampling strategy lets SWAD estimate a more accurate approximation of flat minima: showing 0.8pp degeneration in the average out-of-domain accuracy (SWAD$_{\text{w/o Dense}}$). When we take an average from t_s to the final iteration, the out-of-domain performance degrades by 0.6pp (SWAD$_{\text{w/o Overfit}}$). Similarly, a fixed scheduling without the overfit-aware scheduling only shows very marginal improvements from the vanilla SWA (SWAD$_{\text{w/o Opt-Overfit}}$). We also evaluate SWAD$_{\text{fit-on-val}}$ that uses the range achieving the best performances on the validation set, but it becomes overfitted to the validation, results in lower performances than SWAD. The results demonstrate the benefits of combining "dense" and "overfit-aware" sampling strategies of SWAD.

4.4 Exploring the other applications: ImageNet robustness

Table 6: **ImageNet robustness benchmarks.** We show the ImageNet generalization performances on ImageNet-C, background challenge (BGC), and ImageNet-R.

Method	ImageNet (%) ↑	ImageNet-C (mCE) ↓	BGC (%) ↑	ImageNet-R (%) ↑
ERM	76.5	57.6	8.7	36.7
SWA	76.9	56.8	10.9	37.5
SWAD (ours)	**77.0**	**55.7**	**11.8**	**38.8**

Since SWAD does not rely on domain labels, it can be applied to other robustness tasks not containing domain labels. Table 6 show the generalizability of SWAD on ImageNet [42] and its shifted benchmarks, namely, ImageNet-C [59], ImageNet-R [60], and background challenge (BGC) [61]. SWAD consistently improves robustness performances against the ERM baseline and the SWA baseline. These results support that our method is robustly and widely applicable to improve both in-domain and out-of-domain generalizability. The detailed setup is provided in Appendix B.6.

5 Discussion and Limitations

Despite many benefits from SWAD, such as the significant performance improvements, model selection-free property, working plug-and-play manner for various methods, there are some potential limitations. Here, we discuss the limitations of SWAD for further improvements.

Confidence error in Theorem 1. While the confidence error in Theorem 1 tells the effect of γ on generalization error bound, there exists a limitation in that the confidence error term shows improper behavior with respect to γ if γ is close to zero. The behavior we expect is that the confidence error of RRM converges to the confidence error of ERM as γ decreases to zero, however, the current theorem does not show such tendency since the confidence bound diverges to infinity when γ goes to zero. However, we would like to note that this limitation is not a drawback of RRM, but it is caused by the looseness of the union bound which is a mathematical technique used to derive the confidence error of RRM. Our RRM formulation has a similarity to previous works [26, 34] and we note that the counter-intuitive behavior of the confidence bound and γ also appears in Foret et al. [26].

SWAD is not a perfect flatness-aware optimization method. Note that SWAD is not a perfect and theoretically guaranteed solver for flat minima, but a heuristic approximation with empirical benefits. However, even if a better flatness-aware optimization method is proposed, our theoretical contribution still holds: showing the relationship between flat minima and DG.

SWAD does not strongly utilize domain-specific information. In Theorem 2, the domain generalization gap is bounded by three factors: flat minima, domain discrepancy, and confidence bound. Most of the existing approaches focus on domain discrepancy, reducing the difference between the source domains and the target domain by domain invariant learning [8–12]. SWAD focuses on the first factor, the flat minima. While the domain labels are used to construct a mini-batch, SWAD does not strongly utilize domain-specific information. It implies that if one can consider both flatness and domain discrepancy, better domain generalization can be achievable. Table 4 gives us a clue: the combination of CORAL (utilizing domain-specific information) and SWAD (seeking flat minima) shows the best performance among all comparison methods. As a future research direction, we encourage studying a method that can achieve both flat optima and small domain discrepancy.

6 Concluding Remarks

In this paper, we theoretically and empirically demonstrate that domain generalization (DG) is achievable by seeking flat minima. We propose SWAD that captures flatter minima than the vanilla SWA does. The extensive experiments on five DG benchmarks show superior performances of SWAD compared with existing DG methods. In addition, combinations of SWAD and existing DG methods even show better performances than the vanilla SWAD. We theoretically and empirically observe that seeking flat minima can achieve better generalizability to both in-domain and out-of-domain, while strong in-domain generalization methods without consideration of flatness, *e.g.*, Mixup or CutMix, cannot guarantee to achieve out-of-domain generalizability in both theory and practice. This study first brings the concept of flatness into DG tasks, and shows strong empirical performances not only in DG but also in ImageNet benchmarks. We hope that this study promotes a new research direction of seeking flat minima for domain generalization and other robustness tasks.

Acknowledgement

NAVER Smart Machine Learning (NSML) [62] and Kakao Brain Cloud platform have been used in experiments. This work was supported by IITP grant funded by the Korea government (MSIT) (No. 2021-0-01341, AI Graduate School Program, CAU).

References

[1] Dengxin Dai and Luc Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3819–3824. IEEE, 2018.

[2] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

[3] Terrance de Vries, Ishan Misra, Changhan Wang, and Laurens van der Maaten. Does object recognition work for everyone? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 52–59, 2019.

[4] Kaiyu Yang, Klint Qinami, Li Fei-Fei, Jia Deng, and Olga Russakovsky. Towards fairer datasets: Filtering and balancing the distribution of the people subtree in the imagenet hierarchy. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 547–558, 2020.

[5] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

[6] 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.

[7] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *IEEE International Conference on Computer Vision*, pages 5542–5550, 2017.

[8] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013.

[9] 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.

[10] Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain generalization via conditional invariant representations. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[11] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning (ICML)*, 2020.

[12] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. *Neural Information Processing Systems*, 33, 2020.

[13] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[14] Qi Dou, Daniel C Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *Neural Information Processing System*, 2019.

[15] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Neural Information Processing Systems*, 31:998–1008, 2018.

[16] 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.

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

[18] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018.

[19] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

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

[21] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). *arXiv preprint arXiv:2003.00688*, 2020.

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

[23] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[24] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Neural Information Processing Systems*, 2018.

[25] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *Conference on Uncertainty in Artificial Intelligence*, 2018.

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

[27] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Uncertainty in Artificial Intelligence*, 2017.

[28] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[29] V Vapnik. Statistical learning theory. *NY: Wiley*, 1998.

[30] Seonguk Seo, Yumin Suh, Dongwan Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. *European Conference on Computer Vision*, 2020.

[31] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[32] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8690–8699, 2021.

[33] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *European Conference on Computer Vision*, pages 301–318. Springer, 2020.

[34] Matthew Norton and Johannes O Royset. Diametrical risk minimization: Theory and computations. *arXiv preprint arXiv:1910.10844*, 2019.

[35] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[36] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*, 2019.

[37] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1): 151–175, 2010.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[39] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12): 124018, 2019.

[40] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

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

[43] Chen Fang, Ye Xu, and Daniel N Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *IEEE International Conference on Computer Vision*, pages 1657–1664, 2013.

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

[45] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *European Conference on Computer Vision*, pages 456–473, 2018.

[46] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[47] Oren Nuriel, Sagie Benaim, and Lior Wolf. Permuted adain: Reducing the bias towards global statistics in image classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[48] Shujun Wang, Lequan Yu, Caizi Li, Chi-Wing Fu, and Pheng-Ann Heng. Learning from extrinsic and intrinsic supervisions for domain generalization. In *European Conference on Computer Vision*, pages 159–176. Springer, 2020.

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

[50] 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*, volume 34, pages 6502–6509, 2020.

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

[52] Yufei Wang, Haoliang Li, and Alex C Kot. Heterogeneous domain generalization via domain mixup. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3622–3626. IEEE, 2020.

[53] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[54] 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.

[55] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. *European Conference on Computer Vision*, 2, 2020.

[56] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[57] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2018.

[58] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017.

[59] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

[60] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020.

[61] Kai Yuanqing Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. In *International Conference on Learning Representations*, 2020.

[62] Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*, 2018.

[63] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[64] Han Zhao, Shanghang Zhang, Guanhang Wu, José M. F. Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

A Potential Societal Impacts

In this study, we theoretically and empirically demonstrate that domain generalization (DG) is achievable by seeking flat minima, and propose SWAD to find flat minima. With SWAD, researchers and developers can make a model robust to domain shift in a real deployment environment, without relying on a task-dependent prior, a modified objective function, or a specific model architecture. Accordingly, SWAD has potential positive impacts by developing machines less biased towards ethical aspects, as well as potential negative impacts, *e.g.*, improving weapon or surveillance systems under unexpected environment changes.

B Implementation Details

B.1 Hyperparameters of SWAD

The evaluation protocol by Gulrajani and Lopez-Paz [22] is computationally too expensive; it requires about 4,142 models for every DG algorithm. Hence, we reduce the search space of SWAD for computational efficiency; batch size and learning rate are set to 32 for each domain and 5e-5, respectively. We set dropout probability and weight decay to zero. We only search N_s, N_e and r. N_s and N_e are searched in `PACS` dataset, and the searched values are used for all experiments, while r is searched in [1.2, 1.3] depending on dataset. As a result, we use $N_s = 3$, $N_e = 6$, and $r = 1.2$ for `VLCS` and $r = 1.3$ for the others. We initialize our model by ImageNet-pretrained ResNet-50 and batch normalization statistics are frozen during training. The number of total iterations is $15,000$ for DomainNet and $5,000$ for others, which are sufficient numbers to be converged. Finally, we slightly modify the evaluation frequency because it should be set to small enough to detect the moments that the model is optimized and overfitted. However, too small frequency brings large evaluation overhead, thus we compromise between exactness and efficiency: 50 for `VLCS`, 500 for `DomainNet`, and 100 for others.

B.2 Hyperparameter search protocol for reproduced results

Table 7: **Hyperparameter search space comparison.** U and list indicate Uniform distribution and random choice, respectively.

Parameter	Default value	DomainBed	Ours
batch size	32	$2^{\mathrm{U}(3,5.5)}$	32
learning rate	5e-5	$10^{\mathrm{U}(-5,-3.5)}$	[1e-5, 3e-5, 5e-5]
ResNet dropout	0	[0.0, 0.1, 0.5]	[0.0, 0.1, 0.5]
weight decay	0	$10^{\mathrm{U}(-6,-2)}$	[1e-4, 1e-6]

We evaluate recently proposed methods, SAM [26] and Mixstyle [17], and compare them with previous results. For a fair comparison, we follow the hyperparameter (HP) search protocol proposed by Gulrajani and Lopez-Paz [22], with a modification to reduce computational resources. They searched HP by training a total of 58,000 models, corresponding to about 4,142 runs for each algorithm. It is too much computational burden to train 4,142 models whenever evaluate a new algorithm. Therefore, we re-design the HP search protocol efficiently and effectively. In the HP search protocol of DomainBed [22], training domains and algorithm-specific parameters are included in the HP search space, and HP is found for every data split independently by random search. Instead, we do not sample training domains, use HP found in the first data split to the other splits, search algorithm-specific HP independently, and conduct grid search on the more effectively designed HP space as shown in Table 7. Through the proposed protocol, we find HP for an algorithm under only 396 runs. Although the number of total runs is reduced to about 10% ($4,142 \rightarrow 396$), the results of reproduced ERM is improved 0.9pp in average ($63.3\% \rightarrow 64.2\%$). It demonstrates both the effectiveness and the efficiency of our search protocol.

B.3 Algorithm-specific hyperparameters

We search the algorithm-specific hyperparameters independently in PACS dataset, based on the values suggested from each paper. For Mixstyle [17], we insert Mixstyle block with domain label after the 1st, 2nd, and 3rd residual blocks with $\alpha = 0.1$ and $p = 0.5$. We train SAM [26] with $\rho = 0.05$, and VAT [57] with $\epsilon = 1.0$ and $\alpha = 1.0$. In Π-model [58], $w_{max} = 1$ is chosen among various w_{max} values such as 1, 10, 100, and 300. We use EMA [56] with $decay = 0.99$, Mixup [35] with $\alpha = 0.2$, and CutMix [36] with $\alpha = 1.0$ and $p = 0.5$.

B.4 Pseudo code

Algorithm 1: Stochastic Weight Averaging Densely

```
Input: initial weight $\theta_0$, constant learning rate $\alpha$, tolerance rate $r$, optimum patience $N_s$, overfit
       patience $N_e$, total number of iterations $T$, training loss $\mathcal{E}^{(i)}_{\text{train}}$, validation loss $\mathcal{E}^{(i)}_{\text{val}}$
Output: averaged weight $\theta^{\text{SWAD}}$ from $t_s$ to $t_e$
$t_s \leftarrow 0$                                          // start iteration for averaging
$t_e \leftarrow T$                                          // end iteration for averaging
$l \leftarrow$ None                                         // loss threshold
for $i \leftarrow 1$ to $T$ do
    $\theta_i \leftarrow \theta_{i-1} - \alpha \nabla \mathcal{E}^{(i-1)}_{\text{train}}$
    if $l =$ None then
        if $\mathcal{E}^{(i-N_s+1)}_{val} = \min_{0 \leq i' < N_s} \mathcal{E}^{(i-i')}_{val}$ then
            $t_s \leftarrow i - N_s + 1$
            $l \leftarrow \frac{r}{N_s} \sum_{i'=0}^{N_s-1} \mathcal{E}^{(i-i')}_{\text{val}}$
    else if $l < \min_{0 \leq i' < N_e} \mathcal{E}^{(i-i')}_{val}$ then
        $t_e \leftarrow i - N_e$
        break
$\theta^{\text{SWAD}} \leftarrow \frac{1}{t_e - t_s + 1} \sum_{i'=t_s}^{t_e} \theta^{i'}$
```

B.5 Loss surface visualization

Following Garipov et al. [24], we choose three model weights $\theta_1, \theta_2, \theta_3$ and define two dimensional weight plane from the weights:

$$u = \theta_2 - \theta_1, \qquad v = \frac{(\theta_3 - \theta_1) - \langle \theta_3 - \theta_1, \theta_2 - \theta_1 \rangle}{\|\theta_2 - \theta_1\|^2 \cdot (\theta_2 - \theta_1)}, \tag{3}$$

where $\hat{u} = u / \|u\|$ and $\hat{v} = v / \|v\|$ are orthonormal bases of the weight plane. Then, we build Cartesian grid near the weights on the plane. For each grid point, we calculate the weight corresponding to the point and compute loss from the weight. The results are visualized as a contour plot, as shown in Figure 4 in the main text.

B.6 ImageNet robustness experiments

We investigate the extensibility of SWAD via three robustness benchmarks (Section 4.4 in the main text), namely ImageNet-C [59], ImageNet-R [60], and background challenge (BGC) [61]. ImageNet-C measures the robustness against common corruptions such as Gaussian noise, blur, or weather changes. We follow Hendrycks and Dietterich [59] for measuring mean corruption error (mCE). The lower ImageNet-C implies that the model is robust against corruption noises. BGC evaluates the robustness against background manipulations as well as the adversarial robustness. The BGC dataset has two groups, foreground and background. BGC manipulates images by combining the foregrounds and backgrounds, and measures whether the model predicts a consistent prediction with any manipulated image. ImageNet-R tests the robustness against different domains. ImageNet-R collects very different domain images of ImageNet, such as art, cartoons, deviantart, graffiti, embroidery, graphics, origami, paintings, patterns, plastic objects, plush objects, sculptures, sketches, tattoos, toys, and video game renditions. Showing better performances in ImageNet-R leads to the same conclusion as other domain generalization benchmarks.

Experiment details. We use ResNet-50 architecture and mostly follow standard training recipes. We use SGD optimizer with momentum of 0.9, base learning rate of 0.1 with linear scaling rule [63] and polynomial decay, 5 epochs gradual warmup, batch size of 2048, and total epochs of 90. For SWA, the learning rate is decayed to 1/20 until 80% of training (72 epochs), and the cyclic learning rate with 3 epochs cycle length is used for the left 20% of training. SWAD follows the same learning rate decay until 80% of training, but averages every weight from every iteration after 80% of training with constant learning rate.

C Proof of Theorems

C.1 Technical Lemmas

Consider an instance loss function $\ell(y_1, y_2)$ such that $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ and $\ell(y_1, y_2) = 0$ if and only if $y_1 = y_2$. Then, we can define a functional error as $\mathcal{E}_\mathcal{P}(f(\cdot;\theta), h) := \mathbb{E}_\mathcal{P}[\ell(f(x;\theta), h(x))]$. Note that if we set h as a true label function which generates the label of inputs, $y = h(x)$, then, it becomes a population loss $\mathcal{E}_\mathcal{P}(\theta) = \mathcal{E}_\mathcal{P}(f(\cdot;\theta), h)$. Given two distributions, $\mathcal{P}$ and $\mathcal{Q}$, the following lemma shows that the difference between the error with $\mathcal{P}$ and the error with $\mathcal{Q}$ is bounded by the divergence between $\mathcal{P}$ and $\mathcal{Q}$.

Lemma 1. $|\mathcal{E}_\mathcal{P}(h_1, h_2) - \mathcal{E}_\mathcal{Q}(h_1, h_2)| \le \frac{1}{2}\mathbf{Div}(\mathcal{P}, \mathcal{Q})$

Proof. We employ the same technique in Zhao et al. [64] for our loss function ℓ. From the Fubini's theorem, we have,

$$\mathbb{E}_{x\sim\mathcal{P}}[\ell(h_1(x), h_2(x))] = \int_0^\infty \mathbb{P}_\mathcal{P}\left(\ell(h_1(x), h_2(x)) > t\right) dt \tag{4}$$

By using this fact,

$$\begin{aligned}
&|\mathbb{E}_{x\sim\mathcal{P}}[\ell(h_1(x), h_2(x))] - \mathbb{E}_{x\sim\mathcal{Q}}[\ell(h_1(x), h_2(x))]| &(5)\\
&= \left|\int_0^\infty \mathbb{P}_\mathcal{P}\left(\ell(h_1(x), h_2(x)) > t\right) dt - \int_0^\infty \mathbb{P}_\mathcal{Q}\left(\ell(h_1(x), h_2(x)) > t\right) dt\right| &(6)\\
&\le \int_0^\infty \left|\mathbb{P}_\mathcal{P}\left(\ell(h_1(x), h_2(x)) > t\right) - \mathbb{P}_\mathcal{Q}\left(\ell(h_1(x), h_2(x)) > t\right)\right| dt &(7)\\
&\le M \sup_{t\in[0,M]} \left|\mathbb{P}_\mathcal{P}\left(\ell(h_1(x), h_2(x)) > t\right) - \mathbb{P}_\mathcal{Q}\left(\ell(h_1(x), h_2(x)) > t\right)\right| &(8)\\
&\le M \sup_{h_1,h_2} \sup_{t\in[0,M]} \left|\mathbb{P}_\mathcal{P}\left(\ell(h_1(x), h_2(x)) > t\right) - \mathbb{P}_\mathcal{Q}\left(\ell(h_1(x), h_2(x)) > t\right)\right| &(9)\\
&\le M \sup_{\bar{h}\in\bar{\mathcal{H}}} \left|\mathbb{P}_\mathcal{P}\left(\bar{h}(x) = 1\right) - \mathbb{P}_\mathcal{Q}\left(\bar{h}(x) = 1\right)\right| &(10)\\
&\le M \sup_A \left|\mathbb{P}_\mathcal{P}(A) - \mathbb{P}_\mathcal{Q}(A)\right| &(11)
\end{aligned}$$

where $\bar{\mathcal{H}} := \{\mathbb{I}[\ell(h(x), h'(x)) > t] | h, h' \in \mathcal{H}, t \in [0, M]\}$. □

Lemma 2. *Consider a distribution $\mathcal{S}$ on input space and global label function $f : \mathcal{X} \to \mathcal{Y}$. Let $\left\{\Theta_k \subset \mathbb{R}^d, k = 1, \cdots, N\right\}$ be a finite cover of a parameter space Θ which consists of closed balls with radius $\gamma/2$ where $N := \left\lceil (diam(\Theta)/\gamma)^d \right\rceil$. Let $\theta_k \in \arg\max_{\Theta_k\cap\Theta} \mathcal{E}_\mathcal{S}(\theta)$ be a local maximum in the k-th ball. Let a VC dimension of Θ_k be v_k. Then, for any $\theta \in \Theta$, the following bound holds with probability at least $1 - \delta$.*

$$\mathcal{E}_S(\theta) - \hat{\mathcal{E}}_S^\gamma(\theta) \le \max_k \sqrt{\frac{(v_k\left[\ln(n/v_k) + 1\right] + \ln(N/\delta))}{2n}} \tag{12}$$

where $\hat{\mathcal{E}}_S^\gamma(\theta_k)$ is an empirical robust risk with n samples.

Proof. We first show that the following inequality holds for the local maximum of N covers,

$$\begin{aligned}
\mathbb{P}\left(\max_k \left[\mathcal{E}_S(\theta_k) - \hat{\mathcal{E}}_S(\theta_k)\right] > \epsilon\right) &\leq \sum_{k=1}^{N} \mathbb{P}\left(\mathcal{E}_S(\theta_k) - \hat{\mathcal{E}}_S(\theta_k) > \epsilon\right) && (13)\\
&\leq \sum_{k=1}^{N} \mathbb{P}\left(\sup_{\theta\in\Theta_k} \left[\mathcal{E}_S(\theta) - \hat{\mathcal{E}}_S(\theta)\right] > \epsilon\right) && (14)\\
&\leq \sum_{k=1}^{N} \left(\frac{en}{v_k}\right)^{v_k} e^{-2n\epsilon^2}. && (15)
\end{aligned}$$

Now, we introduce a confidence error bound $\epsilon_k := \sqrt{\frac{(v_k[\ln(n/v_k)+1]+\ln(N/\delta))}{2n}}$. Then, we set $\epsilon := \max_k \epsilon_k$. Then, we get,

$$\begin{aligned}
\mathbb{P}\left(\max_k \left[\mathcal{E}_S(\theta_k) - \hat{\mathcal{L}}_S(\theta_k)\right] > \epsilon\right) &\leq \sum_{k=1}^{N} \left(\frac{en}{v_k}\right)^{v_k} e^{-2n\epsilon^2} && (16)\\
&\leq \sum_{k=1}^{N} \left(\frac{en}{v_k}\right)^{v_k} e^{-2n\epsilon_k^2} && (17)\\
&= \sum_{k=1}^{N} \frac{\delta}{N} = \delta, && (18)
\end{aligned}$$

since $\epsilon > \sqrt{\frac{(v_k[\ln(n/v_k)+1]+\ln(N/\delta))}{2n}}$ for all k. Hence, the inequality holds with probability at least $1-\delta$.

Based on this fact, let us consider the set of events such that $\max_k \left[\mathcal{E}_S(\theta_k) - \hat{\mathcal{E}}_S(\theta_k)\right] \leq \epsilon$. Then, for any θ, there exists k' such that $\theta \in \Theta_{k'}$. Then, we get

$$\begin{aligned}
\mathcal{E}_S(\theta) - \hat{\mathcal{E}}_S^{\gamma}(\theta) &\leq \mathcal{E}_S(\theta) - \hat{\mathcal{E}}_S(\theta_{k'}) && (19)\\
&\leq \mathcal{E}_S(\theta) - \mathcal{E}_S(\theta_{k'}) + \epsilon && (20)\\
&\leq \mathcal{E}_S(\theta_{k'}) - \mathcal{E}_S(\theta_{k'}) + \epsilon = \epsilon, && (21)
\end{aligned}$$

where the second inequality holds since $\mathcal{E}_S(\theta_{k'}) - \hat{\mathcal{E}}_S(\theta_{k'}) \leq \max_k \left[\mathcal{E}_S(\theta_k) - \hat{\mathcal{E}}_S(\theta_k)\right] \leq \epsilon$ and the final inequality holds since $\theta_{k'}$ is the local maximum in $\Theta_{k'}$. In this regards, we know that $\max_k \left[\mathcal{E}_S(\theta_k) - \hat{\mathcal{E}}_S(\theta_k)\right] \leq \epsilon$ implies $\mathcal{E}_S(\theta) - \hat{\mathcal{E}}_S^{\gamma}(\theta) \leq \epsilon$. Consequently, $\mathcal{E}_S(\theta) - \hat{\mathcal{E}}_S^{\gamma}(\theta) \leq \epsilon$ holds with probability at least $1-\delta$. □

C.2 Proof of Theorem 1

Proof. The proof consists of two parts. First, we show that the following inequality holds with high probability.

$$\mathcal{E}_{\mathcal{T}}(\theta) \leq \hat{\mathcal{E}}_{\mathcal{S}}^{\gamma}(\theta) + \frac{1}{2}\mathbf{Div}\left(\mathcal{S}, \mathcal{T}\right) + \max_k \sqrt{\frac{\left(v_k\left[\ln\left(n/v_k\right)+1\right]+\ln\left(N/\delta\right)\right)}{2n}}.$$

Then, secondly, we apply the inequality for multiple source domains.

The first part can be proven by simply combining Lemma 1 and Lemma 2. Then, we get,

$$\begin{aligned}
\mathcal{E}_{\mathcal{T}}(\theta) &\leq \mathcal{E}_{\mathcal{S}}(\theta) + \frac{1}{2}\mathbf{Div}\left(\mathcal{S}, \mathcal{T}\right) && (22)\\
&\leq \hat{\mathcal{E}}_{\mathcal{S}}^{\gamma}(\theta) + \frac{1}{2}\mathbf{Div}\left(\mathcal{S}, \mathcal{T}\right) + \max_k \sqrt{\frac{\left(v_k\left[\ln\left(n/v_k\right)+1\right]+\ln\left(N/\delta\right)\right)}{2n}} && (23)
\end{aligned}$$

where $\mathbf{Div}\left(\mathcal{S}, \mathcal{T}\right)$ is a divergence between $\mathcal{S}$ and $\mathcal{T}$.

For the second part, we set $\mathcal{D} := \sum_{i=1}^{I} \mathcal{D}_i / I$ which is a mixture of source distributions. Then, by applying $\mathcal{D}$ to the first part, we obtain the following inequality,

$$\mathcal{E}_{\mathcal{T}}(\theta) \leq \hat{\mathcal{E}}_{\mathcal{D}}^{\gamma}(\theta) + \frac{1}{2}\mathbf{Div}\left(\mathcal{D}, \mathcal{T}\right) + \max_{k} \sqrt{\frac{\left(v_k \left[\ln\left(In/v_k\right) + 1\right] + \ln\left(N/\delta\right)\right)}{2In}} \tag{24}$$

$$\leq \hat{\mathcal{E}}_{\mathcal{D}}^{\gamma}(\theta) + \frac{1}{2I}\sum_{i=1}^{I}\mathbf{Div}\left(\mathcal{D}_i, \mathcal{T}\right) + \max_{k} \sqrt{\frac{\left(v_k \left[\ln\left(In/v_k\right) + 1\right] + \ln\left(N/\delta\right)\right)}{2In}} \tag{25}$$

where the total number of training data set is In and, for the second inequality, we use the fact that $\frac{1}{2}\mathbf{Div}\left(\mathcal{D}, \mathcal{T}\right) \leq \frac{1}{2I}\sum_{i=1}^{I}\mathbf{Div}\left(\mathcal{D}_i, \mathcal{T}\right)$, which has been proven in [64]. □

C.3 Proof of Theorem 2

Proof. First, let $\bar{\theta} \in \arg\max_{\theta\in\Theta} \mathcal{E}_{\mathcal{T}}(\theta)$. Then, from generalization error bound of $\mathcal{E}_{\mathcal{D}}(\bar{\theta})$, the following inequality holds with probability at most $\frac{\delta}{2}$,

$$\hat{\mathcal{E}}_{\mathcal{D}}(\bar{\theta}) - \mathcal{E}_{\mathcal{D}}(\bar{\theta}) > \sqrt{\frac{v\ln\left(In/v\right) + \ln\left(2/\delta\right)}{In}}, \tag{26}$$

where v is a VC dimension of Θ. Furthermore, from Theorem 1, we have the following inequality with probability at most $\frac{\delta}{2}$,

$$\mathcal{E}_{\mathcal{T}}(\hat{\theta}^{\gamma}) > \mathcal{E}_{\mathcal{D}}^{\gamma}(\hat{\theta}^{\gamma}) + \frac{1}{2}\mathbf{Div}(\mathcal{D}, \mathcal{T}) + \max_{k\in[1,N]} \sqrt{\frac{v_k \ln\left(In/v_k\right) + \ln(2N/\delta)}{In}}. \tag{27}$$

Finally, let us consider the set of event such that $\hat{\mathcal{E}}_{\mathcal{D}}(\bar{\theta}) - \mathcal{E}_{\mathcal{D}}(\bar{\theta}) \leq \sqrt{\frac{v\ln(In/v)+\ln(2/\delta)}{In}}$ and $\mathcal{E}_{\mathcal{T}}(\hat{\theta}^{\gamma}) \leq \mathcal{E}_{\mathcal{D}}^{\gamma}(\hat{\theta}^{\gamma}) + \frac{1}{2}\mathbf{Div}(\mathcal{D}, \mathcal{T}) + \max_{k\in[1,N]} \sqrt{\frac{v_k \ln(In/v_k)+\ln(2N/\delta)}{In}}$ whose probability is at least greater than $1-\delta$. Then, under this set of event, we have,

$$\min_{\theta'} \hat{\mathcal{E}}_{\mathcal{D}}(\theta') \leq \hat{\mathcal{E}}_{\mathcal{D}}(\bar{\theta}) \leq \mathcal{E}_{\mathcal{D}}(\bar{\theta}) + \sqrt{\frac{v\ln\left(In/v\right) + \ln\left(2/\delta\right)}{In}} \tag{28}$$

$$\leq \mathcal{E}_{\mathcal{T}}(\bar{\theta}) + \frac{1}{2}\mathbf{Div}(\mathcal{D}, \mathcal{T}) + \sqrt{\frac{v\ln\left(In/v\right) + \ln\left(2/\delta\right)}{In}} \tag{29}$$

$$\leq \min_{\theta'} \mathcal{E}_{\mathcal{T}}\left(\theta'\right) + \frac{1}{2}\mathbf{Div}(\mathcal{D}, \mathcal{T}) + \sqrt{\frac{v\ln\left(In/v\right) + \ln\left(2/\delta\right)}{In}} \tag{30}$$

Consequently, we have,

$$\begin{aligned}
&\mathcal{E}_{\mathcal{T}}(\hat{\theta}^{\gamma}) - \min_{\theta'} \mathcal{E}_{\mathcal{T}}\left(\theta'\right) \\
&\leq \mathcal{E}_{\mathcal{D}}^{\gamma}(\hat{\theta}^{\gamma}) - \min_{\theta'} \hat{\mathcal{E}}_{\mathcal{D}}\left(\theta'\right) + \mathbf{Div}(\mathcal{D}, \mathcal{T}) + \max_{k\in[1,N]} \sqrt{\frac{v_k \ln\left(In/v_k\right) + \ln(2N/\delta)}{In}} \\
&\quad + \sqrt{\frac{v\ln\left(In/v\right) + \ln\left(2/\delta\right)}{In}} \qquad (31) \\
&\leq \mathcal{E}_{\mathcal{D}}^{\gamma}(\hat{\theta}^{\gamma}) - \min_{\theta'} \hat{\mathcal{E}}_{\mathcal{D}}\left(\theta'\right) + \frac{1}{I}\sum_{i=1}^{I}\mathbf{Div}(\mathcal{D}_i, \mathcal{T}) + \max_{k\in[1,N]} \sqrt{\frac{v_k \ln\left(In/v_k\right) + \ln(2N/\delta)}{In}} \\
&\quad + \sqrt{\frac{v\ln\left(In/v\right) + \ln\left(2/\delta\right)}{In}} \qquad (32)
\end{aligned}$$

□

D Additional Experiments

D.1 Comparison of flatness-aware solvers

Interestingly, the average performance ranking of flatness-aware solvers is the same as the results of the local flatness test (See Figure 3 in the main text). In both experiments, SWAD performs best,

Table 8: **Flatness-aware solvers comparison.** SWAs collect 10 weights from the last 20% of training.

Algorithm	PACS	VLCS	OfficeHome	TerraInc	DomainNet	Avg.
ERM (baseline)	85.5 ±0.2	77.5 ±0.4	66.5 ±0.3	46.1 ±1.8	40.9 ±0.1	63.3
SAM	85.8 ±0.2	**79.4** ±0.1	69.6 ±0.1	43.3 ±0.7	44.3 ±0.0	64.5
SWA$_{\text{w/ cyclic}}$	87.1 ±0.1	76.5 ±0.2	68.5 ±0.2	49.6 ±1.0	45.6 ±0.0	65.5
SWA$_{\text{w/ const}}$	86.9 ±0.2	76.6 ±0.1	69.3 ±0.3	49.2 ±1.2	45.9 ±0.0	65.6
SWAD	**88.1** ±0.1	79.1 ±0.1	**70.6** ±0.2	**50.0** ±0.3	**46.5** ±0.1	**66.9**

followed by SWAs, SAM, and ERM. It is another evidence of our claim that domain generalization is achievable by seeking flat minima.

On the other hand, comparing SWAs and SWAD demonstrates the effectiveness of the proposed dense and overfit-aware sampling strategy. SWAD improves average performance up to 1.4pp, and surpasses both SWAs on every benchmark.

E Full Results

In this section, we show detailed results of Table 2 in the main text. † and ‡ indicate results from DomainBed's and our HP search protocols, respectively. Standard errors are reported from three trials, if available.

E.1 PACS

Table 9: **Out-of-domain accuracies (%) on PACS.**

Algorithm	**A**	**C**	**P**	**S**	**Avg**
CDANN†	84.6 ±1.8	75.5 ±0.9	96.8 ±0.3	73.5 ±0.6	82.6
MASF	82.9	80.5	95.0	72.3	82.7
DMG	82.6	78.1	94.5	78.3	83.4
IRM†	84.8 ±1.3	76.4 ±1.1	96.7 ±0.6	76.1 ±1.0	83.5
MetaReg	87.2	79.2	97.6	70.3	83.6
DANN†	86.4 ±0.8	77.4 ±0.8	97.3 ±0.4	73.5 ±2.3	83.7
ERM‡	85.7 ±0.6	77.1 ±0.8	97.4 ±0.4	76.6 ±0.7	84.2
GroupDRO†	83.5 ±0.9	79.1 ±0.6	96.7 ±0.3	78.3 ±2.0	84.4
MTL†	87.5 ±0.8	77.1 ±0.5	96.4 ±0.8	77.3 ±1.8	84.6
I-Mixup	86.1 ±0.5	78.9 ±0.8	97.6 ±0.1	75.8 ±1.8	84.6
MMD†	86.1 ±1.4	79.4 ±0.9	96.6 ±0.2	76.5 ±0.5	84.7
VREx†	86.0 ±1.6	79.1 ±0.6	96.9 ±0.5	77.7 ±1.7	84.9
MLDG†	85.5 ±1.4	80.1 ±1.7	97.4 ±0.3	76.6 ±1.1	84.9
ARM†	86.8 ±0.6	76.8 ±0.5	97.4 ±0.3	79.3 ±1.2	85.1
RSC†	85.4 ±0.8	79.7 ±1.8	97.6 ±0.3	78.2 ±1.2	85.2
Mixstyle‡	86.8 ±0.5	79.0 ±1.4	96.6 ±0.1	78.5 ±2.3	85.2
ER	87.5	79.3	**98.3**	76.3	85.3
pAdaIN	85.8	81.1	97.2	77.4	85.4
ERM†	84.7 ±0.4	80.8 ±0.6	97.2 ±0.3	79.3 ±1.0	85.5
EISNet	86.6	81.5	97.1	78.1	85.8
CORAL†	88.3 ±0.2	80.0 ±0.5	97.5 ±0.3	78.8 ±1.3	86.2
SagNet†	87.4 ±1.0	80.7 ±0.6	97.1 ±0.1	80.0 ±0.4	86.3
DSON	87.0	80.6	96.0	**82.9**	86.6
Ours	**89.3** ±0.2	**83.4** ±0.6	97.3 ±0.3	82.5 ±0.5	**88.1**

E.2 VLCS

Table 10: **Out-of-domain accuracies (%) on `VLCS`.**

Algorithm	**C**	**L**	**S**	**V**	**Avg**
GroupDRO†	97.3 ±0.3	63.4 ±0.9	69.5 ±0.8	76.7 ±0.7	76.7
RSC†	97.9 ±0.1	62.5 ±0.7	72.3 ±1.2	75.6 ±0.8	77.1
MLDG†	97.4 ±0.2	65.2 ±0.7	71.0 ±1.4	75.3 ±1.0	77.2
MTL†	97.8 ±0.4	64.3 ±0.3	71.5 ±0.7	75.3 ±1.7	77.2
ERM‡	98.0 ±0.3	64.7 ±1.2	71.4 ±1.2	75.2 ±1.6	77.3
I-Mixup	98.3 ±0.6	64.8 ±1.0	72.1 ±0.5	74.3 ±0.8	77.4
ERM†	97.7 ±0.4	64.3 ±0.9	73.4 ±0.5	74.6 ±1.3	77.5
MMD†	97.7 ±0.1	64.0 ±1.1	72.8 ±0.2	75.3 ±3.3	77.5
CDANN†	97.1 ±0.3	65.1 ±1.2	70.7 ±0.8	77.1 ±1.5	77.5
ARM†	98.7 ±0.2	63.6 ±0.7	71.3 ±1.2	76.7 ±0.6	77.6
SagNet†	97.9 ±0.4	64.5 ±0.5	71.4 ±1.3	77.5 ±0.5	77.8
Mixstyle‡	98.6 ±0.3	64.5 ±1.1	72.6 ±0.5	75.7 ±1.7	77.9
VREx†	98.4 ±0.3	64.4 ±1.4	74.1 ±0.4	76.2 ±1.3	78.3
IRM†	98.6 ±0.1	64.9 ±0.9	73.4 ±0.6	77.3 ±0.9	78.6
DANN†	**99.0** ±0.3	65.1 ±1.4	73.1 ±0.3	77.2 ±0.6	78.6
CORAL†	98.3 ±0.1	**66.1** ±1.2	73.4 ±0.3	77.5 ±1.2	78.8
Ours	98.8 ±0.1	63.3 ±0.3	**75.3** ±0.5	**79.2** ±0.6	**79.1**

E.3 OfficeHome

Table 11: **Out-of-domain accuracies (%) on `OfficeHome`.**

Algorithm	**A**	**C**	**P**	**R**	**Avg**
Mixstyle‡	51.1 ±0.3	53.2 ±0.4	68.2 ±0.7	69.2 ±0.6	60.4
IRM†	58.9 ±2.3	52.2 ±1.6	72.1 ±2.9	74.0 ±2.5	64.3
ARM†	58.9 ±0.8	51.0 ±0.5	74.1 ±0.1	75.2 ±0.3	64.8
RSC†	60.7 ±1.4	51.4 ±0.3	74.8 ±1.1	75.1 ±1.3	65.5
CDANN†	61.5 ±1.4	50.4 ±2.4	74.4 ±0.9	76.6 ±0.8	65.7
DANN†	59.9 ±1.3	53.0 ±0.3	73.6 ±0.7	76.9 ±0.5	65.9
GroupDRO†	60.4 ±0.7	52.7 ±1.0	75.0 ±0.7	76.0 ±0.7	66.0
MMD†	60.4 ±0.2	53.3 ±0.3	74.3 ±0.1	77.4 ±0.6	66.4
MTL†	61.5 ±0.7	52.4 ±0.6	74.9 ±0.4	76.8 ±0.4	66.4
VREx†	60.7 ±0.9	53.0 ±0.9	75.3 ±0.1	76.6 ±0.5	66.4
ERM†	61.3 ±0.7	52.4 ±0.3	75.8 ±0.1	76.6 ±0.3	66.5
MLDG†	61.5 ±0.9	53.2 ±0.6	75.0 ±1.2	77.5 ±0.4	66.8
ERM‡	63.1 ±0.3	51.9 ±0.4	77.2 ±0.5	78.1 ±0.2	67.6
I-Mixup	62.4 ±0.8	54.8 ±0.6	76.9 ±0.3	78.3 ±0.2	68.1
SagNet†	63.4 ±0.2	54.8 ±0.4	75.8 ±0.4	78.3 ±0.3	68.1
CORAL†	65.3 ±0.4	54.4 ±0.5	76.5 ±0.1	78.4 ±0.5	68.7
Ours	**66.1** ±0.4	**57.7** ±0.4	**78.4** ±0.1	**80.2** ±0.2	**70.6**

E.4 TerraIncognita

Table 12: **Out-of-domain accuracies (%) on `TerraIncognita`.**

Algorithm	**L100**	**L38**	**L43**	**L46**	**Avg**
MMD†	41.9 ±3.0	34.8 ±1.0	57.0 ±1.9	35.2 ±1.8	42.2
GroupDRO†	41.2 ±0.7	38.6 ±2.1	56.7 ±0.9	36.4 ±2.1	43.2
Mixstyle‡	54.3 ±1.1	34.1 ±1.1	55.9 ±1.1	31.7 ±2.1	44.0
ARM†	49.3 ±0.7	38.3 ±2.4	55.8 ±0.8	38.7 ±1.3	45.5
MTL†	49.3 ±1.2	39.6 ±6.3	55.6 ±1.1	37.8 ±0.8	45.6
CDANN†	47.0 ±1.9	41.3 ±4.8	54.9 ±1.7	39.8 ±2.3	45.8
ERM†	49.8 ±4.4	42.1 ±1.4	56.9 ±1.8	35.7 ±3.9	46.1
VREx†	48.2 ±4.3	41.7 ±1.3	56.8 ±0.8	38.7 ±3.1	46.4
RSC†	50.2 ±2.2	39.2 ±1.4	56.3 ±1.4	**40.8** ±0.6	46.6
DANN†	51.1 ±3.5	40.6 ±0.6	57.4 ±0.5	37.7 ±1.8	46.7
IRM†	54.6 ±1.3	39.8 ±1.9	56.2 ±1.8	39.6 ±0.8	47.6
CORAL†	51.6 ±2.4	42.2 ±1.0	57.0 ±1.0	39.8 ±2.9	47.7
MLDG†	54.2 ±3.0	44.3 ±1.1	55.6 ±0.3	36.9 ±2.2	47.8
I-Mixup	**59.6** ±2.0	42.2 ±1.4	55.9 ±0.8	33.9 ±1.4	47.9
SagNet†	53.0 ±2.9	43.0 ±2.5	57.9 ±0.6	40.4 ±1.3	48.6
ERM‡	54.3 ±0.4	42.5 ±0.7	55.6 ±0.3	38.8 ±2.5	47.8
Ours	55.4 ±0.0	**44.9** ±1.1	**59.7** ±0.4	39.9 ±0.2	**50.0**

E.5 DomainNet

Table 13: **Out-of-domain accuracies (%) on `DomainNet`.**

Algorithm	**clip**	**info**	**paint**	**quick**	**real**	**sketch**	**Avg**
MMD†	32.1 ±13.3	11.0 ±4.6	26.8 ±11.3	8.7 ±2.1	32.7 ±13.8	28.9 ±11.9	23.4
GroupDRO†	47.2 ±0.5	17.5 ±0.4	33.8 ±0.5	9.3 ±0.3	51.6 ±0.4	40.1 ±0.6	33.3
VREx†	47.3 ±3.5	16.0 ±1.5	35.8 ±4.6	10.9 ±0.3	49.6 ±4.9	42.0 ±3.0	33.6
IRM†	48.5 ±2.8	15.0 ±1.5	38.3 ±4.3	10.9 ±0.5	48.2 ±5.2	42.3 ±3.1	33.9
Mixstyle‡	51.9 ±0.4	13.3 ±0.2	37.0 ±0.5	12.3 ±0.1	46.1 ±0.3	43.4 ±0.4	34.0
ARM†	49.7 ±0.3	16.3 ±0.5	40.9 ±1.1	9.4 ±0.1	53.4 ±0.4	43.5 ±0.4	35.5
CDANN†	54.6 ±0.4	17.3 ±0.1	43.7 ±0.9	12.1 ±0.7	56.2 ±0.4	45.9 ±0.5	38.3
DANN†	53.1 ±0.2	18.3 ±0.1	44.2 ±0.7	11.8 ±0.1	55.5 ±0.4	46.8 ±0.6	38.3
RSC†	55.0 ±1.2	18.3 ±0.5	44.4 ±0.6	12.2 ±0.2	55.7 ±0.7	47.8 ±0.9	38.9
I-Mixup	55.7 ±0.3	18.5 ±0.5	44.3 ±0.5	12.5 ±0.4	55.8 ±0.3	48.2 ±0.5	39.2
SagNet†	57.7 ±0.3	19.0 ±0.2	45.3 ±0.3	12.7 ±0.5	58.1 ±0.5	48.8 ±0.2	40.3
MTL†	57.9 ±0.5	18.5 ±0.4	46.0 ±0.1	12.5 ±0.1	59.5 ±0.3	49.2 ±0.1	40.6
ERM†	58.1 ±0.3	18.8 ±0.3	46.7 ±0.3	12.2 ±0.4	59.6 ±0.1	49.8 ±0.4	40.9
MLDG†	59.1 ±0.2	19.1 ±0.3	45.8 ±0.7	13.4 ±0.3	59.6 ±0.2	50.2 ±0.4	41.2
CORAL†	59.2 ±0.1	19.7 ±0.2	46.6 ±0.3	13.4 ±0.4	59.8 ±0.2	50.1 ±0.6	41.5
MetaReg	59.8	**25.6**	50.2	11.5	64.6	50.1	43.6
DMG	65.2	22.2	50.0	15.7	59.6	49.0	43.6
ERM‡	63.0 ±0.2	21.2 ±0.2	50.1 ±0.4	13.9 ±0.5	63.7 ±0.2	52.0 ±0.5	44.0
Ours	**66.0** ±0.1	22.4 ±0.3	**53.5** ±0.1	**16.1** ±0.2	**65.8** ±0.4	**55.5** ±0.3	**46.5**

F Assets

In this section, we discuss about licenses, copyrights, and ethical issues of our assets, such as code and datasets.

F.1 Code

Our work is built upon DomainBed [22][4], which is released under the MIT license.

F.2 Datasets

While we use public datasets only, we track how the datasets were built to discuss licenses, copyrights, and potential ethical issues. For `DomainNet` [46] and `OfficeHome` [44], we use the datasets for non-profit academic research only following their fair use notice. `TerraIncognita` [45] is a subset of Caltech Camera Traps (CCT) dataset, distributed under the Community Data License Agreement (CDLA) license. `PACS` [7] and `VLCS` [43] datasets have images collected from the web and we could not find any statements about licenses, copyrights, or whether consent was obtained. Considering that both datasets contain person class and images of people, there may be potential ethical issues.

G Reproducibility

To provide details of our algorithm and guarantee reproducibility, we provide the source code[5] publicly. The code also specifies detailed environments, dependencies, how to download datasets, and instructions to reproduce the main results (Table 1 and 2 in the main text).

G.1 Infrastructures

Every experiment is conducted on a single NVIDIA Tesla P40 or V100, Python 3.8.6, PyTorch 1.7.0, Torchvision 0.8.1, and CUDA 9.2.

G.2 Runtime Analysis

The total runtime varies depending on datasets and the moment detected to overfit. It takes about 4 hours for `PACS` and `VLCS`, 8 hours for `OfficeHome`, 8.5 hours for `TerraIncognita`, and 56 hours for `DomainNet` on average, when using a single NVIDIA Tesla P40 GPU. Each experiment includes the leave-one-out cross-validations for all domains in each dataset.

G.3 Complexity Analysis

The only additional time overhead incurs from stochastic weights selection, which requires further evaluations. To analyze the overhead, let the forward time t_f, backward time t_b, training and validation split ratio $r = |X^{train}|/|X^{valid}|$, total in-domain samples n, and evaluation frequency v that indicates how many evaluations are conducted for each epoch. For conciseness, we assume $t = t_f = t_b$ and do not consider early stopping.

For one epoch, training time is $2tnr/(r+1)$, and evaluation time is $vtn/(r+1)$. The total runtime for one epoch is $tn(2r+v)/(r+1)$. Final overhead ratio is $(2r+v)/(2r+v_b)$ where v_b is the evaluation frequency of a baseline. In our main experiments, we use $r = 4$. Compared to the default parameters of DomainBed [22], we use $v = 2v_b$ for `DomainNet`, $v = 6v_b$ for `VLCS`, and $v = 3v_b$ for the others. Then, the total runtime of our algorithm takes from 1.07 (`PACS`) to 1.27 (`DomainNet`) times more than the ERM baseline. In practice, it can be improved by conducting approximated evaluations using sub-sampled validation set.

In terms of memory complexity, our method does not require additional GPU memory. Instead, we leverage CPU memory to minimize training time overhead, which takes up to $\max(N, M)$ times more than the baseline.

[4] `https://github.com/facebookresearch/DomainBed`

[5] `https://github.com/khanrc/swad`