Flatness-Aware Minimization for Domain Generalization

Xingxuan Zhang, Renzhe Xu, Han Yu, Yancheng Dong, Pengfei Tian, Peng Cui*
Department of Computer Science, Tsinghua University
xingxuanzhang@hotmail.com, cuip@tsinghua.edu.cn

Abstract

Domain generalization (DG) seeks to learn robust models that generalize well under unknown distribution shifts. As a critical aspect of DG, optimizer selection has not been explored in depth. Currently, most DG methods follow the widely used benchmark, DomainBed, and utilize Adam as the default optimizer for all datasets. However, we reveal that Adam is not necessarily the optimal choice for the majority of current DG methods and datasets. Based on the perspective of loss landscape flatness, we propose a novel approach, Flatness-Aware Minimization for Domain Generalization (FAD), which can efficiently optimize both zeroth-order and first-order flatness simultaneously for DG. We provide theoretical analyses of the FAD’s out-of-distribution (OOD) generalization error and convergence. Our experimental results demonstrate the superiority of FAD on various DG datasets.

1. Introduction

Current neural networks are expected to generalize to unseen distributions in real-world applications, which can break the independent and identically distributional (I.I.D.) assumption of traditional machine learning algorithms [129, 99]. The distribution shift between training and test data may largely deteriorate most current approaches in practice. Hence, instead of generalization within the training distribution, the ability to generalize under distribution shift, namely domain generalization (DG) [73], has attracted increasing attention [99].

Various methods have been proposed to address the DG problem recently, many of which have shown promising performance. As a learning problem mainly in the computer vision field, the DG task relies highly on the chosen optimizer, yet the effect of the optimizer in DG has hardly been studied. Most current algorithms, following the well-known benchmark DomainBed [32], default to optimize models with Adam [48] without considering other optimizers. However, the in-distribution generalization ability of popular optimizers has been investigated from the perspective of loss landscape recently [28, 29] and Adam is found to achieve inferior generalization compared with other optimizers such as SGD [76] due to the sharp minima Adam selected. Thus, investigating the impact of optimizers in DG and clarifying whether Adam should be considered as the default optimizer is of significance.

In this paper, we empirically compare the performance of current optimizers, including Adam, SGD, AdaBelief [131], YOGI [117], AdaHessian [116], SAM [28], and GAM[121]. Through extensive experiments on current DG benchmarks, we show that Adam can hardly surpass other optimizers on most DG datasets. The best choice of the optimizer for different DG methods varies across different datasets, and choosing the right optimizer can help exceed the current leaderboards’ limitations.

Recently, the connection between the flatness of the loss landscape and in-distribution generalization ability has been widely studied and verified both theoretically[130] and empirically[25]. Some works [13, 87] show that flatness also leads to superior OOD generalization. However, none of the previous works consider the optimization in DG nor provide theoretical assurance of their approach.
Among flatness-aware optimization methods, sharpness-Aware Minimization (SAM) [28] and its variants [130, 53, 126, 47], which optimize the zeroth-order flatness, achieve SOTA performance on various in-distribution image classification tasks [53, 130]. Most recently, Gradient Norm Aware Minimization (GAM) [121] shows that minimizing the first-order flatness provides a more substantial penalty on the sharpness of minima than zeroth-order flatness yet requires more computation overhead. Due to the first-order Taylor expansions adopted to optimize zeroth-order and first-order flatness, both SAM and GAM lose some of their penalty strength, and thus they can be mutually reinforced by each other. To accelerate the optimization of first-order flatness and combine zero-order flatness, we propose a unified optimization framework, Flatness-Aware minimization for Domain generalization (FAD), which eliminates the considerable computation overhead for Hessian or Hessian-vector products. We theoretically show that the proposed FAD controls the dominant eigenvalue of Hessian of the training loss, which indicates the sharpness of the loss landscape along its most critical direction [42]. We present an OOD generalization bound to show that the proposed FAD guarantees the generalization error on test data and convergence analysis. Through extensive experiments, we evaluate FAD and other optimizers on various DG benchmarks.

We summarize the main contribution as follows.

- We propose a unified optimization framework, Flatness-Aware minimization for Domain generalization (FAD), to optimize zeroth-order and first-order flatness simultaneously. Without calculating Hessian and Hessian-vector products, FAD considerably reduces the computation overhead of first-order flatness.
- We theoretically analyze the OOD generalization error and convergence of FAD. We show that FAD controls the dominant eigenvalue of Hessian and thus the flatness of learned minima.
- We empirically show that Adam, the default optimizer for most current DG benchmarks, can hardly be the optimal choice for most DG datasets. We present the superiority of FAD compared with current optimizers through extensive experiments.
- We empirically validate that FAD finds flatter optima with lower Hessian spectra compared with zeroth- and first-order flatness-aware optimization methods.

2. Related Works

2.1. Optimizers

Some works [107, 28] have studied the connection between current optimization approaches, such as SGD [76], Adam [49], AdamW [65] and others [26, 63] and in-distribution generalization. Some literature shows that Adam is more vulnerable to sharp minima than SGD [104], which may result in worse generalization [106, 33, 35]. Some follow-ups [66, 16, 107, 117] propose generalizable optimizers to address this problem. However, the generalization ability and convergence speed is often a trade-off [44, 107, 63, 117, 26]. Different optimizers may favor different tasks and network architectures (e.g., SGD is often chosen for ResNet [34] while AdamW [65] for ViTs [21]). Selecting the right optimizer is critical for performance and convergence while the understanding of its relationship to model generalization remains nascent [28]. [74] finds that fine-tuned SGD outperforms Adam on OOD tasks. Yet all the previous works do not consider the effect of optimizers in current DG benchmarks and evaluation protocols. In this paper, we discuss the selection of the default optimizer in DG benchmarks.

2.2. Domain Generalization and Optimization

Domain generalization (DG) aims to improve the generalization ability to novel domains [31, 45, 50]. A common approach is to learn domain-invariant or causal features over multiple source domains [22, 36, 57, 59, 72, 80, 89, 120, 110, 118, 101, 123] or aggregating domain-specific modules [67, 68]. Some works propose to enlarge the input space by augmentation of training data [12, 98, 81, 91, 127, 128, 109]. Exploit regularization with meta-learning [22, 58] and Invariant Risk Minimization (IRM) framework [2] are also proposed for DG. Recently, several works propose to ensemble model weights for better generalization [13, 4, 84, 105, 86, 62, 18] and achieve outstanding performance.

2.3. Flatness of loss landscape

Recent works study the relationship between flatter minima lead and better generalization on in-distribution data [44, 130, 41, 79]. [42] reviews the literature related to generalization and the sharpness of minima. It highlights the role of maximum Hessian eigenvalue in deciding the sharpness of minima [44, 103]. Some simple strategies are proposed to optimize maximum Hessian eigenvalue, such as choosing a large learning rate [54, 19, 40] and smaller batch size [94, 54, 39]. Sharpness-Aware Minimization (SAM) [28] and its variants [130, 53, 24, 64, 25, 71, 126, 47, 46], which are representative training algorithm to seek zeroth-order flatness, show outstanding performance on in-distribution generalization. Most recently, several works discuss the relationship between generalization and gradient norm [8, 125]. GAM [121] proposes to optimize first-order flatness for better generalization. [13, 87] show that flatness minima also lead to superior OOD generalization. However, none of the previous works consider the optimization in DG nor provide theoretical assurance of their approach.
3. Method

In in-distribution generalization tasks, sharpness-aware optimization methods show outstanding performance [28]. We theoretically show that optimizing both zeroth-order and first-order flatness strengthens out-of-distribution performance via a generalization bound. We propose a unified optimization framework to optimize zeroth-order and first-order flatness effectively and efficiently. We present the connection between Hessian spectra and our regularization term and give the convergence analysis of our method.

Notations We use $\mathcal{X}$ and $\mathcal{Y}$ to denote the space of input $X$ and outcome $Y$, respectively. We use $\Delta_Y$ to denote a distribution on $\mathcal{Y}$. Let $S = \{(x_i, y_i)\}_{i=1}^n$ denote the training dataset with $n$ data points drawn independently from the training distribution $D_u$. We consider the covariate shift scenario [10, 92, 111] and assume that the conditional probabilities are invariant across training and test distributions (i.e., $P_u(Y|X) = P_e(Y|X)$). The invariant conditional probability is denoted as $P(Y|X)$. A prediction model $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ parametrized by $\theta \in \Theta \subseteq \mathbb{R}^d$ predicts an input to a simplex on $\mathcal{Y}$, which indicates the predicted probability of each class. $\Theta$ is the hypothesis set. We further assume that the ground truth model $\theta^*$ that characterizes $P(Y|X)$ is in the hypothesis set $\Theta$.

Let $\ell : \Delta_Y \times \Delta_Y \rightarrow \mathbb{R}_+$ define a loss function over $\Delta_Y$. For any hypotheses $\theta_1, \theta_2 \in \Theta$, the expected loss $\mathcal{L}_D(\theta_1, \theta_2)$ for distribution $D$ is given as $\mathcal{L}_D(\theta_1, \theta_2) = \mathbb{E}_{x \sim D} [\ell(f_{\theta_1}(x), f_{\theta_2}(x))]$. To simplify the notations, we use $\mathcal{L}_u$ and $\mathcal{L}_w$ to denote the expected loss $\mathcal{L}_{D_u}$ and $\mathcal{L}_{D_w}$ in training and test distributions, respectively. In addition, we use $\mathcal{E}_u(\theta) = \mathcal{L}_u(\theta, \theta^*)$ and $\mathcal{E}_w(\theta) = \mathcal{L}_w(\theta, \theta^*)$ to denote the loss of a function $\theta \in \Theta$ w.r.t. the true labeling function $\theta^*$ in the two distributions. We use $\mathcal{E}_w(\theta)$ to denote the empirical loss function w.r.t. the $n$ samples $S$ from the training distribution. In addition, we assume that $\mathcal{E}_u(\theta)$ is twice differentiable throughout the paper and we use $\nabla \mathcal{E}_w(\theta)$ and $\nabla^2 \mathcal{E}_w(\theta)$ to denote the derivative and Hessian matrix, respectively.

3.1. A General Flatness-Aware Optimization for DG

In in-distribution generalization tasks, it is shown that the zeroth-order flatness can be not strong enough in cases where $\rho$ covers multiple minima or the maximum loss in $\rho$ is misaligned with the up trend of loss and first-order flatness can help improve it [121]. But both zeroth-order and first-order flatness require first-order approximations in practice. Thus optimizing both zeroth-order and first-order flatness together leads to stronger generalization than both of them [121]. Yet optimizing first-order flatness requires the calculation of Hessian or Hessian-vector products, which introduces heavy computation overhead. As a result, we aim to accelerate the optimization in a Hessian-free way for OOD generalization.

Formally, the zeroth-order and first-order flatness are defined as follows [28, 130, 121].

**Definition 3.1** ($\rho$-zeroth-order flatness). For any $\rho > 0$, the $\rho$-zeroth-order flatness $R^{(0)}_{\rho}(\theta)$ of function $\mathcal{E}_w(\theta)$ at a point $\theta$ is defined as

$$R^{(0)}_{\rho}(\theta) \triangleq \max_{\theta' \in B(\theta, \rho)} \left( \mathcal{E}_w(\theta') - \mathcal{E}_w(\theta) \right), \quad \forall \theta \in \Theta. \quad (1)$$

Here $\rho$ is the perturbation radius that controls the magnitude of the neighborhood.

**Definition 3.2** ($\rho$-first-order flatness). For any $\rho > 0$, the $\rho$-first-order flatness $R^{(1)}_{\rho}(\theta)$ of function $\mathcal{E}_w(\theta)$ at a point $\theta$ is defined as

$$R^{(1)}_{\rho}(\theta) \triangleq \rho \cdot \max_{\theta' \in B(\theta, \rho)} \left\| \nabla \mathcal{E}_w(\theta') \right\|, \quad \forall \theta \in \Theta. \quad (2)$$

Here $\rho$ is the perturbation radius similar to Definition 3.1.

Based on these definitions, we propose to jointly optimize zeroth-order and first-order flatness through the following linear combination of the two flatness metrics.

$$R_{\rho, \alpha}^{\text{FAD}}(\theta) = \alpha R^{(0)}_{\rho}(\theta) + (1 - \alpha) R^{(1)}_{\rho}(\theta), \quad (3)$$

where $\alpha$ is a hyper-parameter that controls the trade-off between zeroth-order and first-order flatness. Since both flatness are related to the maximal eigenvalue of the Hessian matrices [130, 121], we can derive that $R_{\rho, \alpha}^{\text{FAD}}(\theta)$ is also related to the maximal eigenvalue. Specifically, when a point $\theta^*$ is a local minimum of $\mathcal{E}_w(\theta)$ and $\mathcal{E}_w(\theta)$ can be second-order Taylor approximated in the neighbourhood of $\theta^*$, then

$$\lambda_{\max} \left( \nabla^2 \mathcal{E}_w(\theta^*) \right) = \frac{R_{\rho, \alpha}^{\text{FAD}}(\theta^*)}{\rho^2 \left( 1 - \frac{\alpha}{2} \right)} \cdot (4)$$

See Appendix A for details. Since the maximal eigenvalue is a proper measure of the curvature of minima [42, 43] and related to generalization abilities [121, 17, 39], optimizing Equation (3) would hopefully improve the DG performances, which we demonstrate in Section 3.2.

Then the overall objective of our method, Flatness-Aware minimization for Domain generalization (FAD), is as follows.

$$\mathcal{L}_{\text{FAD}}(\theta) = \mathcal{E}_w(\theta) + \beta \cdot R_{\rho, \alpha}^{\text{FAD}}(\theta). \quad (5)$$

The strength of FAD regularization is controlled by a hyperparameter $\beta$. 

\[5191\]
3.2. Flatness Penalty and DG Performance

We give a generalization bound to demonstrate that FAD controls the generalization error on test distribution as follows.

**Proposition 3.1.** Suppose the per-data-point loss function \( \ell \) is differentiable, symmetric, bounded by \( M \), and obeys the triangle inequality. Suppose \( \theta^* \in \Theta \). Fix \( \rho > 0 \) and \( \theta \in \Theta \). Then with probability at least \( 1 - \delta \) over training set \( S \) generated from the distribution \( D \),

\[
\begin{align*}
E_{s_i \sim N(0,\nu^2/(\theta+\sqrt{\nu^2}))}[\hat{E}_u(\theta + \epsilon)] &\leq \hat{E}_u(\theta) + R_{FAD}(\theta) \\
+ \sup_{\theta_1, \theta_2} |\mathcal{L}_u(\theta_1, \theta_2) - \mathcal{L}_u(\theta_1, \theta_2)| + \frac{M}{\sqrt{n}} \\
&+ \sqrt{\frac{1}{4} d \log \left(1 + \frac{\|\theta\|^2 (\sqrt{\theta+\sqrt{\nu^2}})^2}{d \nu} \right) + \frac{2}{3} + \log \left(\frac{6n \nu}{2} + 3d\right)}
\end{align*}
\]

\[
\tag{6}
\]

**Remark.** The term \( \sup_{\theta_1, \theta_2} |\cdot| \) corresponds to the discrepancy distance [69], which measures the covariate shift between the training domain and the unknown test domain. Please note that in DG/OOD generalization tasks, the information on test distribution is unavailable in the optimization phase. Thus one cannot intervene in the third term on RHS. Thus optimizing the regularization of FAD (the second term on RHS) with ERM loss on training data leads to better generalization on test data.

3.3. Algorithm Details

Although the regularization of FAD controls the OOD generalization error, directly optimizing it suffers from significant computational costs. Previous works [114, 121] propose to calculate second order gradient with Hessian-vector product, yet the cost increases fast as the model dimension grows. Inspired by [93, 125], we approximate second-order gradient with first-order gradients and optimize zeroth-order and first-order flatness simultaneously as follows.

**Optimization and acceleration** At each round, suppose the perturbation radius is set to \( \rho_t \), then the gradient of \( \mathcal{L}_{all}(\theta_t) \) at the point \( \theta_t \) in Equation (5) is approximated as follows.

The gradient of \( R_{\rho_t}^{(0)}(\theta_t) \). Following [28], let

\[
\begin{align*}
\tilde{\theta}_{t,1} &= \theta_t + \rho_t \cdot \frac{g_{t,0}}{\|g_{t,0}\|} \quad \text{where} \quad g_{t,0} = \nabla \hat{E}_u(\theta_t), \\
g_{t,1} &= \nabla \hat{E}_u(\tilde{\theta}_{t,1}).
\end{align*}
\]

Then the gradient of \( R_{\rho_t}^{(0)}(\theta_t) \) is approximated by

\[
\nabla R_{\rho_t}^{(0)}(\theta_t) \approx g_{t,1} - g_{t,0}.
\]

**Algorithm 1 Flatness-Aware Minimization for Domain Generalization (FAD)**

1. **Input:** Batch size \( b \), Learning rate \( \eta_t \), Perturbation radius \( \rho_t \), Trade-off coefficients \( \alpha, \beta \), Small constant \( \xi \)
2. \( t \leftarrow 0, \theta_0 \leftarrow \text{initial parameters} \)
3. **while** \( \theta_t \) not converged **do**
4. Sample \( W_t \) from the training data with \( b \) instances
5. \( g_{t,0} \leftarrow \nabla \hat{E}_u(\theta_t) \)
6. \( \theta_{t,1} \leftarrow \theta_t + \rho_t \cdot \frac{g_{t,0}}{\|g_{t,0}\| + \xi} \)
7. \( g_{t,1} \leftarrow \nabla \hat{E}_u(\tilde{\theta}_{t,1}) \)
8. \( h_{t,0} \leftarrow g_{t,1} - g_{t,0} \quad \text{\( \triangleright \) gradient of} \ R_{\rho_t}^{(0)}(\theta_t) \)
9. \( \theta_{t,2} \leftarrow \theta_t + \rho_t \cdot h_{t,0}/(\|h_{t,0}\| + \xi) \)
10. \( g_{t,2} \leftarrow \nabla \hat{E}_u(\tilde{\theta}_{t,2}) \)
11. \( \theta_{t,3} \leftarrow \theta_{t,2} + \rho_t \cdot g_{t,2}/(\|g_{t,2}\| + \xi) \)
12. \( g_{t,3} = \nabla \hat{E}_u(\tilde{\theta}_{t,3}) \)
13. \( h_{t,1} \leftarrow g_{t,3} - g_{t,2} \quad \text{\( \triangleright \) gradient of} \ R_{\rho_t}^{(1)}(\theta_t) \)
14. \( \theta_{t+1} \leftarrow \theta_t - \eta_t (g_{t,0} + \beta (a h_{t,0} + (1 - \alpha) h_{t,1})) \)
15. \( t \leftarrow t + 1 \)
16. **end while**
17. **return** \( \theta_t \)

The gradient of \( R_{\rho_t}^{(1)}(\theta_t) \), [121] propose to approximate the gradient through the following procedure

\[
\nabla R_{\rho_t}^{(1)}(\theta_t) \approx \rho_t \cdot \nabla \|\nabla \hat{E}_u(\theta_t)\| \quad \theta_t^{\text{opt}} = \theta_t + \rho_t \cdot \frac{\nabla \|\nabla \hat{E}_u(\theta_t)\|}{\nabla \|\nabla \hat{E}_u(\theta_t)\|}.
\]

One can approximate \( \nabla \|\nabla \hat{E}_u(\theta_t)\| \) with first-order gradient as follows.

\[
\nabla \|\nabla \hat{E}_u(\theta_t)\| \approx \frac{\nabla \hat{E}_u(\theta_{t,1}) - \nabla \hat{E}_u(\theta_{t})}{\rho_t} = \frac{g_{t,1} - g_{t,0}}{\rho_t},
\]

Thus we can approximate the gradient of \( R_{\rho_t}^{(1)}(\theta_t) \) efficiently and rewrite the term as follows. The details of the derivation can be found in Appendix A.

\[
\nabla R_{\rho_t}^{(1)}(\theta_t) \approx g_{t,3} - g_{t,2}; \quad \text{where} \quad \\
\theta_{t,2} = \theta_t + \rho_t \cdot \frac{g_{t,1} - g_{t,0}}{\|g_{t,1} - g_{t,0}\|}, \quad g_{t,2} = \nabla \hat{E}_u(\tilde{\theta}_{t,2}) \quad \text{and} \quad \\
\theta_{t,3} = \theta_{t,2} + \rho_t \cdot \frac{g_{t,2}}{\|g_{t,2}\|}, \quad g_{t,3} = \nabla \hat{E}_u(\tilde{\theta}_{t,3}) \quad \text{.} \]

Please note that the terms \( g_{t,1} \) and \( g_{t,2} \) for optimizing zeroth-order flatness \( R_{\rho_t}^{(0)}(\theta_t) \) are preserved in the approximation of \( R_{\rho_t}^{(1)}(\theta_t) \) and thus causes no overhead. Combining the gradient approximation steps as shown in Equations (8) and (9), we can obtain the unified optimization framework as shown in Algorithm 1.
**Convergence analysis** To analyze the convergence property of Algorithm 1, we first introduce the Lipschitz smooth assumption that are adopted commonly in optimization-related literature [1, 112, 130].

**Definition 3.3.** For a function \( J : \Theta \rightarrow \mathbb{R} \), \( J \) is \( \gamma_2 \)-Lipschitz smooth if
\[
\forall \theta_1, \theta_2 \in \Theta, \| \nabla J(\theta_1) - \nabla J(\theta_2) \| \leq \gamma_2 \| \theta_1 - \theta_2 \|. \tag{10}
\]

Let \( \Delta_t = g_{t,0} + \beta(\alpha h_{t,0} + (1 - \alpha) h_{t,1}) \) be the estimated gradient at round \( t \) as shown in Line 14 in Algorithm 1.

Now we can prove that the algorithm would eventually converge by the following theorem.

**Theorem 3.2.** Suppose \( \hat{E}_t(\theta) \) is \( \gamma \)-Lipschitz smooth and bounded by \( M \). For any timestamp \( t \in \{0, 1, \ldots, T\} \) and any \( \theta \in \Theta \), suppose we can obtain noisy and bounded observations \( f_t(\theta) \) of \( \nabla \hat{E}_t(\theta) \) such that
\[
\mathbb{E}[f_t(\theta)] = \nabla \hat{E}_t(\theta), \quad \| f_t(\theta) \| \leq G. \tag{11}
\]

Then with learning rate \( \eta_t = \eta_0/\sqrt{t} \) and perturbation radius \( \rho_t = \rho_0/\sqrt{t} \),
\[
\sum_{t=1}^{T} \mathbb{E}[\| \Delta_t \|^2] \leq C_1 + C_2 \log T \frac{T}{\sqrt{T}}, \tag{12}
\]

for some constants \( C_1 \) and \( C_2 \) that only depend on \( \gamma_1, \gamma_2, G, M, \eta_0, \rho_0, \alpha, \beta \).

Thus, the convergence of FAD is guaranteed, and we show FAD’s convergence rate empirically in Section 4.4.

## 4. Experiments

### 4.1. Experimental Settings

We evaluate our optimization method and current optimizers on various DG benchmarks including PACS [55], VLCS [27], OfficeHome [97], TerraIncognita [9], DomainNet [78] and NICO++ [122]. Detailed introductions of these datasets are in Appendix. We consider the generalization accuracy, training time, and hessian spectra of found minima as the evaluation metrics. We show how optimizers contribute when training with ERM and current SOTA DG algorithms. Generally, Adam is hardly the best choice for DG benchmarks.

For a fair comparison, we follow the basic training and evaluation protocol introduced in [32], where the information of test data is unavailable for hyperparameter search. We train all the models on DomainNet for 15,000 iterations as suggested in [13], 10,000 iterations on NICO++, and 5,000 iterations on other datasets unless otherwise noted. For datasets except for NICO++, we follow the leave-one-out protocol in [32] where one domain is chosen as the target domain and the remaining domains as the training domain. For NICO++, we choose two domains as target domains in each run and train models on the remaining four domains. Following the official combination [122], we select \{autumn, rock\}, \{dim, grass\}, \{outdoor, water\} as the target domain pairs. A detailed description of the split of training and test domains is in Appendix B. For all datasets, 20% samples in the training data are used for validation and model selection. ImageNet [20] pretrained ResNet-50 [34] is adopted as the initial model. The search space for the initial learning rate, weight decay, and other hyperparameters is in Appendix B.

### 4.2. Comparison of Optimizers on DG

Here we investigate how the optimizers affect the generalization under domain shifts. We first conduct experiments with various optimizers following the validation with training data protocol in DomainBed [32] to study the generalization of current optimizers and the robustness of the choice of hyperparameters. The results are shown in Table 1. Detailed results of optimizers on all the splits of reported datasets are in Appendix B.

Despite SGD remaining the most popular optimizer for most CNN-based models on in-distribution image recognition benchmarks, such as CIFAR [51], and ImageNet due to its outstanding performance, it fails to show advantages compared with other optimizers on current benchmarks for DG. The previous work [74] shows that SGD with momentum outperforms adaptive-based optimizers such as Adam with an exhaustive hyperparameter search (hundreds of trials for each optimizer on each dataset) to select hyperparameters with good in-distribution performance. Yet for fair comparisons with previous works, we follow the model selection protocol in DomainBed, where only 20 trials of search are conducted for each evaluation, resulting in inadequate searches. In our experiments, SGD fails to outperform adaptive-based optimizers (e.g., AdamW and AdBelief). This may be because SGD is more sensitive to the choice of hyperparameters [48] and the hyperparameter search space in evaluation protocols for DG is too large to be traversed and fine-tune the hyperparameters within several random runs. The empirical demonstration of the sensitivity of optimizers to hyperparameters is in Appendix B.

Furthermore, it is important to note that different datasets may require different optimization strategies. For example, Adam outperforms SGD considerably on PACS, while the opposite stands for VLCS, OfficeHome, and NICO++. Among none-flatness-aware optimizers, AdBelief shows the best average performance. AdamW outperforms Adam on 5 out of 6 datasets and achieves higher average accuracy. Thus, Adam is hardly the best choice for DG and choosing the right optimizer can help exceed current leadersboards’ limitations.

Among all optimizers, flatness-aware optimizers (e.g., SAM, GAM, and FAD) show promising results across all the datasets. FAD considerably outperforms all its coun-
terparts on all the benchmarks, indicating that optimizing zeroth-order and first-order flatness simultaneously learns flatter minima and better generalization.

4.3. Ensemble with current DG algorithms

As a base optimizer, FAD can be combined with current DG methods with various objective functions. The compatibility with DG methods is critical for the choice of optimizers. In this subsection, we investigate the compatibility of GAM with current DG algorithms and compare it with other optimizers. We also report the results of other DG methods for better comparisons.

Please note that we only consider a single pretrained model (i.e., ImageNet pretrained ResNet-50) as the initial model for fair comparisons in this paper, so we do not combine optimizers with some SOTA methods, such as SIMPLE [62], MIRO [14] with RegNetY-16GF [83], and CARFT [70] with CLIP [82] which involve the knowledge of pretrained models other than ImageNet pretrained ResNet-50. We consider SOTA and representative methods including CORAL [95], SWAD [13], and Fishr [85] as the base approach to compare optimizers and leave more methods, including EoA [4] and RSC [37] in Appendix B.

The results are shown in Table 2. Different optimizers seem to favor different DG methods. AdamW achieves the best performance when combined with SWAD and CORAL, while SGD surpasses Adam and AdamW with Fishr. This indicates that fixing a single optimizer in DG benchmarks may be unfair for some methods that are not favored, and choosing the proper optimizer can further improve DG methods’ performance.

FAD consistently outperforms other optimizers with all the methods across various datasets. Compared with the default optimizer Adam, FAD further improves SWAD by 1.7% on PACS and 1.2% on average and surpasses previous SOTA methods.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PACS</th>
<th>VLCS</th>
<th>OfficeHome</th>
<th>TerraInc</th>
<th>DomainNet</th>
<th>NICO++</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam [48]</td>
<td>84.2±0.6</td>
<td>77.3±1.3</td>
<td>67.6±0.4</td>
<td>44.4±0.8</td>
<td>43.0±0.1</td>
<td>76.9±0.0</td>
<td>65.6</td>
</tr>
<tr>
<td>AdamW [65]</td>
<td>83.6±1.5</td>
<td>77.4±0.8</td>
<td>68.8±0.6</td>
<td>45.2±1.4</td>
<td>43.4±0.1</td>
<td>77.5±0.1</td>
<td>66.0</td>
</tr>
<tr>
<td>SGD [76]</td>
<td>79.9±1.4</td>
<td>78.1±0.2</td>
<td>68.5±0.3</td>
<td>44.9±1.8</td>
<td>43.2±0.1</td>
<td>77.2±0.2</td>
<td>65.3</td>
</tr>
<tr>
<td>YOGI [117]</td>
<td>81.2±0.4</td>
<td>77.6±0.6</td>
<td>68.3±0.3</td>
<td>45.4±0.5</td>
<td>43.5±0.0</td>
<td>77.9±0.2</td>
<td>65.7</td>
</tr>
<tr>
<td>AdaBelief [131]</td>
<td>84.6±0.6</td>
<td>78.4±0.4</td>
<td>68.0±0.9</td>
<td>45.2±2.0</td>
<td>43.5±0.1</td>
<td>77.4±0.1</td>
<td>66.2</td>
</tr>
<tr>
<td>AdaHessian [116]</td>
<td>84.5±1.0</td>
<td>78.6±0.8</td>
<td>68.4±0.9</td>
<td>44.4±0.5</td>
<td>44.4±0.1</td>
<td>77.7±0.1</td>
<td>66.3</td>
</tr>
<tr>
<td>SAM [28]</td>
<td>85.3±1.0</td>
<td>78.2±0.5</td>
<td>68.0±0.8</td>
<td>45.7±0.9</td>
<td>43.4±0.1</td>
<td>77.9±0.1</td>
<td>66.4</td>
</tr>
<tr>
<td>GAM [121]</td>
<td>86.1±0.6</td>
<td>78.5±0.4</td>
<td>68.2±1.0</td>
<td>45.2±0.6</td>
<td>43.8±0.1</td>
<td>78.0±0.2</td>
<td>66.6</td>
</tr>
<tr>
<td>FAD (Ours)</td>
<td>88.2±0.5</td>
<td>78.9±0.8</td>
<td>69.2±0.5</td>
<td>45.7±1.0</td>
<td>44.4±0.1</td>
<td>79.0±0.1</td>
<td>67.6</td>
</tr>
</tbody>
</table>

Figure 2. Comparison of computation overhead. Red points on folding lines indicate models trained with FAD for various ratios of iterations (From left to right in each figure, the proportions are 100%, 50%, 10%, 5%).

4.4. Computation Overhead

Since flatness-aware optimization methods require more computation overhead than traditional optimizers, we report the comparison of FAD with other optimizers with more training iterations. We train current optimizers on DG benchmarks for 5,000, 10,000, and 15,000 iterations. We report the performance and required average training time in Table 3. Most optimizers achieve noticeable improvements in performance with more training iterations on PACS, VLCS, and OfficeHome. We show that FAD still outperforms current optimizers with fewer training iterations. For example, models trained with FAD for 5,000 iterations on PACS and NICO++ outperform those trained with other optimizers for 15,000 iterations. This indicates the effectiveness and efficiency of FAD.

Furthermore, as discussed in Section 3.3, the FAD gradient can be efficiently calculated via first-order gradient approximation. Yet it still introduces extra computation when calculated in each iteration. Here we investigate optimizing the FAD term only for a subset of iterations in each epoch and optimize the remaining iterations simply with SGD on PACS and NICO++. We change the ratio of the subset with FAD from 5% to 100% and report the performance and training speed. We set the batch size to 96 (32×3) for PACS and 128 (32×4) for NICO++ for all the methods for a fair
Table 2. Ensemble with domain generalization methods. Numbers for methods marked with * and all the combinations of DG methods with optimizers other than Adam are reproduced results. Other results are from the original literature and DomainBed (donated with †).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PACS</th>
<th>VLCS</th>
<th>OfficeHome</th>
<th>TerraInc</th>
<th>DomainNet</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASF [23]</td>
<td>82.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DMC [15]</td>
<td>83.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>43.6</td>
<td>-</td>
</tr>
<tr>
<td>MetaReg [7]</td>
<td>83.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>43.6</td>
<td>-</td>
</tr>
<tr>
<td>ER [124]</td>
<td>85.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pAdalN [77]</td>
<td>85.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EISNet [100]</td>
<td>85.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DSON [90]</td>
<td>86.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ERM† [96]</td>
<td>85.5</td>
<td>77.5</td>
<td>66.5</td>
<td>46.1</td>
<td>40.9</td>
<td>63.3</td>
</tr>
<tr>
<td>ERM* (with Adam)</td>
<td>84.2</td>
<td>77.3</td>
<td>67.6</td>
<td>44.4</td>
<td>43.0</td>
<td>63.3</td>
</tr>
<tr>
<td>IRM† [3]</td>
<td>83.5</td>
<td>78.6</td>
<td>64.3</td>
<td>47.6</td>
<td>33.9</td>
<td>61.6</td>
</tr>
<tr>
<td>GroupDRO† [88]</td>
<td>84.4</td>
<td>76.7</td>
<td>66.0</td>
<td>43.2</td>
<td>33.3</td>
<td>60.7</td>
</tr>
<tr>
<td>I-Mixup† [108, 113, 102]</td>
<td>84.6</td>
<td>77.4</td>
<td>68.1</td>
<td>47.9</td>
<td>39.2</td>
<td>63.4</td>
</tr>
<tr>
<td>MLDG† [56]</td>
<td>84.9</td>
<td>77.2</td>
<td>66.8</td>
<td>47.8</td>
<td>41.2</td>
<td>63.6</td>
</tr>
<tr>
<td>MMD† [60]</td>
<td>84.7</td>
<td>77.5</td>
<td>66.4</td>
<td>42.2</td>
<td>23.4</td>
<td>58.8</td>
</tr>
<tr>
<td>DANN† [30]</td>
<td>83.7</td>
<td>78.6</td>
<td>65.9</td>
<td>46.7</td>
<td>38.3</td>
<td>62.6</td>
</tr>
<tr>
<td>CDANN† [61]</td>
<td>82.6</td>
<td>77.5</td>
<td>65.7</td>
<td>45.8</td>
<td>38.3</td>
<td>62.0</td>
</tr>
<tr>
<td>MTL† [11]</td>
<td>84.6</td>
<td>77.2</td>
<td>66.4</td>
<td>45.6</td>
<td>40.6</td>
<td>62.9</td>
</tr>
<tr>
<td>SagNet† [75]</td>
<td>86.3</td>
<td>77.8</td>
<td>68.1</td>
<td>48.6</td>
<td>40.3</td>
<td>64.2</td>
</tr>
<tr>
<td>ARM† [119]</td>
<td>85.1</td>
<td>77.6</td>
<td>64.8</td>
<td>45.5</td>
<td>35.5</td>
<td>61.7</td>
</tr>
<tr>
<td>VREx† [52]</td>
<td>84.9</td>
<td>78.3</td>
<td>66.4</td>
<td>46.4</td>
<td>33.6</td>
<td>61.9</td>
</tr>
<tr>
<td>RSC† [38]</td>
<td>85.2</td>
<td>77.1</td>
<td>65.5</td>
<td>46.6</td>
<td>38.9</td>
<td>62.7</td>
</tr>
<tr>
<td>Mixstyle [129]</td>
<td>85.2</td>
<td>77.9</td>
<td>60.4</td>
<td>44.0</td>
<td>34.0</td>
<td>60.3</td>
</tr>
<tr>
<td>MIRO* [14]</td>
<td>85.4</td>
<td>78.9</td>
<td>69.5</td>
<td>45.4</td>
<td>44.0</td>
<td>64.6</td>
</tr>
</tbody>
</table>

Comparison. As shown in Figure 2, FAD consistently outperforms SGD, Adam, and AdamW by considerable margins and surpasses SAM and GAM with less training time when setting the ratio of FAD to 10% and 50%. Detailed results are in Appendix B.

4.5. The Hessian spectra of selected minima

Sharpness-aware optimization methods are shown to decrease the curvature of loss landscape [28, 121]. We investigate how FAD affects the flatness of minima. We consider the maximum eigenvalue of Hessian, which indicates the sharpness across the most critical direction, and the Hessian trace, which measures the expected loss increase under random perturbations to the weights [42] as the measures of flatness.

We show the Hessian spectra of ResNet-50 trained on art, painting, cartoon, and photo domains of PACS for 5,000 iterations with SGD, Adam, AdamW, GAM, SAM, and FAD. Following [121], we adopt power iteration [115] to compute the top eigenvalues of Hessian and Hutchinson’s method [6, 5, 114] to compute the Hessian trace. We report the distribution of the top-50 Hessian eigenvalues and Hessian trace for each method at convergence in Figure 3. None-flatness-aware optimizers, including SGD, Adam, and AdamW show high Hessian trace and dominant eigenvalues. In contrast, flatness-aware optimizers find
Table 3. The performance and average training time of current optimizers and FAD when training for 5,000, 10,000, 15,000 iterations. Training Time is the average time for finishing each dataset with the given iterations.

<table>
<thead>
<tr>
<th>Optimizers</th>
<th>Iterations</th>
<th>Training Time /s</th>
<th>PACS</th>
<th>VLCS</th>
<th>OfficeHome</th>
<th>NICO++</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>5,000</td>
<td>1613.9</td>
<td>84.2</td>
<td>77.3</td>
<td>67.6</td>
<td>76.2</td>
<td>76.3</td>
</tr>
<tr>
<td>Adam</td>
<td>10,000</td>
<td>3049.5</td>
<td>82.9</td>
<td>77.7</td>
<td>67.4</td>
<td>76.9</td>
<td>76.2</td>
</tr>
<tr>
<td>Adam</td>
<td>15,000</td>
<td>4478.6</td>
<td>84.0</td>
<td>78.4</td>
<td>69.6</td>
<td>77.3</td>
<td>77.3</td>
</tr>
<tr>
<td>AdamW</td>
<td>5,000</td>
<td>2145.2</td>
<td>83.6</td>
<td>77.4</td>
<td>68.8</td>
<td>76.7</td>
<td>76.7</td>
</tr>
<tr>
<td>AdamW</td>
<td>10,000</td>
<td>4632.8</td>
<td>83.3</td>
<td>77.1</td>
<td>66.9</td>
<td>77.5</td>
<td>76.2</td>
</tr>
<tr>
<td>AdamW</td>
<td>15,000</td>
<td>6995.3</td>
<td>83.7</td>
<td>76.3</td>
<td>68.0</td>
<td>77.5</td>
<td>76.4</td>
</tr>
<tr>
<td>SGD</td>
<td>5,000</td>
<td>1864.1</td>
<td>79.9</td>
<td>78.1</td>
<td>68.5</td>
<td>75.8</td>
<td>75.4</td>
</tr>
<tr>
<td>SGD</td>
<td>10,000</td>
<td>3438.9</td>
<td>81.4</td>
<td>78.1</td>
<td>68.7</td>
<td>77.2</td>
<td>76.4</td>
</tr>
<tr>
<td>SGD</td>
<td>15,000</td>
<td>4992.0</td>
<td>82.6</td>
<td>78.7</td>
<td>69.6</td>
<td>77.0</td>
<td>77.0</td>
</tr>
<tr>
<td>YOGI</td>
<td>5,000</td>
<td>1388.9</td>
<td>81.2</td>
<td>77.6</td>
<td>68.3</td>
<td>76.1</td>
<td>76.1</td>
</tr>
<tr>
<td>YOGI</td>
<td>10,000</td>
<td>2882.9</td>
<td>81.2</td>
<td>77.8</td>
<td>70.2</td>
<td>77.9</td>
<td>76.8</td>
</tr>
<tr>
<td>YOGI</td>
<td>15,000</td>
<td>4434.2</td>
<td>83.1</td>
<td>78.1</td>
<td>67.9</td>
<td>77.5</td>
<td>76.7</td>
</tr>
<tr>
<td>SAM</td>
<td>5,000</td>
<td>3325.6</td>
<td>85.3</td>
<td>78.2</td>
<td>68.0</td>
<td>77.0</td>
<td>77.1</td>
</tr>
<tr>
<td>SAM</td>
<td>10,000</td>
<td>6506.8</td>
<td>85.6</td>
<td>78.3</td>
<td>68.6</td>
<td>77.9</td>
<td>77.6</td>
</tr>
<tr>
<td>SAM</td>
<td>15,000</td>
<td>9960.2</td>
<td>86.1</td>
<td>78.9</td>
<td>68.8</td>
<td>78.1</td>
<td>78.0</td>
</tr>
<tr>
<td>GAM</td>
<td>5,000</td>
<td>8073.6</td>
<td>86.1</td>
<td>78.5</td>
<td>68.2</td>
<td>77.4</td>
<td>77.6</td>
</tr>
<tr>
<td>GAM</td>
<td>10,000</td>
<td>16210.5</td>
<td>86.0</td>
<td>79.2</td>
<td>68.5</td>
<td>78.0</td>
<td>78.0</td>
</tr>
<tr>
<td>GAM</td>
<td>15,000</td>
<td>23855.1</td>
<td>86.5</td>
<td>78.6</td>
<td>69.0</td>
<td>77.8</td>
<td>78.0</td>
</tr>
<tr>
<td>FAD (Ours)</td>
<td>5,000</td>
<td>5057.8</td>
<td>88.2</td>
<td>78.9</td>
<td>69.2</td>
<td>78.5</td>
<td>78.7</td>
</tr>
<tr>
<td>FAD (Ours)</td>
<td>10,000</td>
<td>10450.8</td>
<td>88.5</td>
<td>79.5</td>
<td>69.6</td>
<td>79.0</td>
<td>79.1</td>
</tr>
<tr>
<td>FAD (Ours)</td>
<td>15,000</td>
<td>16255.2</td>
<td>88.5</td>
<td>79.0</td>
<td>69.0</td>
<td>79.1</td>
<td>78.9</td>
</tr>
</tbody>
</table>

Figure 3. The distribution of top eigenvalues and the trace of Hessian at convergence on PACS with SGD, Adam, AdamW, GAM, SAM, and FAD. Trace indicates Hessian trace and $\lambda_{\text{max}}$ indicates the maximum eigenvalue of Hessian.

minima with significantly lower Hessian spectra. FAD leads to both the lowest Hessian maximum eigenvalue and trace and thus the lowest curvature compared with other optimizers. More results are shown in Appendix B.

5. Discussions

Despite the wide usage in DG benchmarks, Adam is hardly the optimal choice of default optimizer. We show that the best optimizer for different DG methods varies across different datasets. Choosing the right optimizer can help exceed the current benchmarks’ limitations. We propose a unified optimization method, flatness-Aware minimization for Domain generalization (FAD), to optimize zeroth-order and first-order flatness efficiently to address the problem. We further show that FAD controls the OOD generalization error and convergences fast. Compared with current optimizers, flatness-aware optimization approaches, including SAM, GAM, and FAD, leads to flatter minima and better generalization. FAD outperforms its counterparts significantly on most DG datasets.

FAD still has the following limitations which could lead to potential future work. First, the term $\sup_{\theta_1, \theta_2 \in \Theta} |L_{\text{tr}}(\theta_1, \theta_2) - L_{\text{te}}(\theta_1, \theta_2)|$ in Equation (6) indicates the discrepancy between training and test data. Without knowledge of test distribution and extra assumptions, the term cannot be optimized and leads to the optimization of the flatness term. Proper assumptions on the connection between training and test data or the geometry of data may help to learn the connection between flatness and the discrepancy, and thus control the discrepancy for better generalization. Second, We show that Adam is hardly an optimal choice of default optimizer in DG. However, selecting a proper optimizer for each DG method leads to massive overhead. Designing a better DG protocol
or learning an indicator for the fitness of optimizers and methods can significantly contribute.

Acknowledgement

This work was supported in part by National Key R&D Program of China (No. 2018AAA0102004, No. 2020AAA0106300), National Natural Science Foundation of China (No. U1936219, 62141607), Beijing Academy of Artificial Intelligence (BAAI).

References


[78] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source


[130] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha C Dvornek, James s Duncan, Ting Liu, et al. Surrogate gap minimization improves sharpness-