Adaptively Sampling-Reusing-Mixing Decomposed Gradients to Speed Up Sharpness Aware Minimization

Jiaxin Deng, Junbiao Pang

Affiliation

email@example.com

Abstract

Sharpness-Aware Minimization (SAM) improves model generalization but doubles the computational cost of Stochastic Gradient Descent (SGD) by requiring twice the gradient calculations per optimization step. To mitigate this, we propose Adaptively sampling-Reusing-mixing decomposed gradients to significantly accelerate SAM (ARSAM). Concretely, we firstly discover that SAM's gradient can be decomposed into the SGD gradient and the Projection of the Second-order gradient onto the First-order gradient (PSF). Furthermore, we observe that the SGD gradient and PSF dynamically evolve during training, emphasizing the growing role of the PSF to achieve a flat minima. Therefore, ARSAM is proposed to the reused PSF and the timely updated PSF still maintain the model's generalization ability. Extensive experiments show that ARSAM achieves state-of-the-art accuracies comparable to SAM across diverse network architectures. On CIFAR-10/100, ARSAM is comparable to SAM while providing a speedup of about 40%. Moreover, ARSAM accelerates optimization for the various challenge tasks (*e.g.*, human pose estimation, and model quantization) without sacrificing performance, demonstrating its broad practicality.

1 Introduction

The powerful generalization ability of Deep Neural Networks (DNNs) has led to significant success in many fields [Chaudhari *et al.*, 2019][Izmailov *et al.*, 2018]. Several studies have verified the relationship between flat minima and the generalization ability [Dinh *et al.*, 2017] [Li *et al.*, 2018] [Jiang *et al.*, 2019][Liu *et al.*, 2020] [Sun *et al.*, 2021] [Yue *et al.*, 2023]. Among these studies, Jiang et al. [Jiang *et al.*, 2019] explored over 40 complexity measures and demonstrated that a sharpness-based measure exhibits the highest correlation with the generalization.

Based on the connection between sharpness of the loss landscape and model generalization, Foret et al. proposed Sharpness Aware Minimization (SAM) [Foret *et al.*, 2021] to
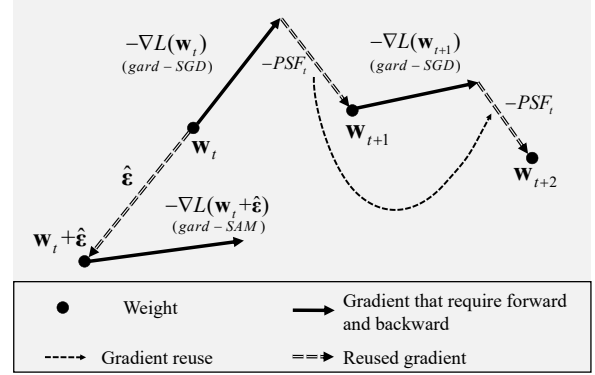

Figure 1: ARSAM speed up SAM by adaptively reusing the decomposed PSF for the next several iterations.

find parameter values whose entire neighborhoods have a uniformly low training loss value. Specifically, SAM minimizes both the loss value and the loss sharpness to obtain a flat minimum. SAM and its variants have demonstrated State-Of-The-Art (SOTA) performances across various applications, such as classification [Zhou *et al.*, 2023][Kwon *et al.*, 2021], transfer learning [Du *et al.*, 2022a][Zhuang *et al.*, 2022], domain generalization [Dong *et al.*, 2024][Xie *et al.*, 2024] and federated learning [Dai *et al.*, 2023]. However, SAM requires two forward and backward passes per iteration, which reduce its optimization speed to half that of SGD.

In this paper, we observe that SAM's gradient (hereinafter referred to as grad-SAM) can be decomposed into two components: the SGD gradient (hereinafter referred to as grad-SGD) which searches for a local minimum, and the Projection of the Second-order gradient onto the First-order gradient (PSF) which guides SAM toward a flat region. By reusing historical PSF, we approximate the current grad-SAM and thereby accelerate SAM. Fig. 1 illustrates how our method accelerates SAM by reusing the decomposed PSF from the t iteration to update the weights at the $t+1$ iteration. Therefore, only one forward and one backward are needed to update the weights $\mathbf{w}_{t+1}$, thus improving the optimization speed.

Importantly, we trained Resnet-18, WideResNet-28-10 and PyramidNet-110 on the CIFAR-100 by SAM and recorded the changes in the L2-norm of the SGD (hereinafter referred to as L2-SGD) (*i.e.*, $||\nabla L_i^{SGD}||$) and that of the the L2-norm
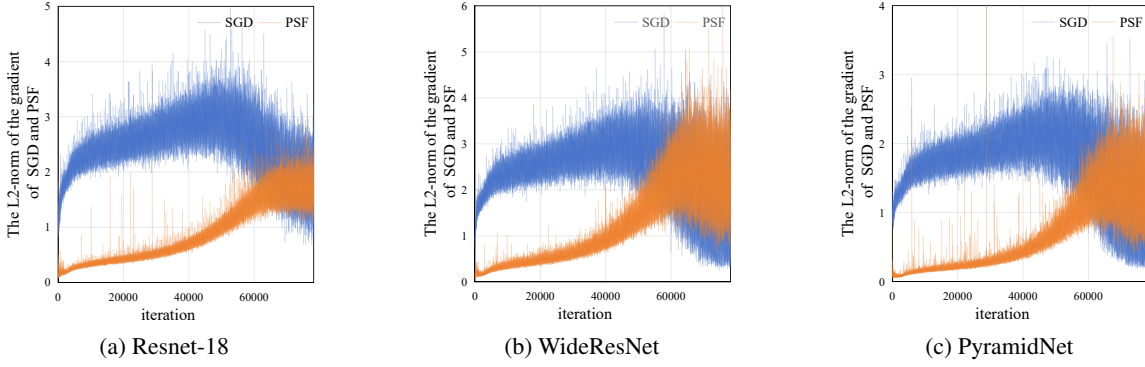
Figure 2: The trends of $||\nabla L_i^{SGD}||$ and $||\nabla L_i^{PSF}||$ are similar across the different networks (*i.e.*, Resnet-18, WideResNet-28-10, PyramidNet-110) during training (best viewed in color).

of PSF (hereinafter referred to as L2-PSF) (*i.e.*, $||\nabla L_i^{PSF}||$) at the i-th iteration in Fig.2. Two observations can be drawn from Fig.2 as follows:

- Both L2-SGD and L2-PSF change dynamically during training, indicating that the contributions of grad-SGD and PSF dynamically evolve throughout the learning process. In contrast, AE-SAM [Jiang *et al.*, 2023] only considers changes in L2-SGD during training.

- L2-PSF increases faster than L2-SGD with the increase of the iteration, especially at the later training stage. This indicates that PSF plays a key role in achieving a flat loss landscape at the later stage of training.

Inspired by the above observations, we propose accelerating SAM by adaptively reusing the decomposed PSF based on the contribution of the PSF. We utilize the ratio between the L2-PSF and L2-SGD as an indicator to reflect the contribution of the PSF during training.

The other important question is how to sample which iteration is used to decompose grad-SAM into the grad-SGD and the PSF. We built an autoregressive model based on the proposed indicator to adaptively adjust the sampling frequency. Specially, when the contribution of the PSF decreases, the autoregressive model reduces the computation frequency and reuses the historical PSF; otherwise, the autoregressive model increases the computation frequency of the PSF to improve optimization accuracy. Compared to uniform sampling (*e.g.*, SAM-5), non-uniform sampling highlights the role of the PSF at the different training stages. Unlike AE-SAM, we consider the different roles of PSF and grad-SGD at each iteration. The mixture of the reused PSF and the timely updated PSF would maintain the model's generalization ability.

In summary, our contributions are as follows:

- To the best of our knowledge, we firstly uncover that the grad-SAM can be decomposed into the grad-SGD and PSF. The discovery suggests that the gradient norm [Zhao *et al.*, 2022] is equivalent to SAM, except for the detailed optimization rule.

- We also observe that the L2-PSF increases gradually during training across different networks, motivating us

to adaptively compute the PSF to accelerate SAM. Thus, we introduce ARSAM, which adaptively reuses the decomposed PSF to not only preserve the model's generalization ability but also speed up SAM.

- The empirical results demonstrate that ARSAM achieves a 40% acceleration in speed compared to SAM while ensuring almost no decrease in model generalization ability. Additionally, ARSAM can be applied to various tasks, such as human pose estimation, and model quantization.

2 Related Works

2.1 Background of SAM

Foret et al. [Foret *et al.*, 2021] introduced an effective approach to improving the model's generalization ability. The optimization process of SAM can be viewed as addressing a minimax optimization problem as follows:

$$
\min_{\mathbf{w}} L^{SAM}(\mathbf{w}) + \lambda||\mathbf{w}||_2^2 \\
where \; L^{SAM}(\mathbf{w}) = \max_{||\boldsymbol{\varepsilon}|| \le \rho} L(\mathbf{w} + \boldsymbol{\varepsilon}), \quad (1)
$$

where $\mathbf{w}$ represents the weights of the network, $\boldsymbol{\varepsilon}$ represents the perturbation of weights $\mathbf{w}$ in a Euclidean ball with the radius ρ ($\rho > 0$), $L(\cdot)$ is the loss function, and $\lambda||\mathbf{w}||_2^2$ is a standard L2 regularization term.

SAM utilizes Taylor expansion to search for the maximum perturbed loss ($L^{SAM}(\mathbf{w})$) in local parameter space as follows:

$$
\arg\max_{||\boldsymbol{\varepsilon}|| \le \rho} L(\mathbf{w} + \boldsymbol{\varepsilon}) \approx \arg\max_{||\boldsymbol{\varepsilon}|| \le \rho} L(\mathbf{w}) + \boldsymbol{\varepsilon}^\top \nabla_{\mathbf{w}} L(\mathbf{w}) \\
= \arg\max_{||\boldsymbol{\varepsilon}|| \le \rho} \boldsymbol{\varepsilon}^\top \nabla_{\mathbf{w}} L(\mathbf{w}). \quad (2)
$$

By solving Eq. (2), SAM obtains the perturbation as follows:

$$
\hat{\boldsymbol{\varepsilon}} = \rho \nabla_{\mathbf{w}} L(\mathbf{w}) / ||\nabla_{\mathbf{w}} L(\mathbf{w})||. \quad (3)
$$

$\hat{\boldsymbol{\varepsilon}}$ can maximize the perturbed loss. SAM computes the gradient $\nabla_{\mathbf{w}} L(\mathbf{w})$ to obtain the perturbation. Substituting the

perturbation $\hat{\varepsilon}$ back into Eq. (1) and differentiating, we then have:

$$\begin{aligned}
\nabla_{\mathbf{w}} L^{SAM}(\mathbf{w}) &\approx \nabla_{\mathbf{w}} L(\mathbf{w} + \hat{\varepsilon}(\mathbf{w})) \\
&= \frac{d(\mathbf{w} + \hat{\varepsilon}(\mathbf{w}))}{d\mathbf{w}} \nabla_{\mathbf{w}} L(\mathbf{w})|_{\mathbf{w}+\hat{\varepsilon}(\mathbf{w})} \\
&= \nabla_{\mathbf{w}} L(\mathbf{w})|_{\mathbf{w}+\hat{\varepsilon}(\mathbf{w})} + \frac{d\hat{\varepsilon}(\mathbf{w})}{d\mathbf{w}} \nabla_{\mathbf{w}} L(\mathbf{w})|_{\mathbf{w}+\hat{\varepsilon}(\mathbf{w})}.
\end{aligned} \quad (4)$$

By dropping the second-order terms in Eq.(4), SAM calculates the gradient at $\mathbf{w} + \hat{\varepsilon}$ as follows:

$$\nabla_{\mathbf{w}} L^{SAM}(\mathbf{w}) \approx \nabla_{\mathbf{w}} L(\mathbf{w})|_{\mathbf{w}+\hat{\varepsilon}}. \quad (5)$$

Finally, SAM computes gradient at the perturbed point $\nabla_{\mathbf{w}} L(\mathbf{w})|_{\mathbf{w}+\hat{\varepsilon}}$ for optimization. From the above process, we observe that SAM necessitates two forward and backward operations to update weights once.

2.2 Efficient optimization methods for SAM

Methods for speeding up SAM are broadly categorized into two groups: 1) accelerating the gradient computation process and 2) reducing gradient computation.

In the first category, Du et al. propose SAF and MESA [Du *et al.*, 2022b], which employ trajectory loss to approximate sharpness through a surrogate sharpness measure. However, SAF requires significant memory to store all output results, while MESA needs to maintain an exponential moving average model during training. Du et al. propose ESAM [Du *et al.*, 2022a], which employs Stochastic Weight Perturbation (SWP) and Sharpness Sensitive Data Selection (SDS) strategy to reduce computation. SWP approximates weight perturbation using a subset of model weights, while SDS focuses on training data that most impact sharpness. Deng et al. propose AUSAM [Deng *et al.*, 2024] to sample a subset of data from the mini-batch based on the estimated average gradient norm, reducing the forward and backward time.

In the second category, Liu et al. propose LookSAM [Liu *et al.*, 2022] which periodically calculates grad-SAM every 5 iterations and reuses previous gradients in other iterations. However, this uniform gradient sampling fails to reflect the importance of sharpness optimization in the overall optimization process. Subsequently, Jiang et al. claims that the SAM update is more useful in sharp regions than in flat regions. As a result, they designs AE-SAM [Jiang *et al.*, 2023] to adaptively employ SAM based on the loss landscape geometry. ARSAM belongs to this research line. Unlike the approaches mentioned above, ARSAM adaptively computes grad-SAM to accelerate optimization and reuses PSF to ensure the model's generalization performance.

3 Method

3.1 Gradient Composition of SAM

We rewrite Eq. (5) by Taylor expansion and substitute $\hat{\varepsilon}$ to obtain the following:

$$\begin{aligned}
\nabla_{\mathbf{w}} L(\mathbf{w} + \hat{\varepsilon}) &\approx \nabla_{\mathbf{w}}(L(\mathbf{w}) + \hat{\varepsilon}\nabla_{\mathbf{w}} L(\mathbf{w})) \\
&= \nabla_{\mathbf{w}}(L(\mathbf{w}) + \rho||\nabla_{\mathbf{w}} L(\mathbf{w})||).
\end{aligned} \quad (6)$$

Eq. (6) indicates that the grad-SAM can be considered as a combination of the grad-SGD $\nabla_{\mathbf{w}} L(\mathbf{w})$, and the gradient of the L2-norm of SGD's gradient $\nabla_{\mathbf{w}}\rho||\nabla_{\mathbf{w}} L(\mathbf{w})||$.

We expand the second term in Eq. (6) as follows:

$$\nabla_{\mathbf{w}}(\rho||\nabla_{\mathbf{w}} L(\mathbf{w})||) = \rho\frac{\nabla_{\mathbf{w}} L(\mathbf{w})\nabla_{\mathbf{w}}^2 L(\mathbf{w})}{||\nabla_{\mathbf{w}} L(\mathbf{w})||}. \quad (7)$$

The gradient of the L2-norm of SGD's gradient (hereinafter referred to as grad-L2-SGD) can be transformed into the Projection of the Second-order gradient onto the First-order gradient, which we refer to as PSF. Eq. (6) and Eq. (7) indicate that SAM's gradient can be decomposed into the SGD gradient and the PSF. Naturally, the PSF prompts SAM to find a flat minima.

Connect PSF to gradient norm [Zhao *et al.*, 2022]: By comparing Eq.(6) in [Zhao *et al.*, 2022] ($\nabla_\theta L(\theta) = \nabla_\theta L_S(\theta) + \lambda \cdot \nabla_\theta^2 L_S(\theta)\frac{\nabla_\theta L_S(\theta)}{||\nabla_\theta L_S(\theta)||}$) with Eq. (7) in our paper, we discover that the PSF ($\frac{\nabla_{\mathbf{w}} L(\mathbf{w})\nabla_{\mathbf{w}}^2 L(\mathbf{w})}{||\nabla_{\mathbf{w}} L(\mathbf{w})||}$) is consistent with the second term on the right side of Eq. (6) in [Zhao *et al.*, 2022]. This indicates that our proposed acceleration method is applicable not only to SAM but also to the gradient norm [Zhao *et al.*, 2022].

Theorem 1. *Let $\nabla_{\mathbf{w}}^2 L(\mathbf{w})$ be a hessian matrix with n eigenvalues, then the L2-PSF has an upper bound as follows:*

$$||\nabla_{\mathbf{w}}(\rho||\nabla_{\mathbf{w}} L(\mathbf{w})||)|| \leq \rho\sum_{i=1}^{n}\delta_i, \quad (8)$$

where δ_i is the i-th eigenvalue.

Remark. Theorem 1 discovers that the L2-PSF is related to the eigenvalues of the matrix $\nabla_{\mathbf{w}}^2 L(\mathbf{w})$. As the sum of the eigenvalues decreases, the L2-PSF becomes smaller.

Without considering the direction of the PSF, we empirically studied the behavior of the L2-PSF in Fig. 2 and observed that the L2-PSF (*i.e.*, $||\nabla L_i^{PSF}|| = ||\nabla L_i^{SAM} - \nabla L_i^{SGD}||$) gradually increases during training. This phenomenon reflects that the significance of the PSF dynamically changes during the training process. One natural question is: When and how to adaptively sample and reuse the PSF based on its behavior to accelerate optimization while guaranteeing generalization?

3.2 Sampling-Reusing-Mixing Decomposed Gradient

Sampling the PSF based on the ratio
Firstly, we use the ratio of L2-PSF to L2-SGD as an indicator to evaluate the contribution of PSF in finding a local minimum, as follows:

$$c_i = \frac{||\nabla L_i^{PSF}||}{||\nabla L_i^{SGD}||}, \quad (9)$$

where $||\nabla L_i^{PSF}||$ and $||\nabla L_i^{SGD}||$ are the L2-PSF and the L2-SGD at the i-th iteration, respectively. We only considered the magnitude of the gradients, not their directions, as Theorem 1 discovers that the magnitude is enough to calibrate the contribution of the PSF and SGD during the optimization.

We have observed that the distribution of the c_i at the same iteration from the different sizes and architecture of networks is very different, as illustrated in Fig. 2. If we directly use c_i in Eq. (9), it becomes difficult to keep its range under control across different networks. Therefore, we consider the relative change of the indicator c_i at the current iteration, which is normalized by the previous iteration as follows:

$$r_i = \frac{c_i - c_{i-1}}{c_{i-1}}. \tag{10}$$

The aim of Eq. (10) is to model the change in c_i through $c_i - c_{i-1}$ and normalize the magnitude of c_i across different models and datasets using $1/c_i$. The normalization in (10) helps different networks have the same characteristic as illustrated in Fig. 2.

Furthermore, to smooth r_i, we apply the Exponential Moving Average (EMA) strategy to smooth r_i as follows:

$$\hat{r}_i = \beta \cdot \hat{r}_{i-1} + (1 - \beta) \cdot r_i, \tag{11}$$

where β is the weighted weight, typically set to 0.9.

We construct an autoregressive model to transform the smoothed $\hat{r}_i$ into the number of PSF to be sampled. We chose the autoregressive model because of its simplicity and generality; in addition, it is easy to implement and does not impose a heavy computational burden. We divide the total N iterations into a series of segments where each length is M. In other words, for every M iteration, we update the number of times the PSF will be sampled in the next segment as follows:

$$s_{j+1} = s_j \cdot (1 + \alpha \cdot \hat{r}_k), \tag{12}$$

where s_{j+1} represent that how many PSF would be sampled in the next M iterations, $k = j \cdot M$ represent the index of iteration in a training, the hyperparameter α adjusts the increase/decrease ratio of s_{j+1}. We set s_0 to 1 in our experiments. That is, only 1 PFS is computed in the first segment. Finally, the sampling probability of PSF for the next M iterations is defined as follows:

$$p_{j+1} = \frac{s_{j+1}}{M}. \tag{13}$$

where $j = 1, \ldots, M$. We perform gradient sampling with the probability p_{j+1} for the next $1, \ldots, M$ iterations.

Approximate the L2-PSF and L2-SGD: As the model size increases, the computational cost of the L2-norm grows. As shown in Appendix 1.5, we observe that the L2-SGD and L2-PSF in the last few layers of the model also capture the changes in the gradient of SGD and PSF. Thus, we replace $||\nabla L^{SGD}||$ and $||\nabla L^{PSF}||$ with the L2-SGD and the L2-PSF in the last few layers of the model, respectively.

Advantages over LookSAM and AE-SAM: In contrast to updating grad-SAM uniformly in LookSAM [Liu *et al.*, 2022], ARSAM computes the PSF in a non-uniform manner to reflect its importance during training. This approach helps schedule the frequency of PSF computation more effectively, thus maintaining the model's generalization ability.

In contrast to AE-SAM, which establishes a distribution about L2-SGD to decide whether to replace SAM with SGD, ARSAM uses the ratio between L2-PSF and L2-SGD (*i.e.* Eq. (9)) as an indicator to reuse PSF. If the reuse error is small

---

Algorithm 1 Pseudocode of the proposed ARSAM

Require: The training dataset, the learning rate η, parameters α, ρ, M, s_1 and I_{start}.
1: **for** $i = 1, 2, \cdots$ **do**
2:     Calculate the grad-SGD ∇L_i^{SGD};
3:     **if** $i <= I_{start}$ **or** sampling **then**
4:         Calculate the grad-SAM ∇L_i^{SAM} in Eq. (5);
5:         Calculate the PSF ∇L_i^{PSF};
6:         Update the weights according to Eq. (14);
7:     **else**
8:         Update the weights according to Eq. (15);
9:     **end if**
10:    **if** $i > I_{start}$ **and** $i \% M = 0$ **then**
11:       Update the sampling probability by Eq. (13);
12:    **end if**
13: **end for**

---

enough, it can be treated as minor noise in the gradient. As the [Keskar *et al.*, 2017] explains, this noise helps push the iterates away from sharp minimizers and toward flatter ones, potentially allowing ARSAM to find a flatter minimum.

Reusing-Mixing decomposed gradients
When the computation probability of PSF is obtained, we determine whether to compute PSF in each iteration based on this probability. The weight update process is as follows:

- When computing the PSF, the optimization rule can be written as follows:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \nabla L_i^{SAM}, \tag{14}$$

where η is the learning rate. Eq. (14) is essentially the optimization rule for SAM.

- When PSF is not computed, we reuse the PSF from the last computed iteration, the optimization rule can be written as follows:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta(\nabla L_i^{SGD} + \nabla L_{\hat{i}}^{PSF}), \tag{15}$$

where $\nabla L_{\hat{i}}^{PSF}$ is the PSF at the last computed iteration.

Algorithm 1 shows the overall proposed algorithm.

Convergence analysis
We further analyze the convergence properties of ARSAM as follows.

Theorem 2. *Suppose* $L^{SGD}(\mathbf{w})$ *is* τ-*Lipschitz smooth and* $|L^{SGD}(\mathbf{w})|$ *is bounded by* U. *For any* $t \in \{0, ..., T\}$ *and any* $\mathbf{w} \in W$, *suppose we can obtain bounded observations as follows:*

$$||\hat{\boldsymbol{\xi}}_t||^2 \leq G_1, \ \mathbb{E}[\nabla L_t^{SGD}(\mathbf{w})] = \mathbb{E}[\hat{\mathbf{g}}_t] = \mathbf{g}_t, \ ||\hat{\mathbf{g}}_t|| \leq G_2,$$

$$\mathbb{E}[\nabla L_t^{PSF}(\mathbf{w})] = \mathbb{E}[\hat{\mathbf{h}}_t] = \mathbf{h}_t, \ ||\hat{\mathbf{h}}_t|| \leq G_3. \tag{16}$$

Then with learning rate $\eta_t = \frac{\eta_0}{\sqrt{t}}$, *we have the following bound for ARSAM:*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[||\mathbf{g}_t + \mathbf{h}_t + \boldsymbol{\xi}_t||^2] \leq \frac{2D_1 + 2D_2 InT}{\sqrt{T}} + 2\tau, \tag{17}$$

where $\boldsymbol{\xi}_t$ can be specifically defined as follows:

$$\boldsymbol{\xi}_t = \begin{cases} -\mathbf{h}_t & When\ updating\ with\ the\ gradient\ of\ SGD \\ 0 & When\ updating\ with\ gradient\ reusing \\ \hat{\mathbf{h}}_t - \mathbf{h}_t & When\ updating\ with\ the\ gradient\ of\ SAM \end{cases} \quad (18)$$

where the true PSF is $\mathbf{h}_t$ and the reused PSF is $\hat{\mathbf{h}}_t$.

Remark. Theorem 2 uncovers that the convergence of ARSAM is affected by $D1$, $D2$, T, and τ. Concretely, $D1$ and $D2$ are dependent on factors such as τ, η_0, G_1, G_2, G_3 and U, all of which must be controlled within a certain range. Therefore, the bounded gradients mean that the optimizer would converge to a local minimum, as discussed in [Andriushchenko and Flammarion, 2022].

The difference between the reuse PSF and the true PSF
Theorem 3. *Assume that the weight at t-th iteration is $\mathbf{w}_t$, and the index of the reused gradients is $t + n(n > 0)$, i.e., $\mathbf{w}_{t+n}$. Assuming that $\frac{\nabla L_B(\mathbf{w}_t)}{\|\nabla L_B(\mathbf{w}_t)\|}$ is Lipschitz continuous and τ is Lipschitz constant. The expected error between the reused PSF with the real PSF is bounded as follows:*

$$\left\| \mathbb{E}_B \left[\nabla^2 L_B(\mathbf{w}_{t+n}) \frac{\nabla L_B(\mathbf{w}_{t+n})}{\|\nabla L_B(\mathbf{w}_{t+n})\|} - \nabla^2 L_B(\mathbf{w}_t) \frac{\nabla L_B(\mathbf{w}_t)}{\|\nabla L_B(\mathbf{w}_t)\|} \right] \right\|$$
$$\leq n\eta\tau \mathbb{E} \left[Tr(\nabla^2 L(\mathbf{w}_t)) \right] \|\mathbb{E}[\nabla L(\mathbf{w})]\| \quad (19)$$

where B represents a mini-batch data.

Remark. The Lipschitz constant and the trace are associated with the flatness of the loss landscape. Theorem 3 shows that as the optimizer reaches a flat loss landscape, the upper bound in Eq. (19) would become small. Moreover, we could reduce the upper bound of the error by controlling the distance between $\mathbf{w}_t$ and $\mathbf{w}_{t+n}$. Therefore, the gradient reuse in Section 3.2 is empirically applied at the two consecutive iterations after sampling iteration.

Acceleration ratio of ARSAM over SAM As shown in Appendix 1.4, we discover that the mean of the indicator $\hat{r}_i$ gradually increases with the index of iterations and approximately follows a quadratic function. To simplify and formalize the acceleration of ARSAM, we assume that the average trend of the indicator $\hat{r}_i$ follows the function $\hat{r}_i = \gamma \cdot i^2$, where i represents the iteration index, and γ reflects the curvature of the function. The total number of times ARSAM samples the PSF during the entire training process can be expressed as:

$$s^* = \sum_{j=1}^{I/M} s_j, \quad where\ s_j = s_{j-1}(1 + \alpha \cdot \gamma \cdot (j \cdot M)^2), \quad (20)$$

where α controls the variation of n_j, and M indicates that $\hat{r}_i$ is sampled every M intervals, I represents the total number of iterations. Assume that gradient computation dominates the optimization time, with other factors neglected, and that each gradient computation takes the same amount of time. The optimization speed of ARSAM compared to SAM is approximately as follows:

$$v = 2I/(I + s^*). \quad (21)$$

These demonstrate that ARSAM effectively accelerates SAM. More details can be found in Appendix 1.4.

4 Experimental Results

4.1 Setup

Datasets and Models
To verify the effectiveness of ARSAM, we conduct experiments on CIFAR-10, CIFAR-100 [Krizhevsky *et al.*, 2009] and Tiny-ImageNet [Le and Yang, 2015] image classification benchmark datasets. We employ a variety of architectures to evaluate the performance and training efficiency, i.e. ResNet-18 [He *et al.*, 2016], WideResNet-28-10 [Zagoruyko and Komodakis, 2016] and PyramidNet-110 [Han *et al.*, 2017] on CIFAR-10 and CIFAR-100, ResNet-18 and MobileNets [Howard *et al.*, 2017] on Tiny-ImageNet.

Baselines and the State-Of-The-Arts (SOTAs)
We take the vanilla SGD and SAM [Foret *et al.*, 2021] as baselines. To comprehensively evaluate the performance of ARSAM, we have also chosen some SOTA methods, ESAM [Du *et al.*, 2022a], SAM-5 [Liu *et al.*, 2022], LookSAM [Liu *et al.*, 2022], SAF [Du *et al.*, 2022b], MESA [Du *et al.*, 2022b] and AE-SAM [Jiang *et al.*, 2023] for comparison. These efficient methods are the follow-up works of SAM that aim to enhance efficiency.

Implementation details
On CIFAR-10 and CIFAR-100, we train all the models 200 epochs using a batch size of 128 with cutout regularization [DeVries and Taylor, 2017] and cosine learning rate decay [Loshchilov and Hutter, 2016]. For Tiny-ImageNet, we also train ResNet-18 and MobileNets [Howard *et al.*, 2017] 200 epochs using a batch size of 128 with cosine learning rate decay. For ResNet-18, we use the 64×64 resolution images and transform the first convolution of the models to a 3×3 convolution. For MobileNets, we resize the original image to 224×224. M was 50 for all experiments. We implement ARSAM in Pytorch and train models on a single NVIDIA GeForce RTX 3090 three times with different seed and reported the average results and standard deviation. The implementation details can be found in Appendix.

In this paper, optimization efficiency is quantified by the Average number of Images processed by the model per Second (AIS) as follows:

$$AIS = (D \cdot E)/T, \quad (22)$$

where D represents the amount of data for one epoch of training, T is the total training time in seconds, including both data loading and model optimization time, and E represents the training epoch. Additionally, we report the proportion of times the gradient of SAM was calculated during the training iterations, denoted as %SAM. Since the runtime of the program varies across different devices, we use %SAM and AIS for fair comparison. The same evaluation method is also used in [Jiang *et al.*, 2023].

4.2 Parameter Studies

We study the effect of α using the WideResNet-28-10 model on the CIFAR-100 dataset. The results in Table 1 show that as $alpha$ increases, the number of times to compute PSF also

α	Accuracy	%SAM	AIS
SAM	84.71	100%	351(100%)
0.1	83.07	7.4%	**620(177%)**
0.2	83.63	16.5%	581(166%)
0.3	84.39	27.7%	523(149%)
0.4	**84.67**	38.2%	486(139%)
0.5	84.65	40.4%	482(137%)

Table 1: Parameter Study of α on CIFAR-100 datasets with WideResNet. The best accuracy and efficiency are bolded.

	CIFAR-10		CIFAR-100	
Methods	Acc.	AIS	Acc.	AIS
SAM	<u>96.79</u>	1855(100%)	<u>81.04</u>	1866(100%)
SAM-5	96.42	2741(148%)	80.59	2715(146%)
ARSAM	**96.63**	2591(140%)	**80.92**	2537(136%)
ARSAM-A	96.61	2542(137%)	80.60	2459(134%)

Table 2: Ablation Study of ARSAM on CIFAR-10 and CIFAR-100 with ResNet-18. The best accuracy is in bold and the second best is underlined.

increases, slowing the training speed. However, accuracy improves with more times to compute PSF. Based on our empirical experience, we recommend setting α between 0.1 and 1. For challenge datasets and models with large parameters, a higher α is preferable. In this paper, we set $\alpha = 0.2$ for CIFAR-10, $\alpha = 0.4$ for CIFAR-100 to achieve a 40% improvement in optimization speed compared to SAM, and $\alpha = 0.8$ for the more challenging Tiny-ImageNet dataset.

4.3 Comparison to SOTAs

We trained ResNet-18, WideResNet-28-10, and PyramidNet-110 using different optimizer on CIFAR-10 and CIFAR-100, with results shown in Table 3. For Tiny-ImageNet, we trained ResNet-18 and MobileNets using SGD, SAM, and ARSAM, with results in Table 6.

Accuracy. From Table 3, ARSAM achieves significantly higher accuracy than SGD and SAM-5, and its accuracy is comparable to SAM. Compared to LookSAM, ESAM, and AE-SAM, ARSAM outperforms in most cases, demonstrating its ability to preserve generalization during training. Because ARSAM calculates the PSF in more appropriate optimization iterations, it promotes generalization. ARSAM reuses the PSF also guarantees the generalization ability of the model. ARSAM outperforms SAF and MESA because these methods optimize sharpness over several iterations, which may overlook local sharpness, as multiple local sharpness often occurs during multiple optimization iterations [Zhang *et al.*, 2023]. For Tiny-ImageNet, ARSAM achieves results comparable to SAM on ResNet-18 and MobileNets, showing its versatility across datasets and models.

Efficiency. In Table 3, SAM's optimization speed is about half that of SGD for WideResNet-28-10 and PyramidNet-110, but slightly faster than half for ResNet-18 due to data loading time being significant for small model but negligible

for larger model. ARSAM achieves about 40% faster training than SAM while maintaining comparable accuracy. SAM-5 enhance optimization speed but suffer a significant decrease in accuracy. In some cases, ARSAM is faster than Look-SAM and ESAM. Because LookSAM includes extra computations like gradient projection, and ESAM still requires two gradient computations per optimization step. SAF sacrifices memory for optimization efficiency, achieving nearly the same speed as SGD. MESA accelerates training by utilizing an EMA model and eliminating the need for backward when updating the EMA model. Although ARSAM may not match the speed of SAF and MESA, it excels at preserving the model's generalization ability. ARSAM also achieves comparable performance to AE-SAM with fewer SAM updates. For Tiny-ImageNet, the training efficiency of ARSAM is higher than that of SAM.

4.4 Ablation Study

To better understand the impact of gradient reuse on performance and efficiency, we introduce a variant of ARSAM that does not utilize gradient reuse, referred to as ARSAM-A. The results are shown in Table 2. Comparing ARSAM with ARSAM-A, we observe that the gradient reuse strategy can prevent the model's generalization ability without increasing the computational cost.

4.5 Robustness to Label Noise

Previous research has shown that SAM is robust to label noise. This subsection explores the effectiveness of ARSAM in the classic noisy-label setting for CIFAR-10 and CIFAR-100. We simulate symmetric label noise by randomly flipping labels [Huang *et al.*, 2019]. In these experiments, we set $\alpha = 0.3$ for CIFAR-10 and $\alpha = 0.4$ for CIFAR-100.

As shown in Table 5, for CIFAR-10, models optimized with SGD experience a sharp accuracy drop, falling to 28.91% at 80% label noise. In contrast, SAM and ARSAM models maintain around 70% accuracy even at 80% noise. A similar trend is observed for CIFAR-100, though SAM performs worse than SGD at 80% noise, likely due to SAM's difficulty in converging under high label noise, as noted in [Foret *et al.*, 2021]. However, ARSAM consistently outperforms both SGD and SAM, demonstrating superior robustness in high-noise conditions. This can be attributed to AR-SAM's predominant use of SGD in the early stages of training to efficiently identify local minima and learn the clean samples. In the later stages, ARSAM transitions to leveraging SAM to find flatter minima, thereby enhancing generalization. Additionally, ARSAM achieves approximately a 25% speedup compared to SAM.

4.6 Application to Semantic Segmentation

We conduct semantic segmentation experiments on the PASCAL VOC 2012 dataset [Everingham *et al.*, 2010], which includes 20 object categories and one background class. PSPNet [Zhao *et al.*, 2017] and DeepLabv3+ [Chen *et al.*, 2018] are selected as representative models. Following [Zhao *et al.*, 2017], we adopt the augmented annotations from [Hariharan *et al.*, 2011], resulting in 10,582 training, 1,449 validation, and 1,456 test images.

		CIFAR-10			**CIFAR-100**		
		Accuracy ↑	%SAM ↓	AIS ↑	Accuracy ↑	%SAM ↓	AIS ↑
ResNet-18	SGD	$96.23_{\pm0.11}$	\	3153(170%)	$79.84_{\pm0.45}$	\	3181(171%)
	SAM	$\mathbf{96.79_{\pm0.03}}$	100%	1855(100%)	$\mathbf{81.04_{\pm0.29}}$	100%	1866(100%)
	SAM-5	$96.42_{\pm0.18}$	20%	2741(148%)	$80.59_{\pm0.18}$	10%	2715(146%)
	LookSAM-5	$96.47_{\pm0.13}$	20%	2134(115%)	$80.48_{\pm0.24}$	20%	2115(113%)
	ESAM	$96.58_{\pm0.13}$	\	1936(104%)	$80.47_{\pm0.31}$	\	1953(105%)
	ESAM[1]	$96.56_{\pm0.08}$	\	2409(140%)	$80.41_{\pm0.10}$	\	2423(140%)
	SAF	$96.19_{\pm0.10}$	\	2733(147%)	$80.03_{\pm0.43}$	\	2736(147%)
	SAF[2]	$96.37_{\pm0.02}$	\	3213(194%)	$80.06_{\pm0.05}$	\	3248(192%)
	MESA[2]	$96.24_{\pm0.02}$	\	2780(168%)	$79.79_{\pm0.09}$	\	2793(165%)
	AE-SAM[3]	$\underline{96.63_{\pm0.04}}$	50.1%	\	$80.48_{\pm0.11}$	49.8%	\
	ARSAM	$\underline{96.63_{\pm0.09}}$	27.7%	2591(140%)	$\underline{80.92_{\pm0.01}}$	23.6%	2537(136%)
WideResNet-28-10	SGD	$96.91_{\pm0.06}$	\	680(195%)	$82.47_{\pm0.3}$	\	685(195%)
	SAM	$\mathbf{97.44_{\pm0.04}}$	100%	349(100%)	$\mathbf{84.71_{\pm0.21}}$	100%	351(100%)
	SAM-5	$97.16_{\pm0.03}$	20%	553(158%)	$83.06_{\pm0.08}$	20%	549(156%)
	LookSAM-5	$97.13_{\pm0.04}$	20%	537(154%)	$83.52_{\pm0.09}$	20%	552(157%)
	ESAM	$\underline{97.41_{\pm0.07}}$	\	502(144%)	$\underline{84.71_{\pm0.32}}$	\	494(141%)
	ESAM[1]	$97.29_{\pm0.11}$	\	550(139%)	$84.51_{\pm0.01}$	\	545(139%)
	SAF	$96.96_{\pm0.09}$	\	639(183%)	$82.57_{\pm0.18}$	\	639(182%)
	SAF[2]	$97.08_{\pm0.15}$	\	727(198%)	$83.81_{\pm0.04}$	\	729(197%)
	MESA[2]	$97.16_{\pm0.23}$	\	617(168%)	$83.59_{\pm0.24}$	\	625(169%)
	AE-SAM[3]	$97.30_{\pm0.10}$	49.5%	\	$84.51_{\pm0.11}$	49.6%	\
	ARSAM	$97.29_{\pm0.08}$	42.5%	468(134%)	$84.64_{\pm0.17}$	37.5%	470(134%)
PyramidNet-110	SGD	$97.14_{\pm0.08}$	\	548(195%)	$83.38_{\pm0.21}$	\	544(196%)
	SAM	$97.69_{\pm0.09}$	100%	281(100%)	$\mathbf{86.06_{\pm0.16}}$	100%	278(100%)
	SAM-5	$97.57_{\pm0.05}$	20%	447(159%)	$84.25_{\pm0.05}$	20%	457(164%)
	LookSAM-5	$97.22_{\pm0.05}$	20%	339(121%)	$83.76_{\pm0.45}$	20%	350(126%)
	ESAM	$97.59_{\pm0.17}$	\	322(115%)	$85.32_{\pm0.03}$	\	321(116%)
	ESAM[1]	$\underline{97.81_{\pm0.01}}$	\	401(139%)	$85.56_{\pm0.05}$	\	381(138%)
	SAF	$96.96_{\pm0.05}$	\	501(178%)	$83.66_{\pm0.34}$	\	502(181%)
	SAF[2]	$97.34_{\pm0.06}$	\	391(202%)	$84.71_{\pm0.01}$	\	397(200%)
	MESA[2]	$97.46_{\pm0.09}$	\	332(171%)	$84.73_{\pm0.14}$	\	339(171%)
	AE-SAM[3]	$\mathbf{97.90_{\pm0.05}}$	50.3%	\	$85.58_{\pm0.10}$	49.8%	\
	ARSAM	$97.49_{\pm0.08}$	37.2%	411(146%)	$\underline{85.67_{\pm0.06}}$	34.5%	395(142%)

1 We report the results in [Du *et al.*, 2022a].
2 We report the results in [Du *et al.*, 2022b].
3 We report the results in [Jiang *et al.*, 2023].

Table 3: The results of the proposed method and the comparison methods on CIFAR-10 and CIFAR-100 dataset. The numbers in parentheses (·) indicate the ratio of corresponding method's training speed to SAM's. ↑ means that the larger the reported results are better, and ↓ means that the smaller the results are better. The best accuracy is in bold and the second best is underlined.

	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Mean	Mean@0.1	AIS
Resnext-152	96.248	94.956	88.358	83.518	88.298	84.807	80.846	88.662	33.391	\
Rle+ResNet50	95.805	94.582	86.893	78.346	87.468	80.355	73.453	86.024	26.313	\
SimCC+HRNet+Adam	96.828	$\underline{95.771}$	$\underline{89.654}$	$\underline{84.272}$	88.679	85.151	$\underline{81.436}$	89.318	37.395	106(183%)
SimCC+HRNet+SAM	**96.965**	95.703	89.637	84.222	88.402	$\underline{85.513}$	**81.862**	89.388	$\underline{37.721}$	58(100%)
SimCC+HRNet+ARSAM	$\underline{96.555}$	**95.788**	**89.671**	**84.701**	**89.268**	**85.614**	81.577	**89.537**	**37.799**	70(121%)

Table 4: Results of training SimCC on MPII with Adam, SAM and ARSAM. The best accuracy is in bold and the second best is underlined.

	Noise rate	20%		40%		60%		80%	
	Methods	Acc.	AIS	Acc.	AIS	Acc.	AIS	Acc.	AIS
CIFAR-10	SGD	85.94	2673(143%)	70.25	2673(143%)	48.50	2575(139%)	28.91	2646(142%)
	SAM	90.44	1873(100%)	79.71	1876(100%)	67.22	1858(100%)	76.94	1863(100%)
	ARSAM	**92.27**	2298(123%)	**89.08**	2315(123%)	**77.67**	2283(123%)	**77.93**	2267(122%)
	Noise rate	20%		40%		60%		80%	
	Methods	Acc.	AIS	Acc.	AIS	Acc.	AIS	Acc.	AIS
CIFAR-100	SGD	65.42	2537(140%)	48.91	2535(137%)	31.46	2539(137%)	12.32	2533(138%)
	SAM	69.10	1815(100%)	55.06	1848(100%)	35.59	1849(100%)	9.82	1833(100%)
	ARSAM	**69.68**	2237(123%)	**55.55**	2288(124%)	**48.98**	2264(122%)	**19.18**	2227(122%)

Table 5: Test accuracy and AIS for ResNet-18 trained on CIFAR-10 and CIFAR-100 with different rate of noisy labels. The best accuracy is in bold.

Models	Methods	Acc. ↑	%SAM ↓	AIS ↑
ResNet-18	SGD	61.88±0.31	\	1456(199%)
	SAM	64.48±0.15	100%	732(100%)
	ARSAM	**64.71±0.10**	61.4%	892(122%)
MobileNet	SGD	58.12±0.40	\	547(189%)
	SAM	58.49±0.38	100%	289(100%)
	ARSAM	**59.29±0.28**	50.2%	704(122%)

Table 6: The results of SGD, SAM and ARSAM on Tiny-ImageNet. The best accuracy is in bold and the second best is underlined.

Methods	PSPNet		DeepLabv3+	
	Acc.	AIS	Acc.	AIS
SGD	78.89	37(195%)	77.12	152(190%)
SAM	**79.20**	19(100%)	77.25	80(100%)
ARSAM	79.10	25(132%)	**77.56**	105(131%)

Table 7: Results of semantic segmentation with SGD, SAM, and ARSAM on the VOC 2012. The best accuracy is in bold and the second best is underlined.

	ResNet-18		MobileNets	
Methods	Acc.	AIS	Acc.	AIS
Full prec.	88.72	\	85.81	\
LSQ+SGD	88.86	1515(174%)	84.04	1244(153%)
LSQ+SAM	**89.75**	870(100%)	84.72	816(100%)
LSQ+ARSAM	89.63	1386(159%)	**84.77**	1199(147%)

Table 8: Results of optimizing LSQ using ARSAM on the Cifar-10 dataset. The best accuracy is in bold and the second best is underlined.

We conducted experiments on the MPII dataset [Andriluka *et al.*, 2014], which contains 40k person samples with 16 joints labels. Following the evaluation procedure outlined in [Li *et al.*, 2022], we performed experiments at an image resolution of 256×256, with a batch size of 64. For comparison, we also report the results of ResNeXt-152 [Xie *et al.*, 2017] and RLE+ResNet-50 [Li *et al.*, 2021], sourced from the MMPose project [Contributors, 2020]. As shown in Table 4, ARSAM achieves performance comparable to SAM while significantly reducing training time. This demonstrates that ARSAM is simple, versatile, and effective. Moreover, ARSAM remains effective when using different base optimizers, such as Adam.

4.8 Application to Quantization-Aware Training

Neural quantization reduces the computational workload and memory requirements of deep neural networks by compressing the precision of weights and activation values into low-bit formats [Esser *et al.*, 2020; Wei *et al.*, 2021; Nagel *et al.*, 2022]. To demonstrate ARSAM's broader applicability, we use it as the optimizer for QAT [Jacob *et al.*, 2018].

We used SGD, SAM, and ARSAM to quantize the parameters of ResNet-18 and MobileNets to W4A4 on the CIFAR-10 dataset. The results, shown in Table 8, indicate that AR-SAM achieves about 50% faster training speeds compared to SAM while maintaining comparable performance. This demonstrates that ARSAM is versatile and satisfies the practical requirements of various applications.

Table 7 shows the semantic segmentation results on VOC 2012 using PSPNet and DeepLabv3+ with SGD, SAM, and ARSAM. On PSPNet, SAM achieves the highest accuracy of 79.20%, while ARSAM closely follows with 79.10%, outperforming SGD (78.89%). On DeepLabv3+, ARSAM achieves the best accuracy of 77.56%. ARSAM improves the optimization speed by approximately 32% on PSPNet and 31% on DeepLabv3+ compared to SAM, while still maintaining competitive accuracy.

4.7 Application to Human Pose Estimation

2D Human Pose Estimation (HPE) focuses on localizing body joints within a single image. To evaluate the generality of our methods, we apply SAM and ARSAM to human pose estimation. SimCC [Li *et al.*, 2022] is a SOTA method for HPE. The key idea of SimCC is to treat HPE as two classification tasks.

We employ SAM and ARSAM as optimization methods for the SimCC, using Adam as the base optimizer for both.

5 Conclusions

In this paper, we reveal that the gradient of SAM can be decomposed into the gradient of SGD and the PSF. We propose an adaptive sampling-reusing-mixing policy for the decomposed gradients, aimed at reducing the computational cost of the PSF and approximating the true PSF. Using this policy, we introduce ARSAM, an efficient optimization method that adaptively adjusts the gradient sampling rate of the PSF to improve optimization efficiency. It reuses the PSF in non-sampling iterations, thereby maintaining the model's generalization ability. We theoretically and empirically analyzed the convergence of ARSAM. Experimental results demonstrate that ARSAM achieves similar generalization performance to SAM while significantly improving optimization speed. Furthermore, we applied ARSAM to human pose estimation and model quantization tasks. The results confirm that ARSAM enhances optimization speed without compromising performance, showcasing its broad applicability.

In some cases, ARSAM achieves even higher accuracy than SAM. This indicates that the reuse of the PSF could reach the a flatter region than that of SAM. Investigating the underlying reasons for this phenomenon remains a future research direction. Besides, the autoregressive approach faces challenges in conveniently controlling the acceleration ratio of ARSAM relative to SAM. In the future, we aim to explore improved methods for modeling the relationship between the importance of the PSF and the sampling ratio to address this limitation.

References

[Andriluka *et al.*, 2014] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[Andriushchenko and Flammarion, 2022] Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 639–668. PMLR, 2022.

[Chaudhari *et al.*, 2019] 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.

[Chen *et al.*, 2018] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[Contributors, 2020] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020.

[Dai *et al.*, 2023] Rong Dai, Xun Yang, Yan Sun, Li Shen, Xinmei Tian, Meng Wang, and Yongdong Zhang. Fedgamma: Federated learning with global sharpness-aware minimization. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2023. doi:10.1109/TNNLS.2023.3304453.

[Deng *et al.*, 2024] Jiaxin Deng, Junbiao Pang, and Baochang Zhang. Asymptotic unbiased sample sampling to speed up sharpness-aware minimization. *ArXiv*, abs/2406.08001, 2024.

[DeVries and Taylor, 2017] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[Dinh *et al.*, 2017] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1019–1028. PMLR, 2017.

[Dong *et al.*, 2024] Mingrong Dong, Yixuan Yang, Kai Zeng, Qingwang Wang, and Tao Shen. Implicit sharpness-aware minimization for domain generalization. *Remote Sensing*, 16(16):2877, 2024.

[Du *et al.*, 2022a] Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent YF Tan. Efficient sharpness-aware minimization for improved training of neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[Du *et al.*, 2022b] Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:23439–23451, 2022.

[Esser *et al.*, 2020] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[Everingham *et al.*, 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[Foret *et al.*, 2021] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[Han *et al.*, 2017] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5927–5935, 2017.

[Hariharan *et al.*, 2011] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 international conference on computer vision*, pages 991–998. IEEE, 2011.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[Howard *et al.*, 2017] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Huang *et al.*, 2019] Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. O2u-net: A simple noisy label detection approach for deep neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, pages 3326–3334, 2019.

[Izmailov *et al.*, 2018] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 876–885, 2018.

[Jacob *et al.*, 2018] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018.

[Jiang *et al.*, 2019] 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.

[Jiang *et al.*, 2023] Weisen Jiang, Hansi Yang, Yu Zhang, and James Kwok. An adaptive policy to employ sharpness-aware minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[Keskar *et al.*, 2017] 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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[Kwon *et al.*, 2021] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5905–5914. PMLR, 2021.

[Le and Yang, 2015] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[Li *et al.*, 2018] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6391–6401, 2018.

[Li *et al.*, 2021] Jiefeng Li, Siyuan Bian, Ailing Zeng, Can Wang, Bo Pang, Wentao Liu, and Cewu Lu. Human pose regression with residual log-likelihood estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11025–11034, 2021.

[Li *et al.*, 2022] Yanjie Li, Sen Yang, Peidong Liu, Shoukui Zhang, Yunxiao Wang, Zhicheng Wang, Wankou Yang, and Shu-Tao Xia. Simcc: A simple coordinate classification perspective for human pose estimation. In *European Conference on Computer Vision*, pages 89–106. Springer, 2022.

[Liu *et al.*, 2020] Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:21476–21487, 2020.

[Liu *et al.*, 2022] Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12360–12370, 2022.

[Loshchilov and Hutter, 2016] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[Nagel *et al.*, 2022] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 16318–16330. PMLR, 2022.

[Sun *et al.*, 2021] Xu Sun, Zhiyuan Zhang, Xuancheng Ren, Ruixuan Luo, and Liangyou Li. Exploring the vulnerability of deep neural networks: A study of parameter corruption. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 11648–11656, 2021.

[Wei *et al.*, 2021] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[Xie *et al.*, 2017] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[Xie *et al.*, 2024] Tianci Xie, Tao Li, and Ruoxue Wu. Adaptive sharpness-aware minimization for adversarial domain generalization. *IEEE Transactions on Computational Social Systems*, 2024.

[Yue *et al.*, 2023] Xubo Yue, Maher Nouiehed, and Raed Al Kontar. Salr: Sharpness-aware learning rate scheduler for improved generalization. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–10, 2023. doi:10.1109/TNNLS.2023.3263393.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[Zhang *et al.*, 2023] Xingxuan Zhang, Renzhe Xu, Han Yu, Hao Zou, and Peng Cui. Gradient norm aware minimization seeks first-order flatness and improves generalization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20247–20257, 2023.

[Zhao *et al.*, 2017] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[Zhao *et al.*, 2022] Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 26982–26992. PMLR, 2022.

[Zhou *et al.*, 2023] Yixuan Zhou, Yi Qu, Xing Xu, and Hengtao Shen. Imbsam: A closer look at sharpness-aware minimization in class-imbalanced recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11345–11355, 2023.

[Zhuang *et al.*, 2022] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.