EFFICIENT SHARPNESS-AWARE MINIMIZATION FOR IMPROVED TRAINING OF NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Overparametrized Deep Neural Networks (DNNs) often achieve astounding performances, but may potentially result in severe generalization error. Recently, the relation between the sharpness of the loss landscape and the generalization error has been established by Foret et al. (2020), in which the Sharpness Aware Minimizer (SAM) was proposed to mitigate the degradation of the generalization. Unfortunately, SAM's computational cost is roughly double that of base optimizers, such as Stochastic Gradient Descent (SGD). This paper thus proposes *Efficient Sharpness Aware Minimizer* (ESAM), which boosts SAM's efficiency at no cost to its generalization performance. ESAM includes two novel and efficient training strategies—Stochastic Weight Perturbation and Sharpness-Sensitive Data Selection. In the former, the sharpness measure is approximated by perturbing a stochastically chosen set of weights in each iteration; in the latter, the SAM loss is optimized using only a judiciously selected subset of data that is sensitive to the sharpness. We provide theoretical explanations as to why these strategies perform well. We also show, via extensive experiments on the CIFAR and ImageNet datasets, that ESAM enhances the efficiency over SAM from requiring 100% extra computational overhead to 40% vis-à-vis base optimizers, while test accuracies are preserved or even improved.

1 INTRODUCTION

Deep learning has achieved astounding performances in many fields by relying on larger numbers of parameters and increasingly sophisticated optimization algorithms. However, DNNs with far more parameters than training samples are more prone to poor generalization. Generalization is arguably the most fundamental and yet mysterious aspect of deep learning.

Several studies have been conducted to better understand the generalization of DNNs and to train DNNs that generalize well across the natural distribution (Keskar et al., 2016; Neyshabur et al., 2017; Chaudhari et al., 2019; Zhang et al., 2019; Wu et al., 2020; Foret et al., 2020; Zhang et al., 2021). For example, Keskar et al. (2016) investigate the effect of batch size on neural networks' generalization ability. Zhang et al. (2019) propose an optimizer for training DNNs with improved generalization ability. Specifically, Hochreiter & Schmidhuber (1995), Li et al. (2017) and Dinh et al. (2017) argue that the geometry of the loss landscape affects generalization and DNNs with a flat minimum can generalize better.

The recent work by Foret et al. (2020) proposes an effective training algorithm *Sharpness Aware Minimizer (SAM)* for obtaining a flat minimum. SAM employs a base optimizer such as Stochastic Gradient Descent (Nesterov, 1983) or Adam (Kingma & Ba, 2014) to minimize both the vanilla training loss and the sharpness. The sharpness, which describes the flatness of a minimum, is characterized using eigenvalues of the Hessian matrix by Keskar et al. (2016). SAM quantifies the sharpness as the maximized change of training loss when a constraint perturbation is added to current weights. As a result, SAM leads to a flat minimum and significantly improves the generalization ability of the trained DNNs. SAM and its variants have been shown to outperform the state-of-the-art across a variety of deep learning benchmarks (Kwon et al., 2021; Chen et al., 2021; Galatolo et al., 2021; Zheng et al., 2021). Regrettably though, SAM and its variants achieve such remarkable performance at the expense of doubling the computational overhead of the given base optimizers, which minimize the training loss with a single forward and backward propagation step. SAM requires an

additional propagation step compared to the base optimizers to resolve the weight perturbation for quantifying the sharpness. The extra propagation step requires the same computational overhead as the single propagation step used by base optimizers, resulting in SAM's computational overhead being *doubled* (2×). As demonstrated in Figure 1, SAM achieves higher test accuracy (i.e., 84.46% vs. 81.89%) at the expense of sacrificing half of the training speed of the base optimizer (i.e., 276 imgs/s vs. 557 imgs/s).

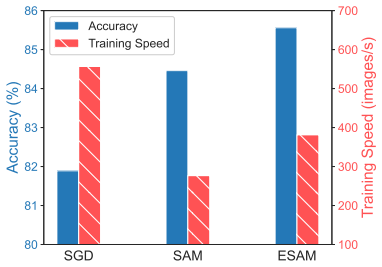

Figure 1: Training Speed vs. Accuracy of SGD, SAM and ESAM evaluated by PyramidNet on CIFAR100. ESAM improves the efficiency with better accuracy compared to SAM.

In this paper, we aim to improve the efficiency of SAM but preserve its superior performance in generalization. We propose *Efficient Sharpness Aware Minimizer (ESAM)*, which consists of two training strategies *Stochastic Weight Perturbation (SWP)* and *Sharpness-sensitive Data Selection (SDS)*, both of which reduce computational overhead and preserve the performance of SAM. On the one hand, SWP approximates the sharpness by searching weight perturbation within a stochastically chosen neighborhood of the current weights. SWP preserves the performance by ensuring that the expected weight perturbation is identical to that solved by SAM. On the other hand, SDS improves efficiency by approximately optimizing weights based on the sharpness-sensitive subsets of batches. These subsets consist of samples whose loss values increase most w.r.t. the weight perturbation and consequently can better quantify the sharpness of DNNs. As a result, the sharpness calculated over the subsets can serve as an upper bound of the SAM's sharpness, ensuring that SDS's performance is comparable to that of SAM's.

We verify the effectiveness of ESAM on the CIFAR10, CIFAR100 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009) datasets with five different DNN architectures. The experimental results demonstrate that ESAM obtains flat minima at a cost of only 40% (vs. SAM's 100%) extra computational overhead over base optimizers. More importantly, ESAM achieves better performance in terms of the test accuracy compared to SAM. In a nutshell, our contributions are as follows:

- We propose two novel and effective training strategies *Stochastic Weight Perturbation (SWP)* and *Sharpness-sensitive Data Selection (SDS)*. Both strategies are designed to improve efficiency without sacrificing performance. The empirical results demonstrate that both of the proposed strategies can improve both the efficiency and effectiveness of SAM.
- We introduce the ESAM, which integrates SWP and SDS. ESAM improves the generalization ability of DNNs with marginally additional computational cost compared to standard training.

The rest of this paper is structured in this way. Section 2.1 introduces SAM and its computational issues. Section 2.2 and Section 2.3 discuss how the two proposed training strategies SWP and SDS are designed respectively. Section 3 verifies the effectiveness of ESAM across a variety of datasets and DNN architectures. Section 4 presents the related work and Section 5 concludes this paper.

2 Methodology

We start with recapitulating how SAM achieves a flat minimum with small sharpness, which is quantified by resolving a maximization problem. To compute the sharpness, SAM requires additional forward and backward propagation and results in the doubling of the computational overhead compared to base optimizers. Following that, we demonstrate how we derive and propose ESAM, which integrates SWP and SDS, to maximize efficiency while maintaining the performance. We introduce SWP and SDS in Sections 2.2 and 2.3 respectively. Algorithm 1 shows the overall proposed ESAM algorithm.

Throughout this paper, we denote a neural network f with weight parameters θ as f_θ. The weights are contained in the vector $\theta = (\theta_1, \theta_2, \dots, \theta_N)$, where N is the number of weight units in the neural network. Given a training dataset $\mathbb{S}$ that contains samples i.i.d. drawn from a distribution $\mathcal{D}$,

Algorithm 1 Efficient SAM (ESAM)

Input: Network f_θ, $\theta = (\theta_1, \theta_2, \ldots, \theta_N)$, Training set $\mathbb{S}$, Batch size b, Learning rate $\eta > 0$, Neighborhood size $\rho > 0$, Iterations A, SWP hyperparameter β, SDS hyperparameter γ.
Output: A flat minimum solution $\hat{\theta}$.

```
for a = 1 to A do
    Sample a mini-batch B ⊂ S with size b.
    for n = 1 to N do
        if θ_n is chosen by probability β then
            ε_n ← (ρ/(1−β)) ∇_{θ_n} L_B(f_θ)                    ▷ SWP in B_1
        else
            ε_n ← 0
        ε̂ ← (ε_1, ..., ε_N)                                     ▷ Assign Weight Perturbation
        Compute ℓ(f_{θ+ε̂}, x_i, y_i) and construct B^+ with selection ratio γ (Equation 6)
        Compute gradients g = ∇_θ L_{B^+}(f_{θ+ε̂})              ▷ SDS in B_2
        Update weights θ ← θ − ηg
```

the network is trained to obtain optimal weights $\hat{\theta}$ via *empirical risk minimization* (ERM), i.e.,

$$\hat{\theta} = \arg\min_{\theta} \left\{ L_{\mathbb{S}}(f_\theta) = \frac{1}{|\mathbb{S}|} \sum_{(x_i, y_i) \in \mathbb{S}} \ell(f_\theta, x_i, y_i) \right\} \tag{1}$$

where ℓ can be an arbitrary loss function. We take ℓ to be the cross entropy loss in this paper. The *population loss* is defined as $L_{\mathcal{D}}(f_\theta) \triangleq \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}}(\ell(f_\theta, x_i, y_i))$. In each training iteration, optimizers sample a mini-batch $\mathbb{B} \subset \mathbb{S}$ with size b to update parameters.

2.1 Sharpness-aware minimization and its computational drawback

To improve the generalization capability of DNNs, Foret et al. (2020) proposed the SAM training strategy for searching flat minima. SAM trains DNNs by solving the following min-max optimization problem,

$$\min_{\theta} \max_{\epsilon : \|\epsilon\|_2 \leq \rho} L_{\mathbb{S}}(f_{\theta+\epsilon}). \tag{2}$$

Given θ, the inner optimization attempts to find a weight perturbation ϵ in Euclidean ball with radius ρ that maximizes the empirical loss. The maximized loss at weights θ is the sum of the empirical loss and the sharpness, which is defined to be $R_{\mathbb{S}}(f_\theta) = \max_{\epsilon : \|\epsilon\|_2 < \rho} [L_{\mathbb{S}}(f_{\theta+\epsilon}) - L_{\mathbb{S}}(f_\theta)]$. This sharpness is quantified by the maximal change of empirical loss when a constraint perturbation ϵ is added to θ. The min-max problem encourages SAM to find flat minima.

For a certain set of weights θ, Foret et al. (2020) theoretically justifies that the population loss of DNNs can be upper-bounded by the sum of sharpness, empirical loss, and a regularization term on the norm of weights (refer to Equation 3). Thus, by minimizing the sharpness together with the empirical loss, SAM produces optimized solutions for DNNs with flat minima, and the resultant models can thus generalize better (Foret et al., 2020; Chen et al., 2021; Kwon et al., 2021).

$$L_{\mathcal{D}}(f_\theta) \leq R_{\mathbb{S}}(f_\theta) + L_{\mathbb{S}}(f_\theta) + \lambda \|\theta\|_2^2 = \max_{\epsilon : \|\epsilon\|_2 \leq \rho} L_{\mathbb{S}}(f_{\theta+\epsilon}) + \lambda \|\theta\|_2^2 \tag{3}$$

In practice, SAM first approximately solves the inner optimization by means of a single-step gradient descent method, i.e.,

$$\hat{\epsilon} = \arg\max_{\epsilon : \|\epsilon\|_2 < \rho} L_{\mathbb{S}}(f_{\theta+\epsilon}) \approx \rho \nabla_\theta L_{\mathbb{S}}(f_\theta). \tag{4}$$

The sharpness at weights θ is approximated by $R_{\mathbb{S}}(f_\theta) = L_{\mathbb{S}}(f_{\theta+\hat{\epsilon}}) - L_{\mathbb{S}}(f_\theta)$. Then, a base optimizer, such as SGD (Nesterov, 1983) or Adam (Kingma & Ba, 2014), updates the DNNs' weights to minimize $L_{\mathbb{S}}(f_{\theta+\hat{\epsilon}})$. We refer to $L_{\mathbb{S}}(f_{\theta+\hat{\epsilon}})$ as the *SAM loss*. Overall, SAM requires two forward and two backward operations to update weights once. We refer to the forward and backward propagation for approximating $\hat{\epsilon}$ as F_1 and B_1 and those for updating weights by base optimizers as F_2 and B_2

respectively. Although SAM can effectively improve the generalization of DNNs, it additionally requires one forward and one backward operation (F_1 and B_1) in each training iteration. Thus, SAM results in a *doubling* of the computational overhead compared to the use of base optimizers.

To improve the efficiency of SAM, we propose ESAM, which consists of two strategies—SWP and SDS, to accelerate the sharpness approximation phase and the weight updating phase. Specifically, on the one hand, when estimating $\hat{\epsilon}$ around weight vector θ, SWP efficiently approximates $\hat{\epsilon}$ by randomly selecting each parameter with a given probability to form a subset of weights to be perturbed. The reduction of the number of perturbed parameters results in lower computational overhead during the backward propagation. SWP rescales the resultant weight perturbation so as to assure that the expected weight perturbation equals to $\hat{\epsilon}$, and the generalization capability thus will not be significantly degraded. On the other hand, when updating weights via base optimizers, instead of computing the upper bound $L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}})$ over a whole batch of samples, SDS selects a *subset* of samples, $\mathbb{B}^+$, whose loss values increase the most with respect to the perturbation $\hat{\epsilon}$. Optimizing the weights based on a fewer number of samples decreases the computational overhead (in a linear fashion). We further justify that $L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}})$ can be upper bounded by $L_{\mathbb{B}^+}(f_{\theta+\hat{\epsilon}})$ and consequently the generalization capability can be preserved. In general, ESAM works much more efficiently and performs as well as SAM in terms of its generalization capability.

2.2 Stochastic Weight Perturbation

This section elaborates on the first efficiency enhancement strategy, SWP, and explains why SWP can effectively reduce computational overhead while preserving the generalization capability.

To efficiently approximate $\hat{\epsilon}(\theta, \mathbb{S})$ during the sharpness estimation phase, SWP randomly chooses a subset $\tilde{\theta} = \{\theta_{I_1}, \theta_{I_2}, \ldots\}$ from the original set of weights $\theta = (\theta_1, \ldots, \theta_N)$ to perform backpropagation B_1. Each parameter is selected to be in the subvector $\tilde{\theta}$ with some probability β, which can be tuned as a hyperparameter. SWP approximates the weight perturbation with $\rho \nabla_{\tilde{\theta}} L_{\mathbb{S}}(f_\theta)$. To be formal, we introduce a *gradient mask* $\mathbf{m} = (m_1, \ldots, m_N)$ where $m_i \overset{\text{i.i.d.}}{\sim} \text{Bern}(\beta)$ for all $i \in \{1, \ldots, N\}$. Then, we have $\rho \nabla_{\tilde{\theta}} L_{\mathbb{S}}(f_\theta) = \mathbf{m}^\top \hat{\epsilon}(\theta, \mathbb{B})$. To ensure the expected weight perturbation of SWP equals to $\hat{\epsilon}$, we scale $\rho \nabla_{\tilde{\theta}} L_{\mathbb{S}}(f_\theta)$ by a factor of $\frac{1}{\beta}$. Finally, SWP produces an approximate solution of the inner maximization as

$$\boldsymbol{a}(\theta, \mathbb{B}) = \frac{\mathbf{m}^\top \hat{\epsilon}(\theta, \mathbb{B})}{\beta}. \tag{5}$$

Computation Ideally, SWP reduces the overall computational overhead in proportion to $1 - \beta$ in B_1. However, there exists some parameters not included in $\tilde{\theta}$ that are still required to be updated in the backpropagation step. This additional computational overhead is present due to the use of the chain rule, which calculates the entire set of gradients with respect to the parameters along a propagation path. This additional computational overhead slightly increases in deeper neural networks. Thus, the amount of reduction in the computational overhead is positively correlated to $1 - \beta$. In practice, β is tuned to maximize SWP's efficiency while maintaining a generalization performance comparable to SAM's.

Generalization We will next argue that SWP's generalization performance can be preserved when compared to SAM by showing that the expected weight perturbation $\boldsymbol{a}(\theta, \mathbb{B})$ of SWP equals to the original SAM's perturbation $\hat{\epsilon}(\theta, \mathbb{B})$ in the sense of the ℓ_2 norm and direction. We denote the expected SWP perturbation by $\bar{\boldsymbol{a}}(\theta, \mathbb{B})$, where

$$\bar{\boldsymbol{a}}(\theta, \mathbb{B})_{[i]} = \mathbb{E}[\boldsymbol{a}(\theta, \mathbb{B})_{[i]}] = \frac{1}{\beta} \cdot \beta \hat{\epsilon}(\theta, \mathbb{B})_{[i]} = \hat{\epsilon}(\theta, \mathbb{B})_{[i]},$$

for $i \in \{1, \ldots, N\}$. Thus, it holds that

$$\|\bar{\boldsymbol{a}}(\theta, \mathbb{B})\|_2 = \|\hat{\epsilon}(\theta, \mathbb{B})\|_2 \quad \text{and} \quad \text{CosSim}\Big(\bar{\boldsymbol{a}}(\theta, \mathbb{B}), \hat{\epsilon}(\theta, \mathbb{B})\Big) = 1,$$

showing that the expected weight perturbation of SWP is the same as that of SAM's.

2.3 SHARPNESS-SENSITIVE DATA SELECTION

In this section, we introduce the second efficiency enhancement technique, SDS, which reduces computational overhead of SAM linearly as the number of selected samples decreases. We also explain why the generalization capability of SAM is preserved by SDS.

In the sharpness estimation phase, we obtain the approximate solution $\hat{\epsilon}$ of the inner maximization. Perturbing weights along this direction significantly increases the average loss over a batch $\mathbb{B}$. To improve the efficiency but still control the upper bound $L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}})$, we select a subset of samples from the whole batch. The loss values of this subset of samples increase most when the weights are perturbed by $\hat{\epsilon}$. To be specific, SDS splits the mini-batch $\mathbb{B}$ into the following two subsets

$$\begin{aligned} \mathbb{B}^{+} &:= \{(x_i, y_i) \in \mathbb{B} : \ell(f_{\theta+\hat{\epsilon}}, x_i, y_i) - \ell(f_{\theta}, x_i, y_i) > \alpha\}, \\ \mathbb{B}^{-} &:= \{(x_i, y_i) \in \mathbb{B} : \ell(f_{\theta+\hat{\epsilon}}, x_i, y_i) - \ell(f_{\theta}, x_i, y_i) < \alpha\}, \end{aligned} \tag{6}$$

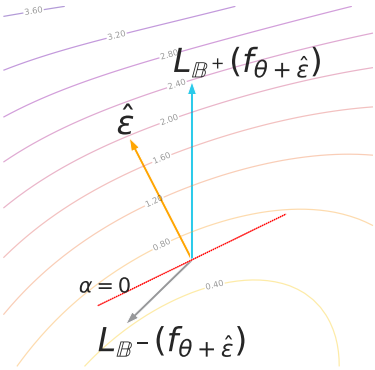

Figure 2: Illustration on the loss changes of samples in $\mathbb{B}^+$ and $\mathbb{B}^-$ along the weight perturbation $\hat{\epsilon}$. The average loss of samples in $\mathbb{B}^+$ increases the most along the perturbation $\hat{\epsilon}$.

where $\mathbb{B}^+$ is termed as the sharpness-sensitive subset and the threshold α controls the size of $\mathbb{B}^+$. We let $\gamma = |\mathbb{B}^+|/|\mathbb{B}|$ be the ratio of the number of selected samples with respect to the batch size. In practice, γ determines the exact value of α and serves as a predefined hyperparameter of SDS. As illustrated in Figure 2, when $\alpha = 0$, the gradient of the weights evaluated on $\mathbb{B}^+$ *aligns* with the direction of $\hat{\epsilon}$ and the loss values of the samples in $\mathbb{B}^+$ will increase with respect to the weight perturbation $\hat{\epsilon}$.

Computation SDS reduces the computational overhead in F_2 and B_2. The reduction is linear in $1 - \gamma$. The hyperparameter γ can be tuned to meet up distinct requirements in efficiency and performance. SDS is configured the same as SWP for maximizing efficiency with comparable performance to SAM.

Generalization For the generalization capability, we now justify that the SAM loss computing over the batch $\mathbb{B}$, $L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}})$, can be approximately upper bounded by the corresponding loss evaluated only on $\mathbb{B}^+$, $L_{\mathbb{B}^+}(f_{\theta+\hat{\epsilon}})$. From Equation 3, we have

$$\begin{aligned} L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}}) &= \gamma L_{\mathbb{B}^+}(f_{\theta+\hat{\epsilon}}) + (1-\gamma) L_{\mathbb{B}^-}(f_{\theta+\hat{\epsilon}}) \\ &= L_{\mathbb{B}^+}(f_{\theta+\hat{\epsilon}}) + (1-\gamma)[R_{\mathbb{B}^-}(f_\theta) - R_{\mathbb{B}^+}(f_\theta) + L_{\mathbb{B}^-}(f_\theta) - L_{\mathbb{B}^+}(f_\theta)]. \end{aligned} \tag{7}$$

On the one hand, since $R_{\mathbb{B}}(f_\theta) = \frac{1}{|\mathbb{B}|}\sum_{(x_i,y_i)\in\mathbb{B}}[\ell(f_{\theta+\hat{\epsilon}}, x_i, y_i) - \ell(f_\theta, x_i, y_i)]$ represents the average sharpness of the batch $\mathbb{B}$, by Equation 6, we have $R_{\mathbb{B}^-}(f_\theta) \leq R_{\mathbb{B}}(f_\theta) \leq R_{\mathbb{B}^+}(f_\theta)$, and

$$R_{\mathbb{B}^-}(f_\theta) - R_{\mathbb{B}^+}(f_\theta) \leq 0. \tag{8}$$

On the other hand, $\mathbb{B}^+$ and $\mathbb{B}^-$ are constructed sorting by $\ell(f_{\theta+\hat{\epsilon}}, x_i, y_i) - \ell(f_\theta, x_i, y_i)$, which is positively correlated to $l(f_\theta, x_i, y_i)$ (Li et al., 2019) (more details can be found in the Appendix A.2). Thus, we have

$$L_{\mathbb{B}^-}(f_\theta) - L_{\mathbb{B}^+}(f_\theta) \leq 0. \tag{9}$$

Therefore, by Equation 8 and Equation 9, we have

$$L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}}) \leq L_{\mathbb{B}^+}(f_{\theta+\hat{\epsilon}}). \tag{10}$$

Experimental results in Figure 5 corroborate that $R_{\mathbb{B}^-}(f_\theta) - R_{\mathbb{B}^+}(f_\theta) < 0$ and $L_{\mathbb{B}^-}(f_\theta) - L_{\mathbb{B}^+}(f_\theta) < 0$. Besides, Figure 6 verifies that the selected batch $\mathbb{B}^+$ is sufficiently representative to mimic the gradients of $\mathbb{B}$ since $\mathbb{B}^+$ has a significantly higher cosine similarity with $\mathbb{B}$ compared to $\mathbb{B}^-$ in terms of the computed gradients. According to Equation 10, one can utilize $L_{\mathbb{B}^+}(f_{\theta+\hat{\epsilon}})$ as a proxy to the real objective to minimize of the overall loss $L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}})$ with a smaller number of samples. As a result, SDS improves SAM's efficiency without performance degradation.

3 EXPERIMENTS

This section demonstrates the effectiveness of our proposed ESAM algorithm. We conduct experiments on several benchmark datasets: CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009), using various model architectures: ResNet (He et al., 2016), Wide ResNet (Zagoruyko & Komodakis, 2016), and PyramidNet (Han et al., 2017). We demonstrate the proposed ESAM improves the efficiency of vanilla SAM by speeding up to 40.3% computational overhead with better generalization performance. We report the main results in Table 1 and Table 2. Besides, we perform an ablation study on the two proposed strategies of ESAM (i.e., SWP and SDS). The experimental results in Table 3 and Figure 3 indicate that both strategies improve SAM's efficiency and performance.

3.1 RESULTS

CIFAR10 and CIFAR100 We start from evaluating ESAM on the CIFAR-10 and CIFAR-100 image classification datasets. The evaluation is carried out on three different model architectures: ResNet-18 (He et al., 2016), WideResNet-28-10 (Zagoruyko & Komodakis, 2016) and PyramidNet-110 (Han et al., 2017). We set all the training settings, including the maximum number of training epochs, iterations per epoch, and data augmentations, the same for fair comparison among SGD, SAM and ESAM. Additionally, the other hyperparameters of SGD, SAM and ESAM have been tuned separately for optimal test accuracies using grid search.

We train all the models with 3 different random seeds using a batch size of 128, weight decay 10^{-4} and cosine learning rate decay (Loshchilov & Hutter, 2016). The training epochs are set to be 200 for ResNet-18 (He et al., 2016), WideResNet-28-10 (Zagoruyko & Komodakis, 2016), and 300 for PyramidNet-110 (Han et al., 2017). We set $\beta = 0.6$ and $\gamma = 0.5$ for ResNet-18 and PyramidNet-110 models; and set $\beta = 0.5$ and $\gamma = 0.5$ for WideResNet-28-10. The above-mentioned β and γ are optimal for efficiency with comparable performance compared to SAM. The details of training setting are listed in Appendix A.7. We record the best test accuracies obtained by SGD, SAM and ESAM in Table 1.

The experimental results indicate that our proposed ESAM can increase the training speed by up to 40.30% in comparison with SAM. Concerning the performance, ESAM outperforms SAM in the six sets of experiments. The best efficiency of ESAM is reported in CIFAR10 trained with ResNet-18 (Training speed 140.3% vs. SAM 100%). The best accuracy is reported in CIFAR100 trained with PyramidNet110 (Accuracy 85.56% vs. SAM 84.46%). ESAM improves efficiency and achieves better performance compared to SAM in CIFAR10/100 benchmarks.

Table 1: Classification accuracies and training speed on the CIFAR-10 and CIFAR-100 datasets. Computational overhead is quantified by #images processed per second (images/s). The numbers in parentheses $(\cdot)$ indicate the ratio of ESAM's training speed w.r.t. SAM.

	CIFAR-10		**CIFAR-100**	
ResNet-18	Accuracy	images/s	Accuracy	images/s
SGD	95.41 ± 0.03	3,387	78.17 ± 0.05	3,483
SAM	96.52 ± 0.13	1,717(100.0%)	80.17 ± 0.17	1,730 (100.0%)
ESAM	**96.56** ± 0.08	2,409 (140.3%)	**80.41** ± 0.10	2,423 (140.0%)
Wide-28-10	Accuracy	images/s	Accuracy	images/s
SGD	96.34 ± 0.12	801	81.56 ± 0.13	792
SAM	97.27 ± 0.11	396 (100.0%)	83.42 ± 0.04	391 (100.0%)
ESAM	**97.29** ± 0.11	550 (138.9%)	**84.51** ± 0.01	545 (139.4%)
PyramidNet-110	Accuracy	images/s	Accuracy	images/s
SGD	96.62 ± 0.10	580	81.89 ± 0.17	555
SAM	97.30 ± 0.10	289 (100.0%)	84.46 ± 0.04	276 (100.0%)
ESAM	**97.81** ± 0.01	401 (138.7%)	**85.56** ± 0.05	381 (137.9%)

ImageNet To evaluate ESAM's effectiveness on a large-scale benchmark dataset, we conduct experiments on ImageNet Datasets.The 1000 class ImageNet dataset contains roughly 1.28 million

Table 2: Classification accuracies and training speed on the ImageNet dataset. The numbers in parentheses (·) indicate the ratio of ESAM's training speed w.r.t. SAM's. Results with * are referred to Chen et al. (2021)

	ResNet-50		**ResNet-101**	
ImageNet	Accuracy	images/s	Accuracy	images/s
SGD	76.00*	1,327	77.80*	891
SAM	76.70*	654 (100.0%)	78.60*	438 (100.0%)
ESAM	77.05	846 (129.3%)	79.09	564 (128.7%)

training images and $50,000$ validation images with 469×387 averaged resolution. The ImageNet dataset is more representative (of real-world scenarios) and persuasive (of a method's effectiveness) than CIFAR datasets. We resize the images on ImageNet to 224×224 resolution to train ResNet-50 and ResNet-110 models. We train 90 epochs and set the optimal hyperparameters for SGD, SAM and ESAM as suggested by Chen et al. (2021), and the details are listed in appendix A. We use $\beta = 0.6$ and $\gamma = 0.7$ for ResNet-50 and ResNet-101. We employ the m-sharpness strategy for both SAM and ESAM with $m = 128$, which is the same as that suggested in Zheng et al. (2021).

The experimental results are reported in Table 2. The results indicate that the performance of ESAM on large-scale datasets is consistent with the two (smaller) CIFAR datasets. ESAM outperforms SAM by 0.35% to 0.49% in accuracy and, more importantly, enjoys 28.7% faster training speed compared to SAM. As the γ we used here is larger than the one used in CIFAR datasets, the training speed of ESAM here is slightly slower than that in the CIFAR datasets.

These experiments demonstrate that ESAM outperforms SAM on a variety of benchmark datasets for widely-used DNNs' architectures in terms of training speed and classification accuracies.

3.2 Ablation and Parameter Studies

To better understand the effectiveness of SWP and SDS in improving the performance and efficiency compared to SAM, we conduct four sets of ablation studies on CIFAR-10 and CIFAR-100 datasets using ResNet-18 and WideResNet-28-10 models, respectively. We consider two variants of ESAM: (i) only with SWP, (ii) only with SDS. The rest of the experimental settings are identical to the settings described in Section 3.1. We conduct grid search over the interval $[0.3, 0.9]$ for β and the interval $[0.3, 0.9]$ for γ, with a same step size of 0.1. We report the grid search results in Figure 3. We use $\beta = 0.6$,$\gamma = 0.5$ for ResNet-18; and set $\beta = 0.5$, $\gamma = 0.5$ for WideResNet-28-10 in the four sets of ablation studies. The ablation study results are reported in Table 3.

Table 3: Ablation Study of ESAM on CIFAR-10 and CIFAR100. The numbers in brackets [·] represent the accuracy improvement in comparison to SGD. The numbers in parentheses (·) indicate the ratio of ESAM's training speed to SAM's. Green color indicates improvement compared to SAM, whereas red color suggests a degradation.

	CIFAR-10		**CIFAR-100**	
ResNet-18	Accuracy	images/s	Accuracy	images/s
SGD	95.41	3,387	78.17	3,438
SAM	96.52 [+1.11]	1,717 (100.0%)	80.17 [+2.00]	1,730 (100.0%)
+ ESAM-SWP	**96.74** [+1.33]	1,896 (110.5%)	**80.53** [+2.36]	1,887 (109.1%)
+ ESAM-SDS	96.45 [+1.04]	2,105 (122.6%)	80.38 [+2.21]	2,103 (121.5%)
ESAM	96.56 [+1.15]	**2,409** (140.3%)	80.41 [+2.24]	**2,423** (140.9%)
Wide-28-10	Accuracy	images/s	Accuracy	images/s
SGD	96.34	801	81.56	792
SAM	97.27 [+0.93]	396 (100.0%)	83.42 [+1.86]	391 (100.0%)
+ ESAM-SWP	**97.37** [+1.03]	430 (108.5%)	84.44 [+2.88]	423 (108.3%)
+ ESAM-SDS	97.24 [+0.90]	495 (124.8%)	84.46 [+2.90]	492 (125.8%)
ESAM	97.29 [+0.95]	**551** (138.9%)	**84.51** [+2.95]	**545** (139.4%)

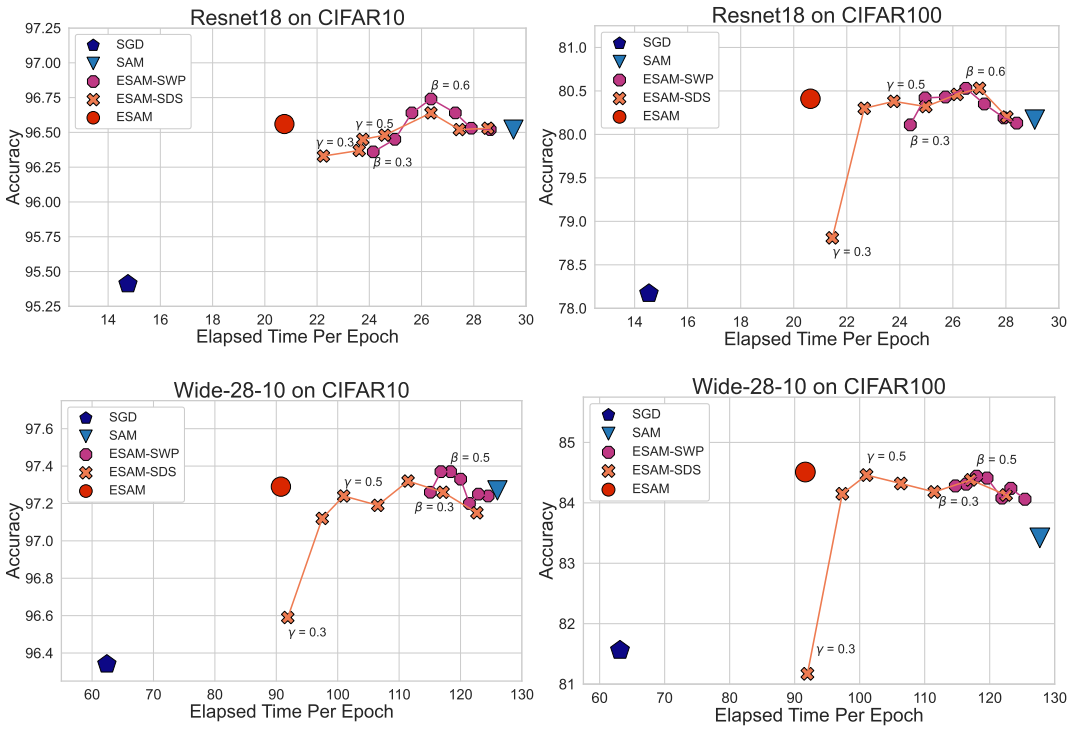

Figure 3: Parameter study of SWP and SDS. The connected dots refer to SWP and SDS with different parameters; the isolated dots refer to the final results of SGD, SAM, and ESAM.

ESAM-SWP As shown in Table 3, SWP improves SAM's training speed by 8.3% to 10.5%, and achieves better performance at the same time. SWP can further improve the efficiency by using a smaller β. The best performance of SWP is obtained when $\beta = 0.6$ for ResNet-18 and $\beta = 0.5$ for WideResNet-28-10. The four sets of experiments indicate that β is consistent among different architectures and datasets. Therefore, we set $\beta = 0.6$ for PyramidNet in CIFAR10/100 datasets and ResNet on ImageNet datasets.

ESAM-SDS SDS also significantly improves the efficiency by 21.5% to 25.8% compared to SAM. It outperforms SAM's performance in CIFAR100 datasets, and achieves comparable performance in CIFAR10 datasets. SDS can outperform SAM in both datasets with both architectures with little degradation to the efficiency, as demonstrated in Figure 3. Across all experiments, $\gamma = 0.5$ is the smallest value that is optimal for efficiency while maintaining comparable performance to SAM.

Visualization of Loss Landscapes To visualize the sharpness of the flat minima obtained by ESAM, we plot the loss landscapes trained with SGD, SAM and ESAM on the ImageNet dataset. We display the loss landscapes in Figure 4, following the plotting algorithm in Li et al. (2017). The x- and y-axes represent two random sampled orthogonal Gaussian perturbations. We sampled 100×100 points for 10 groups random Gaussian perturbations. The displayed loss landscapes are the results we obtained by averaging over ten groups of random perturbations. It can be clearly seen that both SAM and ESAM improve the sharpness significantly in comparison to SGD.

To summarize, SWP and SDS both reduce the computational overhead and accelerate training compared to SAM. Most importantly, both these strategies achieve a comparable or better performance than SAM. In practice, by configuring the β and γ, ESAM can meet a variety of user-defined efficiency and performance requirements.

4 Related Work

The concept of regularizing sharpness for better generalization dates back to (Hochreiter & Schmidhuber, 1995). By using an MDL-based argument, which clarifies that a statistical model with fewer

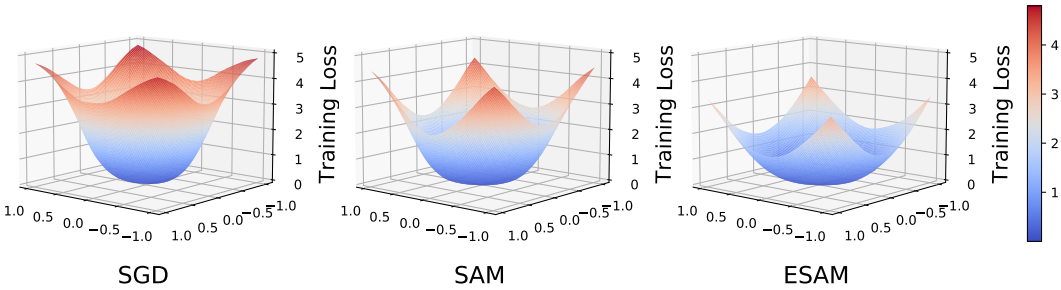

Figure 4: Cross-entropy loss landscapes of the ResNet50 model on the ImageNet dataset trained with SGD, SAM, and ESAM.

bits to describe can have better generalization ability, Hochreiter & Schmidhuber (1995) claim that a flat minimum can alleviate overfitting issues. Following that, more studies were proposed to investigate the connection between the flat minima with the generalization abilities (Keskar et al., 2016; Dinh et al., 2017; Liu et al., 2020; Li et al., 2017; Dziugaite & Roy, 2017; Jiang et al., 2019; Moosavi-Dezfooli et al., 2019). Keskar et al. (2016) starts by investigating the phenomenon that training with a larger batch size results in worse generalization ability. The authors found that the sharpness of the minimum is critical in accounting for the observed phenomenon. Keskar et al. (2016) and Dinh et al. (2017) both argue that the sharpness can be characterized using the eigenvalues of the Hessian. Although they also define specific notions and methods to quantify sharpness, they do not propose complete training strategies to find minima that are relative "flat".

SAM (Foret et al., 2020) leverages the connection between "flat" minima and the generalization error to train DNNs that generalize well across the natural distribution. Inspired by Keskar et al. (2016) and Dinh et al. (2017), SAM first proposes the quantification of the sharpness, which is achieved by solving a maximization problem. Then, SAM proposes a complete training algorithm to improve the generalization abilities of DNNs. SAM is demonstrated to achieve state-of-the-art performance in a variety of deep learning benchmarks, including image classification, natural language processing, and noisy learning (Foret et al., 2020; Chen et al., 2021; Kwon et al., 2021; Pham et al., 2021; Yuan et al., 2021; Jia et al., 2021).

A series of SAM-related works has been proposed. A work that was done contemporaneously SAM (Wu et al., 2020) also regularizes the sharpness term in adversarial training and achieves much more robust generalization performance against adversarial attacks. Many works focus on combining SAM with other training strategies or architectures (Chen et al., 2021; Wang et al., 2021; Tseng et al., 2021), or apply SAM on other tasks (Zheng et al., 2021; Damian et al., 2021; Galatolo et al., 2021). Kwon et al. (2021) improves SAM's sharpness by adaptively scaling the size of the nearby search space ρ in relation to the size of parameters. However, most of these works overlook the fact that SAM improves generalization at the expense of the doubling the computational overhead. As a result, most of the SAM-related works suffer from the same efficiency drawback as SAM. This computational cost prevents SAM from being widely used in large-scale datasets and architectures, particularly in real-world applications, which motivates us to propose ESAM to *efficiently* improve the generalization ability of DNNs.

5 Conclusion

In this paper, we propose the *Efficient Sharpness Aware Minimizer* (ESAM) to enhance the efficiency of vanilla SAM. The proposed ESAM integrates two novel training strategies, namely, SWP and SDS, both of which are derived based on theoretical underpinnings and are evaluated over a variety of datasets and DNN architectures. Both SAM and ESAM are two-step training strategies consisting of sharpness estimation and weight updating. In each step, gradient back-propagation is performed to compute the weight perturbation or updating. In future research, we will explore how to combine the two steps into one by utilizing the information of gradients in previous iterations so that the computational overhead of ESAM can be reduced to the same as base optimizers.

Reproducibility Statement To ensure the reproducibility of our proposed ESAM formulation and algorithm, we will submit our source codes implemented in the PyTorch framework (Paszke et al., 2017) together with the supplementary material. The source codes contains the ESAM training algorithm codes, documents on how to apply ESAM, and examples of the application of ESAM on the CIFAR benchmark datasets. We will also release our ESAM codes that embedded into TIMM (Wightman, 2019) framework to the public.

REFERENCES

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.

Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pretraining or strong data augmentations. *arXiv preprint arXiv:2106.01548*, 2021.

Alex Damian, Tengyu Ma, and Jason Lee. Label noise sgd provably prefers flat global minimizers. *arXiv preprint arXiv:2106.06530*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Alessio Galatolo, Alfred Nilsson, Roderick Karlemstrand, and Yineng Wang. Using early-learning regularization to classify real-world noisy data. *arXiv preprint arXiv:2105.13244*, 2021.

Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5927–5935, 2017.

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

Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pp. 529–536, 1995.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021.

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.

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. *arXiv preprint arXiv:1609.04836*, 2016.

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

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6(1):1, 2009.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. *arXiv preprint arXiv:2102.11600*, 2021.

Buyu Li, Yu Liu, and Xiaogang Wang. Gradient harmonized single-stage detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8577–8584, 2019.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

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. *arXiv preprint arXiv:2006.08403*, 2020.

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

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9078–9086, 2019.

Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. *arXiv preprint arXiv:1706.08947*, 2017.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11557–11568, 2021.

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. *arXiv preprint arXiv:1907.02610*, 2019.

Ching-Hsun Tseng, Shin-Jye Lee, Jia-Nan Feng, Shengzhong Mao, Yu-Ping Wu, Jia-Yu Shang, Mou-Chung Tseng, and Xiao-Jun Zeng. Upanets: Learning from the universal pixel attention networks. *arXiv preprint arXiv:2103.08640*, 2021.

Pichao Wang, Xue Wang, Hao Luo, Jingkai Zhou, Zhipeng Zhou, Fan Wang, Hao Li, and Rong Jin. Scaled relu matters for training vision transformers. *arXiv preprint arXiv:2109.03810*, 2021.

Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *arXiv preprint arXiv:2004.05884*, 2020.

Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng, and Shuicheng Yan. Volo: Vision outlooker for visual recognition. *arXiv preprint arXiv:2106.13112*, 2021.

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

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Michael R Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. *arXiv preprint arXiv:1907.08610*, 2019.

Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8156–8165, 2021.

A APPENDIX

A.1 THE ALGORITHM OF SAM

The algorithm of SAM is demonstrated in Algorithm 2.

Algorithm 2 SGD vs. SAM

Input: Network f_θ, $\theta = (\theta_1, \theta_1, \ldots, \theta_N)$, Training set $\mathbb{S}$, Batch size b, Learning rate η, Neighborhood size ρ, Iterations A.

Output: A minimum solution $\tilde{\theta}$.

for $a = 1$ to A **do**
  Sample $\mathbb{B}$ with size b that $\mathbb{B} \subset \mathbb{S}$,
  if SGD **then**
    $\hat{\epsilon} \leftarrow 0$ ▷ No addtional computational overhead for $\hat{\epsilon}$
  else if SAM **then**
    $\hat{\epsilon} \leftarrow \nabla_\theta L_{\mathbb{B}}(f_\theta)$ ▷ additional F_1 and B_1 to compute $\hat{\epsilon}$
  Compute $g = \nabla_\theta L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}})$ ▷ F_2 and B_2
  update the weights $\theta \leftarrow \theta - \eta g$

A.2 OPTIMIZING OVER SUBSET $\mathbb{B}^+$ IS REPRESENTATIVE

The sharpness-sensitive subset $\mathbb{B}^+$ are constructed sorting by $\ell(f_{\theta+\hat{\epsilon}}, x_i, y_i) - \ell(f_\theta, x_i, y_i)$, which is positively correlated to $l(f_\theta, x_i, y_i)$. By the first-order Taylor series approximation,

$$\ell(f_{\theta+\hat{\epsilon}}, x_i, y_i) - \ell(f_\theta, x_i, y_i) = \hat{\epsilon} \cdot \nabla_\theta \ell(f_\theta, x_i, y_i) + o(\|\hat{\epsilon}\|)$$

By Equation 4, $\hat{\epsilon}$ is the aggregated gradients of each instance in the complete dataset $\mathbb{B}$, i.e.,

$$\hat{\epsilon} = \underset{\epsilon:\|\epsilon\|_2<\rho}{\arg\max}\, L_{\mathbb{S}}(f_{\theta+\epsilon}) \approx \rho \nabla_\theta L_{\mathbb{S}}(f_\theta) = \sum_{i=1}^{|\mathbb{B}|} \nabla_\theta \ell(f_\theta, x_i, y_i)$$

which indicates that $\ell(f_{\theta+\hat{\epsilon}}, x_i, y_i) - \ell(f_\theta, x_i, y_i)$ is positively correlated to the gradient $\nabla_\theta \ell(f_\theta, x_i, y_i)$. Li et al. (2019) claims that the hard examples in deep learning (the training samples with high training loss) produce gradients with larger magnitudes. Therefore, $\ell(f_{\theta+\hat{\epsilon}}, x_i, y_i) - \ell(f_\theta, x_i, y_i)$ is positively correlated to $l(f_\theta, x_i, y_i)$. We also demonstrate the correlation empirically.

We conduct experiments to verify Equation 9 and Equation 10. In Figure 5, We plot the four losses, $L_{\mathbb{B}^+}(f_\theta)$, $L_{\mathbb{B}^-}(f_\theta)$, $L_{\mathbb{B}^+}(f_{\theta+\hat{\epsilon}})$, and $L_{\mathbb{B}^-}(f_{\theta+\hat{\epsilon}})$ w.r.t the epochs. The experimental results verify that Equation 9 and Equation 10 hold for every training epoch.

Moreover, we conduct experiments to demonstrate that optimizing over the subset $\mathbb{B}^+$ is much more representative than the subset $\mathbb{B}^-$. We compare the updating gradients computed from $\mathbb{B}^+$,$\mathbb{B}^-$ and a random subset $\mathbb{B}_{\text{rand}}$ that $|\mathbb{B}_{\text{rand}}| = |\mathbb{B}^+| = |\mathbb{B}^-|$ to those computed from $\mathbb{B}$ by calculating the cosine similarity, i.e.

$$\text{CosSim}(\nabla_\theta L_{\mathbb{B}^+}(f_{\theta+\hat{\epsilon}}), \nabla_\theta L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}}))$$
$$\text{CosSim}(\nabla_\theta L_{\mathbb{B}^-}(f_{\theta+\hat{\epsilon}}), \nabla_\theta L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}}))$$
$$\text{CosSim}(\nabla_\theta L_{\mathbb{B}_{\text{rand}}}(f_{\theta+\hat{\epsilon}}), \nabla_\theta L_{\mathbb{B}}(f_{\theta+\hat{\epsilon}}))$$

In Figure 6, we plot the cosine similarities in each training epoch evaluated with ResNet-18, Wide-28-10 on CIFAR10. In terms of the computed gradients, the experimental results show that $\mathbb{B}^+$ has the highest cosine similarities with $\mathbb{B}$ than $\mathbb{B}^-$ and the random set $\mathbb{B}_{\text{rand}}$.

A.3 LINEARITY MEASUREMENT OF SWP

As the experimental results in Figure 3 demonstrated, SWP can also improve the accuracy of ESAM compared to SAM. We will investigate the advantage of SWP in terms of generalization in the future research. Here we provide a discussion about the accuracy improvement contributed by SWP.

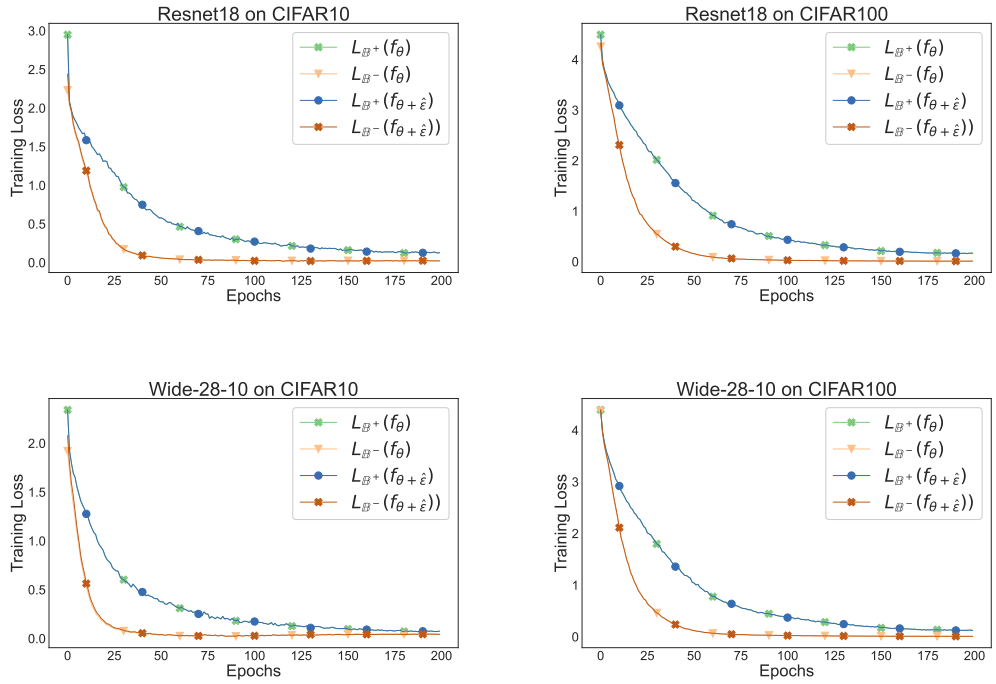

Figure 5: The SAM loss and the empirical loss calculated over the selected subsets $\mathbb{B}^+$, $\mathbb{B}^-$, w.r.t the epochs, as evaluated with ResNet-18, Wide-28-10 on CIFAR10 and CIFAR100. The subset $\mathbb{B}^+$ selected by SDS has much higher SAM loss and empirical loss than $\mathbb{B}^-$ among all the four groups of experiments.

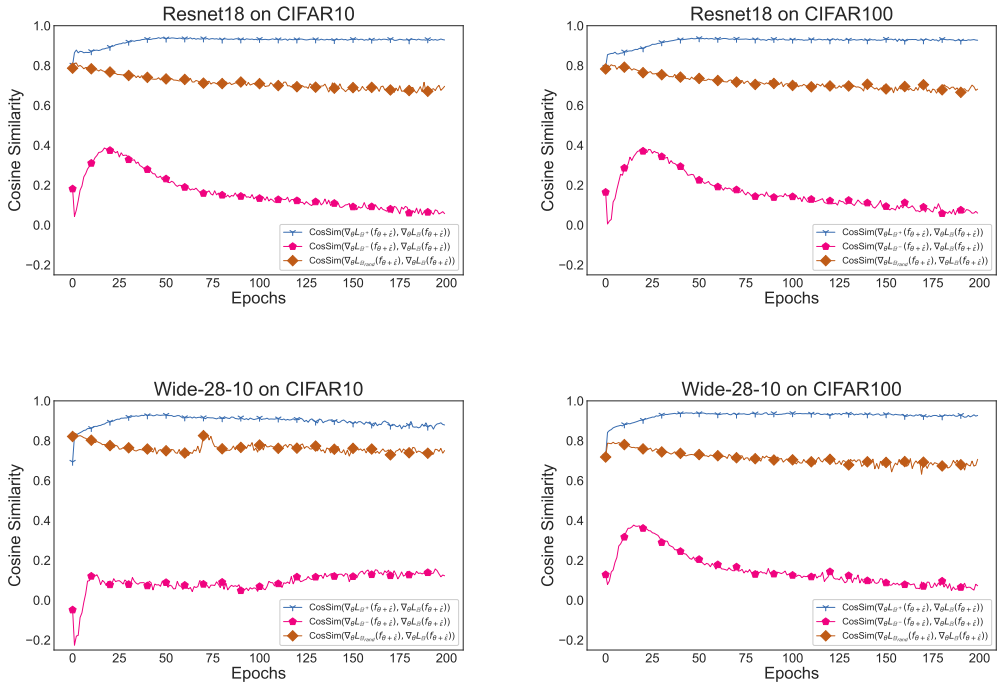

Figure 6: The cosine similarity between the gradients computed from subsets $\mathbb{B}^+$, $\mathbb{B}^-$, and $\mathbb{B}_{\text{rand}}$ with $\mathbb{B}$, as evaluated with ResNet-18, Wide-28-10 on CIFAR10 and CIFAR100. The gradients from the subset $\mathbb{B}^+$ selected by SDS has much higher cosine similarity with the gradients from $\mathbb{B}$ than the gradients from $\mathbb{B}^-$ and $\mathbb{B}_{\text{rand}}$ among all the four groups of experiments.

A plausible reason for such improvement is that SWP leads to a better inner maximization solved in equation 2. The current solution of $\hat{\epsilon}$ is approximated by assuming $L_{\mathbb{S}}(f_\theta)$ is a linear function. Therefore, the $\hat{\epsilon}$ would result in a better inner maximization if $L_{\mathbb{S}}(f_\theta)$ is more linear with respect to θ. Inspired by (Qin et al., 2019), we measure the linearity of the loss function by

$$\zeta(\epsilon, \mathbb{B}) = |L_{\mathbb{B}}(f_{\theta+\epsilon}) - L_{\mathbb{B}}(f_\theta) - \epsilon^\top \nabla_\theta L_{\mathbb{B}}(f_\theta)|$$

We conduct experiments on the CIFAR10 dataset with ResNet-18 model to verify that SWP can improve the linearity $\zeta(\epsilon, \mathbb{B})$ of the loss function $L_{\mathbb{S}}(f_\theta)$. We compare the linearity of ESAM with the β ranging from $\{0.2, 0.3, ..., 0.9\}$ to the SAM. The results are demonstrated in Figure 7. It can

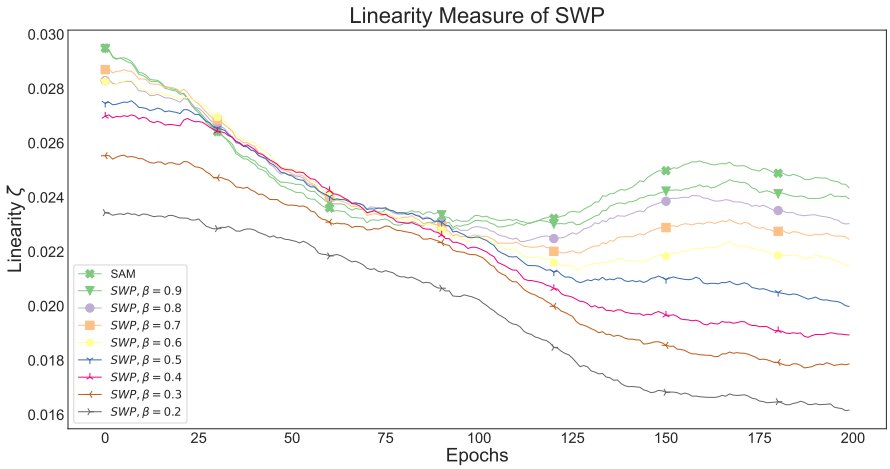

Figure 7: The linearity measurement of SWP evaluated with the ResNet18 model on the CIFAR10 dataset compared with SAM. The experimental results indicate that a smaller β can result in better linearity.

be shown that SWP will result in a better linearity as the β decreases. However, decreasing β will also reduce the magnitude of $\hat{\epsilon}$ and thus result in a worse inner maximization in equation 2. We observe that $\beta = \{0.5, 0.6\}$ is optimal to balance the accuracy and efficiency of ESAM.

A.4 Reduced computational overhead contributed by SWP

Formulation We formulate the saved computational overhead contributed by SWP here. We discuss and examine the computational overhead in the PyTorch framework. Suppose that the NN we discussed here has N layers. The most unit of parameters ($D \times C \times H \times W$) is the entire parameters of a layer in NN, where D is the number of kernels, C is the number of channels, H and W are the height and width of the input. SWP select each basic parameter unit to compute gradients (i.e. requries_grad = True) with probability β. We use $g(N, \beta)$ to measure the saved computational overhead in terms of percentage contributed by SWP compared to the vanilla SAM.

The computational overhead $g(N, \beta)$ is reduced not just by calculating gradients, but also by the storing and hooking gradients. Because the storing and hooking operations of each parameter's gradients are independent to each other, the computational overhead saved from storing and hooking are proportional to $1 - \beta$ and irrelevant to the depth N of NN. In addition, the storing and hooking gradients are the dominant factor that result in the reduced computational overhead according to our toy example in the following.

Next we discuss the saved computational overhead that stems from the calculation operations in the general case. Some layers in the DNNs with complicated architectures such as DenseNet and ViT, may have multiple basic parameter units in the same layer and be connected across any other layers. Suppose the n_{th} layer has $K(n)$ basic parameter units, we have $p^n = (1-\beta)^{K(n)} \cdot p^{n-1}$. Therefore, $p^n = (1-\beta)^{\sum_{j=1}^{n} K(j)} \approx (1-\beta)^{n\bar{K}}$, where $\bar{K} = \frac{1}{N}\sum_{j=1}^{N} K(j)$. Assumed that the computations

of each layer is the same, by summing up the saved computational overhead of all parameters, we have

$$g(N,\beta) = k_1(1-\beta)+k_2\sum_{n=i}^{N}\frac{p^n}{N} \approx k_1(1-\beta)+\frac{k_2(1-\beta)}{N[1-(1-\beta)^{\bar{K}}]} \approx k_1(1-\beta)+\frac{k_2^{'}(1-\beta)}{N\beta} \quad (11)$$

where k_1, k_2 are determined by the computing time of calculation, $k_2^{'} = \frac{k_2\beta}{[1-(1-\beta)^{\bar{K}}]}$.

However, the commonly used DNNs' architectures such as ResNet only have one basic parameter unit in each layer. Besides, each layer of them is connected in serial with the next layer. Then, let p^n be the calculation-free rate of a certain parameter unit in n_{th} layer, we have $p^n = (1-\beta)\cdot p^{n-1}$. Therefore, $p^n = (1-\beta)^n$. The saved calculation contributed by the parameter unit is $\frac{1}{N}p^n$. By summing up the saved computational overhead of all parameters, we have

$$g(N,\beta) = k_1(1-\beta) + k_2\sum_{n=i}^{N}\frac{p^n}{N} = k_1(1-\beta) + \frac{k_2}{N}\frac{1-\beta-(1-\beta)^N}{\beta} \approx k_1(1-\beta) + \frac{k_2(1-\beta)}{N\beta} \quad (12)$$

where k_1, k_2 are determined by the computing time of calculation, storing and hooking gradients.

Toy Example We conducted a toy example in Cifar10 dataset with two MLPs. Each fully connected layer of MLP is the same with a size of $3,000$ for both in and out features. The first mlp is for the special case that $\bar{K} = 1$, where each layer has only one basic parameter unit and is connected in serial with the next layer. We examined $N = \{50, 75, 100, 125\}$ and $\beta = \{0.1, ..., 0.9, 1.0\}$ to record the saved computational overhead in percentage. Part of the results are reported in Table A.4. By linear regression, we have $k1 = 0.3185, k2 = 0.1310$, and the returned $R^2 = 0.9983$. The second mlp is for the general case that $\bar{K} > 1$, where each layer may have multiple basic parameter units in the same layer and be connected across any other layers. We examined $N = \{35, 50, 65, 75\}$ and $\beta = \{0.1, ..., 0.9, 1.0\}$ to record the saved computational overhead in percentage. Part of the results are reported in Table A.4. By linear regression, we have $k1 = 0.3143, k2 = 0.0737$, and the returned $R^2 = 0.9989$. The above experimental results verify the formulation of the reduced computational overhead contributed by SWP in equation 12 and equation 11.

N	β	$g(N,\beta)$
50	0.9	3.42%
50	0.5	16.28%
50	0.1	30.08%
75	0.9	3.44%
75	0.5	15.79%
75	0.1	30.42%
100	0.9	3.37%
100	0.5	15.86%
100	0.1	29.56%
125	0.9	2.94%
125	0.5	14.87%
125	0.1	29.42%

Table 4: The special case that $\bar{K} = 1$. By linear regression, $k_1 = 0.3185, k_2 = 0.1310$, and the returned $R^2 = 0.9983$.

N	β	$g(N,\beta)$
25	0.9	2.88%
25	0.5	15.54%
25	0.1	30.03%
50	0.9	2.34%
50	0.5	15.10%
50	0.1	29.83%
65	0.9	2.23%
65	0.5	14.95%
65	0.1	29.22%
75	0.9	2.22%
75	0.5	15.10%
75	0.1	28.68%

Table 5: The general case that $\bar{K} > 1$. By linear regression, $k_1 = 0.3143, k_2^{'} = 0.0737$, and the returned $R^2 = 0.9989$.

A.5 Visualization of Loss Landscapes with respect to adversarial weight perturbations

We visualize the sharpness of the flat minima with respect to adversarial weight perturbations of SGD,SAM and ESAM on the Cifar10 dataset. The x- and y-axes represent two orthogonal adversarial weight perturbations, which are $\eta\nabla_\theta L_{\mathbb{B}_x}(f_\theta)$ and $\eta\nabla_\theta L_{\mathbb{B}_y}(f_\theta)$ respectively, where η is the learning rate during training. $\mathbb{B}_x$ and $\mathbb{B}_y$ are the randomly sampled subsets of batch $\mathbb{B}$, and $|\mathbb{B}_x| = |\mathbb{B}_y| = \frac{1}{2}|\mathbb{B}|$, $\mathbb{B}_x \cup \mathbb{B}_y = \mathbb{B}$. We display the loss landscape in Figure 8, which demonstrates that both SAM and ESAM improve the sharpness significantly in comparison to SGD.

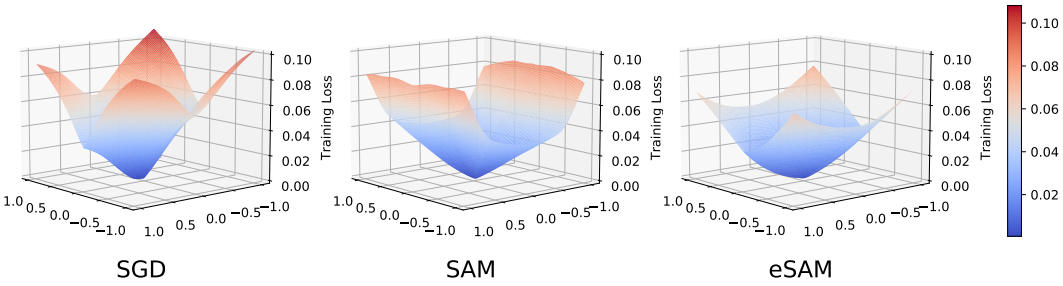

Figure 8: Cross-entropy loss landscapes of the ResNet18 model respect to adversarial weight perturbations on the CIFAR10 dataset trained with SGD, SAM, and ESAM.

A.6 Evaluation of ESAM on ViT-S/16

SAM has also been demonstrated to be effective on the new vision Transformer(ViT) architecture (Chen et al., 2021). Therefore, we also evaluate ESAM with ViT-S/16 on ImageNet Datasets. We use $\beta = 0.5$ and $\gamma = 0.7$ for ViT-S/16, which share the same hyperparameters as ResNet-50 and ResNet-101 in section 3.1. The results are reported in Table 6, which indicate that ESAM can still be effective to improve efficiency in ViT-S/16 architecture. In particular, ESAM-SWP achieves much better accuracy than SAM(80.88% v.s. 80.34%).

Table 6: Classification accuracies and training speed of ViT-S/16 on the ImageNet dataset.

	ViT-S/16	
ImageNet	Accuracy	images/s
SGD	79.72	1,133
SAM	80.34	581
ESAM-SWP	80.88	616
ESAM-SDS	79.97	693
ESAM	80.46	734

A.7 Training Details

We tune the training parameters of SGD, SAM, and ESAM, by using grid searches. The learning rate is chosen from the set $\{0.01, 0.05, 0.1, 0.2\}$, the weight decay from the set $\{5 \times 10^{-4}, 1 \times 10^{-3}\}$, and the batch size from the set $\{64, 128, 256\}$. This is done to attain the best accuracies. The exact training hyperparameters are reported in Table 7. On the ImageNet datasets, limited by the computing resource, we follow and slightly modify the optimal hyperparameters as suggested by Chen et al. (2021) for SGD, SAM and ESAM. The exact training hyperparameters are reported in Table 8.

Table 7: Hyperparameters for training from scratch on CIFAR10 and CIFAR100

	CIFAR-10			**CIFAR-100**		
ResNet-18	SGD	SAM	ESAM	SGD	SAM	ESAM
Epoch		200			200	
Batch size		128			128	
Data augmentation		Basic			Basic	
Peak learning rate		0.05			0.05	
Learning rate decay		Cosine			Cosine	
Weight decay	5×10^{-4}	1×10^{-3}	1×10^{-3}	5×10^{-4}	1×10^{-3}	1×10^{-3}
ρ	-	0.05	0.05	-	0.05	0.05
Wide-28-10	SGD	SAM	ESAM	SGD	SAM	ESAM
Epoch		200			200	
Batch size		256			256	
Data augmentation		Basic			Basic	
Peak learning rate		0.05			0.05	
Learning rate decay		Cosine			Cosine	
Weight decay	5×10^{-4}	1×10^{-3}	1×10^{-3}	5×10^{-4}	1×10^{-3}	1×10^{-3}
ρ	-	0.1	0.1	-	0.1	0.1
PyramidNet-110	SGD	SAM	ESAM	SGD	SAM	ESAM
Epoch		300			300	
Batch size		256			256	
Data augmentation		Basic			Basic	
Peak learning rate		0.1			0.1	
Learning rate decay		Cosine			Cosine	
Weight decay		5×10^{-4}			5×10^{-4}	
ρ	-	0.2	0.2	-	0.2	0.2

Table 8: Hyperparameters for training from scratch on ImageNet

	ResNet-50			**ResNet-110**		
ImageNet	SGD	SAM	ESAM	SGD	SAM	ESAM
Epoch		90			90	
Batch size		512			512	
Data augmentation		Inception-style			Inception-style	
Peak learning rate		0.2			0.2	
Learning rate decay		Cosine			Cosine	
Weight decay		1×10^{-4}			1×10^{-4}	
ρ	-	0.05	0.05	-	0.05	0.05
Input resolution		224×224			224×224	