Enhancing Sharpness-Aware Minimization by Learning Perturbation Radius

Xuehao Wang[1,†], Weisen Jiang[1,2,†], Shuai Fu[1], and Yu Zhang[1] (✉)

[1] Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China

[2] The Hong Kong University of Science and Technology, Hong Kong, China

{xuehaowangfi,waysonkong,fus.jayce,yu.zhang.ust}@gmail.com

Abstract. Sharpness-aware minimization (SAM) is to improve model generalization by searching for flat minima in the loss landscape. The SAM update consists of one step for computing the perturbation and the other for computing the update gradient. Within the two steps, the choice of the perturbation radius is crucial to the performance of SAM, but finding an appropriate perturbation radius is challenging. In this paper, we propose a bilevel optimization framework called LEarning the perTurbation radiuS (LETS) to learn the perturbation radius for sharpness-aware minimization algorithms. Specifically, in the proposed LETS method, the upper-level problem aims at seeking a good perturbation radius by minimizing the squared generalization gap between the training and validation losses, while the lower-level problem is the SAM optimization problem. Moreover, the LETS method can be combined with any variant of SAM. Experimental results on various architectures and benchmark datasets in computer vision and natural language processing demonstrate the effectiveness of the proposed LETS method in improving the performance of SAM.

Keywords: Sharpness-Aware Minimization · Bilevel Optimization · Hyperparameter Optimization

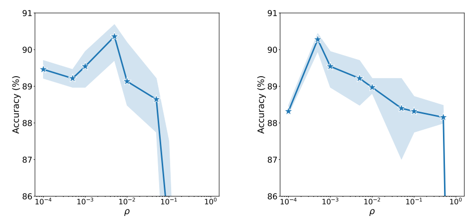
1 Introduction

Deep neural networks have demonstrated remarkable performance across various fields [17,48], but they tend to overfit on the training data with poor ability of generalization due to overparameterization [49]. The loss function landscape is intricate and non-linear, characterized by numerous local minima with varying generalization capabilities. Several studies [19,27,26] have explored the connection between the geometry of the loss function surface and the generalization ability of neural networks, and have revealed that flatter minima tend to result in better generalization performance than sharper minima [10,39,27,26].

Sharpness-aware minimization (SAM) [12] is an optimization method that solves a min-max optimization problem to seek flat minima. Specifically, SAM

[†] Equal Contribution.

aims to find a model parameterized by $\boldsymbol{\theta}$ such that its neighbors in parameter space also have good performance. SAM first computes the worst-case perturbation $\boldsymbol{\epsilon}$ that maximizes the training loss within a neighborhood specified by a perturbation radius ρ, and then minimizes the training loss w.r.t. the perturbed model $\boldsymbol{\theta} + \boldsymbol{\epsilon}$. Many variants of SAM are proposed to improve its effectiveness [53,31,50,36] and efficiency [35,51,24].



(a) SAM. (b) ASAM.

Fig. 1. Classification accuracy of SAM and ASAM with different ρ's on *MRPC* using *DeBERTa*. As shown, the performance is sensitive to ρ.

The perturbation radius ρ controls the strength of penalizing sharp minima and plays an important role in SAM [12,2,31,53]. Figure 1 shows the classification accuracy of SAM and ASAM [31] w.r.t. different ρ's on the *MRPC* dataset using the *DeBERTa* network, where the corresponding experimental setups are shown in Section 4.4. As can be seen, SAM and ASAM prefer different ρ's, and their performance is sensitive to ρ, emphasizing the crucial need for careful selection of ρ. To deal with this issue, recent attempts [12,31,53] perform grid search on ρ, which is straightforward but time-consuming.

To learn the perturbation radius ρ more efficiently, in this paper, we propose a LEarning the perTurbation radiuS (LETS) method by formulating the learning of ρ as a bilevel optimization problem. Specifically, in the lower-level problem, a SAM model is obtained based on the training data and a given ρ, while in the upper-level problem, ρ is updated by minimizing the gap between the validation and training losses based on the obtained SAM model, which is a function of ρ. As the lower-level problem is usually nonconvex, we propose a gradient-based algorithm for updating model parameters and ρ alternatively. Experiments conducted on several benchmark datasets across diverse fields demonstrate that the proposed LETS is effective in learning a suitable radius.

In summary, our contributions are three-fold: (i) We formulate the problem of learning the perturbation radius as bilevel optimization and propose a gradient-based algorithm (called LETS-SAM) to adjust the radius for SAM. (ii) We perform extensive experiments on various datasets across computer vision and natural language process tasks as well as various network architectures across convolution-based and transformer-based networks to verify that LETS-SAM performs better than SAM. (iii) LETS is general and can be combined with any SAM algorithm. We integrate it into ASAM to propose LETS-ASAM. Experimental results show that LETS-ASAM achieves better performance than ASAM, demonstrating the proposed LETS is effective.

Notations. Lowercase and uppercase boldface letters denote vectors (e.g., $\mathbf{x}$) and matrices (e.g., $\mathbf{X}$), respectively. ℓ_2-norm of $\mathbf{x}$ is denoted by $\|\mathbf{x}\|$. $\mathrm{diag}(\mathbf{v})$ constructs a diagonal matrix with the vector $\mathbf{v}$ on the diagonal. For a vector $\mathbf{v} \in \mathbb{R}^d$, $[\mathbf{v}]^2 \equiv [v_1^2, \ldots, v_d^2]$ (resp. $|\mathbf{v}| \equiv [|v_1|, \ldots, |v_d|]$) denotes the elementwise square (resp. absolute) of $\mathbf{v}$. $\mathbf{I}$ is the identity matrix. $\mathbf{1}_d$ is a d-dimensional all-ones vector. $\mathcal{D}^{tr} = \{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N^{tr}}$, $\mathcal{D}^{vl} = \{(\mathbf{x}_i^{vl}, y_i^{vl})\}_{i=1}^{N^{vl}}$ and $\mathcal{D}^{ts} = \{(\mathbf{x}_i^{ts}, y_i^{ts})\}_{i=1}^{N^{ts}}$ represent the training, validation, and testing datasets, respectively. $f(\mathbf{x}; \boldsymbol{\theta})$ denotes a model parameterized by $\boldsymbol{\theta}$. $\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \ell(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i)$ denotes the loss on data set $\mathcal{D}$ using model $\boldsymbol{\theta}$, where $\ell(\cdot, \cdot)$ denotes a loss function (e.g., cross-entropy loss for classification). $\nabla \mathcal{L}(\mathcal{D}; \boldsymbol{\theta})$ and $\nabla^2 \mathcal{L}(\mathcal{D}; \boldsymbol{\theta})$ denote the gradient and Hessian of $\mathcal{L}(\mathcal{D}; \boldsymbol{\theta})$ w.r.t $\boldsymbol{\theta}$, respectively.

2 Related Work

Generalization and Loss Landscape. As deep neural networks are powerful enough to memorize all training data, seeking a model with better generalization ability is crucial to mitigate overfitting. Recently, various works [26,10,27] conduct extensive experiments on various datasets to study the relationship between loss landscape and generalization, and found that a flatter minima results in a better generalization. Therefore, several algorithms are proposed to improve the generalization ability of models by seeking flatter minima. For example, [4,52] add noise to model parameters, SWA and its variants [7,21] average model parameters during training, [50] penalizes gradient norm, and sharpness-aware minimization (SAM) as well as its variants [36,53,12,31] solves a min-max problem to search flat minima explicitly. These approaches have demonstrated superior results in various fields, including supervised learning [4,40,50,12,21], transfer learning [12,53], domain generalization [7], federated learning [40], and natural language processing [3].

Sharpness-Aware Minimization (SAM). SAM [12] seeks flat minima via solving a min-max optimization problem as

$$\min_{\boldsymbol{\theta}} \max_{\|\boldsymbol{\epsilon}\| \leq \rho} \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta} + \boldsymbol{\epsilon}), \tag{1}$$

where $\rho > 0$ is a perturbation radius. Intuitively, SAM aims to find a model $\boldsymbol{\theta}$ such that all its neighbor models (within an ℓ_2 ball of radius ρ) have low losses. Due to the infeasible computation cost of solving the inner maximization problem for nonconvex losses, SAM approximates it via the first-order Taylor approximation and obtains the update rule at iteration t as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla \mathcal{L}\left(\mathcal{D}^{tr}; \boldsymbol{\theta}_t + \rho \frac{\nabla \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_t)}{\|\nabla \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_t)\|}\right),$$

where η denotes the step size. Note that SAM involves two gradient calculations at each iteration. To improve its efficiency, many methods have been proposed to reduce the computation cost of SAM. For example, Look-SAM [35] and RST [51] employ the SAM update periodically or randomly, respectively, while AE-SAM [24] proposes an adaptive policy to employ SAM only when the model is in sharp regions. ESAM [9] proposes to perturb the chosen part of the model and only uses a selected subset of samples to compute the gradients. To improve the effectiveness of SAM, GSAM [53] proposes to minimize a surrogate gap $\max_{\|\boldsymbol{\epsilon}\|\leq\rho} \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}+\boldsymbol{\epsilon}) - \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta})$, while RSAM [36] simply injects Gaussian noises to perturb model parameters and ASAM [31] designs an adaptive sharpness measure by re-scaling.

For most of the SAM-based methods, the perturbation radius ρ is crucial to their performance [12,2,31,53]. Instead of performing a grid search over ρ by cross-validation, which is time-consuming, in this paper, we propose a gradient-based method to learn ρ.

Bilevel Optimization. Bilevel optimization is first introduced in [6] and successfully used in a variety of areas, for example, hyperparameter optimization [33,11,13], meta-learning [13,22,23,47], prompt tuning [25], and reinforcement learning [44]. Bilevel optimization consists of two problems: a lower-level problem and an upper-level one. The former acts as a constraint for the latter. When the lower-level problem is convex, one approach is to reformulate the bilevel problem as a single-level problem by replacing the lower-level problem with the first-order optimality condition [1,43]. However, in deep neural networks, problems are usually nonconvex. Recently, gradient-based first-order methods [14,20,32] for bilevel optimization have become popular due to their efficiency and effectiveness.

3 The LETS Method

In this section, we formulate the objective function of the LETS method to learn the perturbation radius ρ as bilevel optimization (i.e., Section 3.1) and propose a gradient-based algorithm to learn ρ (i.e., Section 3.2). Considering the generality of the proposed method, it can be combined with any SAM algorithm, and an example of integrating it into ASAM [31] is shown in Section 3.3.

3.1 Problem Formulation

We consider the SAM optimization in problem (1). Let $\boldsymbol{\theta}^\star(\rho)$ be the solution, which is a function of the perturbation radius ρ. Though SAM has shown to be effective, its generalization performance is sensitive to the choice of ρ (i.e., Figure 1). Instead of using grid search, which is simple but time-consuming, we propose to learn ρ in an end-to-end manner. To achieve this, we formulate the problem of learning ρ into bilevel optimization, where the lower-level objective is the SAM problem and the upper-level objective is a generalization metric for $\boldsymbol{\theta}^\star(\rho)$. The choice of generalization metric is flexible, for example, the loss value on the validation set $\mathcal{L}(\mathcal{D}^{vl}; \boldsymbol{\theta}^\star(\rho))$, the generalization gap $\mathcal{L}(\mathcal{D}^{vl}; \boldsymbol{\theta}^\star(\rho)) - \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}^\star(\rho))$, or its square $\frac{1}{2}\left(\mathcal{L}(\mathcal{D}^{vl}; \boldsymbol{\theta}^\star(\rho)) - \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}^\star(\rho))\right)^2$. Empirical results (i.e., Table 6 in Section 4.8) show that the last is better and therefore is used. Formally, the objective function of the proposed LETS method is formulated as

$$\min_{\rho\in(0,\infty)} \quad \frac{1}{2}\left(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}^\star(\rho)) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}^\star(\rho))\right)^2 \tag{2}$$

$$\text{s.t.} \quad \boldsymbol{\theta}^\star(\rho) = \arg\min_{\boldsymbol{\theta}} \max_{\|\boldsymbol{\epsilon}\|\le\rho} \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}+\boldsymbol{\epsilon}). \tag{3}$$

3.2 Learning Perturbation Radius for SAM

When the lower-level problem is convex, one can seek the optimal solution $\boldsymbol{\theta}^\star(\rho)$ by solving the low-level problem (3) and update ρ in the upper level by performing one gradient descent step, where hyper-gradient $\nabla_\rho \frac{1}{2}(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}^\star(\rho)) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}^\star(\rho)))^2$ can be computed by iterative differentiation [38] or approximate implicit differentiation [29]. However, in deep neural networks, the lower-level problem is usually nonconvex, thus, seeking $\boldsymbol{\theta}^\star(\rho)$ is computationally infeasible. To address this problem, we propose a gradient-based algorithm for updating the model parameters and ρ alternatively. The detailed procedure is shown in Algorithm 1.

At iteration t, we sample a batch $\mathcal{B}_t^{tr}$ from the training dataset and $\mathcal{B}_t^{vl}$ from the validation dataset (i.e., steps 2 and 3). For the lower-level problem, we take a gradient descent update (i.e., steps 5 and 7) as

$$\boldsymbol{\theta}_{t+1}(\rho_t) = \boldsymbol{\theta}_t - \eta\nabla\mathcal{L}\left(\mathcal{D}^{tr};\boldsymbol{\theta}_t + \rho_t\hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})}\right), \tag{4}$$

where $\hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})} \equiv \frac{\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)}{\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|}$ and η is the step size. Here $\boldsymbol{\theta}_{t+1}(\rho_t)$ is an approximate solution to the SAM problem as we only conduct the gradient descent step once. Obviously $\boldsymbol{\theta}_{t+1}(\rho_t)$ is a function of ρ_t.

In the upper-level problem, we perform a gradient descent step for updating ρ (i.e., step 14) as

$$\rho_{t+1} = \rho_t - \beta\nabla_{\rho_t}\frac{1}{2}(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}_{t+1}(\rho_t)) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1}(\rho_t)))^2,$$

where β is the step size. $\nabla_{\rho_t}\frac{1}{2}(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}_{t+1}(\rho_t)) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1}(\rho_t)))^2$ is computed by the chain rule (i.e., steps 11 and 13) as $-\eta\nabla^\top_{\boldsymbol{\theta}_{t+1}}\frac{1}{2}(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1}))^2\nabla^2\mathcal{L}\left(\mathcal{D}^{tr};\boldsymbol{\theta}_t+\rho_t\hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})}\right)\hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})}$. Details of the derivation are provided in Appendix A. Here the first term of gradient is easy to compute as

$$\nabla_{\boldsymbol{\theta}_{t+1}}\frac{1}{2}(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}_{t+1})-\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1}))^2 = (\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}_{t+1})-\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1})) \\ \left(\nabla_{\boldsymbol{\theta}_{t+1}}\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}_{t+1})-\nabla_{\boldsymbol{\theta}_{t+1}}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1})\right)$$

(i.e., steps 8 and 9). The second term needs to compute a Hessian, which is computationally expensive for large models like deep neural networks. Following

Algorithm 1 LEarning perTurbation radiuS (LETS-SAM and LETS-ASAM).

Require: training set $\mathcal{D}^{tr}$, validation set $\mathcal{D}^{vl}$; stepsizes β and η, #iterations T; model parameter $\boldsymbol{\theta}$; initialization ρ_0 and $\boldsymbol{\theta}_0$; ξ for LETS-ASAM;

1: **for** $t = 0, \ldots, T-1$ **do**
2: sample a mini-batch training data $\mathcal{B}_t^{tr}$ from $\mathcal{D}^{tr}$;
3: sample a mini-batch validation data $\mathcal{B}_t^{vl}$ from $\mathcal{D}^{vl}$;
4: $\mathbf{g}_t^{tr} = \nabla \mathcal{L}(\mathcal{B}_t^{tr}; \boldsymbol{\theta}_t)$;
5: if LETS-SAM: $\hat{\mathbf{g}}_t^{tr} = \nabla \mathcal{L}\left(\mathcal{B}_t^{tr}; \boldsymbol{\theta}_t + \rho_t \frac{\mathbf{g}_t^{tr}}{\|\mathbf{g}_t^{tr}\|}\right)$;
6: if LETS-ASAM: $\hat{\mathbf{g}}_t^{tr} = \nabla \mathcal{L}\left(\mathcal{B}_t^{tr}; \boldsymbol{\theta}_t + \rho_t \frac{\mathbf{T}_{\boldsymbol{\theta}_t}^2 \mathbf{g}_t^{tr}}{\|\mathbf{T}_{\boldsymbol{\theta}_t} \mathbf{g}_t^{tr}\|}\right)$,
where $\mathbf{T}_{\boldsymbol{\theta}_t}$ is computed by Eq. (7);
7: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \hat{\mathbf{g}}_t^{tr}$;
8: $\bar{\mathbf{g}}_t^{tr} = \nabla \mathcal{L}(\mathcal{B}_t^{tr}; \boldsymbol{\theta}_{t+1})$ and $\bar{\mathbf{g}}_t^{vl} = \nabla \mathcal{L}(\mathcal{B}_t^{vl}; \boldsymbol{\theta}_{t+1})$;
9: $\mathbf{g}_a = (\mathcal{L}(\mathcal{B}_t^{vl}; \boldsymbol{\theta}_{t+1}) - \mathcal{L}(\mathcal{B}_t^{tr}; \boldsymbol{\theta}_{t+1}))(\bar{\mathbf{g}}_t^{vl} - \bar{\mathbf{g}}_t^{tr})$;
10: $\mathbf{H} = \mathrm{diag}([\hat{\mathbf{g}}_t^{tr}]^2)$;
11: if LETS-SAM: $\mathbf{g}_b = \mathbf{H} \frac{\bar{\mathbf{g}}_t^{tr}}{\|\bar{\mathbf{g}}_t^{tr}\|}$;
12: if LETS-ASAM: $\mathbf{g}_b = \mathbf{H} \frac{\mathbf{T}_{\boldsymbol{\theta}_t}^2 \bar{\mathbf{g}}_t^{tr}}{\|\mathbf{T}_{\boldsymbol{\theta}_t} \bar{\mathbf{g}}_t^{tr}\|}$;
13: $g_\rho = -\mathbf{g}_a^\top \mathbf{g}_b g_\rho$;
14: $\rho_{t+1} = \rho_t - \beta \eta g_\rho$;
15: **end for**
16: **return** $\boldsymbol{\theta}_T$.

[5,28], the Hessian $\nabla^2 \mathcal{L}\left(\mathcal{D}^{tr}; \boldsymbol{\theta}_t + \rho_t \hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})}\right)$ can be approximated by a first-order derivative (i.e., step 10) as

$$\mathrm{diag}\left(\left[\nabla \mathcal{L}\left(\mathcal{D}^{tr}; \boldsymbol{\theta}_t + \rho_t \hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})}\right)\right]^2\right). \tag{5}$$

As proved in Appendix B , LETS-SAM has a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{T}})$, which is the same as SAM [2] and its variants [40,24] under similar conditions. The details of the theoretical analysis are provided in Appendix B. Hence, adjusting the perturbation radius does not affect the convergence speed.

3.3 LETS-ASAM

As the proposed LETS method is very general and can be integrated into any variant of SAM, we show an example by combining LETS with the recent state-of-the-art method ASAM [31]. The combined algorithm called LETS-ASAM is shown in Algorithm 1.

ASAM defines an adaptive sharpness of the loss function, whose maximization region is determined by the normalization operator. Then, the objective function of ASAM is formulated as

$$\boldsymbol{\theta}^\star(\rho) \equiv \arg\min_{\boldsymbol{\theta}} \max_{\|\mathbf{T}_{\boldsymbol{\theta}}^{-1} \boldsymbol{\epsilon}\| \leq \rho} \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta} + \boldsymbol{\epsilon}), \tag{6}$$

where $\mathbf{T}_{\boldsymbol{\theta}}^{-1}$ is a normalization operator at $\boldsymbol{\theta}$. For example, $\mathbf{T}_{\boldsymbol{\theta}}$ is defined as $\mathbf{T}_{\boldsymbol{\theta}} = \mathrm{diag}(|\boldsymbol{\theta}|) + \xi\mathbf{I}$ for fully-connected layers, where ξ is a positive hyperparameter, and for convolutional layers, $\mathbf{T}_{\boldsymbol{\theta}}$ is defined as

$$\mathbf{T}_{\boldsymbol{\theta}} = \mathrm{diag}\left(\left[\|\mathbf{c}_1\|\mathbf{1}_{d_1}, ..., \|\mathbf{c}_k\|\mathbf{1}_{d_k}, |\tilde{\boldsymbol{\theta}}|\right]\right) + \xi\mathbf{I}, \tag{7}$$

where $\boldsymbol{\theta} = [\mathbf{c}_1, \ldots, \mathbf{c}_k, \tilde{\boldsymbol{\theta}}]$, $\mathbf{c}_i$ is the flattened weight vector of the ith convolution filter with its dimension as d_i, and $\tilde{\boldsymbol{\theta}}$ denote parameters that are not contained in convolution filters. To integrate LETS into ASAM, we replace the lower-level problem (3) with problem (6). At iteration t, the update rule at the lower level (i.e., steps 6 and 7) becomes

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\nabla\mathcal{L}\left(\mathcal{D}^{tr}; \boldsymbol{\theta}_t + \rho_t\hat{\boldsymbol{\epsilon}}_t^{(\mathrm{ASAM})}\right), \tag{8}$$

where $\hat{\boldsymbol{\epsilon}}_t^{(\mathrm{ASAM})} \equiv \frac{\mathbf{T}_{\boldsymbol{\theta}_t}^2\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)}{\|\mathbf{T}_{\boldsymbol{\theta}_t}\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|}$, while the update rule at the upper level (i.e., steps 12 and 13) is

$$\rho_{t+1} = \rho_t + \frac{\beta\eta}{2}\nabla_{\boldsymbol{\theta}_{t+1}}^\top\big(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1})\big)^2\nabla^2\mathcal{L}\left(\mathcal{D}^{tr};\boldsymbol{\theta}_t + \rho_t\hat{\boldsymbol{\epsilon}}_t^{(\mathrm{ASAM})}\right)\hat{\boldsymbol{\epsilon}}_t^{(\mathrm{ASAM})},$$

where the Hessian can be approximately computed by using the first-order derivative as in Eq. (5).

4 Experiments

In this section, we first compare the proposed LETS-SAM and LETS-ASAM methods with state-of-the-art SAM-based methods on computer vision tasks (e.g. *CIFAR-10*, *CIFAR-100*, and *ImageNet*) and natural language processing (e.g., *GLUE* and *IWSLT'14 DE-EN*) tasks on various architectures. Next, we evaluate the robustness of LETS-SAM and LETS-ASAM to label noise. Furthermore, we conduct experiments to study the robustness of LETS to the initialization of ρ (i.e., ρ_0) and the effects of different generalization metrics. To further illustrate the superior performance of LETS, we visualize the loss landscapes of models learned by the LETS methods. Finally, we empirically study the convergence of the proposed LETS method.

Baselines. The proposed methods are compared with ERM, SAM [12], ESAM [9], RST [51], LookSAM [35], AE-SAM [24], AE-LookSAM [24], ASAM [31], and GSAM [53]. ESAM selects some of the training samples to update the model and uses a subset of parameters to compute the perturbation. RST switches between SAM and ERM randomly for each iteration according to a Bernoulli trial with a probability 0.5. LookSAM uses SAM for every five steps. AE-SAM and AE-LookSAM use SAM adaptively. ASAM improves SAM by using an adaptive sharpness measure while GSAM improves SAM by minimizing a surrogate gap. We use official implementations of those baselines.

Table 1. Classification accuracy (%) on *CIFAR-10* using various architectures. The better result in each comparison group is underlined and the best result across all the groups is in **bold**.

	ResNet-18	*WideResNet-28-10*	*PyramidNet-110*	*ViT-S16*
ERM	95.41 ± 0.03	96.34 ± 0.12	96.62 ± 0.10	86.69 ± 0.11
ESAM	96.56 ± 0.08	97.29 ± 0.11	97.81 ± 0.10	84.27 ± 0.11
RST	96.40 ± 0.16	97.09 ± 0.11	97.22 ± 0.10	87.38 ± 0.14
AE-SAM	96.63 ± 0.04	97.30 ± 0.10	97.90 ± 0.09	87.77 ± 0.13
LookSAM	96.32 ± 0.12	97.02 ± 0.12	97.10 ± 0.11	87.12 ± 0.20
AE-LookSAM	96.56 ± 0.21	97.15 ± 0.08	97.22 ± 0.11	87.32 ± 0.11
GSAM	96.61 ± 0.05	97.39 ± 0.08	97.65 ± 0.05	88.33 ± 0.41
SAM	96.52 ± 0.12	97.27 ± 0.11	97.30 ± 0.10	87.37 ± 0.09
LETS-SAM	$\underline{\mathbf{96.81}} \pm 0.02$	$\underline{97.49} \pm 0.08$	$\underline{97.79} \pm 0.06$	$\underline{88.83} \pm 0.17$
ASAM	96.57 ± 0.02	97.28 ± 0.07	97.58 ± 0.06	90.35 ± 0.05
LETS-ASAM	$\underline{96.77} \pm 0.01$	$\underline{\mathbf{97.54}} \pm 0.08$	$\underline{\mathbf{97.91}} \pm 0.01$	$\underline{\mathbf{90.75}} \pm \mathbf{0.37}$

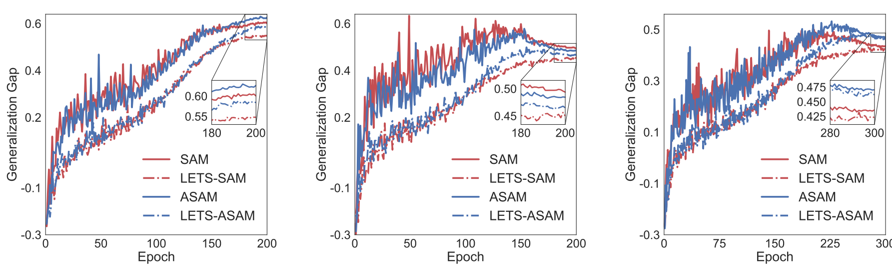
4.1 *CIFAR-10* and *CIFAR-100*

Setups. Experiments are conducted on the *CIFAR-10* and *CIFAR-100* datasets [30], each of which contains 50,000 images for training and 10,000 for testing. We use four network architectures: *ResNet-18* [17], *WideResNet-28-10* [48], *PyramidNet-110* [15], and *ViT-S16* [8]. Following experimental setups in [12,31,24], the batch size is set to 128, and the SGD optimizer with momentum 0.9 and weight decay 0.0005 is used. In the SGD optimizer, for updating model parameters, an initial learning rate 0.1 with the cosine learning rate scheduler is adopted, while we use an initial learning rate of 0.0001 with an exponential learning rate scheduler to update ρ. We train *PyramidNet-100* for 300 epochs, *ViT-S16* for 1200 epochs, and train *ResNet-18* and *WideResNet-28-10* for 200 epochs. As the *CIFAR* datasets do not have a held-out validation set, following the practice introduced in [34], mini-batches of validation data are randomly sampled from the training set. To ensure ρ is positive, an exponential transformation $\exp(\cdot)$ is applied to ρ, i.e., $\rho = \exp(\nu)$, where ν is an unconstrained variable to be learned. Experiments are repeated over three random seeds. All implementation details are summarized in Appendix E.

Results. The experimental results on *CIFAR-10* and *CIFAR-100* are shown in Table 1 and 2, respectively. We can find that, by learning the perturbation radius, LETS-SAM performs better than SAM on all the four architectures. Compared with ASAM, LETS-ASAM is also better, demonstrating the effectiveness of the proposed LETS method. Furthermore, on *CIFAR-100*, LETS-ASAM achieves the highest accuracy on *ResNet-18* and *ViT-S16*, while LETS-SAM is the best on *WideResNet-28-10* and *PyramidNet-110*. On *CIFAR-10*, LETS-SAM achieves the highest accuracy on *ResNet-18*, while LETS-ASAM outperforms all the baseline models on *WideResNet-28-10*, *PyramidNet-110*, and *ViT-S16*.

Table 2. Classification accuracy (%) on *CIFAR-100* using various architectures. The better result in each comparison group is underlined and the best result across all the groups is in **bold**.

	ResNet-18	*WideResNet-28-10*	*PyramidNet-110*	*ViT-S16*
ERM	78.17 ± 0.05	81.56 ± 0.14	81.89 ± 0.15	62.42 ± 0.22
ESAM	80.41 ± 0.10	84.51 ± 0.02	85.39 ± 0.05	62.11 ± 0.15
RST	80.10 ± 0.16	82.89 ± 0.02	84.90 ± 0.05	63.18 ± 0.19
AE-SAM	80.48 ± 0.11	84.51 ± 0.11	85.58 ± 0.10	63.68 ± 0.23
LookSAM	79.89 ± 0.29	83.70 ± 0.12	84.01 ± 0.06	63.52 ± 0.19
AE-LookSAM	80.29 ± 0.37	83.92 ± 0.07	84.80 ± 0.13	64.16 ± 0.23
GSAM	80.27 ± 0.33	83.80 ± 0.08	84.91 ± 0.29	63.21 ± 0.38
SAM	80.17 ± 0.15	83.42 ± 0.05	84.46 ± 0.05	63.23 ± 0.25
LETS-SAM	$\underline{80.71} \pm 0.07$	$\mathbf{\underline{84.78}} \pm 0.27$	$\mathbf{\underline{85.86}} \pm \mathbf{0.23}$	$\underline{64.66} \pm 0.46$
ASAM	80.66 ± 0.16	83.68 ± 0.12	85.13 ± 0.12	66.44 ± 0.26
LETS-ASAM	$\mathbf{\underline{81.42}} \pm 0.07$	$\underline{84.73} \pm 0.05$	$\underline{85.47} \pm 0.10$	$\mathbf{\underline{66.64}} \pm \mathbf{0.43}$



(a) *ResNet-18*. (b) *WideResNet-28-10*. (c) *PyramidNet-110*.

Fig. 2. Generalization gap *w.r.t.* training epochs on *CIFAR-100*. Best viewed in color.

Figure 2 (resp. Figure 7 in Appendix D.3) shows the generalization gap (i.e., $\mathcal{L}(\mathcal{D}^{ts};\boldsymbol{\theta}_t) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)$) w.r.t. training epochs on *CIFAR-100* (resp. *CIFAR-10*) dataset. As shown, LETS-SAM (resp. LETS-ASAM) has a smaller generalization gap than SAM (resp. ASAM) when the training process nearly converges, verifying that learning the perturbation radius can reduce the generalization gap.

4.2 *ImageNet*

Setups. In this section, we conduct experiments on the *ImageNet* dataset [42], which contains $1,281,167$ images for training and $32,702$ images for testing, by using *ResNet-50* [17]. Following the experimental setup in [9,24], we use a batch size of 512, SGD optimizer with momentum 0.9, weight decay 0.0001, an initial learning rate 0.1 with the cosine learning rate scheduler for model parameters, and an initial learning rate 0.0001 with the exponential learning rate scheduler for ρ. The number of training epochs is 90. Mini-batches of validation data

Table 3. Classification accuracy (%) on the *ImageNet* dataset. The better result in each comparison group is underlined and the best result across all the groups is in **bold**.

ERM	77.11 ± 0.14
ESAM	77.25 ± 0.75
RST	77.38 ± 0.06
AE-SAM	77.43 ± 0.06
LookSAM	77.13 ± 0.09
AE-LookSAM	77.29 ± 0.08
GSAM	77.20 ± 0.00
SAM	77.47 ± 0.12
LETS-SAM	$\underline{\mathbf{77.67}} \pm 0.11$
ASAM	77.17 ± 0.15
LETS-ASAM	$\underline{77.61} \pm 0.10$

are randomly sampled from the training set as in Section 4.1. Experiments are repeated over three random seeds.

Results. The experimental results on *ImageNet* are shown in Table 3. We can find that LETS-SAM performs better than all the baseline methods. Compared with ASAM, LETS-ASAM achieves a higher accuracy, demonstrating the effectiveness of the proposed LETS.

4.3 *IWSLT'14 DE-EN*

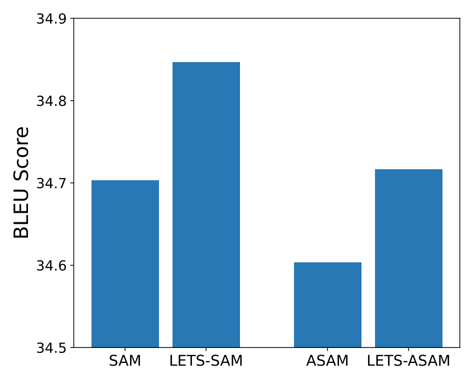


Fig. 3. Experimental results on *IWSLT'14 DE-EN*.

Setups. In this section, we conduct experiments on the *IWSLT'14 DE-EN* dataset, which is a widely used dataset for machine translation. Following experimental setups in [31], we use the widely used machine translation architecture: Transformer architecture [45]. We use the Adam optimizer with $(\beta_1, \beta_2) = (0.9, 0.98)$ and weight decay 0.0001, initial learning rate 0.0005 for model parameters, initial learning rate 0.0001 with the exponential learning rate scheduler for ρ, and a dropout rate 0.3. Label smoothing is adopted with a factor of 0.1. The number of training epochs is 50. Mini-batches of validation data are randomly sampled from the training set as in Section 4.1. Following [31], we use the BLEU score as the evaluation metric (higher is better). Experiments are repeated over three random seeds.

Results. Experimental results on the *IWSLT'14 DE-EN* dataset are shown in Figure 3. We can find that LETS-SAM performs better than SAM and achieves the best performance, while LETS-ASAM also outperforms ASAM, demonstrating the effectiveness of the proposed LETS method.

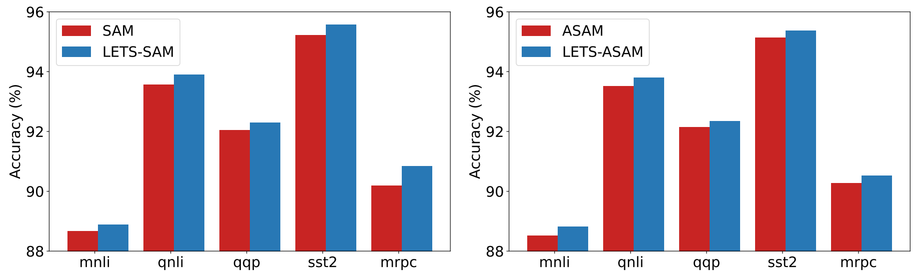
4.4 *GLUE*

Setups. In this section, we conduct experiments on the *GLUE* benchmark [46], which has various corpora and natural language understanding (NLU) tasks. Each task has respective corpora and metrics. The details of the *GLUE* benchmark are summarized in Appendix C. We fine-tune the pre-trained checkpoint of the DeBERTa-base model on the *GLUE* benchmark. Following the experimental setups in [18], we use Adam optimizer with $\epsilon = 10^{-6}$ and $(\beta_1, \beta_2) = (0.9, 0.999)$, linear learning rate scheduler with warmup steps and gradient clipping 1.0. Mini-batches of validation data are randomly sampled from the training set as in Section 4.1. Experiments are repeated over three random seeds.

Results. The experimental results on five NLU tasks of *GLUE* are shown in Figure 4. We can find that LETS-SAM performs better than SAM as shown in Figure 4(a). Compared with ASAM, LETS-ASAM achieves better performance shown in Figure 4(b), demonstrating the effectiveness of the proposed LETS method. Due to page limit, the overall results on the *GLUE* benchmark are reported in Table 9 of Appendix D, which shows the superiority of the proposed LETS method.

4.5 Robustness to Label Noise

Setups. SAM has shown to be robust to label noise in training data [12]. In this section, we follow the experimental setups in [12,24] to study whether learning the perturbation radius can enhance the robustness of SAM. The *ResNet-18* and *ResNet-32* are used. We train the model on a corrupted version of the *CIFAR-10* dataset (with noise levels of 20%, 40%, 60%, and 80%), where the labels of some training data are flipped randomly while the testing set is kept clean. We use batch size 128, SGD optimizer with momentum 0.9 and weight decay 0.0005, initial learning rate 0.1 with the cosine learning rate scheduler for model parameters, and initial learning rate 0.0001 with the exponential learning rate scheduler for ρ. Mini-batches of validation data are randomly sampled from



(a) SAM and LETS-SAM. (b) ASAM and LETS-ASAM.

Fig. 4. Testing accuracy on five datasets from *GLUE*.

the training set as in Section 4.1. The number of training epochs is set to 200. Each experiment is repeated over three random seeds.

Results. The results on *ResNet-18* and *ResNet-32* are shown in Table 4. We can find that LETS-SAM performs the best in all the noise levels. Moreover, LETS-ASAM outperforms ASAM by a large margin. Those results confirm that LETS is an effective method to improve the robustness of SAM and ASAM.

4.6 Robustness to the Initialization of ρ

In this section, we conduct experiments on the *CIFAR-10* and *CIFAR-100* datasets using *ResNet-18* to study the effect of the initialization of ρ (i.e., ρ_0) to the performance of LETS-ASAM. According to results shown in Table 5, we can find that the performance of LETS-ASAM is not so sensitive to a wide range of $\rho_0 \in \{0.01, 0.05, 0.1, 0.5, 1, 1.5, 2\}$. Hence, ρ_0 can be initialized more randomly without compromising the performance of LETS-ASAM, which could imply that learning the perturbation radius is more efficient and effective than using grid search to find the perturbation radius.

4.7 Loss Landscapes

To illustrate the superior performance of the LETS method, we follow [9] to visualize the loss landscapes w.r.t. weight perturbations of SAM, LETS-SAM, ASAM, and LETS-ASAM. Figure 5 (resp. Figure 8 in Appendix D.4) shows the corresponding loss landscapes for different methods built on *ResNet-18* on the *CIFAR-10* (resp. *CIFAR-100*) dataset, respectively. We can find that the model learned by LETS-SAM (resp. LETS-ASAM) has a flatter loss landscape than that of SAM (resp. ASAM). Since the flatness is a measure for generalization, those results could explain why using the proposed LETS method could lead to performance improvement.

Table 4. Classification accuracy (%) on *CIFAR-10* for *ResNet-18* and *ResNet-32* trained with different levels of label noises. The better result in each comparison group is underlined and the best result across all the groups is in **bold**.

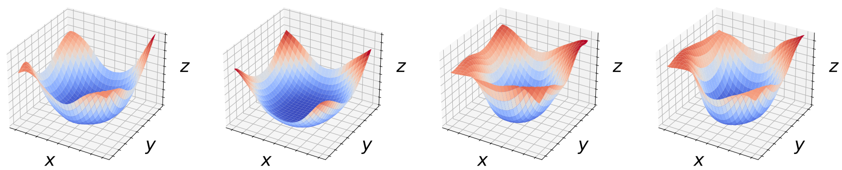
		noise = 20%	noise = 40%	noise = 60%	noise = 80%
ResNet-18	ERM	87.92 ± 0.02	70.82 ± 0.33	49.61 ± 0.39	28.23 ± 0.40
	ESAM	94.19 ± 0.10	91.46 ± 0.49	81.30 ± 0.69	15.00 ± 4.89
	RST	90.62 ± 0.37	77.84 ± 0.56	61.18 ± 0.87	47.32 ± 1.50
	AE-SAM	92.84 ± 0.25	84.17 ± 0.53	73.54 ± 0.50	65.00 ± 2.25
	LookSAM	92.72 ± 0.18	88.04 ± 0.40	72.26 ± 1.75	69.72 ± 1.52
	AE-LookSAM	94.34 ± 0.29	91.58 ± 0.54	87.85 ± 0.23	76.90 ± 0.32
	GSAM	91.72 ± 0.15	87.88 ± 0.50	83.29 ± 0.25	73.16 ± 1.65
	SAM	94.80 ± 0.05	91.50 ± 0.22	88.15 ± 0.23	77.40 ± 0.21
	LETS-SAM	$\underline{\mathbf{95.65}} \pm 0.09$	$\underline{\mathbf{93.84}} \pm 0.19$	$\underline{\mathbf{89.48}} \pm 0.31$	$\underline{\mathbf{77.89}} \pm 0.80$
	ASAM	91.47 ± 0.21	88.28 ± 0.16	83.22 ± 0.38	71.77 ± 1.41
	LETS-ASAM	$\underline{92.77} \pm 0.18$	$\underline{89.72} \pm 0.20$	$\underline{84.94} \pm 0.16$	$\underline{75.00} \pm 0.56$
ResNet-32	ERM	87.43 ± 0.00	70.82 ± 0.98	46.26 ± 0.18	29.00 ± 1.79
	ESAM	93.42 ± 0.50	91.63 ± 0.29	82.73 ± 1.21	10.09 ± 0.10
	RST	89.63 ± 0.26	74.17 ± 0.47	58.40 ± 2.95	59.53 ± 1.63
	AE-SAM	92.87 ± 0.17	82.85 ± 2.16	71.50 ± 0.74	65.43 ± 3.19
	LookSAM	92.49 ± 0.05	86.56 ± 0.92	63.35 ± 0.48	68.01 ± 5.37
	AE-LookSAM	94.70 ± 0.10	91.80 ± 0.87	88.22 ± 0.27	77.03 ± 0.16
	GSAM	92.07 ± 0.13	80.61 ± 0.45	84.08 ± 0.47	72.46 ± 1.85
	SAM	95.08 ± 0.23	91.01 ± 0.41	88.90 ± 0.39	77.32 ± 0.12
	LETS-SAM	$\underline{\mathbf{95.73}} \pm 0.10$	$\underline{\mathbf{93.96}} \pm 0.05$	$\underline{\mathbf{89.71}} \pm 0.17$	$\underline{\mathbf{77.39}} \pm 0.19$
	ASAM	91.61 ± 0.26	88.83 ± 0.76	83.61 ± 0.33	72.32 ± 1.15
	LETS-ASAM	$\underline{92.80} \pm 0.16$	$\underline{89.91} \pm 0.41$	$\underline{85.29} \pm 0.38$	$\underline{75.55} \pm 1.06$

Table 5. Classification accuracy (%) of LETS-ASAM on *CIFAR-10* and *CIFAR-100* for different initializations of ρ.

ρ_0	*CIFAR-10*	*CIFAR-100*
0.01	96.79 ± 0.09	81.37 ± 0.18
0.05	96.73 ± 0.10	81.62 ± 0.14
0.1	96.78 ± 0.08	81.72 ± 0.07
0.5	96.74 ± 0.03	81.51 ± 0.14
1	96.77 ± 0.01	81.36 ± 0.21
1.5	96.79 ± 0.03	81.65 ± 0.06
2	96.79 ± 0.04	81.75 ± 0.15

4.8 Effects of Generalization Metrics

In this section, we conduct experiments on the *CIFAR-10* and *CIFAR-100* datasets using *ResNet-18* to analyze the effects of different generalization metrics (in upper-level problem (2)), including validation loss, the generalization



(a) SAM. (b) LETS-SAM. (c) ASAM. (d) LETS-ASAM.

Fig. 5. Loss landscapes of different methods built on *ResNet-18* for *CIFAR-10*, where x- and y-axes represent two orthogonal weight perturbations, while z-axis represents the loss value.

Table 6. Classification accuracy (%) on *CIFAR-10* and *CIFAR-100* for different generalization metrics on LETS-SAM. The best is in **bold**.

	CIFAR-10	*CIFAR-100*
$\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}^\star(\rho))$	96.61 ± 0.07	80.54 ± 0.06
$\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}^\star(\rho)) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}^\star(\rho))$	96.75 ± 0.18	80.62 ± 0.15
$\frac{1}{2}\left(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}^\star(\rho)) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}^\star(\rho))\right)^2$	$\mathbf{96.81} \pm 0.02$	$\mathbf{80.71} \pm 0.07$

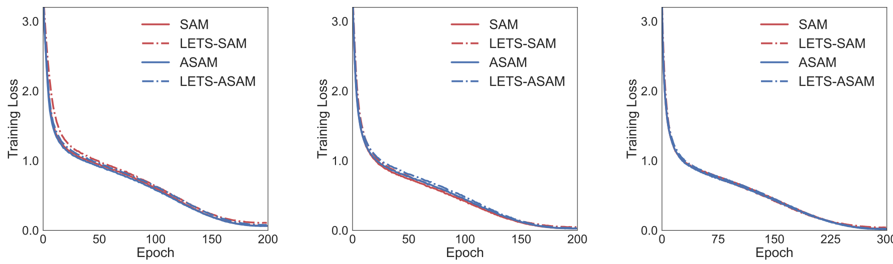
gap, and its square. According to results shown in Table 6, we can find that using $\frac{1}{2}\left(\mathcal{L}(\mathcal{D}^{vl};\boldsymbol{\theta}^\star(\rho)) - \mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}^\star(\rho))\right)^2$ achieves the best performance on both datasets, which suggests that it is a good objective for the upper-level problem.

4.9 Convergence

In this experiment, we study whether the proposed LETS-SAM can converge as suggested in Theorem 4 of Appendix B. Figure 6 (resp. Figure 9 in Appendix D.5) shows the change of the training loss w.r.t. number of epochs for the experiment on *CIFAR-100* (resp. *CIFAR-10*) in Section 4.1. We can find that LETS-SAM and SAM exhibit comparable convergence speeds. Similarly, LETS-ASAM and ASAM empirically enjoy similar convergence rates.

5 Conclusion

In this paper, we study the problem of learning the perturbation radius in sharpness-aware minimization. The proposed LETS method formulates it as a bilevel optimization problem and proposes a gradient-based algorithm to update the model parameters and the radius alternatively. Extensive experiments demonstrate the effectiveness of the proposed LETS method across multiple tasks and various network architectures. The proposed LETS method is general and can be combined with any SAM algorithm, as shown by the success of LETS-ASAM.



(a) *ResNet-18.* (b) *WideResNet-28-10.* (c) *PyramidNet-110.*

Fig. 6. Training loss w.r.t. training epochs on *CIFAR-100*. Best viewed in color.

Acknowledgements

This work is supported by NSFC key grant under grant no. 62136005, NSFC general grant under grant no. 62076118, and Shenzhen fundamental research program JCYJ20210324105000003.

References

1. Allende, G.B., Still, G.: Solving bilevel programs with the KKT-approach. Mathematical Programming (2013)
2. Andriushchenko, M., Flammarion, N.: Towards understanding sharpness-aware minimization. In: ICML (2022)
3. Bahri, D., Mobahi, H., Tay, Y.: Sharpness-aware minimization improves language model generalization. In: ACL (2022)
4. Bisla, D., Wang, J., Choromanska, A.: Low-pass filtering SGD for recovering flat optima in the deep learning optimization landscape. In: AISTATS (2022)
5. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Review (2018)
6. Bracken, J., McGill, J.T.: Mathematical programs with optimization problems in the constraints. Operations Research (1973)
7. Cha, J., Chun, S., Lee, K., Cho, H.C., Park, S., Lee, Y., Park, S.: SWAD: Domain generalization by seeking flat minima. In: NeurIPS (2021)
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
9. Du, J., Yan, H., Feng, J., Zhou, J.T., Zhen, L., Goh, R.S.M., Tan, V.: Efficient sharpness-aware minimization for improved training of neural networks. In: ICLR (2022)
10. Dziugaite, G.K., Roy, D.M.: Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In: UAI (2017)
11. Feurer, M., Hutter, F.: Hyperparameter optimization. In: AutoML (2019)

12. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: ICLR (2021)
13. Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M.: Bilevel programming for hyperparameter optimization and meta-learning. In: ICML (2018)
14. Ghadimi, S., Wang, M.: Approximation methods for bilevel programming. Preprint arXiv:1802.02246 (2018)
15. Han, D., Kim, J., Kim, J.: Deep pyramidal residual networks. In: CVPR (2017)
16. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: ICCV (2015)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
18. He, P., Liu, X., Gao, J., Chen, W.: Deberta: Decoding-enhanced bert with disentangled attention. Preprint arXiv:2006.03654 (2020)
19. Hochreiter, S., Schmidhuber, J.: Simplifying neural nets by discovering flat minima. In: NeurIPS (1994)
20. Hong, M., Wai, H.T., Wang, Z., Yang, Z.: A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. Preprint arXiv:2007.05170 (2020)
21. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. In: UAI (2018)
22. Jiang, W., Kwok, J., Zhang, Y.: Effective meta-regularization by kernelized proximal regularization. In: NeurIPS (2021)
23. Jiang, W., Kwok, J., Zhang, Y.: Subspace learning for effective meta-learning. In: ICML (2022)
24. Jiang, W., Yang, H., Zhang, Y., Kwok, J.: An adaptive policy to employ sharpness-aware minimization. In: ICLR (2023)
25. Jiang, W., Zhang, Y., Kwok, J.: Effective structured prompting by meta-learning and representative verbalizer. In: ICML (2023)
26. Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., Bengio, S.: Fantastic generalization measures and where to find them. In: ICLR (2020)
27. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: ICLR (2017)
28. Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., Srivastava, A.: Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In: ICML (2018)
29. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: ICML (2017)
30. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep. (2009)
31. Kwon, J., Kim, J., Park, H., Choi, I.K.: ASAM: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In: ICML (2021)
32. Liao, R., Xiong, Y., Fetaya, E., Zhang, L., Yoon, K., Pitkow, X., Urtasun, R., Zemel, R.: Reviving and improving recurrent back-propagation. In: ICML (2018)
33. Liu, R., Liu, X., Yuan, X., Zeng, S., Zhang, J.: A value-function-based interior-point method for non-convex bi-level optimization. In: ICML (2021)
34. Liu, S., James, S., Davison, A.J., Johns, E.: Auto-Lambda: Disentangling dynamic task relationships. TMLR (2022)
35. Liu, Y., Mai, S., Chen, X., Hsieh, C.J., You, Y.: Towards efficient and scalable sharpness-aware minimization. In: CVPR (2022)
36. Liu, Y., Mai, S., Cheng, M., Chen, X., Hsieh, C.J., You, Y.: Random sharpness-aware minimization. In: NeurIPS (2022)

37. Mi, P., Shen, L., Ren, T., Zhou, Y., Sun, X., Ji, R., Tao, D.: Make sharpness-aware minimization stronger: A sparsified perturbation approach. In: NeurIPS (2022)
38. Pedregosa, F.: Hyperparameter optimization with approximate gradient. In: ICML (2016)
39. Petzka, H., Kamp, M., Adilova, L., Sminchisescu, C., Boley, M.: Relative flatness and generalization. In: NeurIPS (2021)
40. Qu, Z., Li, X., Duan, R., Liu, Y., Tang, B., Lu, Z.: Generalized federated learning via sharpness aware minimization. In: ICML (2022)
41. Reddi, S.J., Hefny, A., Sra, S., Poczos, B., Smola, A.: Stochastic variance reduction for nonconvex optimization. In: ICML (2016)
42. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. IJCV (2015)
43. Sinha, A., Soun, T., Deb, K.: Using Karush-Kuhn-Tucker proximity measure for solving bilevel optimization problems. Swarm and Evolutionary Computation (2019)
44. Stadie, B., Zhang, L., Ba, J.: Learning intrinsic rewards as a bi-level optimization problem. In: UAI (2020)
45. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
46. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: A multi-task benchmark and analysis platform for natural language understanding. Preprint arXiv:1804.07461 (2018)
47. Ye, F., Lin, B., Yue, Z., Guo, P., Xiao, Q., Zhang, Y.: Multi-objective meta learning. In: NeurIPS (2021)
48. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC (2016)
49. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning (still) requires rethinking generalization. Communications of the ACM (2021)
50. Zhao, Y., Zhang, H., Hu, X.: Penalizing gradient norm for efficiently improving generalization in deep learning. In: ICML (2022)
51. Zhao, Y., Zhang, H., Hu, X.: Randomized sharpness-aware training for boosting computational efficiency in deep learning. Preprint arXiv:2203.09962 (2022)
52. Zhou, M., Liu, T., Li, Y., Lin, D., Zhou, E., Zhao, T.: Toward understanding the importance of noise in training neural networks. In: ICML (2019)
53. Zhuang, J., Gong, B., Yuan, L., Cui, Y., Adam, H., Dvornek, N.C., S. Duncan, J., Liu, T.: Surrogate gap minimization improves sharpness-aware training. In: ICLR (2022)

A Derivation of gradient descent step for updating ρ_{t+1} in LETS-SAM (step 14 in Algorithm 1)

$$\begin{aligned}
\rho_{t+1} &= \rho_t - \beta \nabla_{\rho_t} \frac{1}{2}(\mathcal{L}(\mathcal{D}^{vl}; \boldsymbol{\theta}_{t+1}(\rho_t)) - \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_{t+1}(\rho_t)))^2 \\
\text{(By chain rule)} &= \rho_t - \beta \nabla_{\boldsymbol{\theta}_{t+1}}^\top \frac{1}{2}(\mathcal{L}(\mathcal{D}^{vl}; \boldsymbol{\theta}_{t+1}) - \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_{t+1}))^2 \nabla_{\rho_t} \boldsymbol{\theta}_{t+1} \\
\text{(By chain rule and Eq. (4))} &= \rho_t - \beta \nabla_{\boldsymbol{\theta}_{t+1}}^\top \frac{1}{2}(\mathcal{L}(\mathcal{D}^{vl}; \boldsymbol{\theta}_{t+1}) - \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_{t+1}))^2 \\
&\quad \cdot (-\eta \nabla_{\rho_t} \nabla \mathcal{L}(\left(\mathcal{D}^{tr}; \boldsymbol{\theta}_t + \rho_t \hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})}\right))) \\
\text{(By chain rule)} &= \rho_t + \frac{\beta\eta}{2} \nabla_{\boldsymbol{\theta}_{t+1}}^\top (\mathcal{L}(\mathcal{D}^{vl}; \boldsymbol{\theta}_{t+1}) - \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_{t+1}))^2 \\
&\quad \cdot \nabla^2 \mathcal{L}\left(\mathcal{D}^{tr}; \boldsymbol{\theta}_t + \rho_t \hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})}\right) \hat{\boldsymbol{\epsilon}}_t^{(\mathrm{SAM})}.
\end{aligned}$$

B Theoretical Analysis of Convergence

In this section, we study the convergence of LETS-SAM. The following assumptions on the smoothness and bounded variance of stochastic gradients are standard in the literature on nonconvex optimization [41] and SAM [2,40,24]. The assumption on bounded gradients is also standard in SAM [37].

Assumption 1 (Smoothness) *$\mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta})$ is γ-smooth in $\boldsymbol{\theta}$, i.e., $\|\nabla \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}) - \nabla \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}')\| \leq \gamma \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$.*

Assumption 2 (Bounded variance of stochastic gradients) *$\mathbb{E}_{(\mathbf{x}_i, y_i) \sim \mathcal{D}^{tr}} \| \nabla \ell(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) - \nabla \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta})\|^2 \leq \sigma^2$.*

Assumption 3 (Bounded gradients) *$\|\nabla \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta})\| \leq \zeta$.*

Based on those assumptions, we have the following theorem with the proof in Appendix B.1.

Theorem 4. *Let b be the mini-batch size. If stepsize $\eta = \frac{1}{\gamma\sqrt{T}}$ and $\rho_t \leq \frac{\kappa}{\sqrt{T}}$ (where $\kappa > 0$ is a constant), Algorithm LETS-SAM satisfies*

$$\min_{0 \leq t \leq T-1} \mathbb{E}\|\nabla \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_t)\|^2 \leq \frac{\gamma \mathbb{E}\mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_0)}{2\sqrt{T}} + \frac{\gamma\zeta\kappa + \frac{\sigma^2}{b} + 2\zeta^2 + \gamma\kappa^2}{2\sqrt{T}}, \quad (9)$$

where the expectation is taken over the random training samples.

The $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ convergence rate in Theorem 4 is the same as SAM [2] and its variants [40,24]. Hence, adjusting the perturbation radius does not affect the convergence rate.

B.1 Proof

Proof (Proof of Theorem 4).

To simplify notations, let $\mathbf{g}_{t+\frac{1}{2}} \equiv \nabla\mathcal{L}\left(\mathcal{B}_t^{tr};\boldsymbol{\theta}_t+\rho_t\frac{\nabla\mathcal{L}(\mathcal{B}_t^{tr};\boldsymbol{\theta}_t)}{\|\nabla\mathcal{L}(\mathcal{B}_t^{tr};\boldsymbol{\theta}_t)\|}\right)$ and $\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}} \equiv \nabla\mathcal{L}\left(\mathcal{D}^{tr};\boldsymbol{\theta}_t+\rho_t\frac{\nabla\mathcal{L}(\mathcal{B}_t^{tr};\boldsymbol{\theta}_t)}{\|\nabla\mathcal{L}(\mathcal{B}_t^{tr};\boldsymbol{\theta}_t)\|}\right)$.

By Taylor expansion and Assumption 1, we have

$$
\begin{aligned}
&\mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1})\\
\le&\mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)+\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)(\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_t)+\frac{\gamma}{2}\mathbb{E}\|\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_t\|^2\\
=&\mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)-\eta\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\mathbf{g}_{t+\frac{1}{2}}+\frac{\gamma\eta^2}{2}\mathbb{E}\|\mathbf{g}_{t+\frac{1}{2}}\|^2\\
\le&\mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)-\eta\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\mathbf{g}_{t+\frac{1}{2}}+\gamma\eta^2\mathbb{E}\left(\|\mathbf{g}_{t+\frac{1}{2}}-\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\|^2+\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\|^2\right)\\
=&\mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)-\eta\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\mathbf{g}_{t+\frac{1}{2}}+\frac{\gamma\eta^2\sigma^2}{b}+\gamma\eta^2\mathbb{E}\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\|^2,
\end{aligned}
\tag{10}
$$

where the second inequality follows from the property $\|\mathbf{a}\|^2\le 2(\|\mathbf{a}-\mathbf{b}\|^2+\|\mathbf{b}\|^2)$ for any two vectors $\mathbf{a}$ and $\mathbf{b}$. We bound the second and last terms separately in the following.

Claim 1: $-\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\mathbf{g}_{t+\frac{1}{2}}\le-\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2+\rho_t\gamma\zeta$.

By triangle inequality, we have

$$
\begin{aligned}
&\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\mathbf{g}_{t+\frac{1}{2}}\\
&=\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\left(\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)-\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)+\mathbf{g}_{t+\frac{1}{2}}\right)\\
&=\mathbb{E}\|\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2+\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\left(\mathbf{g}_{t+\frac{1}{2}}-\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\right)\\
&\ge\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2-\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|\|\mathbb{E}_{\mathcal{B}_t^{tr}}\left(\mathbf{g}_{t+\frac{1}{2}}-\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\right)\|\\
&\ge\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2-\rho_t\gamma\zeta,
\end{aligned}
$$

where we use Assumption 3 and $\|\mathbb{E}_{\mathcal{B}_t^{tr}}(\mathbf{g}_{t+\frac{1}{2}}-\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t))\|=\|\mathbb{E}_{\mathcal{B}_t^{tr}}(\mathbf{g}_{t+\frac{1}{2}}-\nabla\mathcal{L}(\mathcal{B}_t^{tr};\boldsymbol{\theta}_t))\|\le\gamma\rho_t\|\mathbb{E}_{\mathcal{B}_t}\frac{\nabla\mathcal{L}(\mathcal{B}_t^{tr};\boldsymbol{\theta}_t)}{\|\nabla\mathcal{L}(\mathcal{B}_t^{tr};\boldsymbol{\theta}_t)\|}\|\le\gamma\rho_t$ to obtain the last inequality.

Claim 2: $\mathbb{E}\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\|^2\le 2\zeta^2+\gamma\rho_t^2-\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2$.

Using the property $\|\mathbf{a}\|^2=\|\mathbf{a}-\mathbf{b}\|^2-\|\mathbf{b}\|^2+2\mathbf{a}^\top\mathbf{b}$, it follows that

$$
\begin{aligned}
\mathbb{E}\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\|^2=&\mathbb{E}\left(\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}-\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2-\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2+2\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\right)\\
=&2\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}+\mathbb{E}\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}-\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2-\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2.
\end{aligned}
\tag{11}
$$

For the first term, it follows that

$$
\mathbb{E}\nabla^\top\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\le\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\|\le\zeta^2,
\tag{12}
$$

where we have used Assumption 3 to obtain the last inequality.

For the second term in Eq. (11), by Assumption 1, it follows that

$$\mathbb{E}\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}} - \nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2 \leq \gamma\|\boldsymbol{\theta}_t + \rho_t \frac{\nabla\mathcal{L}(\mathcal{B}^{tr}_t;\boldsymbol{\theta}_t)}{\|\nabla\mathcal{L}(\mathcal{B}^{tr}_t;\boldsymbol{\theta}_t)\|} - \boldsymbol{\theta}_t\|^2 = \gamma\rho_t^2. \tag{13}$$

Substituting (12) and (13) into (11), we have

$$\mathbb{E}\|\mathbf{g}^{\mathcal{D}}_{t+\frac{1}{2}}\|^2 \leq 2\zeta^2 + \gamma\rho_t^2 - \mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2.$$

Using Claims 1 and 2, for (10), we have

$$\begin{aligned}
&\mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_{t+1}) \\
&\leq \mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t) - \eta(1+\gamma\eta)\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2 + \eta\rho_t\gamma\zeta + \frac{\gamma\eta^2\sigma^2}{b} + \gamma\eta^2(2\zeta^2 + \gamma\rho_t^2) \\
&\leq \mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t) - 2\eta\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2 + \frac{\gamma\zeta\kappa + \frac{\sigma^2}{b} + 2\zeta^2 + \gamma\kappa^2}{\gamma T}.
\end{aligned}$$

Summing over both sides from $t = 1$ to T, and rearranging it, we have

$$2\eta\sum_{t=1}^{T}\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2 \leq \mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_0) + \frac{\gamma\zeta\kappa + \frac{\sigma^2}{b} + 2\zeta^2 + \gamma\kappa^2}{\gamma}$$

Hence,

$$\begin{aligned}
\min_{0\leq t\leq T-1}\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2 &\leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_t)\|^2 \\
&\leq \frac{\gamma\mathbb{E}\mathcal{L}(\mathcal{D}^{tr};\boldsymbol{\theta}_0)}{2\sqrt{T}} + \frac{\gamma\zeta\kappa + \frac{\sigma^2}{b} + 2\zeta^2 + \gamma\kappa^2}{2\sqrt{T}},
\end{aligned}$$

and we finish the proof.

C *GLUE* Benchmark

The General Language Understanding Evaluation (*GLUE*) benchmark is a collection of resources for training, evaluating, and analyzing natural language understanding systems. *GLUE* consists of nine language understanding corpora including natural language inference, question answering, paraphrase detection, sentiment analysis, linguistic acceptability, and text similarity. *GLUE* covers a diverse range of dataset sizes, text genres, and degrees of difficulty. The task and metric of each corpus are shown in Table 7.

D Additional Experimental Results

D.1 *IWSLT'14 DE-EN*

The BLEU scores on *IWSLT'14 DE-EN* are shown in Table 8. We can find that LETS-SAM is better than SAM, while LETS-ASAM outperforms ASAM, demonstrating the effectiveness of the proposed LETS method.

Table 7. Details of the *GLUE* benchmark.

Corpus	Task	Metric
CoLA	Acceptability	Matthews Correlation
SST	Sentiment	Accuracy
MRPC	Paraphrase	Accuracy/F1
STSB	Similarity	Pearson/Spearmanr
QQP	Paraphrase	Accuracy/F1
MNLI	NLI	Accuracy
QNLI	QA/NLI	Accuracy
RTE	NLI	Accuracy

Table 8. BLEU scores on *IWSLT'14 DE-EN*. The best result in each comparison group is underlined and the best result across all the groups is in **bold**.

ERM	34.60 ± 0.04
SAM	34.70 ± 0.07
LETS-SAM	$\underline{\mathbf{34.85}} \pm 0.02$
ASAM	34.60 ± 0.13
LETS-ASAM	$\underline{34.72} \pm 0.10$

Table 9. Experimental results on the GLUE development set. The better result in each comparison group is underlined and the best result across all the groups is in **bold**.

	CoLA	SST	MRPC	STSB	QQP	MNLI	QNLI	RTE
ERM	64.37 ± 0.27	94.99 ± 0.07	89.95 ± 0.42	91.17 ± 0.42	91.77 ± 0.08	88.48 ± 0.10	93.41 ± 0.37	63.54 ± 9.93
SAM	64.40 ± 0.40	95.22 ± 0.17	90.36 ± 0.57	91.22 ± 0.19	92.04 ± 0.17	88.67 ± 0.24	93.58 ± 0.05	56.44 ± 4.71
LETS-SAM	$\underline{65.17} \pm 0.81$	$\underline{\mathbf{95.57}} \pm 0.06$	$\underline{\mathbf{90.85}} \pm 0.14$	$\underline{\mathbf{91.51}} \pm 0.06$	$\underline{92.30} \pm 0.03$	$\underline{\mathbf{88.88}} \pm 0.05$	$\underline{\mathbf{93.90}} \pm 0.12$	$\underline{74.01} \pm 2.96$
ASAM	63.89 ± 0.63	95.26 ± 0.24	90.28 ± 0.28	91.23 ± 0.23	92.14 ± 0.11	88.51 ± 0.11	93.51 ± 0.04	69.80 ± 6.18
LETS-ASAM	$\underline{\mathbf{65.81}} \pm 0.44$	$\underline{95.37} \pm 0.06$	$\underline{90.52} \pm 0.37$	$\underline{91.48} \pm 0.15$	$\underline{\mathbf{92.34}} \pm 0.09$	$\underline{88.82} \pm 0.20$	$\underline{93.80} \pm 0.04$	$\underline{\mathbf{74.85}} \pm 4.15$

D.2 *GLUE*

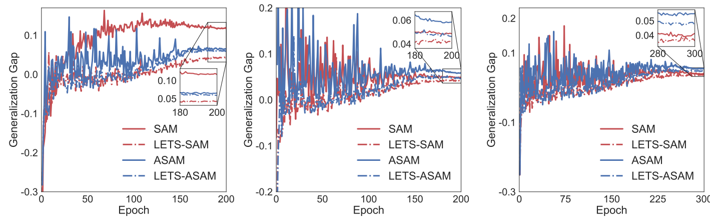
The results on eight corpora of *GLUE* are shown in Table 9. We can find that LETS-SAM is consistently better than SAM. Compared with ASAM, LETS-ASAM always has better performance, demonstrating the effectiveness of the proposed LETS. Furthermore, on *SST*, *MRPC*, *STSB*, *MNLI*, and *QNLI* datasets, LETS-SAM achieves the highest accuracy, while LETS-SAM is the best on the *CoLA*, *QQP* and *RTE* datasets.

D.3 Generalization Gap

Figure 7 show the generalization gap (i.e., $\mathcal{L}(\mathcal{D}^{ts}; \boldsymbol{\theta}_t) - \mathcal{L}(\mathcal{D}^{tr}; \boldsymbol{\theta}_t)$) w.r.t. training epochs on *CIFAR-10* dataset. As shown, LETS-SAM (resp. LETS-ASAM) has a smaller generalization gap than SAM (resp. ASAM) when the training process nearly converges, verifying that learning the perturbation radius can reduce the generalization gap.

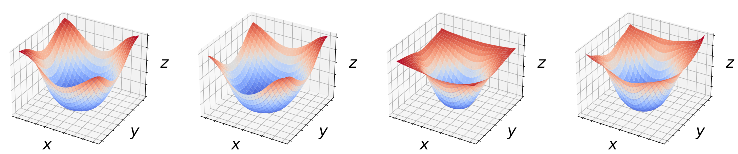
D.4 Loss Landscapes

To illustrate the superior performance of the LETS method, we follow [9] to visualize the loss landscapes w.r.t. weight perturbations of SAM, LETS-SAM,



(a) *ResNet-18*. (b) *WideResNet-28-10*. (c) *PyramidNet-110*.

Fig. 7. Generalization gap w.r.t. training epochs on *CIFAR-10*. Best viewed in color.



(a) SAM. (b) LETS-SAM. (c) ASAM. (d) LETS-ASAM.

Fig. 8. Loss landscapes of different methods built on *ResNet-18* for *CIFAR-100*, where x- and y-axes represent two orthogonal weight perturbations, while z-axis represents the loss value.

ASAM, and LETS-ASAM. Figure 8 shows the corresponding loss landscapes for different methods built on *ResNet-18* on the *CIFAR-100* dataset. We can find that the model learned by LETS-SAM (resp. LETS-ASAM) has a flatter loss landscape than that of SAM (resp. ASAM). Since the flatness is a measure for generalization, those results could explain why using the proposed LETS method could lead to performance improvement.

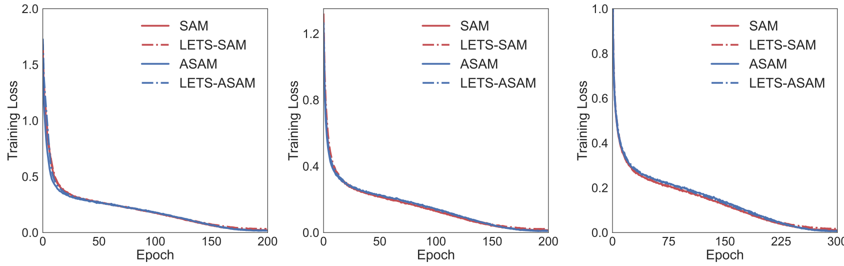
D.5 Convergence

In this experiment, we study whether the proposed LETS-SAM can converge as suggested in Theorem 4. Figure 9 shows the change of the training loss w.r.t. the number of epochs for the experiment on *CIFAR-10* in Section 4.1. As can be seen, LETS-SAM and SAM empirically enjoy similar convergence rates. Moreover, LETS-ASAM exhibits a similar convergence behavior to ASAM.

E Implementation Details

Experiments on *CIFAR-10* and *CIFAR-100* are conducted on NVIDIA GeForce RTX 3090 GPUs. Experiments on *ImageNet*, *IWSLT'14 DE-EN* and *GLUE* are conducted on NVIDIA A100 GPUs.

Table 10 presents the hyperparameters employed in CV tasks. For CV tasks, following experimental setups in [12,31,24], both SAM (resp. ASAM) and LETS-SAM (resp. LETS-ASAM) use the same hyperparameter configurations. Tables 11 and 12 present the hyperparameters employed on the *GLUE* and *IWSLT'14*



(a) *ResNet-18*. (b) *WideResNet-28-10*. (c) *PyramidNet-110*

Fig. 9. Training loss w.r.t. training epochs on *CIFAR-10*. Best viewed in color.

DE-EN datasets, respectively. For the *GLUE* dataset, we adopt the experimental setups in [18]. For the *IWSLT'14 DE-EN* dataset, we follow the experimental setups used in [31].

Table 10. Hyperparameters for CV datasets.

	LETS-SAM	LETS-ASAM
model parameters		
optimizer	SGD optimizer for *ResNet-18*, *WideResNet-28-10*, *PyramidNet-110* and *ResNet-50*, Adam optimizer for *ViT-S16*	
initial learning rate	0.1 for *ResNet-18*, *WideResNet-28-10*, *PyramidNet-110* and *ResNet-50*, 0.0001 for *ViT-S16*	
initialization	*He Initialization* [16]	*He Initialization* [16]
weight decay	0.0005 for *ResNet-18*, *WideResNet-28-10* and *PyramidNet-110*, 0.0001 for *ResNet-50*	
momentum	0.9	0.9
learning rate schedule	Cosine	Cosine
perturbation radius ρ		
optimizer	Adam optimizer	Adam optimizer
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999
initial learning rate	0.0001	0.0001
learning rate schedule	Exponential	Exponential
batch size	128 for *ResNet-18*, *WideResNet-28-10* and *PyramidNet-110*, 512 for *ResNet-50*, 256 for *ViT-S16*	
ξ	-	0.01
#epoch	200 for *ResNet-18* and *WideResNet-28-10*, 300 for *PyramidNet-110*, 90 for *ResNet-50*, 1200 for *ViT-S16*	

Table 11. Hyperparameters for *GLUE*.

	SAM	LETS-SAM	ASAM	LETS-ASAM
model parameters				
dropout rate	$\{0, 0.1, 0.15\}$	$\{0, 0.1, 0.15\}$	$\{0, 0.1, 0.15\}$	$\{0, 0.1, 0.15\}$
warmup steps	$\{50, 100, 500, 1000\}$	$\{50, 100, 500, 1000\}$	$\{50, 100, 500, 1000\}$	$\{50, 100, 500, 1000\}$
learning rate	$\{0, 0.1, 0.15\}$	$\{0, 0.1, 0.15\}$	$\{0, 0.1, 0.15\}$	$\{0, 0.1, 0.15\}$
learning rate schedule	Linear	Linear	Linear	Linear
Adam ϵ	$1e-6$	$1e-6$	$1e-6$	$1e-6$
Adam β_1	0.9	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999	0.999
gradient clipping	1.0	1.0	1.0	1.0
perturbation radius ρ				
optimizer	-	Adam optimizer	-	Adam optimizer
Adam β_1	-	0.9	-	0.9
Adam β_2	-	0.999	-	0.999
initial learning rate	-	$\{0.0001, 0.0005, 0.001\}$	-	$\{0.0001, 0.0005, 0.001\}$
learning rate schedule		Exponential		Exponential
batch size	$\{16, 32, 48, 64\}$	$\{16, 32, 48, 64\}$	$\{16, 32, 48, 64\}$	$\{16, 32, 48, 64\}$
ξ	-	-	0.01	0.01

Table 12. Hyperparameters for *IWSLT'14 DE-EN*.

	SAM	LETS-SAM	ASAM	LETS-ASAM
model parameters				
dropout rate	0.3	0.3	0.3	0.3
learning rate	0.0005	0.0005	0.0005	0.0005
Adam β_1	0.9	0.9	0.9	0.9
Adam β_2	0.98	0.98	0.98	0.98
weight decay	0.0001	0.0001	0.0001	0.0001
perturbation radius ρ				
optimizer	-	Adam optimizer	-	Adam optimizer
Adam β_1	-	0.9	-	0.9
Adam β_2	-	0.999	-	0.999
initial learning rate	-	0.00001	-	0.00001
learning rate schedule	-	Exponential	-	Exponential
ξ	-	-	0.01	0.01
#epoch	50	50	50	50