Sharpness-Aware Minimization with Dynamic Reweighting

Wenxuan Zhou[1], **Fangyu Liu**[2], **Huan Zhang**[3], **Muhao Chen**[1]
[1]University of Southern California; [2]University of Cambridge;
[3]Carnegie Mellon University
{zhouwenx, muhaoche}@usc.edu; fl399@cam.ac.uk;
huan@huan-zhang.com

Abstract

Deep neural networks are often overparameterized and may not easily achieve model generalization. Adversarial training has shown effectiveness in improving generalization by regularizing the change of loss on top of adversarially chosen perturbations. The recently proposed sharpness-aware minimization (SAM) algorithm conducts adversarial weight perturbation, encouraging the model to converge to a flat minima. SAM finds a *common* adversarial weight perturbation *per-batch*. Although *per-instance* adversarial weight perturbations are stronger adversaries and can potentially lead to better generalization performance, their computational cost is very high and thus it is impossible to use per-instance perturbations efficiently in SAM. In this paper, we tackle this efficiency bottleneck and propose sharpness-aware minimization with dynamic reweighting (δ-SAM). Our theoretical analysis motivates that it is possible to approach the stronger, per-instance adversarial weight perturbations using reweighted per-batch weight perturbations. δ-SAM dynamically reweights perturbation within each batch according to the theoretically principled weighting factors, serving as a good approximation to per-instance perturbation. Experiments on various natural language understanding tasks demonstrate the effectiveness of δ-SAM.

1 Introduction

Although deep neural networks (DNNs) have demonstrated promising results in various fields such as natural language understanding (Devlin et al., 2019) and computer vision (Krizhevsky et al., 2012), they are often overparameterized and can easily overfit the training data (Zhang et al., 2021). Adversarial training has been proven effective in improving both model generalization (Zhu et al., 2019; Zhang et al., 2020a) and adversarial robustness (Madry et al., 2018; Zhang et al., 2019). A general approach for adversarial training has been first to augment the inputs with small perturbations

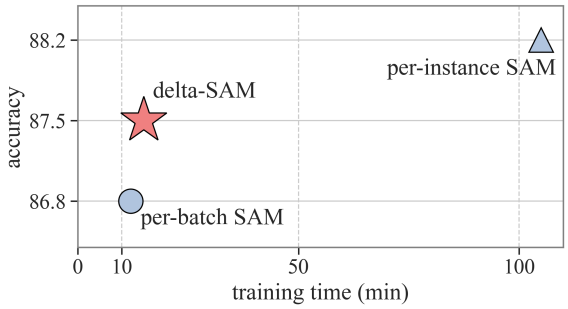

Figure 1: δ-SAM performs close to the computation-intensive per-instance weight perturbation while adding only marginal computation overhead to the standard per-batch SAM. Results shown are from the MRPC dataset. See §5.5 for more detailed results.

that lead to the maximum possible change of loss, and then optimize the model parameters to the direction where the changed amount is minimized.

Besides perturbing inputs, a recent work of sharpness-aware minimization (SAM; Foret et al. 2020) has further proposed to adversarially perturb model weights. Such a method works by first adversarially calculating a weight perturbation that maximizes the empirical risk and then minimizing the empirical risk on the perturbed model. This method demonstrates improved model generalizations across different datasets and models. In principle, each instance in a batch has its own worst-case weight perturbation and the weight perturbations of different instances need to be calculated separately and cannot be done in a single forward/backward pass. This leads to a significant increase in computational and memory cost. To allow a feasible algorithm, SAM approximates *per-instance* perturbations by a single *per-batch* perturbation, where the weight perturbation is calculated on the averaged loss of the batch and shared by all instances in the batch. However, as the per-batch perturbation represents the average of perturbations yielded by different instances, it is a weaker adversary compared to per-instance perturbations, and may hinder

the effectiveness of SAM.

In this paper, we study how to efficiently approximate per-instance weight perturbation for sharpness-aware minimization, while maintaining a similar computational cost to per-batch perturbation. We first theoretically analyze the gradient posed by the optimization of per-instance perturbation, and find that it can be effectively approximated with a *weighted-batch* perturbation under some assumptions, where the instances with a larger rate of gradient change are up-weighted. Based on this motivation, we propose sharpness-aware minimization with dynamic reweighting (δ-SAM). Specifically, we first estimate the Hessian and gradient norm of each instance by perturbing the loss with a random Gaussian noise on model weights. Next, δ-SAM dynamically reweights the loss within each batch of training instances, and then calculate a shared weight perturbation that maximizes the reweighted batch loss. Finally, we update the perturbed model on the original (unweighted) batch. Compared to SAM, δ-SAM only requires extra computation cost in estimation of the rate of gradient change, which can be efficiently performed using three additional forward passes.

We evaluate δ-SAM on finetuning pretrained language models (PLMs). Experiments on standard GLUE benchmark (Wang et al., 2018), self-supervised Semantic Textual Similarity (STS), and abstractive summarization tasks show that besides significantly outperforming base models, δ-SAM also consistently outperforms SAM with only 18% extra computational cost in average.

The main contributions of this paper are threefold. First, we analyze the training objective of per-instance weight perturbation and find that under some assumptions, it can be approximated by a weighted-batch perturbation, where instances are efficiently reweighted according to their caused rates of gradient changes. Second, we propose to use random perturbations as estimations to efficiently realize the weighting scheme. Third, we evaluate δ-SAM on a diverse set of datasets and find consistent improvements across the board.

2 Related Work

Model Generalization. Deep neural networks are often overparameterized and may suffer from poor generalization (Zhang et al., 2021). A lot of efforts have been devoted to improve the generalization of neural models, leading to methods including data augmentation (Sennrich et al., 2016; Wei and Zou, 2019; Sun et al., 2020; Kumar et al., 2020; Thakur et al., 2021), regularization (Loshchilov and Hutter, 2018; Xuhong et al., 2018; Liang et al., 2021), and improved optimization processes (Izmailov et al., 2018; Mobahi et al., 2020; Heo et al., 2021). These methods consider different aspects of generalization and may be combined to achieve better performance. Among them, adversarial training (Goodfellow et al., 2014) has demonstrated its effectiveness in improving model generalization without the need of any extra data or external knowledge, and has been widely attempted to enhance NLP models (Zhu et al., 2019; Jiang et al., 2020a; Pereira et al., 2021; Li and Qiu, 2021). Adversarial training works by adversarially perturbing the input embedding and either minimize the adversarial risk or regularizes the change of risk to be small. Specifically, FreeLB (Zhu et al., 2019) uses projected gradient descent (PGD; Madry et al. 2018) to generate adversarial perturbations on input embedding, and recycles the computed gradients when updating model parameters in adjacent steps (Shafahi et al., 2019) to reduce the computational costs. TAT (Pereira et al., 2021) improves FreeLB by prioritizing the most frequently mispredicted classes in perturbation calculation. TAVAT (Li and Qiu, 2021) uses a token-level accumulated perturbation vocabulary to guide the initialization in PGD. However, these works only consider the robustness on input feature representation, while we consider the robustness of all model weights.

Sharpness-Aware Minimization. Foret et al. (2020) leverage the correlation between flat minima and better model generalization and propose SAM for training deep neural models that are robust to adversarial weight perturbations. It has demonstrated effectiveness in tasks on both vision (Chen et al., 2022; Zheng et al., 2021) and language (Bahri et al., 2022) modalities. Several variants of SAM have been proposed to improve its efficiency or effectiveness. For efficiency, Brock et al. (2021) propose to speed up SAM by perturbing fewer instances in the batch; Du et al. (2022) introduce stochastic weight perturbation and sharpness-sensitive data selection to reduce the computational overhead. For effectiveness; Kwon et al. (2021) propose to adaptively set the SAM's radius such that it is invariant to parameters' scales; Zhuang et al. (2022) introduce a gradient ascent optimization step in the perturbed model's orthogonal direction to achieve better flat-

Algorithm 1: SAM and δ-SAM

Input: model $f_{\boldsymbol{w}}$, training set $\mathcal{S} \triangleq \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{|\mathcal{S}|}$, loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, batch size N, neighborhood size $\rho \in \mathbb{R}^+$, optimizer h.
Output: a flat solution $\hat{\boldsymbol{w}}$.
Initialize model weights $\boldsymbol{w}$.
while *not converge* **do**
  Sample a batch $\mathcal{B} = \{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^{N}$.
  For δ-SAM:
  Reweigh $\mathcal{B}$ by Eq. 5.
  Compute gradient $\nabla l_{\mathcal{B}}(\boldsymbol{w})$ of the (reweighted) batch's empirical risk.
  Perturb the model weights by $\boldsymbol{\epsilon}^* = \rho \nabla l_{\mathcal{B}}(\boldsymbol{w}) / \left\|\nabla l_{\mathcal{B}}(\boldsymbol{w})\right\|_2$.
  Update $\boldsymbol{w}$ w.r.t. the unweighted empirical risk $\frac{1}{N}\sum_{j=1}^{N} l_j(\boldsymbol{w} + \boldsymbol{\epsilon}^*)$ with the optimizer h.

ness. In contrast to the aforementioned studies, this paper, for the first time, tackles how to narrow the gap of per-instance and per-batch weight perturbation. Accordingly, we propose an efficient approximation to per-instance weight perturbation, which shows improved results on several NLP tasks while does not bring much computational overheads.

3 Sharpness-Aware Minimization (SAM)

In this section, we briefly review the principle of SAM and discuss its limitations.

Literature has observed a direct correlation between flat minima and better model generalization, both empirically and theoretically (Keskar et al., 2016; Dziugaite and Roy, 2017; Li et al., 2018; Jiang et al., 2020b). To find a flat loss landscape, SAM (Foret et al., 2020) adversarially perturbs the model weights and optimizes the following min-max objective on a batch of size N:

$$\min_{\boldsymbol{w}} \max_{\boldsymbol{\epsilon}:\|\boldsymbol{\epsilon}\|_2 \le \rho} \frac{1}{N} \sum_{i=1}^{N} l_i(\boldsymbol{w} + \boldsymbol{\epsilon}), \tag{1}$$

where given the model weights $\boldsymbol{w}$, the inner maximization seeks for a perturbation $\boldsymbol{\epsilon}$ with L_2-norm $\le \rho$ that maximizes the empirical risk, and the outer minimization minimizes the empirical risk of the perturbed model. This training objective aims at finding model parameters whose neighborhood has a uniformly low training loss. As finding the exact solution to $\boldsymbol{\epsilon}$ is NP-hard, SAM estimates the solution $\boldsymbol{\epsilon}^*$ of the inner maximization with a single-step gradient descent on the empirical risk of the batch:

$$\begin{aligned} l(\boldsymbol{w}) &= \frac{1}{N}\sum_{i=1}^{N} l_i(\boldsymbol{w}), \\ \boldsymbol{\epsilon}^* &\approx \arg\max_{\boldsymbol{\epsilon}:\|\boldsymbol{\epsilon}\|_2 \le \rho} l(\boldsymbol{w}) + \boldsymbol{\epsilon}^\intercal \nabla l(\boldsymbol{w}) \\ &= \rho \nabla l(\boldsymbol{w}) / \left\|\nabla l(\boldsymbol{w})\right\|_2 . \end{aligned}$$

The outer minimization can be performed with a standalone optimizer (e.g., Adam; Kingma and Ba 2015). SAM roughly doubles the computational cost of training the model, requiring two forward and two backward passes for each batch. The SAM algorithm is outlined in Alg. 1.

Besides perturbing by batches, weight perturbation can also be performed on individual instances:

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} \max_{\boldsymbol{\epsilon}_i:\|\boldsymbol{\epsilon}_i\|_2 \le \rho} l_i(\boldsymbol{w} + \boldsymbol{\epsilon}_i), \tag{2}$$

where $\boldsymbol{\epsilon}_i$ is calculated by single-step gradient descent on individual instances. This approach is similar to many adversarial training methods in NLP, such as VAT (Miyato et al., 2018) and FreeLB (Zhu et al., 2019), except that the perturbation is computed on model weights instead of input embedding only. We refer to the objectives of Eq. 1 and Eq. 2 as *per-batch weight perturbation* and *per-instance weight perturbation*, respectively. It is noted in the same paper by Foret et al. (2020) that per-instance weight perturbation produces a smaller test error and is a better predictor of model generalization.

Despite its effectiveness, per-instance weight perturbation increases the computational and memory cost significantly, requiring $2N$ forward and $2N$ backward passes for a batch of size N. Because per-instance weight perturbation modifies all model weights independently, the perturbation for each individual instance needs to be computed on a distinct model copy. Therefore, per-instance weight perturbation can be computationally unaffordable for large-scale training.

4 SAM with Dynamic Reweighting

In this paper, we seek to improve SAM with a better adversary on weight perturbations. As the per-batch weight perturbation adopted by SAM weakens the adversarial training, we propose a simple yet effective modification of SAM, δ-SAM (SAM with dynamic reweighting), that can approximate

per-instance weight perturbation without requiring much additional computational cost. Our reweighting approach is motivated by a theoretical analysis on approximating the per-instance weight perturbation to justify its superior efficiency. Based on this motivation, we then illustrate how δ-SAM is realized in implementation.

4.1 Theoretical Motivations

In this subsection, we motivate our dynamic reweighting approach by formally analyzing the training objective posed by per-instance perturbation, and show that it can be approximated with a weighted-batch perturbation, which motivates our δ-SAM algorithm.

Preliminary. We motivate our approach from the perspective of sharpness in SAM, which quantifies the flatness of loss landscape as the *increase of loss* in the neighborhood region of model weights. The sharpness of per-batch and per-instance weight perturbations are defined as:

$$\mathcal{R}_{\text{batch}} = \max_{\boldsymbol{\epsilon}:\|\boldsymbol{\epsilon}\|_2\leq\rho} \frac{1}{N}\sum_{i=1}^{N}\left(l_i(\boldsymbol{w}+\boldsymbol{\epsilon}) - l_i(\boldsymbol{w})\right),$$

$$\mathcal{R}_{\text{inst}} = \frac{1}{N}\sum_{i=1}^{N}\max_{\boldsymbol{\epsilon}_i:\|\boldsymbol{\epsilon}_i\|_2\leq\rho}\left(l_i(\boldsymbol{w}+\boldsymbol{\epsilon}_i) - l_i(\boldsymbol{w})\right).$$

Due to non-shared $\boldsymbol{\epsilon}_i$, $\mathcal{R}_{\text{inst}} \geq \mathcal{R}_{\text{batch}}$, suggesting stronger regularization effects of $\mathcal{R}_{\text{inst}}$. However, $\mathcal{R}_{\text{inst}}$ is expensive to compute, since ϵ_i in the N inner maximization problems must be calculated by gradient descent on N individual instances, for which $O(N)$ backward passes through the network are needed. In the analysis below, we show how to approximate the stronger $\mathcal{R}_{\text{inst}}$ with a weighted per-batch weight perturbation.

We start by considering the second-order expansion of a general empirical risk l_i for instance i:

$$l_i(\boldsymbol{w}+\boldsymbol{\epsilon}) = l_i(\boldsymbol{w}) + \nabla l_i(\boldsymbol{w})^\intercal\boldsymbol{\epsilon} + \frac{1}{2}\boldsymbol{\epsilon}^\intercal\boldsymbol{H}_i(\boldsymbol{w})\boldsymbol{\epsilon}.$$

To allow a tractable theoretical analysis, we assume that the Hessian is a low-rank, positive definite matrix $\boldsymbol{H}_i(\boldsymbol{w}) = a_i\nabla l_i(\boldsymbol{w})\nabla l_i(\boldsymbol{w})^\intercal, (a_i > 0)$. Then, we obtain the perturbations in $\mathcal{R}_{\text{batch}}$ and $\mathcal{R}_{\text{inst}}$ under second-order approximation in closed-form with one-step gradient descent, to align with the practice of SAM:

$$\boldsymbol{\epsilon} = \rho\nabla l(\boldsymbol{w}) / \left\|\nabla l(\boldsymbol{w})\right\|_2,$$
$$\boldsymbol{\epsilon}_i = \rho\nabla l_i(\boldsymbol{w}) / \left\|\nabla l_i(\boldsymbol{w})\right\|_2,$$

where $\nabla l(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\nabla l_i(\boldsymbol{w})$ is the average gradient of the batch.

Training Objective. After $\mathcal{R}_{\text{batch}}$ or $\mathcal{R}_{\text{inst}}$ is obtained, SAM will compute $\frac{\partial}{\partial\boldsymbol{w}}\mathcal{R}_{\text{batch}}$ or $\frac{\partial}{\partial\boldsymbol{w}}\mathcal{R}_{\text{inst}}$ and update model weights to minimize the loss. To aim for a more effective perturbation, we seek to align with the gradient $\mathcal{R}_{\text{inst}}$, which determines how model weights will be updated under the strong per-instance adversary. We hope to update the model weights in a "similar" manner as the per-instance adversary, while not explicitly computing the expensive term $\mathcal{R}_{\text{inst}}$. Here "similar" means the cosine similarity between the gradient of $\mathcal{R}_{\text{inst}}$ and our new objective is positive.[1]

We thereby first derive the gradient of per-instance weight perturbation. Specifically, after calculating $\boldsymbol{\epsilon}_i$ in the inner maximization step ($\boldsymbol{\epsilon}_i$ is not differentiated in outer minimization), the gradient of per-instance perturbation is:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{w}}\mathcal{R}_{\text{inst}} &= \frac{1}{N}\sum_{i=1}^{N}\left(\nabla l_i(\boldsymbol{w}+\boldsymbol{\epsilon}_i) - \nabla l_i(\boldsymbol{w})\right)\\ &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{H}_i(\boldsymbol{w})\boldsymbol{\epsilon}_i\\ &= \frac{1}{N}\sum_{i=1}^{N}\rho a_i\left\|\nabla l_i(\boldsymbol{w})\right\|_2\nabla l_i(\boldsymbol{w}). \quad (3)\end{aligned}$$

In this paper, we aim at finding a per-batch perturbation $\boldsymbol{\epsilon}'$ that produces the gradient whose direction is aligned with the per-instance gradient, so model weights will be updated similarly using gradient based optimizers. Specifically, for a shared perturbation $\boldsymbol{\epsilon}'$ and $\mathcal{R}$ defined as $\mathcal{R} := \frac{1}{N}\sum_{i=1}^{N}\left(l_i(\boldsymbol{w}+\boldsymbol{\epsilon}') - l_i(\boldsymbol{w})\right)$, its gradient is:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{w}}\mathcal{R} &= \frac{1}{N}\sum_{i=1}^{N}\left(\nabla l_i(\boldsymbol{w}+\boldsymbol{\epsilon}') - \nabla l_i(\boldsymbol{w})\right)\\ &= \left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{H}_i(\boldsymbol{w})\right)\boldsymbol{\epsilon}'.\end{aligned}$$

Compared to the ordinary $\mathcal{R}_{\text{batch}}$, here we propose to use a different $\boldsymbol{\epsilon}'$. Our goal is that optimizing $\mathcal{R}$ also leads to smaller $\mathcal{R}_{\text{inst}}$; that is, we seek to find an $\boldsymbol{\epsilon}'$ such that $\left(\frac{\partial}{\partial\boldsymbol{w}}\mathcal{R}\right)^\intercal\frac{\partial}{\partial\boldsymbol{w}}\mathcal{R}_{\text{inst}} > 0$. An easy choice would be:

$$\boldsymbol{\epsilon}' = \rho\cdot\left(\frac{\partial}{\partial\boldsymbol{w}}\mathcal{R}_{\text{inst}}\right) \Big/ \left\|\frac{\partial}{\partial\boldsymbol{w}}\mathcal{R}_{\text{inst}}\right\|_2. \quad (4)$$

[1] If the cosine similarity is positive, then optimizing $\mathcal{R}$ also leads to optimized $\mathcal{R}_{\text{inst}}$.

Because each $\boldsymbol{H}_i$ is positive definite under our assumptions, $\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{H}_i(\boldsymbol{w})$ is also positive definite, then we have:

$$\left(\frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}\right)^{\intercal}\frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}_{\text{inst}}$$
$$\propto \left(\frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}_{\text{inst}}\right)^{\intercal}\left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{H}_i(\boldsymbol{w})\right)\left(\frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}_{\text{inst}}\right)$$
$$> 0.$$

Thus, under this specific $\boldsymbol{\epsilon}' = \frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}_{\text{inst}}$, $\frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}$ is aligned with $\frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}_{\text{inst}}$. Although $\frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}_{\text{inst}}$ is the gradient that we aim to approximate and not computable here, the key observation of Eq. 4 is that *per-instance weight perturbation can be optimized by using a perturbation shared by all instances in the batch*. Therefore, we attempt to perturb the model with only a (rough) estimation of $\boldsymbol{\epsilon}'$. Now the next challenge is how to efficiently derive such estimation.

An important observation is that under our assumptions on $\boldsymbol{H}_i(\boldsymbol{w})$ and the use of second order approximations, $\frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}_{\text{inst}}$ can be calculated by using one time backpropagation on a *reweighted batch*. To see this fact, we define weights $g_i = a_i \left\|\nabla l_i(\boldsymbol{w})\right\|_2$ and the reweighted batch is:

$$l_{\text{reweighted}} = \frac{1}{N}\sum_{i=1}^{N} g_i l_i(\boldsymbol{w}).$$

Then, treating g_i as constants, $\frac{\partial}{\partial \boldsymbol{w}} l_{\text{reweighted}} \propto \frac{\partial}{\partial \boldsymbol{w}}\mathcal{R}_{\text{inst}}$, as defined in Eq. 3. Compared to per-batch SAM, this reweighting only requires small extra computation cost on calculating the instance weights g_i. We introduce how to estimate these weights efficiently in the following section.

4.2 Implementation

In δ-SAM, the key problem in implementation is how to efficiently estimate the instance weight g_i. We solve this problem by sampling random perturbations. Specifically, for random perturbation $\boldsymbol{r}$, where $\boldsymbol{r}_i$ follows Gaussian distribution $\mathcal{N}(0, \sigma\boldsymbol{I})$, under the same assumptions as in Section 4.1, we have:

$$\begin{aligned}
&E[(l_i(\boldsymbol{w}+\boldsymbol{r}) - l_i(\boldsymbol{w}-\boldsymbol{r}))^2] \\
&= E[\nabla l_i(\boldsymbol{w})^{\intercal}\boldsymbol{r} \cdot \nabla l_i(\boldsymbol{w})^{\intercal}\boldsymbol{r}] \\
&= \sigma^2 \left\|\nabla l_i(\boldsymbol{w})\right\|_2^2. \\
&E[l_i(\boldsymbol{w}+\boldsymbol{r}) + l_i(\boldsymbol{w}-\boldsymbol{r}) - 2l_i(\boldsymbol{w})] \\
&= E[\boldsymbol{r}^{\intercal}\boldsymbol{H}_i\boldsymbol{r}] \\
&= a_i\sigma^2 \left\|\nabla l_i(\boldsymbol{w})\right\|_2^2.
\end{aligned}$$

Therefore, by sampling random perturbations and take the expectation, we can get unbiased estimations of a_i and the gradient norm $\left\|\nabla l_i(\boldsymbol{w})\right\|_2$. Each estimation takes three forward passes for calculating $l_i(\boldsymbol{w})$, $l_i(\boldsymbol{w}+\boldsymbol{r})$, and $l_i(\boldsymbol{w}-\boldsymbol{r})$. As we do not need to save the intermediate states for backpropagation (`no_grad` in PyTorch), these forward passes are faster than normal ones. In δ-SAM, for the efficiency of the algorithm, we only sample one (shared) $\boldsymbol{r} \sim \mathcal{N}(0, \sigma\boldsymbol{I})$ for each batch in δ-SAM, and then calculate the instance weight g_i by:

$$g_i = \frac{\left|l_i(\boldsymbol{w}+\boldsymbol{r}) + l_i(\boldsymbol{w}-\boldsymbol{r}) - 2l_i(\boldsymbol{w})\right|}{\max\left(\left|l_i(\boldsymbol{w}+\boldsymbol{r}) - l_i(\boldsymbol{w}-\boldsymbol{r})\right|, \eta\right)}, \quad (5)$$

where η is a hyperparameter for avoiding division by zero. After deriving the instance weights, the *weighted-batch* weight perturbation can be computed by:

$$\nabla l_{\mathcal{B}}(\boldsymbol{w}) = \nabla\left(\sum_{i}^{N} g_i l_i(\boldsymbol{w})\right), \quad (6)$$
$$\boldsymbol{\epsilon}^* = \rho\nabla l_{\mathcal{B}}(\boldsymbol{w}) / \left\|\nabla l_{\mathcal{B}}(\boldsymbol{w})\right\|_2. \quad (7)$$

We hereby summarize our algorithm, as outlined in Alg. 1. Modifications made for δ-SAM are highlighted in blue. Given a batch $\mathcal{B}$, we first dynamically reweigh the instances by Eq. 5, then estimate the perturbation $\boldsymbol{\epsilon}^*$ that maximizes the reweighted loss by a single-step gradient descent as shown in Eq. 6 and Eq. 7, and finally minimize the empirical risk of the perturbed model on the original (unweighted) batch.

5 Experiments

This section presents experimental evaluation of δ-SAM based on the GLUE benchmark tasks (§5.2), self-supervised Semantic Textual Similarity (STS) tasks (§5.3), and the CNN/DailyMail abstractive summarization task (§5.4). We also provide additional analyses to further illustrate the performance of δ-SAM (§5.5).

Model	avg.	MNLI Acc-m	QQP Acc	RTE Acc	QNLI Acc	MRPC Acc	CoLA Mcc	SST2 Acc	STS-B Pearson
$\text{BERT}_{\text{BASE}}$	82.9	83.8	91.0	68.2	90.8	85.3	62.3	92.4	89.3
R-Drop (Liang et al., 2021)	84.1	**85.5**	91.4	71.1	**92.0**	87.3	62.6	93.0	89.6
SAM	83.9	85.0	91.6	69.3	91.7	88.2	63.1	93.0	89.4
δ-SAM	**84.7**	85.2	**91.7**	**72.2**	91.5	**89.5**	**63.8**	**93.7**	**89.7**
$\text{RoBERTa}_{\text{LARGE}}$ (Liu et al., 2019)	88.9	90.2	92.2	86.6	94.7	90.9	68.0	96.4	92.4
R-Drop (Liang et al., 2021)	89.7	90.9	92.5	88.4	95.2	91.4	70.0	96.9	92.5
FreeLB (Zhu et al., 2019)	89.8	90.6	**92.6**	88.1	95.0	91.4	71.1	96.7	**92.7**
SMART (Jiang et al., 2020a)	-	**91.1**	92.4	92.0*	**95.6**	89.2*	70.6	96.9	92.8*
R3F (Aghajanyan et al., 2020)	-	**91.1**	92.4	88.5	95.3	91.6	**71.2**	**97.0**	-
SAM	89.6	91.0	92.3	88.5	95.0	91.4	69.2	96.7	92.4
δ-SAM	**90.1**	**91.1**	92.5	**89.2**	95.1	**92.2**	71.1	96.9	**92.7**

Table 1: Results on the development set of the GLUE benchmark. * denotes results derived from the model intermediately trained on the MNLI dataset (not comparable to other results), while others are derived by finetuning the original BERT/RoBERTa. The results of $\text{BERT}_{\text{BASE}}$ are from the reimplementation by Liang et al. (2021).

5.1 Baseline Methods

We compare SAM and δ-SAM to the following baseline methods, which were all proposed for improving the generalization of PLMs:

- **R-Drop** (Liang et al., 2021) enforces the prediction of the same instances augmented by different dropout masks to be similar with a consistency term (KL divergence for classification and mean squared error for regression), which leads to improved performance on various language and vision tasks.
- **R3F** (Aghajanyan et al., 2020) also uses a consistency term to make the prediction of the same instance to be similar. Besides augmenting the instances by different dropout masks, it further adds random uniform or normal noise to input embedding in PLMs. Therefore, R3F can be regarded as an extension to R-Drop.
- **FreeLB** (Zhu et al., 2019) adversarially perturbs the token embedding using a multi-step projected gradient descent (PGD; Madry et al. 2018) to maximize the empirical risk and regularizes the adversarial risk to be small.
- **SMART** (Jiang et al., 2020a) is a framework that combines multiple techniques for improving model generalization, including adversarial training, and improved optimizer and regularization techniques. In terms of adversarial training, it perturbs the input embedding with PGD to maximize the empirical risk. It then uses a consistency term to regularize the change of risk to be small, for which the consistency term is defined as the KL divergence for classification and mean squared loss for regression.

5.2 GLUE Tasks

Task Setup. We first evaluate δ-SAM on the GLUE benchmark (Wang et al., 2018). In this experiment, we use both $\text{BERT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{LARGE}}$ as the encoders. To ensure a fair comparison, for task-specific hyperparameters including batch sizes, optimizers, learning rates, training steps, weight decay, dropout rates, and learning rate scheduling, we strictly replicate the values from R-Drop (Liang et al., 2021). For SAM and δ-SAM, we search ρ in $\{0.01, 0.02, 0.05\}$ and η in $\{1e{-}4, 2e{-}4, 5e{-}4, 1e{-}3\}$. For σ in random perturbations, we find that rescaling the random Gaussian perturbation to an L_2-norm of ρ achieves promising results, so we simply set $\sigma = 1$ and rescale the random perturbation afterwards. Following the evaluation settings of R3F and R-drop, we report the best result on the development set out of 5 runs of training with different random seeds.

Results and Discussion. Results are shown in Table 1. We observe that in average, SAM improves $\text{BERT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{LARGE}}$ by 1.0% and 0.7%, respectively, showing that SAM improves the generalization of PLMs, being consistent with the findings in the recent work (Bahri et al., 2022). However, its performance is still worse than other compared methods. On the other hand, δ-SAM improves $\text{BERT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{LARGE}}$ by 1.8% and 1.2%, respectively, and also achieves better or comparable results compared to other methods, demonstrating its effectiveness. In terms of individual tasks, the performance gain of δ-SAM to

Model	avg.	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R
Mirror-BERT$_{\text{BASE}}$	74.85	67.87	80.98	71.84	81.58	74.41	77.78	69.53
+ FreeLB	75.71	69.78	80.81	72.87	82.55	74.77	79.01	**70.40**
+ R3F	75.27	68.58	80.93	72.51	81.97	75.47	77.62	69.79
+ SAM	75.50	68.37	82.16	72.88	82.22	74.71	77.98	70.20
+ δ-SAM	75.72	68.44	82.30	73.12	82.27	75.12	78.83	69.94
+ SAM w/ random noise	76.44	**70.09**	83.53	74.22	82.67	77.61	**80.16**	66.81
+ δ-SAM w/ random noise	**76.71**	69.73	**83.73**	**74.58**	**83.01**	**77.72**	79.08	69.11
Mirror-RoBERTa$_{\text{BASE}}$	74.98	64.49	81.69	73.32	79.78	77.49	78.53	69.56
+ FreeLB	75.73	65.52	82.41	74.00	80.72	78.60	79.06	69.81
+ R3F	75.32	64.58	82.12	73.62	80.27	77.90	78.66	70.13
+ SAM	75.18	64.86	81.96	73.56	79.82	77.06	78.83	70.21
+ δ-SAM	75.27	64.97	81.89	73.46	80.12	77.62	78.96	69.89
+ SAM w/ random noise	75.90	66.65	82.52	74.10	80.81	78.47	79.02	69.71
+ δ-SAM w/ random noise	**76.31**	**66.94**	**82.86**	**74.46**	**81.13**	**78.91**	**79.52**	**70.32**

Table 2: Results on self-supervised STS tasks. For all datasets, we report the average Spearman's ρ of 5 runs of training using 5 fixed random seeds.

SAM is larger on smaller datasets (e.g., MRPC, RTE, CoLA, SST2), while it becomes less prominent on larger datasets. We hypothesize that due to increased training steps and number of instances in large datasets, the gap between per-batch and per-instance perturbation becomes smaller. It is also possible that smaller datasets need better generalization so δ-SAM helps more. Besides, we observe that the improved performance and generalization by δ-SAM is obtained at a merely little average extra computational cost of 18% to SAM (see Table 6 in Appendix for the running time of models). Taking BERT$_{\text{BASE}}$ and the SST2 dataset as an example, the average running time is 118/132 min for SAM/δ-SAM, respectively, meaning that δ-SAM is only 12% slower than SAM to approximate per-instance perturbation.

5.3 Self-supervised STS

Model. To conduct self-supervised STS evaluation, we apply δ-SAM to the training process of Mirror-BERT$_{\text{BASE}}$ and Mirror-RoBERTa$_{\text{BASE}}$ (Liu et al., 2021), which are SOTA self-supervised sentence embedding frameworks. Similar to SimCSE (Gao et al., 2021), Mirror-BERT embeds a sentence x with the same encoder but different dropout masks to get two sentence embedding h_1 and h_2, and optimizes h_1 and h_2 to be similar using contrastive loss. This training objective resembles R-Drop and SMART. Empirically, we find that applying adversarial training (FreeLB, SAM, and δ-SAM) to only one embedding (e.g. h_2 only) achieves much better results than to the contrastive loss, and we use that strategy in experiments.

Task Setup. We experiment with self-supervised sentence embedding learning on 7 STS datasets including SemEval 2012-2016 datasets (STS12-16, Agirre et al. 2012, 2013, 2014, 2015, 2016), STS Benchmark (STS-B, Cer et al. 2017), and SICK-Relatedness (SICK-R, Marelli et al. 2014). We strictly follow and replicate the model and experimental setup of Mirror-BERT. As the training objective of Mirror-BERT is similar to R-Drop and SMART, we compare with other baselines including R3F[2] and FreeLB on this task. Furthermore, as FreeLB and R3F both use random noise to augment input embedding, we also attempt with adding random uniform noise to input embedding in δ-SAM for fair comparison. Following the evaluation setting of Mirror-BERT, we report the average Spearman's ρ under 5 fixed random seeds for all models. All hyperparameters of FreeLB, R3F, SAM, and δ-SAM are tuned on the development set of STS-B.

Results and Discussion. Results are shown in Table 2. Overall, δ-SAM with random noise achieves the best results on both encoders, outperforming Mirror-BERT$_{\text{BASE}}$ and Mirror-RoBERTa$_{\text{BASE}}$ by 1.86% and 1.33% in terms of Spearman's ρ in average, respectively, and also outperforms other methods. On individual datasets, δ-SAM with random noise achieves the top performance on 11 of 14 setups. Note that both R3F and FreeLB use random noise so the comparison is fair. Applying δ-SAM alone leads to improvements of 0.87% and 0.29% on Mirror-BERT$_{\text{BASE}}$ and Mirror-RoBERTa$_{\text{BASE}}$, respectively. Besides, δ-SAM outperforms SAM on both encoders either w/ or w/o random noise. These

[2] Applying R3F to Mirror-BERT is simply adding random noise to input embedding.

Model	RG-1	RG-2	RG-L
BART (Lewis et al., 2020)	44.16	21.28	40.90
PEGASUS (Zhang et al., 2020b)	44.17	21.41	41.11
BART + R3F (Aghajanyan et al., 2020)	44.38	21.53	41.17
BART + R-Drop (Liang et al., 2021)	44.51	**21.58**	41.24
BART + δ-SAM	**44.70**	21.54	**41.81**

Table 3: Results on CNN/Daily Mail summarization.

Method	$\mathcal{L}_{\text{adv}}$		Acc	
	MRPC	RTE	MRPC	RTE
Vanilla	1.93	2.97	84.6/85.3	67.9/68.2
SAM	0.62	0.78	86.8/88.2	68.6/69.3
δ-SAM	0.59	0.75	87.5/89.5	69.3/**72.2**
Per-instance perturbation	**0.50**	**0.65**	**88.2/89.7**	**70.0**/71.5

Table 4: Adversarial risk and evaluation results on MRPC and RTE datasets. We report the average adversarial risk and the median/max accuracy of 5 runs.

results have demonstrated that besides supervised models, δ-SAM can also effectively improve self-supervised sentence embedding models.

5.4 Summarization

Task Setup. We experiment with the abstractive summarization task on the CNN/DailyMail dataset (Hermann et al., 2015). Using the large version of BART (Lewis et al., 2020) as the encoder-decoder model, we compare δ-SAM to two regularization methods, including R3F and R-Drop, that have experimented on this task. Besides, we also compare to PEGASUS (Zhang et al., 2020b), which introduces a self-supervised training objective specifically designed for summarization. Following the experimental settings of BART, we report metrics including the unigram ROUGE-1 and bigram ROUGE-2 for evaluating the informativeness, and the longest common subsequence ROUGE-L for evaluating the fluency.

Results and Discussion. Results are shown in Table 3. We observe that δ-SAM achieves the best results in ROUGE-1 and ROUGE-L, outperforming the original BART by 0.54% and 0.91%, respectively. As for ROUGE-2, it also outperforms BART by 0.26% and achieves performance comparable to R3F and R-Drop. This experiment shows the effectiveness of δ-SAM on optimizing an encoder-decoder model for abstractive summarization.

5.5 Analysis

The previous experiments have demonstrated that δ-SAM achieves promising improvements on various tasks. In this section, we analyze whether δ-SAM can derive smaller adversarial risk and how well it approximates per-instance weight perturbation.

We assess the adversarial risks and accuracies of four optimization approaches including vanilla training, SAM, δ-SAM, and per-instance weight perturbation. We measure the adversarial risk with Eq. 2 in §3, which is copied as follows:

$$\mathcal{L}_{\text{adv}} = \frac{1}{N}\sum_{i=1}^{N} \max_{\boldsymbol{\epsilon}_i:\|\boldsymbol{\epsilon}_i\|_2\leq\rho} l_i(\boldsymbol{w}+\boldsymbol{\epsilon}_i).$$

For all compared methods, we set $\rho = 0.05$ in $\mathcal{L}_{\text{adv}}$. Due to the high computational cost of per-instance weight perturbation (about 7x of δ-SAM with a batch size of 16), we only conduct experiments on two small datasets: MRPC and RTE.

From the results shown in Table 4, we observe that δ-SAM achieves smaller adversarial risk than SAM, showing that δ-SAM is indeed a better approximation to per-instance weight perturbation than SAM, being consistent with our theoretical motivation (§4.1). When it comes to accuracy, we observe that: (1) Per-instance weight perturbation generally achieves the highest accuracy except for the maximum accuracy on RTE, being consistent with the observation in Foret et al. (2020); (2) Although δ-SAM consistently outperforms SAM, its performance is often slightly lower than the much more costly per-instance weight perturbation, indicating room for further improvements.

6 Conclusion

This paper presents a new sharpness-aware minimization method with dynamic reweighting (δ-SAM). The proposed method represents the first successful attempt in realizing a *per-instance* weighting scheme. We achieve this by prioritizing instances with larger gradient change rate in adversarial weight perturbation, in comparison to previous approaches that adopt *per-batch* weight perturbation. We show that perturbation calculated on reweighted batch can serve as a better approximation to per-instance weight perturbation while requiring only similar computational cost to per-batch perturbation. We conduct extensive experiments on the GLUE, STS, and abstractive summarization benchmarks. Across all 30 experimental setups that compares to SAM, δ-SAM achieves

an consistent improvement over SAM in 27 of them. When compared to a set of other competitive regularization methods, δ-SAM achieves the best performance in 23 out of 33 of the setups. Further, we quantitatively analyze δ-SAM's impact on sharpness, finding that it indeed leads to flatter loss landscape. Future work includes inventing new techniques to further reduce the computational cost of δ-SAM and demonstrating its effectiveness on more tasks such as sequence tagging and question answering.

Limitations

Like SAM and other training methods based on weight perturbation, the improved performance by δ-SAM is at the cost of introducing additional computational overhead to vanilla training. Specifically, although δ-SAM more precisely approximates per-instance weight perturbation with merely 18% extra computational cost to per-batch SAM, both SAM and δ-SAM are still slower than vanilla training by roughly doubling the computational costs in practice. This may limit the application of such optimization algorithms on massive-scale training.

Acknowledgement

We appreciate the anonymous reviewers for their insightful comments and suggestions. This material is supported by the NSF Grant IIS 2105329, a subaward from NSF Cloudbank 1925001 and a Cisco Research Award.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Dara Bahri, Hossein Mobahi, and Yi Tay. 2022. Sharpness-aware minimization improves language model generalization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7360–7371, Dublin, Ireland. Association for Computational Linguistics.

Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. 2021. High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning*, pages 1059–1071. PMLR.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. 2022. When vision transformers outperform resnets without pre-training or strong data augmentations. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent Tan. 2022. Efficient sharpness-aware minimization for improved training of neural networks. In *International Conference on Learning Representations*.

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

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. 2021. AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights. In *International Conference on Learning Representations*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI).

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020a. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. 2020b. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems*, 31.

Linyang Li and Xipeng Qiu. 2021. Token-aware virtual adversarial training in natural language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8410–8418.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: regularized dropout for neural networks. In *Advances in neural information processing systems*.

Fangyu Liu, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2021. Fast, effective, and self-supervised: Transforming masked language models into universal lexical and sentence encoders. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1459, Online and

Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Hossein Mobahi, Mehrdad Farajtabar, and Peter Bartlett. 2020. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 33:3351–3361.

Lis Pereira, Xiaodong Liu, Hao Cheng, Hoifung Poon, Jianfeng Gao, and Ichiro Kobayashi. 2021. Targeted adversarial training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5385–5393, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip Yu, and Lifang He. 2020. Mixup-transformer: Dynamic data augmentation for NLP tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3436–3440, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 296–310, Online. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

LI Xuhong, Yves Grandvalet, and Franck Davoine. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, pages 2825–2834. PMLR.

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

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR.

Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. 2020a. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020b. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

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

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha C Dvornek, sekhar tatikonda, James s Duncan, and Ting Liu. 2022. Surrogate gap minimization improves sharpness-aware training. In *International Conference on Learning Representations*.

Appendices

A Hyperparameters

On the GLUE benchmark, we search ρ in $\{0.01, 0.02, 0.05\}$ and η in $\{\texttt{1e-4}, \texttt{2e-4}, \texttt{5e-4}, \texttt{1e-3}\}$, respectively. The hyperparameters that achieve the best performance on the GLUE benchmark is listed in Table 5. On the STS benchmark, we search ρ in $\{0.01, 0.02, 0.05, 0.1\}$ and η in $\{\texttt{1e-3}, \texttt{1e-2}, 0.1, 1.0\}$, respectively. We search the scale of uniform random noise in $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$. The best hyperparameter of ρ, η, and the scale of uniform random noise is 0.1, 1.0, 0.1 for Mirror-BERT and 0.02, 1.0, 0.2 for Mirror-RoBERTa, respectively. For summarization, we use $\rho = 0.01$ and $\eta = \texttt{1e-4}$. For all models, we use grid search to find the best hyperparameter. For other hyperparameters (e.g., batch size, training steps, learning rate, etc.), we directly use the suggested values in the original papers. Note that for per-instance perturbation, we adopt twice the number of original epochs since we observe they are under trained with default number of epochs.

B Running Time

The running time of SAM and δ-SAM on the GLUE benchmark is shown in Table 6.[3] All experiments are conducted on one RTX2080 GPU.

[3]For easy comparison, the recorded training time is for the same number of epochs across all models, though per-instance perturbation actually takes twice the number of default epochs in the actual experiment.

Hyperparameter	MNLI	QQP	RTE	QNLI	MRPC	CoLA	SST2	STS-B
$\text{BERT}_{\text{BASE}}$								
ρ	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.01
η	1e-4	1e-4	5e-4	1e-4	1e-4	1e-4	1e-4	1e-4
$\text{RoBERTa}_{\text{LARGE}}$								
ρ	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01
η	1e-4	1e-4	5e-4	1e-4	1e-4	1e-4	1e-4	1e-4

Table 5: Hyperparameters for SAM and δ-SAM on the GLUE benchmark.

Running time	MNLI	QQP	RTE	QNLI	MRPC	CoLA	SST2	STS-B
$\text{BERT}_{\text{BASE}}$								
SAM	1240	818	15	349	12	25	118	26
δ-SAM	1350	995	16	450	15	25	132	31
Per-instance weight perturbation	-	-	87	-	105			
$\text{RoBERTa}_{\text{LARGE}}$								
SAM	1056	2591	33	1402	30	38	285	42
δ-SAM	1699	2963	36	1425	35	45	325	58

Table 6: Average running time (in min) for SAM and δ-SAM on the GLUE benchmark.