δ-SAM: Sharpness-Aware Minimization with Dynamic Reweighting

Anonymous ACL submission

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. Unfortunately, due to increased computational cost, adversarial weight perturbation can only be efficiently estimated per-batch instead of per-instance by SAM, leading to degraded performance. In this paper, we tackle this efficiency bottleneck and propose the first instance-based weight perturbation method: sharpness-aware minimization with dynamic reweighting (δ-SAM). δ-SAM dynamically reweights perturbation within each batch by estimated guardedness (i.e. unguarded instances are up-weighted), serving as a better approximation to per-instance perturbation. Experiments on various 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., 2020) and adversarial robustness (Madry et al., 2018; Zhang et al., 2019). A general approach for adversarial training is (1) augmenting the inputs with small perturbations that lead to the maximum possible change of loss, and then (2) optimizing the model to the direction such that the changed amount is minimized.

Besides perturbing inputs, a recent work of sharpness-aware minimization (SAM; Foret et al. 2020) has further considered adversarially perturbing model weights. It works by first adversarially calculating a weight perturbation that maximizes the empirical risk and then minimizing the empirical risk on the perturbed network. This method demonstrates improved model generalizations across different datasets and models. Nevertheless, as the weight perturbation is derived on a large space of all model parameters, adding this mechanism in training leads to a significant increase in computational and memory cost. To mitigate this drawback, SAM speeds up training by calculating perturbations on *per-batch* instead of *per-instance*. However, as per-batch perturbation averages perturbations yielded by different instances, it weakens the derived perturbations (compared to per-instance perturbations) as being less fine-grained, and may lead to a performance drop.

In this paper, we study how to bridge the performance gap to efficiently realize the first per-instance weight perturbation method. Our intuition is that the performance gap from per-batch perturbation is caused by the loss of per-instance characteristics when averaging independent perturbations and can be narrowed by prioritizing *unguarded* instances[1] in perturbation. Based on this intuition, we propose sharpness-aware minimization with dynamic reweighting (δ-SAM). We first estimate how *guarded* each instance is by its change of loss with a random weight perturbation. Next, instead of equally perturbing all instances in the batch, δ-SAM dynamically reweights the perturbation within each batch of training instances, where the perturbations on less guarded instances are up-weighted. Finally, we update the perturbed network on the original (unweighted) batch. Compared to SAM, δ-SAM only requires one extra computation cost in guardedness estimation, which can be

[1] Similar to Zhang et al. (2020), we describe more certain instances that are far from decision boundaries as more *guarded*, or having higher *guardedness*.

efficiently performed using two forward passes.

We evaluate δ-SAM on finetuning pretrained language models (PLMs) using both BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) as the backbone. Experiments on language understanding and unsupervised STS show that besides significantly outperforming base models, δ-SAM also consistently outperforms SAM with only 18% extra computational cost.

2 Methodology

In this section, we briefly review the principle of SAM and present the proposed δ-SAM algorithm.

2.1 Sharpness-Aware Minimization (SAM)

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., 2019). To find a flat loss landscape, SAM (Foret et al., 2020) adversarially perturbs the neural network weights and optimizes the following min-max objective on a batch of size N:

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

where given the network weights $\boldsymbol{w}$, the inner maximization seeks for a perturbation $\boldsymbol{\epsilon}$ with L_2-norm $\leq \rho$ that maximizes the empirical risk, and the outer minimization minimizes the empirical risk of the perturbed network. As finding the exact solution to $\boldsymbol{\epsilon}$ is NP-hard, SAM estimates the solution $\boldsymbol{\epsilon}^*$ to inner maximization with a single-step gradient descent on the empirical risk of the batch:

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

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 network, 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:

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

Algorithm 1: SAM and δ-SAM

Input: network $f_{\boldsymbol{w}}$, training set $\mathcal{S} \triangleq \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{|\mathcal{S}|}$, loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, 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 $\mathcal{B} = \{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^{N}$.
- δ-SAM:
- Estimate guardedness of instances in $\mathcal{B}$ and reweigh $\mathcal{B}$ by Eq. 3 and Eq. 4.
- Rescale the weighting by Eq. 5 and Eq. 6.
- Compute gradient $\nabla l_{\mathcal{B}}(\boldsymbol{w})$ of the (reweighted) batch's empirical risk.
- Perturb the network weights by $\boldsymbol{\epsilon}^* = \rho\nabla l_{\mathcal{B}}(\boldsymbol{w}) / \|\nabla l_{\mathcal{B}}(\boldsymbol{w})\|_2$.
- Update $\boldsymbol{w}$ w.r.t. the empirical risk $\frac{1}{N}\sum_{j=1}^{N} l_j(\boldsymbol{w}+\boldsymbol{\epsilon}^*)$ with the optimizer h.

where $\boldsymbol{\epsilon}_i$ is calculated by 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 network 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 observed 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 network weights, the perturbation for each individual instance needs to be trained on a distinct network copy. Therefore, per-instance weight perturbation can be computationally unaffordable for large-scale training.

2.2 SAM with Dynamic Reweighting (δ-SAM)

In this paper, we seek to adapt SAM to adversarially, and more efficiently, train NLP models. As the per-batch weight perturbation adopted by SAM weakens the adversarial training, we propose a simple yet effective modification of SAM, the δ-SAM (SAM with dynamic reweighting), that can simulate per-instance weight perturbation without requiring much additional computational cost.

Motivation. 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 network 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 $\mathcal{R}_{\text{inst}}$. To bridge the gap between $\mathcal{R}_{\text{batch}}$ and $\mathcal{R}_{\text{inst}}$, we examine the following reweighted sharpness measure:

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

$$s.t. \sum_{i=1}^{N} p_i = 1; p_i \geq 0, \forall i \in \{1,...,N\},$$

where p_i is the instance weight. From $\mathcal{R}_s$, we can observe that: (1) When all instance weights equal $\frac{1}{N}$, $\mathcal{R}_s$ is identical to $\mathcal{R}_{\text{batch}}$, and (2) Assume that instance j is the most unguarded instance in $\mathcal{R}_{\text{inst}}$, i.e., $j = \arg\max_{i\in\{1,...,N\}}\left(l_i(\boldsymbol{w}+\boldsymbol{\epsilon}_i) - l_i(\boldsymbol{w})\right)$, if we set p_j to 1 and other instances' weights to 0, we will have $\mathcal{R}_s = \frac{1}{N}\left(l_j(\boldsymbol{w}+\boldsymbol{\epsilon}_j) - l_j(\boldsymbol{w})\right) \geq \frac{1}{N}\mathcal{R}_{\text{inst}}$, which means that $\mathcal{R}_s$ upper bounds $\frac{1}{N}\mathcal{R}_{\text{inst}}$. Therefore, by assigning larger instance weights to more unguarded instances, $\mathcal{R}_s$ may approximate the per-instance weight perturbation. This intuition leads to the following inner maximization problem:

$$\boldsymbol{\epsilon}_s = \arg\max_{\boldsymbol{\epsilon}:\|\boldsymbol{\epsilon}\|_2\leq\rho}\sum_{i=1}^{N} g_i\left(l_i(\boldsymbol{w}+\boldsymbol{\epsilon}) - l_i(\boldsymbol{w})\right),$$

where g is a measure of the unguardedness of instances. $\boldsymbol{\epsilon}_s$ can be estimated with a single-step gradient descent on the reweighted batch.

Implementation. As explained above, g should be positively correlated to the per-instance sharpness. In this paper, we simply set g proportional to $\max_{\boldsymbol{\epsilon}_i:\|\boldsymbol{\epsilon}_i\|_2\leq\rho}\left(l_i(\boldsymbol{w}+\boldsymbol{\epsilon}_i) - l_i(\boldsymbol{w})\right)$. As we only need the value of g without the weight perturbation, we estimate g by first sampling a random weight perturbation $\hat{\boldsymbol{\epsilon}}$ from the normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and then calculate the change of loss:

$$a_j = l_j(\boldsymbol{w} + \rho\cdot\hat{\boldsymbol{\epsilon}}/\|\hat{\boldsymbol{\epsilon}}\|_2) - l_j(\boldsymbol{w}), \quad (3)$$

$$g_j = |a_j| / \sum_{i=1}^{N}|a_j|. \quad (4)$$

This estimation takes two forward passes. As we do not need to save the intermediate states for back-propagation, these forward passes are faster than the normal ones.

Besides, we observe that some instances have high unguardedness, making the reweighted perturbation focus on very few instances while neglecting others. This leads to inefficient training. Therefore, we set the instances weights as a mixture of g and the uniform instance weights, controlled by a hyperparameter β. Specifically, we first estimate the value of $\mathcal{R}_{\text{batch}}$ and $\mathcal{R}_s$ by $\bar{a} = \frac{1}{N}\sum_{i=1}^{N} a_i$ and $a_s = \sum_{i=1}^{N} g_i a_i$, and then rescale g by:

$$r = \beta\cdot|\bar{a}|/|a_s|, \quad (5)$$

$$g'_i = (g_i - 1/N)\cdot r + 1/N. \quad (6)$$

This rescaling makes the estimated per-batch sharpness $\mathcal{R}_{\text{batch}}$ and the reweighted sharpness $\mathcal{R}_s$ to be close as $\mathcal{R}_s \leq (\beta+1)\cdot\mathcal{R}_{\text{batch}}$. We use the rescaled g'_i as the final instance weights in training.

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, then estimate the perturbation $\boldsymbol{\epsilon}_s$ that maximizes the reweighted loss by a single-step gradient descent, and finally minimize the empirical risk of the perturbed network on the original (unweighted) batch.

3 Experiments

This section presents experimental evaluation of δ-SAM based on GLUE benchmark tasks (Wang et al., 2018) and the unsupervised Semantic Textual Similarity (STS) task. We use BERT$_{\text{BASE}}$ and RoBERTa$_{\text{BASE/LARGE}}$ as base PLMs to evaluate our method. We implement SAM based on an open-source repository[2]. Hyperparameter settings and compared methods are described in Appx. §A.

3.1 GLUE Results

We first evaluate our method on the GLUE benchmark. We use the same set of finetuning hyperparameters as R-Drop (Liang et al., 2021) for BERT and R3F (Aghajanyan et al., 2020) for RoBERTa. Following their work, we report the best development result out of 5 runs of training. Results are shown in Tab. 1. We observe that in average, SAM improves BERT/RoBERTa by $1.06\%/0.63\%$, respectively, showing that SAM enhances the generalization of PLMs, being consistent with the findings

[2] https://github.com/davda54/sam

Method	avg.	MNLI Acc-m	QQP Acc	RTE Acc	QNLI Acc	MRPC Acc	CoLA Mcc	SST2 Acc	STS-B Pearson
$\text{BERT}_{\text{BASE}}$	82.85	83.8	91.0	68.2	90.8	85.3	62.3	92.4	89.3
R-Drop (Liang et al., 2021)	84.06	**85.5**	91.4	**71.1**	**92.0**	87.3	62.6	93.0	89.6
SAM	83.91	85.0	91.6	69.3	91.7	88.2	63.1	93.0	89.4
δ-SAM	**84.54**	85.2	**91.7**	70.8	91.7	**89.7**	**63.8**	**93.4**	**90.0**
$\text{RoBERTa}_{\text{LARGE}}$ (Liu et al., 2019)	88.93	90.2	92.2	86.6	94.7	90.9	68.0	96.4	92.4
R-Drop (Liang et al., 2021)	89.73	90.9	92.5	88.4	95.2	91.4	70.0	96.9	92.5
FreeLB (Zhu et al., 2019)	89.78	90.6	**92.6**	88.1	95.0	91.4	71.1	96.7	92.7
SMART (Jiang et al., 2020)	90.08	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	89.56	91.0	92.3	88.5	95.0	91.4	69.2	96.7	92.4
δ-SAM	**90.14**	91.1	92.5	88.8	95.0	**92.2**	**71.9**	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, while others are derived by finetuning the original BERT/RoBERTa model. The results of $\text{BERT}_{\text{BASE}}$ is from the reimplementation by Liang et al. (2021).

model↓, dataset→	avg.	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R
Mirror-$\text{BERT}_{\text{BASE}}$	74.67	68.02	80.68	71.80	81.46	74.48	76.86	69.41
+ R3F	75.25	**68.53**	80.82	72.36	81.99	**75.57**	77.74	69.78
+ δ-SAM	75.14	68.48	80.66	72.15	82.05	74.45	77.56	70.61
+ R3F & δ-SAM	**75.55**	68.45	**81.03**	**72.63**	**82.33**	75.44	**78.18**	**70.83**
Mirror-$\text{RoBERTa}_{\text{BASE}}$	75.40	65.08	82.02	73.40	80.33	77.81	79.14	69.74
+ R3F	**76.10**	**66.43**	**82.66**	**74.22**	**81.11**	**78.72**	**79.51**	**70.08**
+ δ-SAM	75.54	65.60	82.03	73.48	80.56	78.05	79.15	69.87
+ R3F & δ-SAM	75.92	66.28	82.50	73.93	81.09	78.45	79.29	69.91

Table 2: Unsupervised STS results (metric: Spearman's rho).

in recent work (Bahri et al., 2021). δ-SAM further improves SAM by $0.63\%/0.58\%$, respectively, and also achieves better or comparable results to other compared methods, demonstrating its effectiveness. In terms of individual tasks, δ-SAM sees more performance gain on smaller datasets (e.g., MRPC, RTE, CoLA), while the performance gain 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. Besides, we observe that the improved performance and generalization by δ-SAM is obtained at a merely little average extra computatioal cost of 18% to SAM. Taking $\text{RoBERTa}_{\text{LARGE}}$ and the SST2 dataset as an example, the average running time is 285/348 min for SAM/δ-SAM, respectively, meaning that δ-SAM is only 22% slower than SAM. The complete running time results are given in Appx. §B.

3.2 Unsupervised STS Results

We also experiment with unsupervised 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 the recently proposed Mirror-BERT (Liu et al., 2021), and test Mirror-BERT with and without applying δ-SAM, R3F[3], and a combination of both. We report average performance under five fixed random seeds for all models (incl. baselines). The hyperparameters of both δ-SAM and R3F are tuned on the dev set of STS-B. From the results in Tab. 2, we observe consistent improvements over the baseline with both adversarial perturbation methods, and a combination of both approaches have a synergistic effect, leading to the optimal performance on BERT. However, on RoBERTa, R3F alone achieves the best average score.

4 Conclusion

This paper presents sharpness-aware minimization with dynamic reweighting (δ-SAM), which is a first successful attempt in realizing an instance weighting scheme by prioritizing unguarded instances in adversarial weight perturbation. We show that perturbation calculated on reweighted batch can serve as a better approximation to per-instance network weight perturbation, while requires only similar computational cost to per-batch perturbation. Experiments on the GLUE benchmark demonstrate the effectiveness and efficiency of δ-SAM.

[3] We re-implemented R3F for unsupervised STS and searched its hparameter. See appendix for details.

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. 2021. Sharpness-aware minimization improves language model generalization. *arXiv preprint arXiv:2110.08529*.

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.

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.

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*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. 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. 2019. 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.

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.

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*.

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.

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.

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. 2020. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*.

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*.

Appendices

A More Experiments Settings

Compared methods. We compare SAM and δ-SAM to the following methods, which were all proposed for improving the generalization of PLMs: (1) **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 loss for regression), which leads to improved performance on various NLP and CV tasks. (2) **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 noise to the token embedding in PLMs. (3) **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 increase of the empirical risk to be small. (4) **SMART** (Jiang et al., 2020) is a framework that consists of multiple techniques for improving model generalization. In terms of adversarial training, it adversarially perturbs the input embedding with PGD to maximize the change of loss, defined as the KL divergence for classification and mean squared loss for regression. It then uses a consistency term to regularize the change of loss to be small.

Hyperparameters. For task-specific hyperparameters including batch sizes, optimizers, learning rates, training steps, weight decay, dropout rates, and learning rate scheduling, we directly adopt the values from Liang et al. (2021) for BERT and values from Aghajanyan et al. (2020) for RoBERTa on the GLUE benchmark, and we adopt the original values from Liu et al. (2021) on unsupervised STS tasks.[4] For SAM and δ-SAM, we search ρ in $\{0.01, 0.02, 0.05\}$ and β in $\{3, 5, 10, 20, 50, 70\}$. The hyperprameter values used on the GLUE benchmark are listed in Tab. 3. For unsupervised STS experiments, we use $\rho = 0.05$, $\alpha = 3$ for $\text{BERT}_{\text{BASE}}$ and $\rho = 0.01$,

[4] https://github.com/cambridgeltl/mirror-bert

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
β	3	3	70	3	10	10	3	10
$\text{RoBERTa}_{\text{LARGE}}$								
ρ	0.02	0.02	0.02	0.02	0.02	0.02	0.05	0.01
β	3	3	5	3	5	20	3	10

Table 3: 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	1436	1075	16	415	15	28	130	32
$\text{RoBERTa}_{\text{LARGE}}$								
SAM	1056	2591	33	1402	30	38	285	42
δ-SAM	1192	2831	41	1425	39	45	348	55

Table 4: Average running time (in min) for SAM and δ-SAM on the GLUE benchmark.

$\alpha = 5$ for $\text{RoBERTa}_{\text{BASE}}$. For R3F on unsupervised STS, we searched its uniform noise range in $\{1\text{e-}5, 1\text{e-}4, 1\text{e-}3, 1\text{e-}2, 5\text{e-}2, 1\text{e-}1\}$. We use 5e-2 and 1e-3 for $\text{BERT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{BASE}}$ respectively.

B Running Time

We run experiments with Intel Xeon 5220 and RTX 2080Ti. The average running times on the GLUE benchmark are shown in 4. We observe that in average, δ-SAM increases the running time of SAM roughly by 18%, showing its efficiency in approximating per-instance perturbation without significantly adding computational overhead to per-batch perturbation.