Penalizing Gradient Norm for Efficiently Improving Generalization in Deep Learning

Yang Zhao[1] & Hao Zhang[1] & Xiuyuan Hu[1]

[1] Department of Electronic Engineering, Tsinghua University

Corresponds to: zhao-yan18@mails.tsinghua.edu.cn

- Basic setting

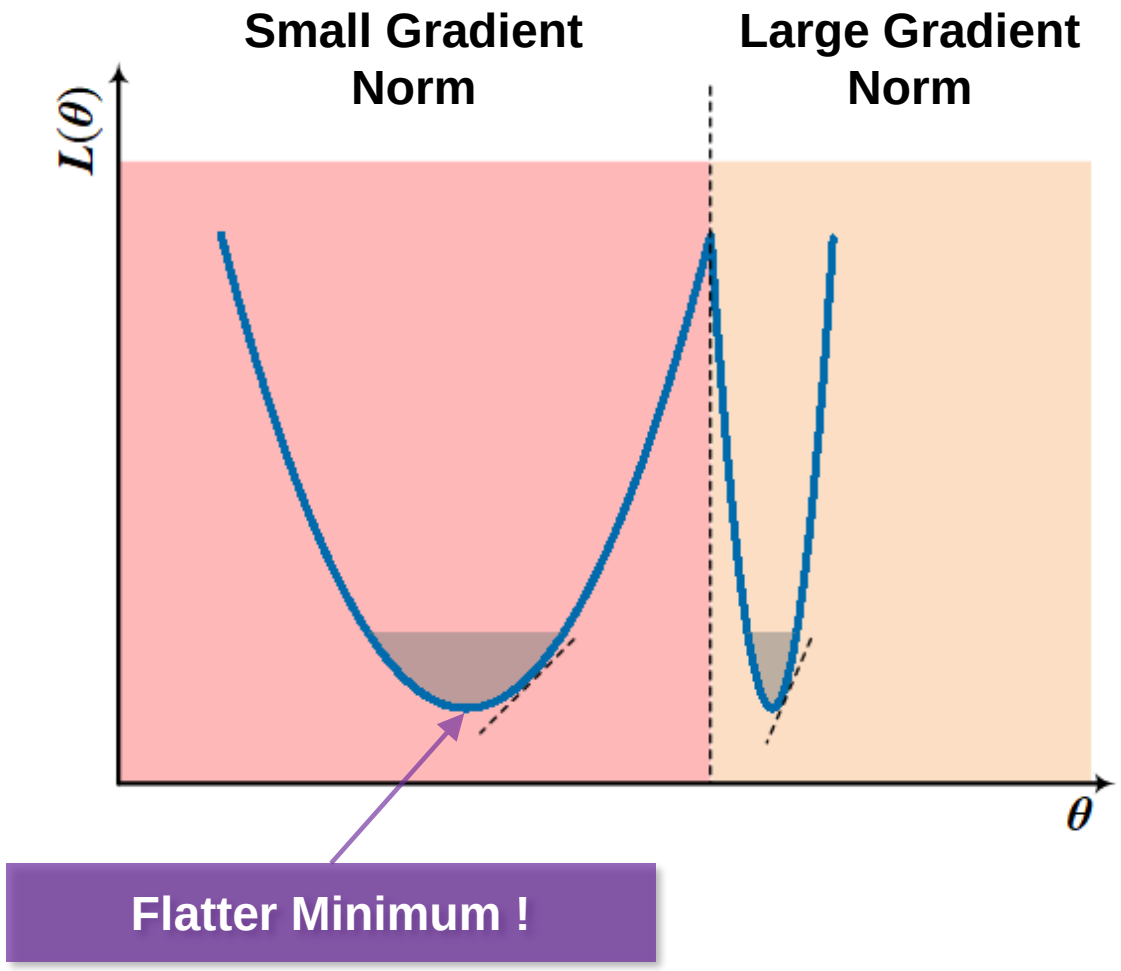
$$L(\boldsymbol{\theta}) = L_{\mathcal{S}}(\boldsymbol{\theta}) + \lambda \cdot ||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||_2$$

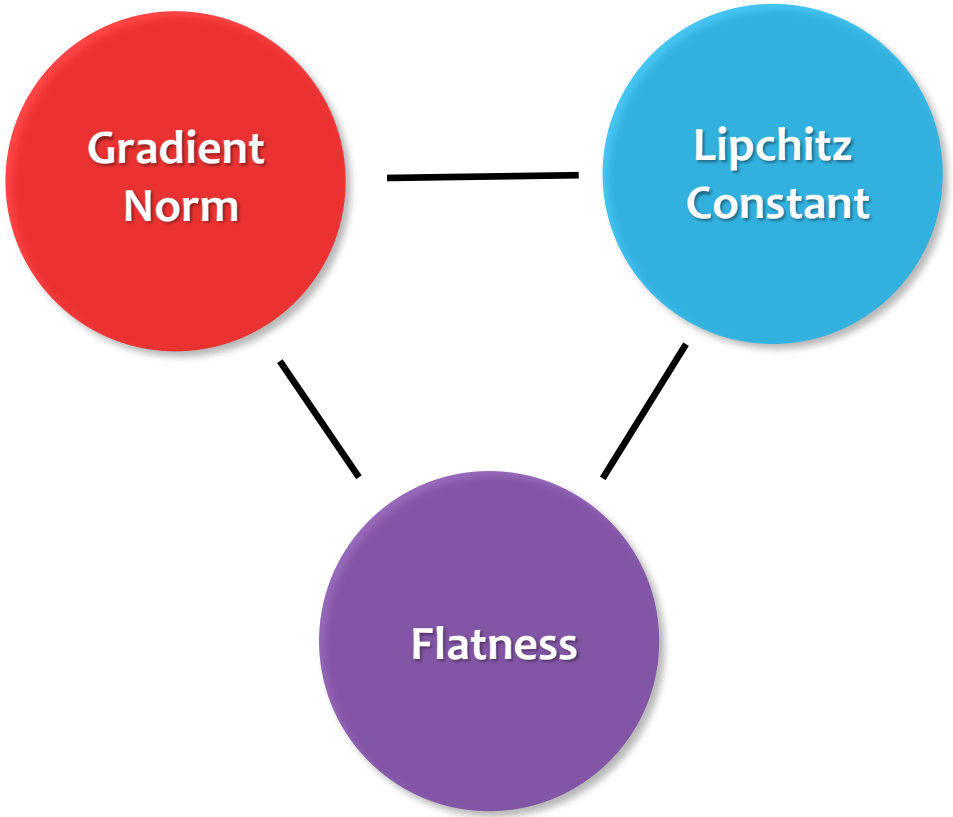
Empirical loss

Penalty coefficient

Gradient norm of empirical loss

- Motivation


Small Gradient Norm
Large Gradient Norm
$L(\theta)$
θ
Flatter Minimum !


Gradient Norm
Lipchitz Constant
Flatness

- Computing Gradient

$$L(\boldsymbol{\theta}) = L_{\mathcal{S}}(\boldsymbol{\theta}) + \lambda \cdot ||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||_2$$

$\nabla_{\boldsymbol{\theta}}$

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \lambda \nabla_{\boldsymbol{\theta}} ||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||_2$$

Chain Rule

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \lambda \cdot \nabla^2_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}$$

Hessian Computation !

- Approximation of $\nabla_{\boldsymbol{\theta}}^2 L_{\mathcal{S}}(\boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}$

Taylor expansion :

$$\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \boldsymbol{H}\Delta\boldsymbol{\theta} + \mathcal{O}(||\Delta\boldsymbol{\theta}||^2)$$

Make $\Delta\boldsymbol{\theta} = r\boldsymbol{v}$

$$\boldsymbol{H}\boldsymbol{v} = \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}+r\boldsymbol{v}) - \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{r} + \mathcal{O}(r)$$

Make $\boldsymbol{v} = \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}$

$$\nabla_{\boldsymbol{\theta}}^2 L_{\mathcal{S}}(\boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||} \approx \frac{\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta} + r\frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}) - \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})}{r}$$

- Practical Implementation

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \frac{\lambda}{r} \cdot (\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta} + r \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}) - \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}))$$

$$= (1-\alpha)\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \alpha \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta} + r \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||})$$

$$\approx (1-\alpha)\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \alpha \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}+r\frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}}$$

Note :

1. The parameter λ denotes the penalty coefficient, controlling the degree of penalty. We should set it properly during training.
2. The parameter *r* is used for approximating the Hessian vector multiplication. It should be set smaller enough to ignore the big O item safely. But on the other side, as *r* becomes smaller, the other term $r\boldsymbol{v}$ will approach zero, causing a loss of precision on $\boldsymbol{v}$ in approximating $\boldsymbol{Hv}$.

- Practical Implementation

Algorithm 1:

Full procedures of our optimization scheme. This algorithm only shows gradient norm penalty with SGD as its base optimizer. For other optimizers or update strategies (such as Adam), one could add specific operations before step 8.

Algorithm 1 Optimization Scheme of Penalizing Gradient Norm

Input: Training set $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=0}^{N}$; loss function $L(\cdot)$; batch size B; learning rate η; total step T; balance coefficient α; approximation scalar r.
Parameter: Model parameters $\boldsymbol{\theta}$
Output: Optimized weight $\hat{\boldsymbol{\theta}}$

1: Parameter initialization $\boldsymbol{\theta}_0$.
2: **for** step $t = 1$ **to** T **do**
3: Get batch data pairs $\mathcal{B} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=0}^{B}$ sampled from training set $\mathcal{S}$.

Step 1

4: Calculate the gradient $\boldsymbol{g}_1 = \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})$ based on the batch samples.

Step 2

5: Add $r \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}$ on the current parameter $\boldsymbol{\theta}_t$, which makes $\boldsymbol{\theta}_t{}' = \boldsymbol{\theta}_t + r \frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}$.
6: Calculate the gradient $\boldsymbol{g}_2 = \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})$ at $\boldsymbol{\theta} = \boldsymbol{\theta}_t{}'$.

Step 3

7: Calculate the final gradient $\boldsymbol{g} = (1 - \alpha)\boldsymbol{g}_1 + \alpha \boldsymbol{g}_2$.
8: (SGD optimizer) Update parameter with final gradient, $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \cdot \boldsymbol{g}$.

9: **end for**
10: **return** Final optimization $\hat{\boldsymbol{\theta}}$.

- Connection with SAM

Gradient Norm Penalty

- Optimization target :

$$\min_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \lambda \cdot ||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||_2$$

- Optimization solution :

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = (1-\alpha)\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \alpha \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}+r\frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}}$$

Sharpness-Aware Minimization[1]

- Optimization target :

$$\min_{\boldsymbol{\theta}} \max_{||\boldsymbol{\epsilon}||_2 \leq r} L_{\mathcal{S}}(\boldsymbol{\theta}+\boldsymbol{\epsilon})$$

- Optimization solution :

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = 0 \cdot \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + 1 \cdot \nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}+r\frac{\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta})||}}$$

Apparently, SAM is a special implementation of gradient norm penalty, where $\alpha = 1$. In other words, SAM will create connections between the two independent parameters, where the coefficient λ will be always set equal to r. This limits our training and models could not get the best performance for $\alpha = 1$ in gradient norm penalty.

[1] Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *ICLR 2021*, 2021

- Training CNN Models on Cifar

Table :

Testing error rate of CNN models on Cifar10 and Cifar100 when training with the standard SGD, the SAM and our scheme, respectively.

It could be found that SAM could outperform the standard scheme, but could not be the best implementation in our scheme..

	Cifar10		Cifar100	
VGG16	Basic	Cutout	Basic	Cutout
Standard	7.07	5.31	28.78	**26.98**
SAM	6.91	**6.17**	28.62	27.13
Ours	**6.72**	**5.19**	**28.48**	27.07
VGG16-BN	Basic	Cutout	Basic	Cutout
Standard	$5.74_{\pm 0.09}$	$4.39_{\pm 0.07}$	$25.22_{\pm 0.31}$	$24.69_{\pm 0.25}$
SAM	$5.24_{\pm 0.08}$	$4.16_{\pm 0.11}$	$24.23_{\pm 0.29}$	$23.35_{\pm 0.33}$
Ours	$\mathbf{4.88_{\pm 0.12}}$	$\mathbf{4.02_{\pm 0.08}}$	$\mathbf{24.04_{\pm 0.18}}$	$\mathbf{23.07_{\pm 0.26}}$
WideResNet-28-10	Basic	Cutout	Basic	Cutout
Standard	$3.53_{\pm 0.10}$	$2.81_{\pm 0.07}$	$18.99_{\pm 0.12}$	$16.92_{\pm 0.10}$
SAM	$2.78_{\pm 0.07}$	$2.43_{\pm 0.13}$	$16.53_{\pm 0.13}$	$14.87_{\pm 0.16}$
Ours	$\mathbf{2.52_{\pm 0.09}}$	$\mathbf{2.16_{\pm 0.11}}$	$\mathbf{16.02_{\pm 0.19}}$	$\mathbf{14.28_{\pm 0.16}}$
WideResNet-SS 2×96	Basic	Cutout	Basic	Cutout
Standard	$2.82_{\pm 0.05}$	$2.39_{\pm 0.06}$	$17.19_{\pm 0.19}$	$15.85_{\pm 0.14}$
SAM	$2.37_{\pm 0.09}$	$2.11_{\pm 0.13}$	$15.22_{\pm 0.19}$	$14.32_{\pm 0.15}$
Ours	$\mathbf{2.28_{\pm 0.13}}$	$\mathbf{2.01_{\pm 0.10}}$	$\mathbf{14.93_{\pm 0.10}}$	$\mathbf{14.03_{\pm 0.17}}$
PyramidNet-SD	Auto Aug + Cutmix		Auto Aug + Cutmix	
Standard	$1.66_{\pm 0.11}$		$10.83_{\pm 0.14}$	
SAM	$1.41_{\pm 0.08}$		$10.33_{\pm 0.13}$	
Ours	$\mathbf{1.30_{\pm 0.07}}$		$\mathbf{10.12_{\pm 0.17}}$	

- Training ViT Models on Cifar

Table :

Testing error rate of ViT models on Cifar10 and Cifar100 when training with the standard SGD, the SAM and our scheme, respectively.

It could be found that SAM could outperform the standard scheme, but again it could not be the best implementation in our scheme..

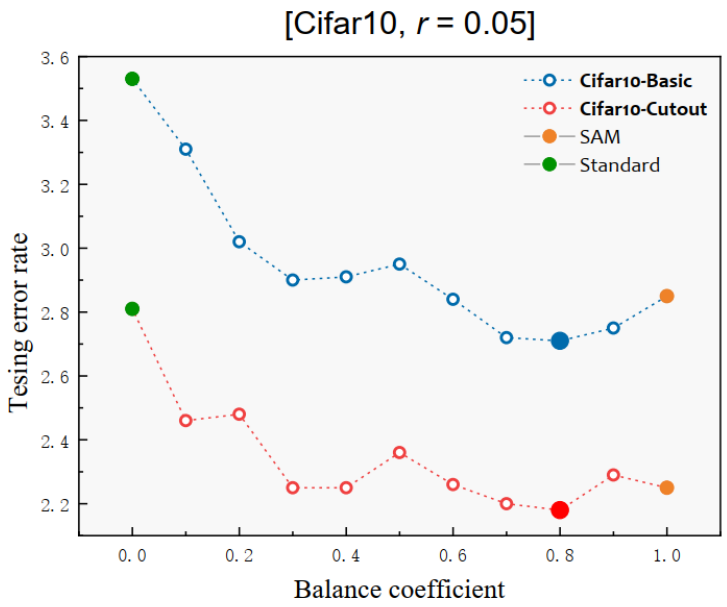
	Cifar10		Cifar100	
ViT-TI16	Basic	Heavy	Basic	Heavy
Standard	$15.92_{\pm 0.17}$	$14.68_{\pm 0.14}$	$40.21_{\pm 0.20}$	$38.93_{\pm 0.28}$
SAM	$15.33_{\pm 0.18}$	$13.77_{\pm 0.12}$	$38.89_{\pm 0.23}$	$37.61_{\pm 0.19}$
Ours	$\mathbf{14.75_{\pm 0.17}}$	$\mathbf{13.52_{\pm 0.21}}$	$\mathbf{38.58_{\pm 0.27}}$	$\mathbf{37.15_{\pm 0.21}}$
ViT-S16	Basic	Heavy	Basic	Heavy
Standard	$14.55_{\pm 0.14}$	$13.31_{\pm 0.11}$	$38.43_{\pm 0.19}$	$37.58_{\pm 0.22}$
SAM	$13.91_{\pm 0.18}$	$12.63_{\pm 0.09}$	$37.98_{\pm 0.23}$	$36.77_{\pm 0.25}$
Ours	$\mathbf{13.66_{\pm 0.16}}$	$\mathbf{12.29_{\pm 0.19}}$	$\mathbf{37.32_{\pm 0.28}}$	$\mathbf{36.59_{\pm 0.22}}$

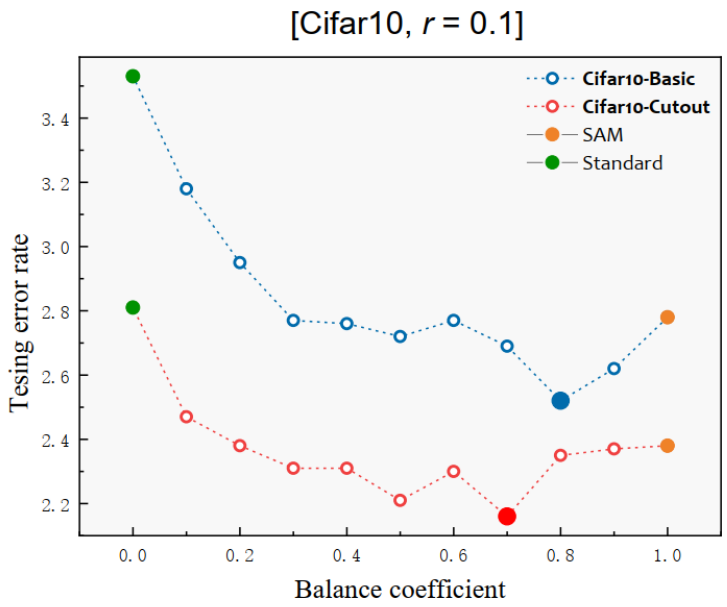
- Parameter Study

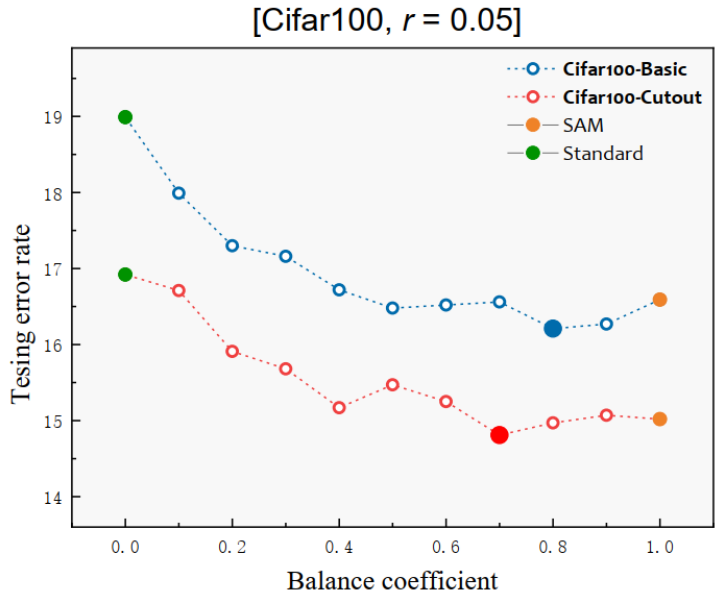
Figure :

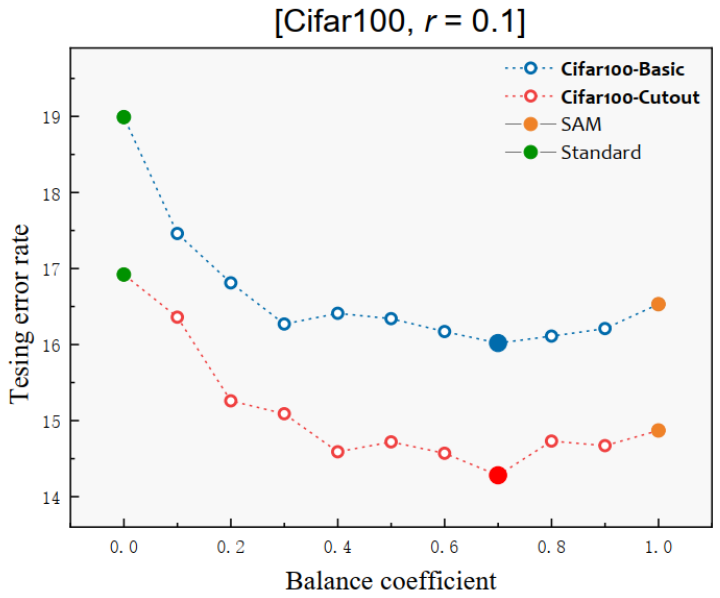
Testing error rate when trained with different hyperparameters. The points colored green are results using the standard scheme, and the points colored orange are results using the SAM scheme.

We recommend to set $\alpha = 0.8$ for achieving the best performance in practice, since most of our training could give the best results at this value.









Thanks for listening!