Sharpness-Aware Minimization (SAM)

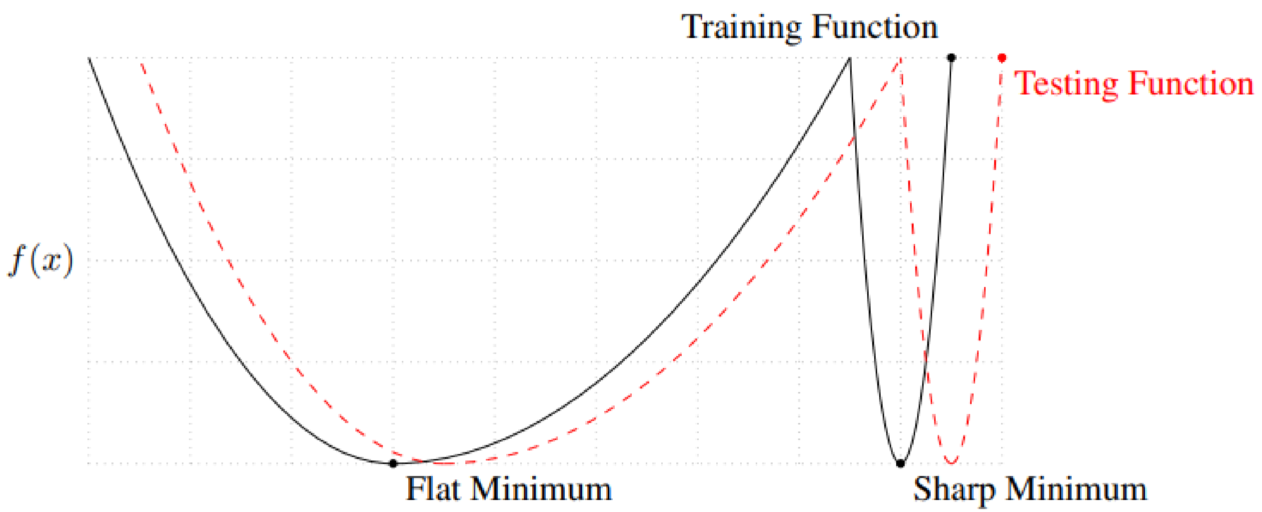
2025.02.13

KangJun Lee

Sharpness-Aware Minimization

Modern Neural Network 학습 과정에서…

Training Data와 Real Data를 같게 볼 수 있다고(i.i.d.) 가정하며 Training data의 Training Loss만 최소화시킴.



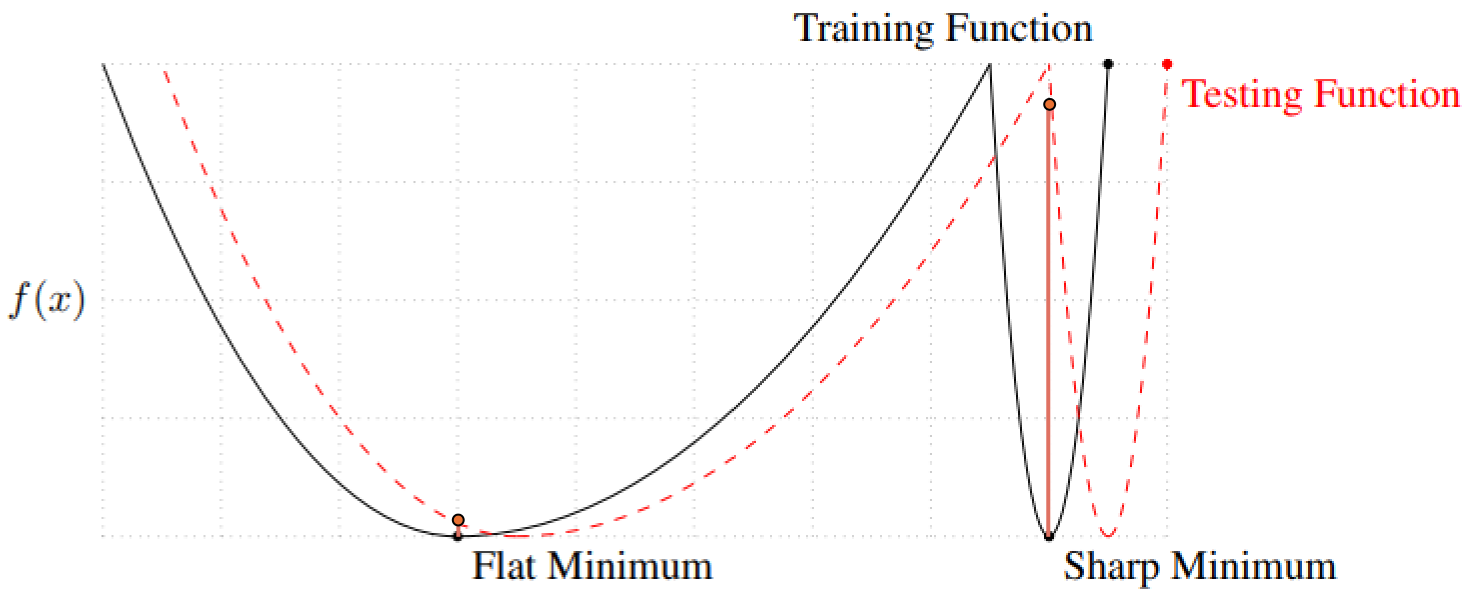
그러나, Training loss와 Real loss의 landscape가 다르며,

이로 인해 Training loss가 좋은 성능을 보장하지 못할 수 있다.

Sharpness-Aware Minimization

만약 Training distribution 이 Test distribution에서 shift 된 형태라면,

Training Loss에서 Flat한 Minima이 Sharp 한 Minima보다 더 좋은 Test Loss를 보일 것이다.



Idea) 그러면 Loss를 최소화할 뿐만 아니라 Flat한 minima를 찾으면 높은 Generalization ability를 보이지 않을까?

Sharpness-Aware Minimization

(Flat한지 인지하면서) (Loss를 Minimizing하는 Optimizer)

SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION

Pierre Foret *
Google Research
pierre.pforet@gmail.com

Ariel Kleiner
Google Research
akleiner@gmail.com

Hossein Mobahi
Google Research
hmobahi@google.com

Idea) Flat한 minima를 찾는 Optimizer는 Generalization ability를 향상시켜줄 것이다.

Sharpness-Aware Minimization

Theorem (stated informally) 1. *For any $\rho > 0$, with high probability over training set $\mathcal{S}$ generated from distribution $\mathscr{D}$,*

$$L_{\mathscr{D}}(\boldsymbol{w}) \leq \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\epsilon}) + h(\|\boldsymbol{w}\|_2^2/\rho^2),$$

where $h : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is a strictly increasing function (under some technical conditions on $L_{\mathscr{D}}(\boldsymbol{w})$).

To make explicit our sharpness term, we can rewrite the right hand side of the inequality above as

$$[\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\epsilon}) - L_{\mathcal{S}}(\boldsymbol{w})] + L_{\mathcal{S}}(\boldsymbol{w}) + h(\|\boldsymbol{w}\|_2^2/\rho^2).$$

-> PAC-Bayesian Bound를 활용해서 증명. (Skip the Proof)

Q. PAC-Bayesian Bound?

PAC (Probablistic Approximately Correct) Bound

Meaning : 높은 확률로 (high Probability), true and empirical risk의 차이를 근사한다. (Approximately).

(Real and Sample data loss)

Example) **Hoeffding Inequality**

Let $Z_1, \ldots, Z_N$ be i.i.d. random variables bounded in $[0,1]$. Then for all $\epsilon > 0$,

$$\mathbb{P}\left[\left|\frac{1}{N}\sum_{n=1}^{N} Z_n - \mathbb{E}[Z_n]\right| > \epsilon\right] \leq 2\exp(-2N\epsilon^2).$$

By writing $\delta = 2\exp(-2N\epsilon^2)$, we get, with probability $1-\delta$,

High Probability

$$\left|\frac{1}{N}\Sigma_{n=1}^{N} Z_n - \mathbb{E}[Z_n]\right| \leq \sqrt{\frac{1}{2N}\log\frac{2}{\delta}}$$

Approximately

PAC(Probablistic Approximately Correct) Bayesian-Bound

PAC 개념으로부터, Bayesian 개념을 차용하여 PAC Bayesian-Bound을 유도할 수 있다.

Recall) **Bayesian Theory**

$$\underbrace{P(A|B)}_{\text{Posterior Probability}} = \frac{\overbrace{P(B|A)}^{\text{Likelihood}}\,\overbrace{P(A)}^{\text{Class Prior Probability}}}{\underbrace{P(B)}_{\text{Predictor Prior Probability}}}$$

-> 기존에 주어진 확률(Prior Probability, Likelihood)를 통해

사후 확률(Posterior Probability)을 구할 수 있다.

PAC(Probablistic Approximately Correct) Bayesian-Bound

여기에서 Prior, Posterior의 개념을 차용해서,

prior P 를 Sample data와 관계없는 hypothesis(predictor)로 정의할 수 있다.

posterior Q(S) = Q 를 Sample data에 종속된 hypothesis(predictor)로 정의할 수 있다.

여기에서, Prior와 Posterior는 (Bayesian과 다르게) 사건이 아닌 Probability Distribution이다.

그러면 모든 hypothesis(predictor), h에 대해 real, empirical risk를 정의할 수 있다.

$$R_Q = \mathbb{E}_{h\sim Q}[R(h)] = \mathbb{E}_{h\sim Q}\left[\mathbb{E}_{(x,y)\sim D}[l((x,y),h)]\right]$$

$$r_{S,Q} = \mathbb{E}_{h\sim Q}[r_S(h)] = \mathbb{E}_{h\sim Q}[\frac{1}{N}\Sigma_{(x,y)\in S}\, l((x,y),h)]$$

PAC(Probablistic Approximately Correct) Bayesian-Bound

이를 이용하여, PAC-Bayesian을 정의할 수 있다.

Probablistic

$$R_Q \leq^{1-\delta} r_{S,Q} + \underline{f(Q,P,N,\delta)}$$

Approximately

Ex) Mcallester's Thm (1999)

For any $\ell \in \{0,1\}$, $D, \mathcal{H}$ and P a probability measure supported on $\mathcal{H}$, for $N \geq 8$,

Probablistic

$$R_Q \overset{1-\delta}{\leq} r_{S,Q} + \underline{\sqrt{\frac{\mathcal{D}_{\mathrm{KL}}[Q||P] + \log\sqrt{N} + \log\frac{2}{\delta}}{2N}}} \qquad (1)$$

Approximately

for all Q probability measures supported on $\mathcal{H}$.

(Mcallester's Thm이 실제 증명에 사용됨.)

Sharpness-Aware Minimization

Theorem (stated informally) 1. *For any $\rho > 0$, with high probability over training set $\mathcal{S}$ generated from distribution $\mathscr{D}$,*

$$L_{\mathscr{D}}(\boldsymbol{w}) \leq \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\epsilon}) + h(\|\boldsymbol{w}\|_2^2/\rho^2),$$

where $h : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is a strictly increasing function (under some technical conditions on $L_{\mathscr{D}}(\boldsymbol{w})$).

To make explicit our sharpness term, we can rewrite the right hand side of the inequality above as

$$\underbrace{[\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\epsilon}) - L_{\mathcal{S}}(\boldsymbol{w})]}_{\text{Sharpness Term}} + \underbrace{L_{\mathcal{S}}(\boldsymbol{w}) + h(\|\boldsymbol{w}\|_2^2/\rho^2)}_{\text{Loss \& Regularization}}.$$

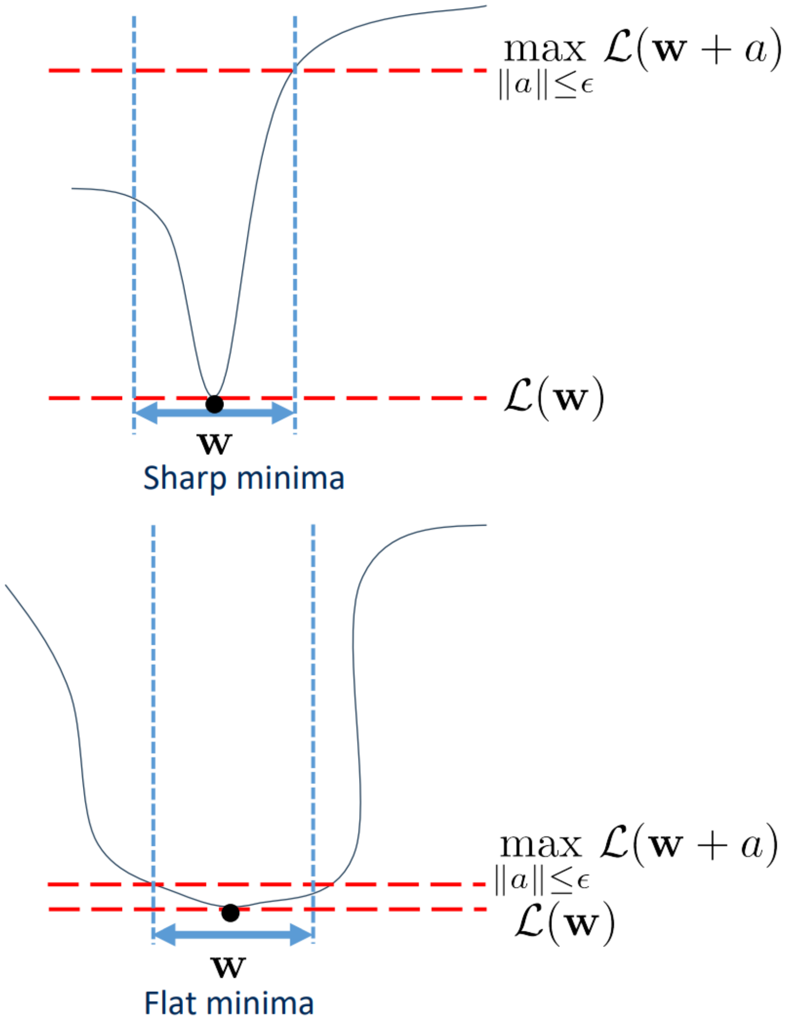
Motivation : loss와 sharpness를 동시에 최소화 하는 것이 실제 데이터에서 좋은 성능을 갖게 한다.

-> Good Generalization ability

Sharpness-Aware Minimization

$$\underbrace{[\max_{\|\boldsymbol{\epsilon}\|_2 \le \rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\epsilon}) - L_{\mathcal{S}}(\boldsymbol{w})]}_{\text{Sharpness Term}} + \underbrace{L_{\mathcal{S}}(\boldsymbol{w}) + h(\|\boldsymbol{w}\|_2^2/\rho^2)}_{\text{Loss \& Regularization}}.$$

Motivation : loss와 sharpness를 동시에 최소화하자!



Sharpness-Aware Minimization

결론적으로 둘 다 줄이기 위해서는, 아래 식을 최소화 해야한다.

$$\min_{w} L_S^{SAM}(w) + \lambda \|w\|_2^2$$

where

$$L_S^{SAM}(w) \triangleq \max_{\|\epsilon\|_p \le \rho} L_S(w+\epsilon)$$

$\max_{\|\epsilon\|_p \le \rho} L_S(w+\epsilon)$를 구하기 위해서, $\max_{\|\epsilon\|_p \le \rho} L_S(w+\epsilon) = L_S(w+\hat{\epsilon})$ 만족하는 $\hat{\epsilon}(w)$를 구해야한다.

$$\hat{\epsilon}(w) = \operatorname*{argmax}_{\|\epsilon\|_p \le \rho} L_S(w+\epsilon) \approx \operatorname*{argmax}_{\|\epsilon\|_p \le \rho} L_S(w) + \epsilon^T \nabla_w L_S(w) = \operatorname*{argmax}_{\|\epsilon\|_p \le \rho} \epsilon^T \nabla_w L_S(w)$$

$$\rightarrow \hat{\epsilon}(w) = \rho \frac{\nabla_w L_S(w)}{\left|\left|\nabla_w L_S(w)\right|\right|_2^2}$$

Gradient Calculation

Sharpness-Aware Minimization

그러면,

$$\nabla_w L_S^{SAM}(w) \approx \nabla_w L_s(w+\hat{\epsilon}) = \frac{d(w+\hat{\epsilon})}{dw} \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}}$$

$$\frac{d(w+\hat{\epsilon})}{dw} \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}} = \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}} + \cancel{\frac{d\hat{\epsilon}}{dw} \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}}}$$

To accelerate Computation

$$\therefore \nabla_w L_S^{SAM}(w) \approx \underline{\nabla_w L_s(w)\Big|_{w+\hat{\epsilon}}}$$

Gradient Calculation

전체 과정에서 Gradient가 2번 계산된다. -> More Computation Time

Sharpness-Aware Minimization

Algorithm)

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots (\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\epsilon}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\epsilon}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Algorithm 1: SAM algorithm

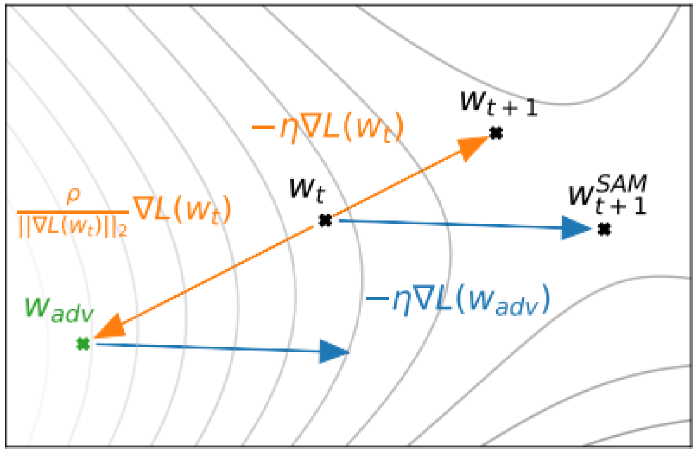


Figure 2: Schematic of the SAM parameter update.

Experimental Result (Image Classification)

Experimental-Setup)

Model	Augmentation
WRN-28-10 (200 epochs)	Basic
WRN-28-10 (200 epochs)	Cutout
WRN-28-10 (200 epochs)	AA
WRN-28-10 (1800 epochs)	Basic
WRN-28-10 (1800 epochs)	Cutout
WRN-28-10 (1800 epochs)	AA
Shake-Shake (26 2x96d)	Basic
Shake-Shake (26 2x96d)	Cutout
Shake-Shake (26 2x96d)	AA
PyramidNet	Basic
PyramidNet	Cutout
PyramidNet	AA
PyramidNet+ShakeDrop	Basic
PyramidNet+ShakeDrop	Cutout
PyramidNet+ShakeDrop	AA

다음의 NN들을 대상으로 SAM과 SGD의 정확도를 비교함.

SAM에서 ρ값은 {0.01, 0.02, 0.05, 0.1, 0.2, 0.5}의 값 중 가장 성능이 좋은 값으로 설정함. (Grid Search)

또한, SAM 학습 과정에서 8대의 Nvidia V100 GPU를 통해 병렬로 계산하여 각각의 data에서 계산한 ϵ 값의 평균을 구하여 update함. (뒤에 나올 m-Sharpness 방식과 유사함.)

모든 데이터에 대해서 basic data augmentations (horizontal flip, padding by four pixels, and random crop) 이 적용됨.

Experimental Result (Image Classification)

		CIFAR-10		CIFAR-100	
Model	Augmentation	SAM	SGD	SAM	SGD
WRN-28-10 (200 epochs)	Basic	**2.7**$_{\pm 0.1}$	3.5$_{\pm 0.1}$	**16.5**$_{\pm 0.2}$	18.8$_{\pm 0.2}$
WRN-28-10 (200 epochs)	Cutout	**2.3**$_{\pm 0.1}$	2.6$_{\pm 0.1}$	**14.9**$_{\pm 0.2}$	16.9$_{\pm 0.1}$
WRN-28-10 (200 epochs)	AA	**2.1**$_{\pm <0.1}$	2.3$_{\pm 0.1}$	**13.6**$_{\pm 0.2}$	15.8$_{\pm 0.2}$
WRN-28-10 (1800 epochs)	Basic	**2.4**$_{\pm 0.1}$	3.5$_{\pm 0.1}$	**16.3**$_{\pm 0.2}$	19.1$_{\pm 0.1}$
WRN-28-10 (1800 epochs)	Cutout	**2.1**$_{\pm 0.1}$	2.7$_{\pm 0.1}$	**14.0**$_{\pm 0.1}$	17.4$_{\pm 0.1}$
WRN-28-10 (1800 epochs)	AA	**1.6**$_{\pm 0.1}$	2.2$_{\pm <0.1}$	**12.8**$_{\pm 0.2}$	16.1$_{\pm 0.2}$
Shake-Shake (26 2x96d)	Basic	**2.3**$_{\pm <0.1}$	2.7$_{\pm 0.1}$	**15.1**$_{\pm 0.1}$	17.0$_{\pm 0.1}$
Shake-Shake (26 2x96d)	Cutout	**2.0**$_{\pm <0.1}$	2.3$_{\pm 0.1}$	**14.2**$_{\pm 0.2}$	15.7$_{\pm 0.2}$
Shake-Shake (26 2x96d)	AA	**1.6**$_{\pm <0.1}$	1.9$_{\pm 0.1}$	**12.8**$_{\pm 0.1}$	14.1$_{\pm 0.2}$
PyramidNet	Basic	**2.7**$_{\pm 0.1}$	4.0$_{\pm 0.1}$	**14.6**$_{\pm 0.4}$	19.7$_{\pm 0.3}$
PyramidNet	Cutout	**1.9**$_{\pm 0.1}$	2.5$_{\pm 0.1}$	**12.6**$_{\pm 0.2}$	16.4$_{\pm 0.1}$
PyramidNet	AA	**1.6**$_{\pm 0.1}$	1.9$_{\pm 0.1}$	**11.6**$_{\pm 0.1}$	14.6$_{\pm 0.1}$
PyramidNet+ShakeDrop	Basic	**2.1**$_{\pm 0.1}$	2.5$_{\pm 0.1}$	**13.3**$_{\pm 0.2}$	14.5$_{\pm 0.1}$
PyramidNet+ShakeDrop	Cutout	**1.6**$_{\pm <0.1}$	1.9$_{\pm 0.1}$	**11.3**$_{\pm 0.1}$	11.8$_{\pm 0.2}$
PyramidNet+ShakeDrop	AA	**1.4**$_{\pm <0.1}$	1.6$_{\pm <0.1}$	**10.3**$_{\pm 0.1}$	10.6$_{\pm 0.1}$

(Error-rate)

SGD와 비교했을 때, 모든 Data Augmentation 기법과, 다른 Network에 대해서도 유의미하게 성능 향상을 나타냄.

(여기에서, SAM의 1 epoch당 시간이 2배나 걸리기 때문에 SAM에서는 200, SGD에서는 epoch=400으로 학습함.)

Experimental Result (Image Classification)

Model	Epoch	SAM Top-1	SAM Top-5	Standard Training (No SAM) Top-1	Standard Training (No SAM) Top-5
ResNet-50	100	**22.5**$_{\pm 0.1}$	6.28$_{\pm 0.08}$	22.9$_{\pm 0.1}$	6.62$_{\pm 0.11}$
	200	**21.4**$_{\pm 0.1}$	5.82$_{\pm 0.03}$	22.3$_{\pm 0.1}$	6.37$_{\pm 0.04}$
	400	**20.9**$_{\pm 0.1}$	5.51$_{\pm 0.03}$	22.3$_{\pm 0.1}$	6.40$_{\pm 0.06}$
ResNet-101	100	**20.2**$_{\pm 0.1}$	5.12$_{\pm 0.03}$	21.2$_{\pm 0.1}$	5.66$_{\pm 0.05}$
	200	**19.4**$_{\pm 0.1}$	4.76$_{\pm 0.03}$	20.9$_{\pm 0.1}$	5.66$_{\pm 0.04}$
	400	**19.0**$_{\pm <0.01}$	4.65$_{\pm 0.05}$	22.3$_{\pm 0.1}$	6.41$_{\pm 0.06}$
ResNet-152	100	**19.2**$_{\pm <0.01}$	4.69$_{\pm 0.04}$	20.4$_{\pm <0.0}$	5.39$_{\pm 0.06}$
	200	**18.5**$_{\pm 0.1}$	4.37$_{\pm 0.03}$	20.3$_{\pm 0.2}$	5.39$_{\pm 0.07}$
	400	**18.4**$_{\pm <0.01}$	4.35$_{\pm 0.04}$	20.9$_{\pm <0.0}$	5.84$_{\pm 0.07}$

Table 2: Test error rates for ResNets trained on ImageNet, with and without SAM.

(Without SAM) / SGD with cosine learning-schedule, momentum with 0.9, weight-decay = 0.005.

앞의 예시보다 비교적 무거운 모델인 ResNet에 대해서도 SAM이 더 좋은 성능을 보임.

심지어, SAM은 epoch가 많아지더라도 overfitting 현상이 일어나지 않은 것을 볼 수 있음. (Regularization Term)

Experimental Result (Fine-Tuning)

Dataset	EffNet-b7 + SAM	EffNet-b7	Prev. SOTA (ImageNet only)	EffNet-L2 + SAM	EffNet-L2	Prev. SOTA
FGVC_Aircraft	$6.80_{\pm 0.06}$	$8.15_{\pm 0.08}$	**5.3** (TBMSL-Net)	$\mathbf{4.82}_{\pm 0.08}$	$5.80_{\pm 0.1}$	5.3 (TBMSL-Net)
Flowers	$\mathbf{0.63}_{\pm 0.02}$	$1.16_{\pm 0.05}$	0.7 (BiT-M)	$\mathbf{0.35}_{\pm 0.01}$	$0.40_{\pm 0.02}$	0.37 (EffNet)
Oxford_IIIT_Pets	$\mathbf{3.97}_{\pm 0.04}$	$4.24_{\pm 0.09}$	4.1 (Gpipe)	$\mathbf{2.90}_{\pm 0.04}$	$3.08_{\pm 0.04}$	4.1 (Gpipe)
Stanford_Cars	$5.18_{\pm 0.02}$	$5.94_{\pm 0.06}$	**5.0** (TBMSL-Net)	$4.04_{\pm 0.03}$	$4.93_{\pm 0.04}$	**3.8** (DAT)
CIFAR-10	$\mathbf{0.88}_{\pm 0.02}$	$0.95_{\pm 0.03}$	1 (Gpipe)	$\mathbf{0.30}_{\pm 0.01}$	$0.34_{\pm 0.02}$	0.63 (BiT-L)
CIFAR-100	$\mathbf{7.44}_{\pm 0.06}$	$7.68_{\pm 0.06}$	7.83 (BiT-M)	$\mathbf{3.92}_{\pm 0.06}$	$4.07_{\pm 0.08}$	6.49 (BiT-L)
Birdsnap	$\mathbf{13.64}_{\pm 0.15}$	$14.30_{\pm 0.18}$	15.7 (EffNet)	$\mathbf{9.93}_{\pm 0.15}$	$10.31_{\pm 0.15}$	14.5 (DAT)
Food101	$7.02_{\pm 0.02}$	$7.17_{\pm 0.03}$	7.0 (Gpipe)	$\mathbf{3.82}_{\pm 0.01}$	$3.97_{\pm 0.03}$	4.7 (DAT)
ImageNet	$15.14_{\pm 0.03}$	15.3	**14.2** (KDforAA)	$\mathbf{11.39}_{\pm 0.02}$	11.8	11.45 (ViT)

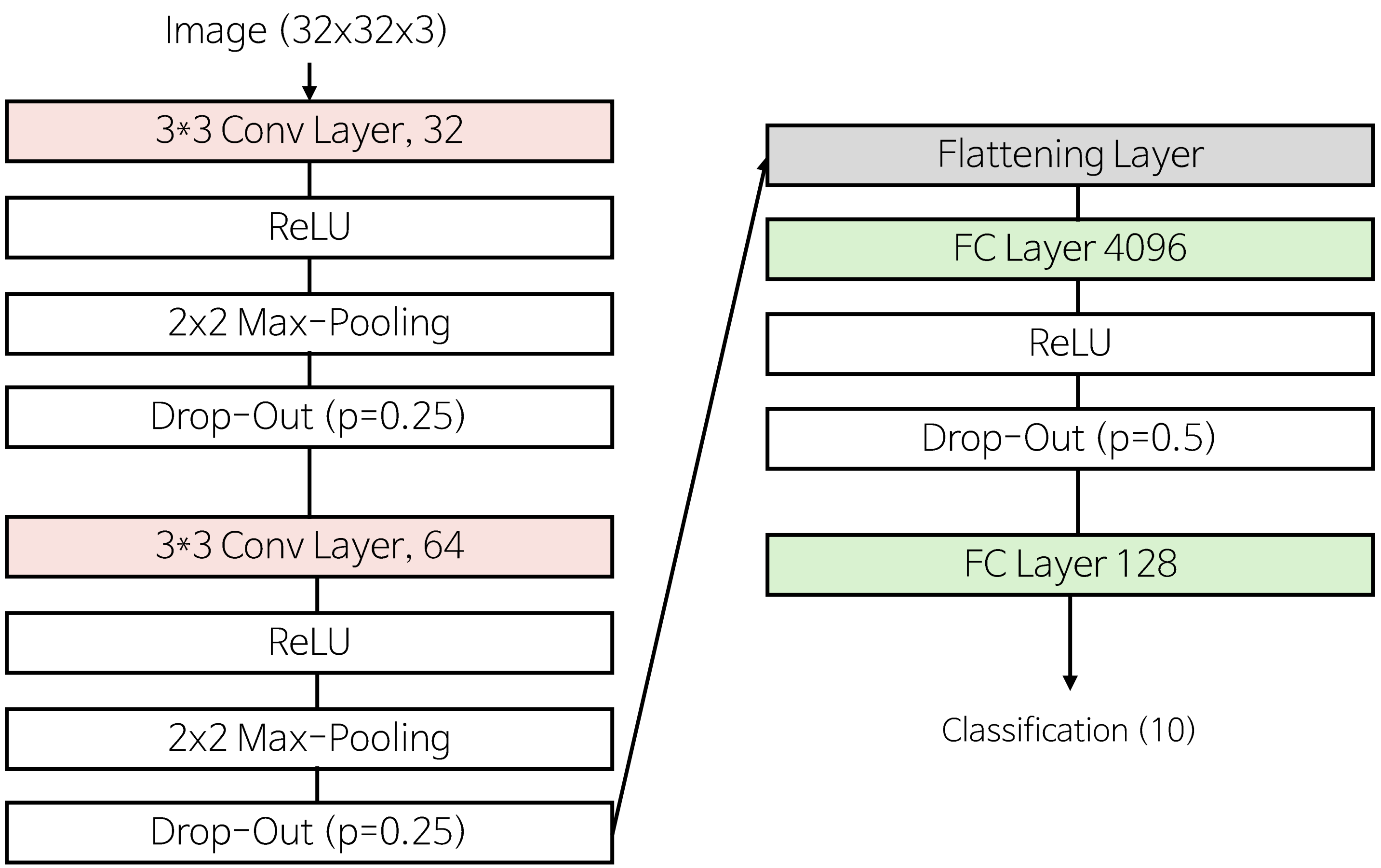
(Error-rate)

Fine-Tuning의 경우에서도 SAM이 기존에 쓰였던 방법에 비해 이점을 보임.

(여기에서, EffNet-b7과 EffNet-L2는 각각 RandAugment와 NoisyStudent 기법으로 학습된 모델의 checkpoint 를 가져와서 Fine-Tuning 했다고 함.)

Experimental Result (Self-Experiment)

Experimental-Setup)



학습은 개인 노트북으로 진행.

(RTX 1650Ti)

Task : CIFAR10 Classification

Github에 있는 코드를 이용하여 학습.

Data augmentation 적용, momentum = 0.9,

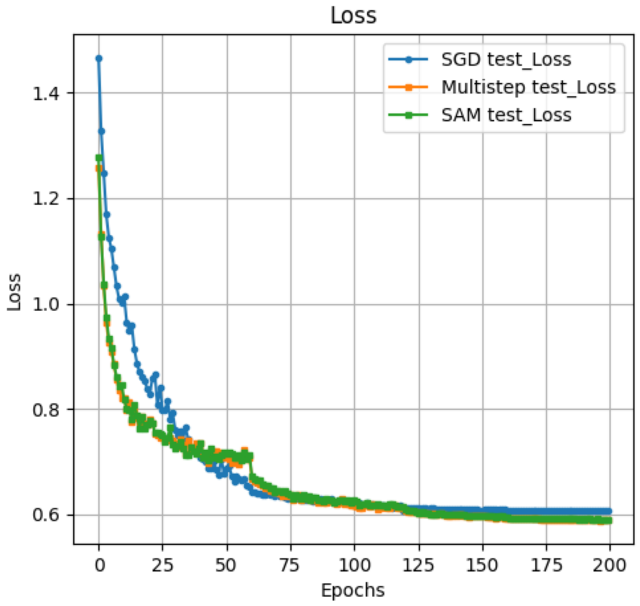
Batch_size = 64

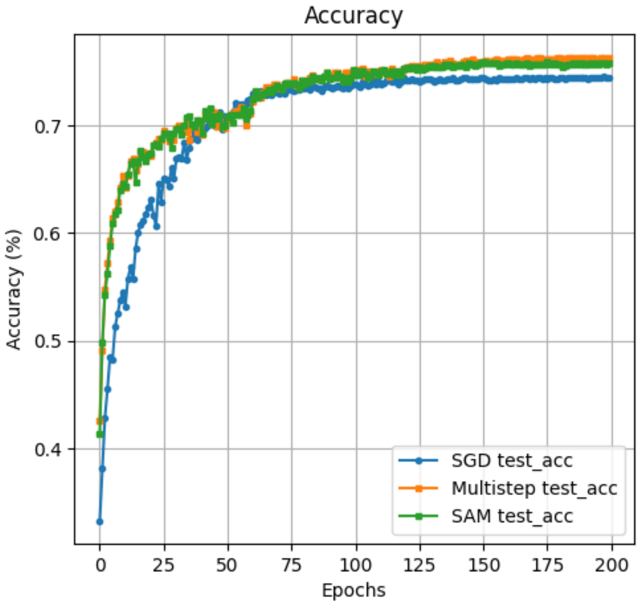
Weight_decay = 0.0005

Step_LR 적용

ρ = 0.5

Experimental Result (Self-Experiment)





Optimizer	Loss	Accuracy(%)
SGD	0.60742	74.42
SAM_single	0.59017	75.74
SAM_multi	0.58964	76.27

앞의 예시와 같이 가벼운 모델에서도 SAM이 유의미한 성능 향상을 보임.

Multi-Step(double) SAM 같은 경우 SAM과 비교하여 약간의 성능 향상을 보임.

특히, 학습 후반부에 Multi-Step SAM이 기존 SAM에 비해 더 좋은 성능을 보임.

Experimental Result (Self-Experiment)

Q. Multi-Step SAM도 Second-order term을 무시할 수 있을까?

$$\nabla L_S^{SAM}(w) \approx \nabla_w L_S(w)\Big|_{w+\sum_{n=1}^{m}\widehat{\epsilon_n}} + \frac{d(\sum_{n=1}^{m}\widehat{\epsilon_n})}{dw}\nabla_w L_S(w)\Big|_{w+\sum_{n=1}^{m}\widehat{\epsilon_n}}$$

$\frac{d(\sum_{n=1}^{m}\widehat{\epsilon_n})}{dw}\nabla_w L_S(w)|_{w+\sum_{n=1}^{m}\widehat{\epsilon_n}}$ 부분도 똑같이 무시될 수 있는데, 이 부분이 Single-Step SAM과 다르게 나타날 수 있지 않을까?

그러나, $||\sum_{n=1}^{m}\widehat{\epsilon_n}|| \le \rho$, 즉 permutation은 항상 $\mathrm{B}_\rho(w)$ 안에 존재한다.

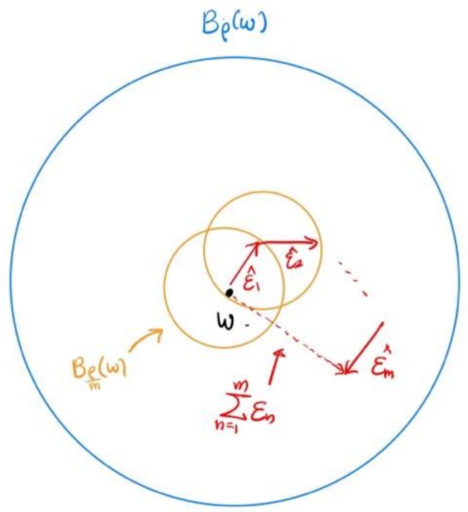
Single-Step에서 $\frac{d\hat{\epsilon}}{dw}\nabla_w L_S(w)|_{w+\hat{\epsilon}}$ 가 무시될 수 있고, $||\hat{\epsilon}|| \le \rho$ 임을 고려하면,

같은 이유로 $\frac{d(\sum_{n=1}^{m}\widehat{\epsilon_n})}{dw}\nabla_w L_S(w)|_{w+\sum_{n=1}^{m}\widehat{\epsilon_n}}$도 무시될 수 있다.

m-step SAM

$\nabla \hat{L}^{SAM}(w) = \nabla \hat{L}(w+\sum_{n=1}^{m}\hat{\epsilon}_n) \approx \frac{d(w+\sum_{n=1}^{m}\hat{\epsilon}_n)}{dw}\nabla \hat{L}(w)|_{w+\sum_{n=1}^{m}\hat{\epsilon}_n}$

=> consider $\frac{d(\sum_{n=1}^{m}\hat{\epsilon}_n)}{dw}\nabla \hat{L}(w)|_{w+\sum_{n=1}^{m}\hat{\epsilon}_n}$. (here, $||\hat{\epsilon}_n|| = \frac{\rho}{m}$)



but $||\sum_{n=1}^{m}\hat{\epsilon}_n|| \le \rho$. => Considering we can ignore $||\hat{\epsilon}|| \le \rho$,

$\frac{d\hat{\epsilon}}{dw}\nabla \hat{L}(w)|_{w+\hat{\epsilon}}$, $\frac{d(\sum_{n=1}^{m}\hat{\epsilon}_n)}{dw}\nabla \hat{L}(w)|_{w+\sum_{n=1}^{m}\hat{\epsilon}_n}$. can be ignored.

Experimental Result (Self-Experiment)

Q. Second-Order Term이 무시될 수 있는 이유?

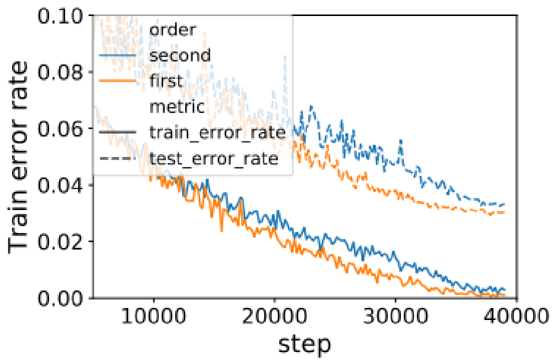


Figure 4: Training and test error for the first and second order version of the algorithm.

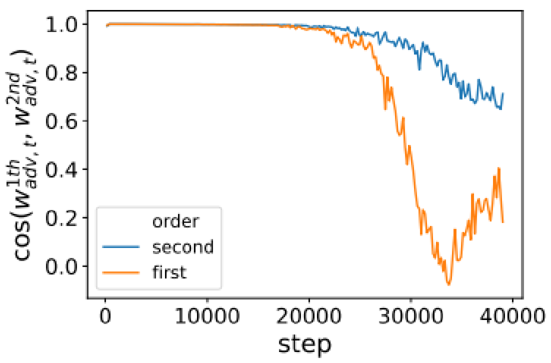


Figure 5: Cosine similarity between the first and second order updates.

논문에서는 실험적으로 보여줌.

학습 중반까지, First order update SAM의 학습 방향이

Second order update SAM과 방향이 거의 같음을 볼 수 있다.

중반 이후에는 방향이 달라지지만,

second-order을 생략한 SAM 모델이 더 낮은 train error를 보임.

Experimental Result (m-Sharpness)

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Algorithm 1: SAM algorithm

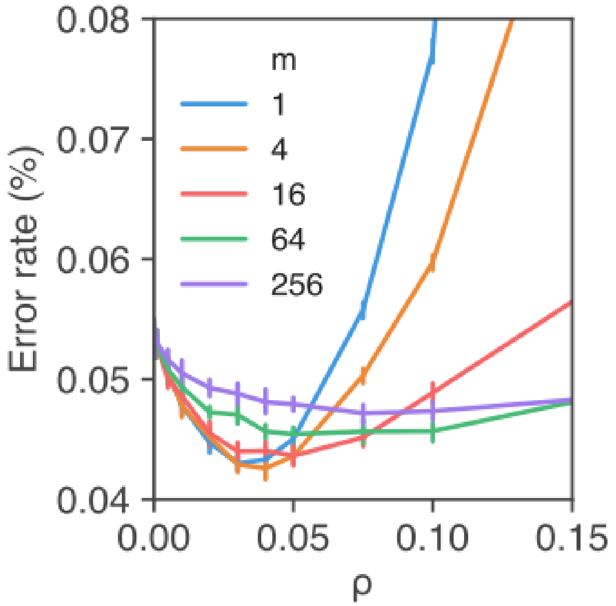
기존 알고리즘)

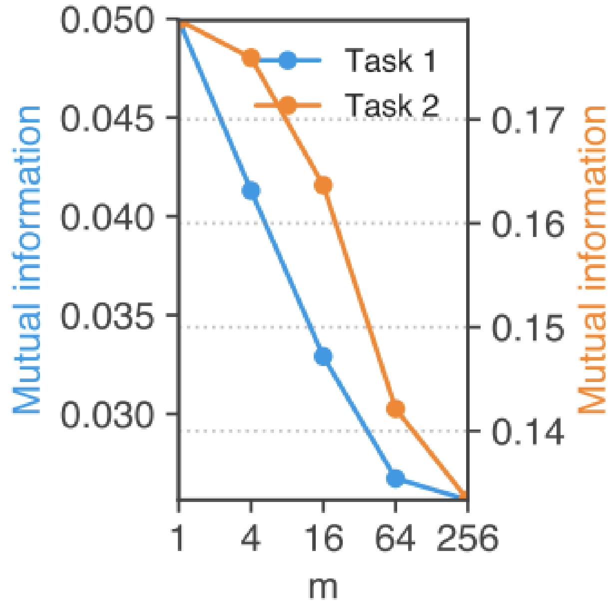
Batch를 나누어서 각 Batch 마다 ϵ을 구한 뒤 weight를 update.

m-Sharpness)

각 배치마다 독립적으로 계산한 뒤 각 배치마다의 ϵ의 평균 pertubation을 구하여 그 값을 바탕으로 weight를 update.

Experimental Result (m-Sharpness)





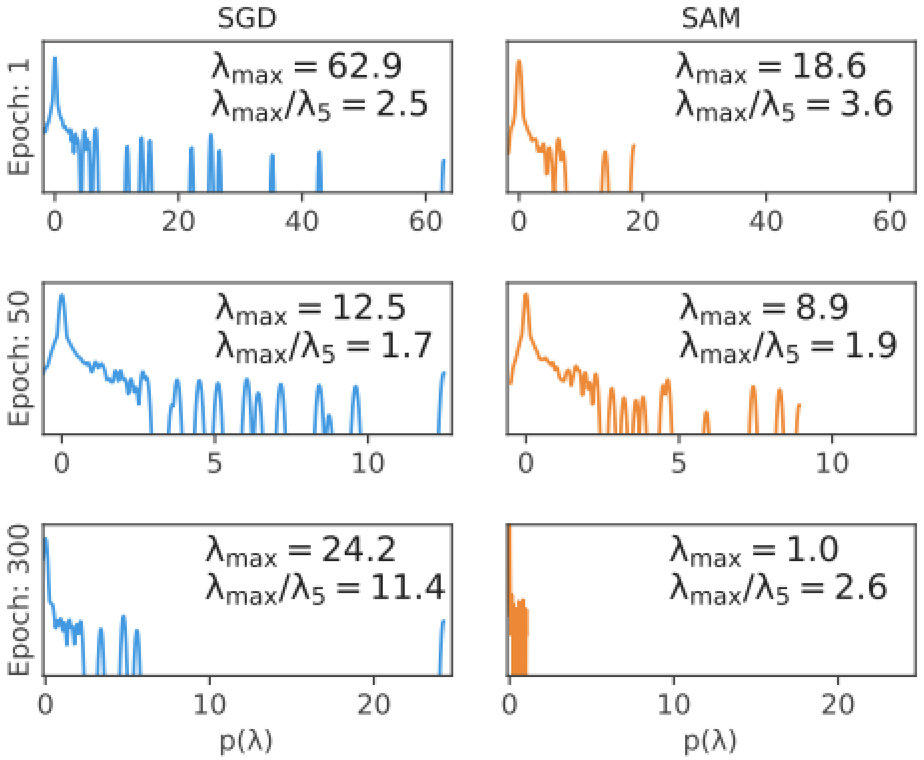
(left) Test error as a function of ρ for different values of m.
(right) Predictive power of m-sharpness for the generalization gap, for different values of m (higher means the sharpness measure is more correlated with actual generalization gap).

배치 크기, m의 값이 작아질수록 더 좋은 generalization ability를 보여줌.

이 사실은 대규모 학습에서 필요한 병렬학습에도 적합하다는 사실을 보여줌.

또한, m의 값이 작을수록 sharpness-term이 generalization에 영향을 더 많이 끼친다는 것을 보여줌.

Experimental Result (Hessian-Spectra)



Evolution of the spectrum of the Hessian during training of a model with standard SGD (lefthand column) or SAM (righthand column).

마지막으로, Epoch에 따라 Hessian Matrix의 eigenvalue를 비교함.

Hessian의 Eigenvalue가 작을수록 평탄하다고 볼 수 있는데, SGD와 비교했을 때 SAM의 eigenvalue가 월등히 작음을 알 수 있음.

그렇기에, loss 뿐만 아니라 Flatness까지 잘 학습되고 있다는 것을 알 수 있음.

Thank you!