Flat Minima Optimizer for generalization performance

Data Mining & Quality Analytics Lab.

정용태

발표자 소개

❖ 정용태 (Yongtae Jeong)

- 고려대학교 산업경영공학과 대학원 재학
- Data Mining & Quality Analytics Lab. (김성범 교수님)
- M. S. Student (2023.03 ~ Present)

❖ Research Interest

- Domain Adaptation
- Domain Generalization

❖ Contact

- yongtc2005@korea.ac.kr

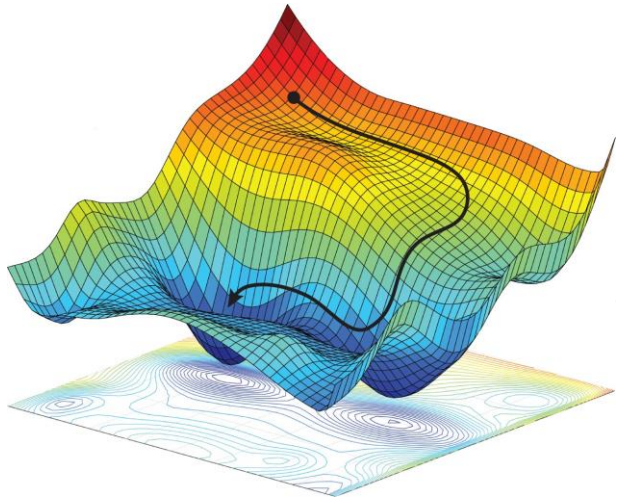
Introduction

Background of Flat Minima

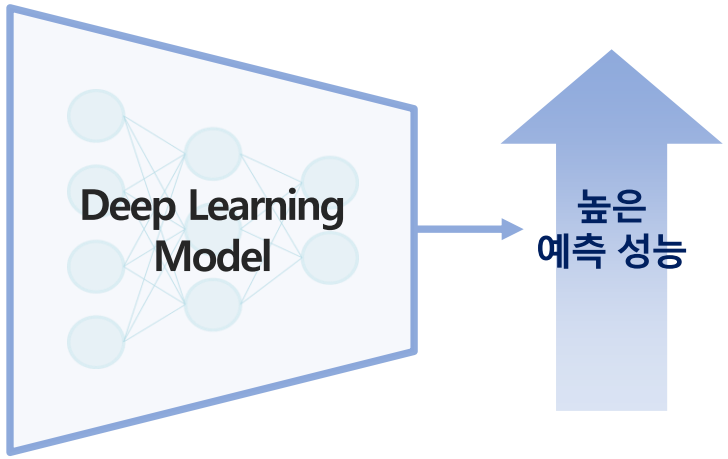
❖ Training of Deep Learning Model

- 딥러닝 모델은 다수의 학습 데이터를 활용해서 높은 예측 성능을 보이는 모델 구축
- 다양한 손실함수 (MSE, cross entropy, etc.) 및 optimizer (SGD, Adam, etc.)를 통해서 모델을 학습하고자 함

Big Data



Loss Function & Optimizer

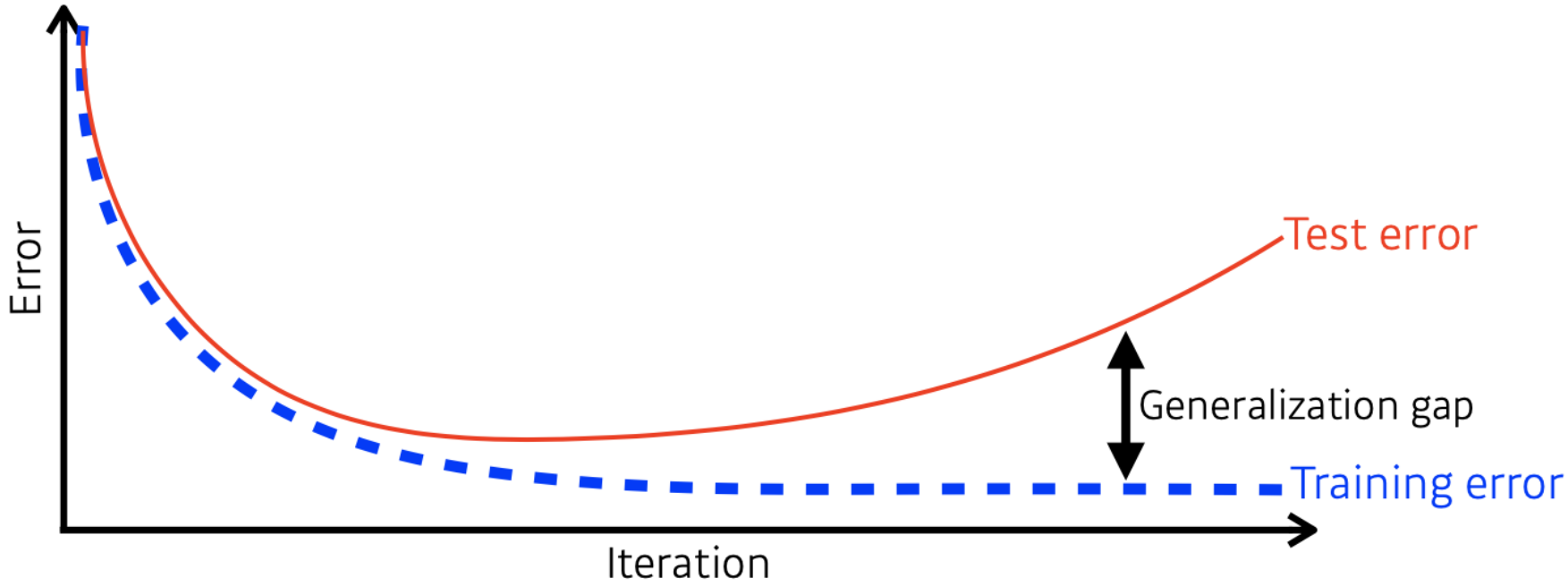


Introduction

Background of Flat Minima

❖ Generalization Gap

- Training error와 test error의 차이로 인해서 학습과 테스트 결과의 차이가 발생함
- Generalization Gap을 줄임으로써 모델의 Generalization 성능 향상이 필요

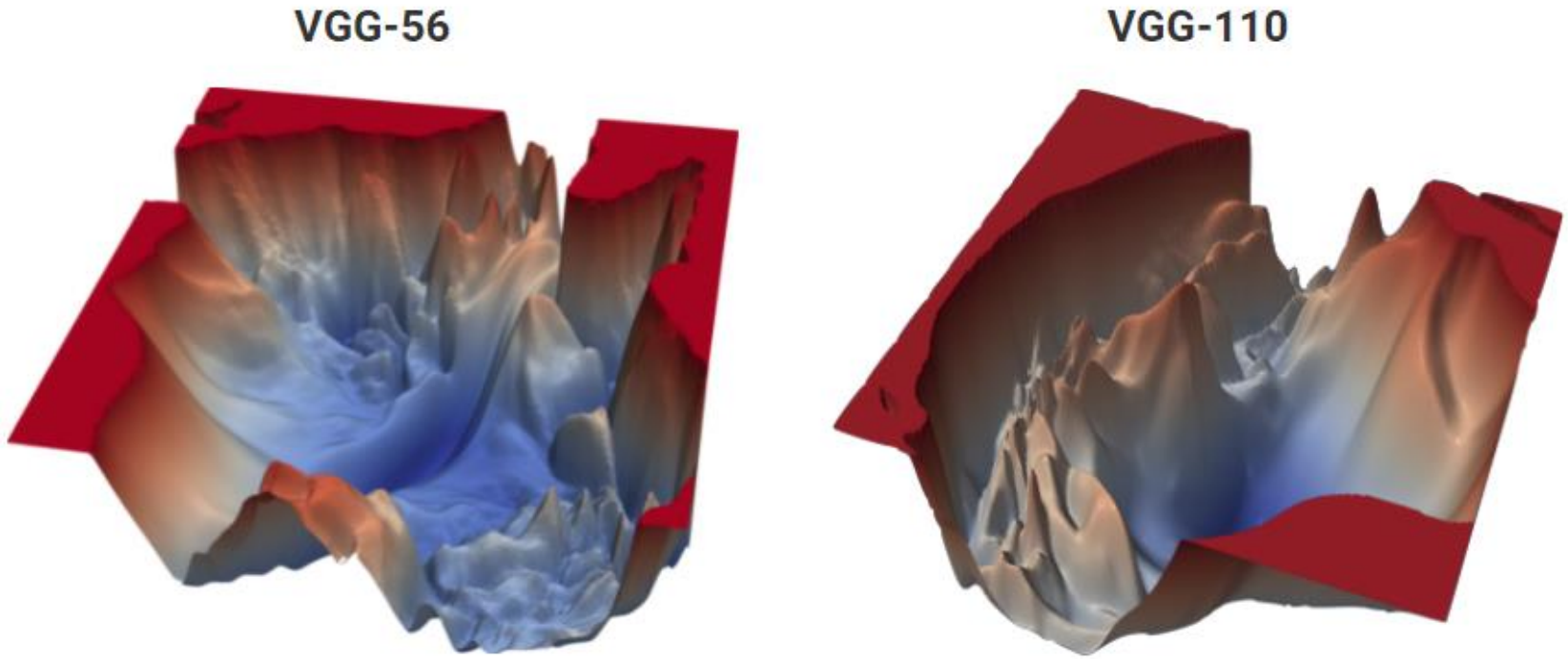


Introduction

Background of Flat Minima

❖ Deep Learning Model

- 하지만, 딥러닝 모델은 loss landscape가 complex하고 non-convex 함
- 다수의 local Minima가 존재하며, 이에 따라 각각의 Generalization 성능이 서로 다름

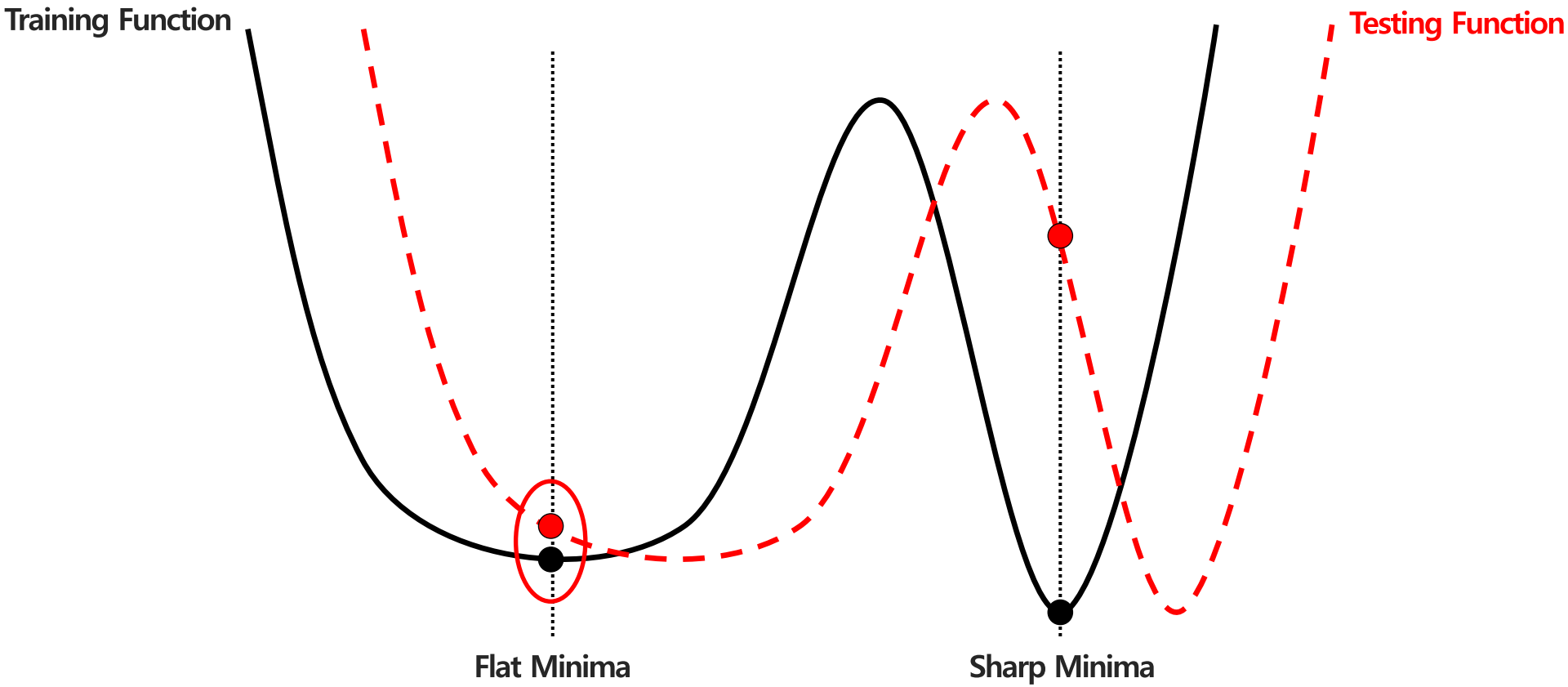


Introduction

Background of Flat Minima

❖ Flat Minima for generalization performance

- 일반화 성능이 높은 local Minima를 찾음으로써, Generalization Gap을 줄일 수 있음
- Flat Minima는 Sharp Minima보다 Generalization Gap이 작은 특징을 갖음



Introduction

❖ Flat Minima Optimizer for generalization performance

- 모델을 학습함에 있어 loss의 Flat Minima를 찾는 것은 모델 일반화 성능 향상에 기여
- 따라서, 본 세미나에서는 대표적인 Flat Minima Optimizer 2개와 벤치마크 적용 결과를 아래와 같이 소개

1. Averaging weights leads to wider optima and better generalization (UAI, 2018)

➢ 모델 가중치 평균화를 통해 Flat minima를 찾음으로써 일반화 성능 향상

2. Sharpness-aware minimization for efficiently improving generalization (ICLR, 2021)

➢ Loss의 Sharpness를 고려한 목적함수를 통해 Flat Minima를 찾음으로써 일반화 성능 향상

3. When Do Flat Minima Optimizers Work? (NeurIPS, 2022)

➢ 다양한 벤치마크 적용 인사이트 제공 및 WASAM 방법론 제안

Averaging weights leads to wider optima and better generalization

Methods

1. Stochastic Weight Averaging

❖ Averaging weights leads to wider optima and better generalization[1]

- 모델 가중치 평균화를 통해 Flat minima를 찾음으로써 일반화 성능 향상 (UAI 2018, 24년 12월 기준 1766회 인용)
- Computational cost의 증가가 거의 없고, 적용이 쉬운 Stochastic Weight Averaging (SWA) 방법론 소개

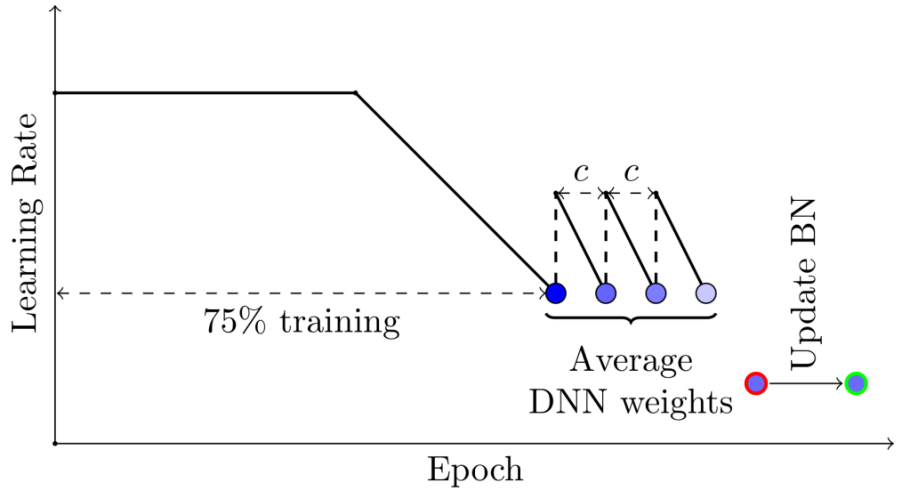
Averaging Weights Leads to Wider Optima and Better Generalization

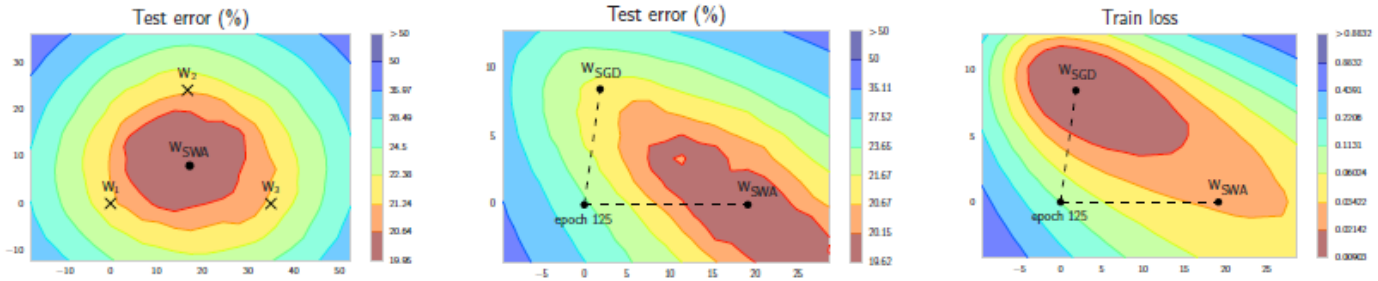
Pavel Izmailov*[1] Dmitrii Podoprikhin*[2,3] Timur Garipov*[4,5] Dmitry Vetrov[2,3] Andrew Gordon Wilson[1]
[1]Cornell University, [2]Higher School of Economics, [3]Samsung-HSE Laboratory,
[4]Samsung AI Center in Moscow, [5]Lomonosov Moscow State University

Abstract

Deep neural networks are typically trained by optimizing a loss function with an SGD variant, in conjunction with a decaying learning rate, until convergence. We show that simple averaging of multiple points along the trajectory of SGD, with a cyclical or constant learning rate, leads to better generalization than conventional training. We also show that this *Stochastic Weight Averaging* (SWA) procedure finds much flatter solutions than SGD, and approximates the recent *Fast Geometric Ensembling* (FGE) approach with a single model. Using SWA we achieve notable improvement in test accuracy over conventional SGD training on a range of state-of-the-art residual networks, PyramidNets, DenseNets, and Shake-Shake networks on CIFAR-10, CIFAR-100, and ImageNet. In short, SWA is extremely easy to implement, improves generalization, and has almost no computational overhead.

we see that the weights of the networks ensembled by FGE are on the periphery of the most desirable solutions. This observation suggests it is promising to average these points in *weight space*, and use a network with these averaged weights, instead of forming an ensemble by averaging the outputs of networks in *model space*. Although the general idea of maintaining a running average of weights traversed by SGD dates back to Ruppert [1988], this procedure is not typically used to train neural networks. It is sometimes applied as an exponentially decaying running average in combination with a decaying learning rate (where it is called an exponential moving average), which smooths the trajectory of conventional SGD but does not perform very differently. However, we show that an equally weighted average of the points traversed by SGD with a cyclical or high constant learning rate, which we refer to as *Stochastic Weight Averaging* (SWA), has many surprising and promising features for training deep neural networks, leading to a better understanding of the geometry of their loss surfaces. Indeed, SWA with cyclical or constant learning rates can be used as a drop-in replacement for standard SGD training of multilayer networks — but with improved generalization and essentially no overhead. In particular:





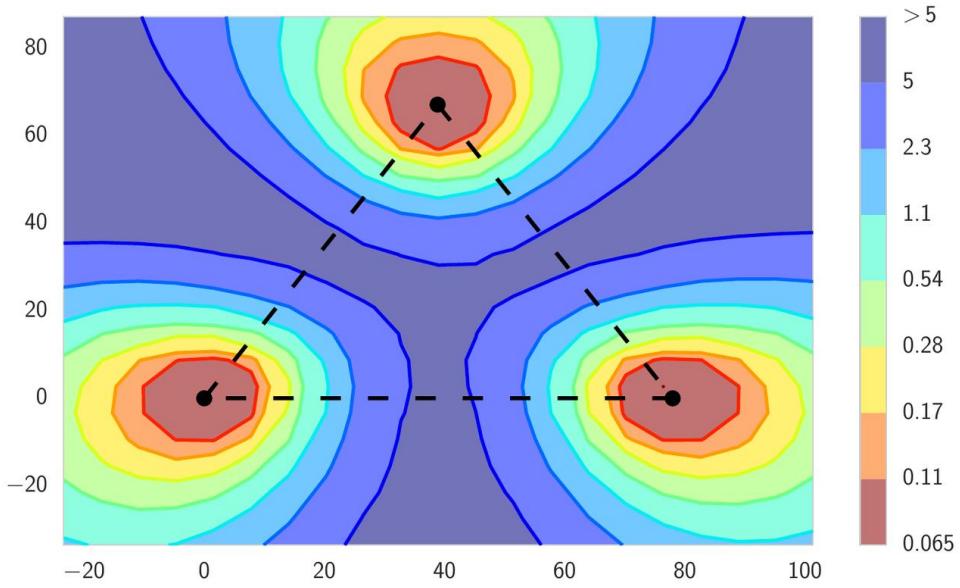
[1] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407.

Methods

1. Stochastic Weight Averaging

❖ Observation from Stochastic Gradient Descent (SGD)

- SGD를 통해 찾은 여러 local minima들은 서로 고립될 것으로 상상됨

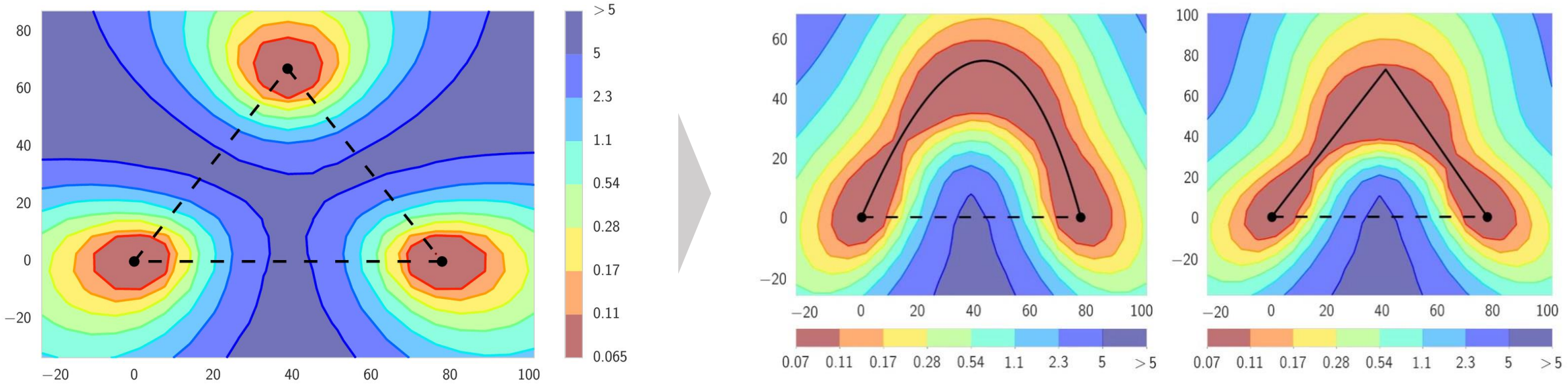


Methods

1. Stochastic Weight Averaging

❖ Observation from Stochastic Gradient Descent (SGD)

- SGD를 통해 찾은 여러 local minima들은 서로 고립될 것으로 상상됨
- **Mode connectivity[2]:** 여러 local minima들은 일정한 경로로 연결됨을 실험적으로 밝힘
- 위 관찰에 근거하여 Fast Geometric Ensembling (FGE) 방법이 제안됨



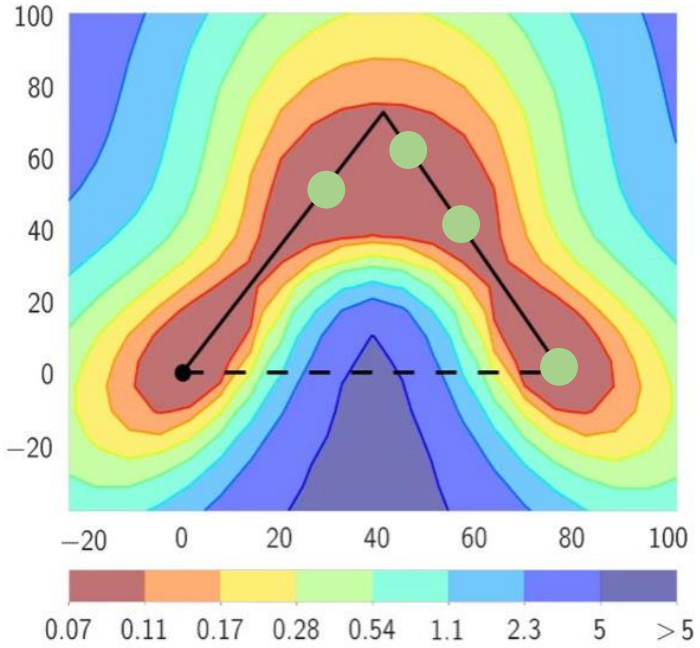
[2] Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., & Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. Advances in neural information processing systems, 31.

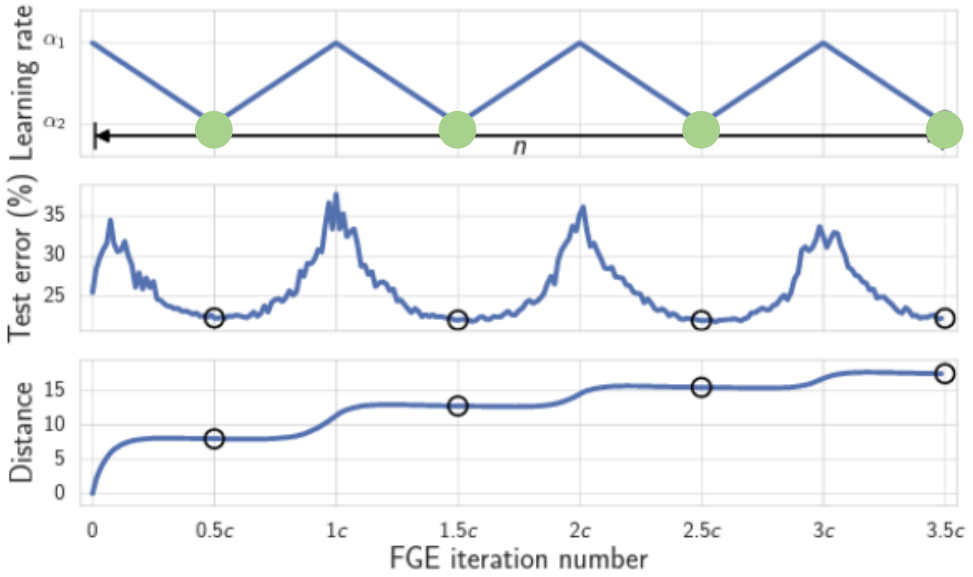
Methods

1. Stochastic Weight Averaging

❖ Related Work: Fast Geometric Ensembling (FGE)[2]

- DNN을 학습시킬 때 weight space에 위치한 여러 인접한 모델들을 sampling하여 앙상블 하는 기법
- piecewise linear cyclical learning rate 기법을 활용하여 local minima를 갖는 여러 모델들을 학습





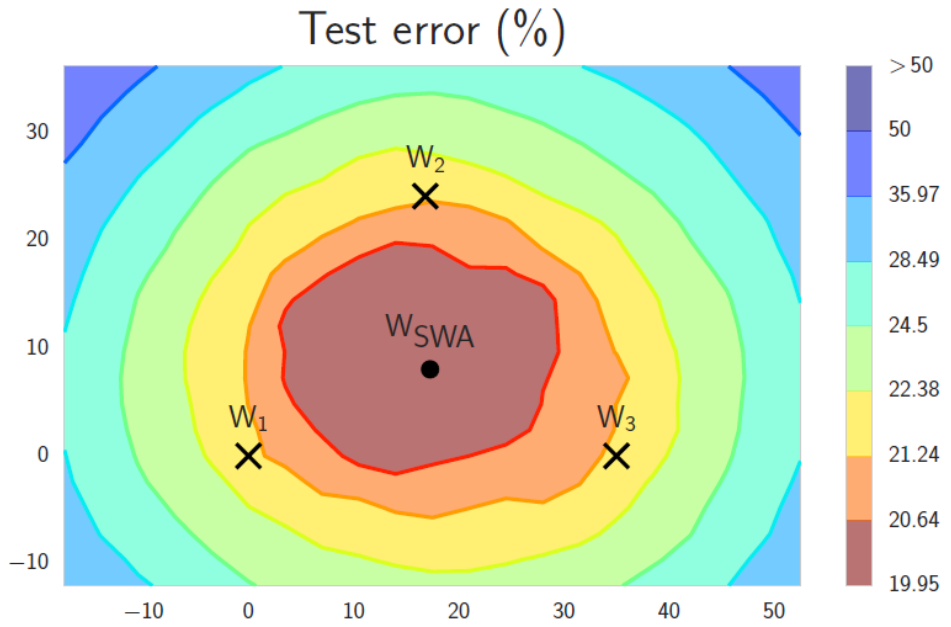
[2] Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., & Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. Advances in neural information processing systems, 31.

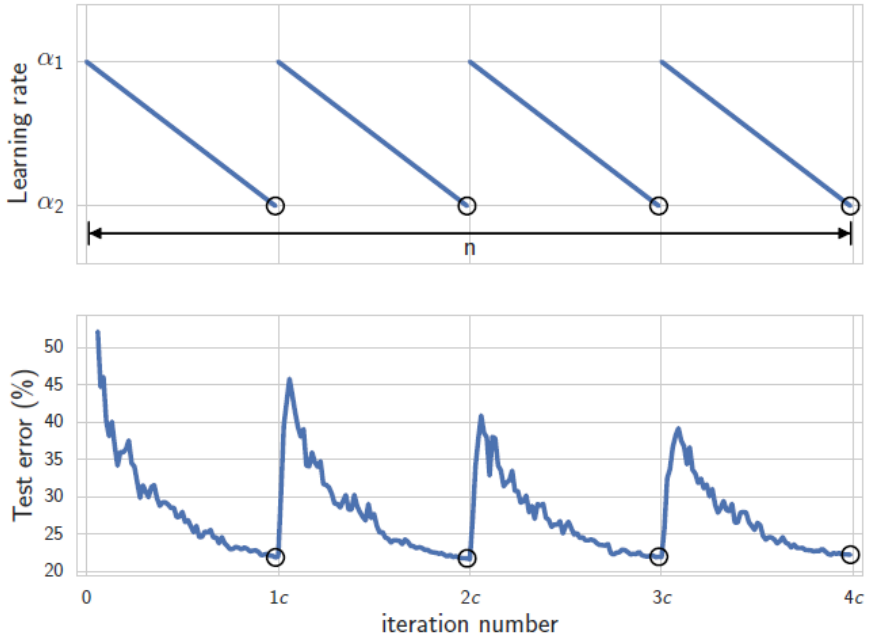
Methods

1. Stochastic Weight Averaging

❖ Stochastic Weight Averaging (SWA)

- SGD로 학습되는 모델 weight를 시간 축으로 누적 평균하여 하나의 모델을 구축
- SGD를 통해 찾은 모델들을 평균함으로써 가장 높은 일반화 성능의 모델을 찾는 것이 가능



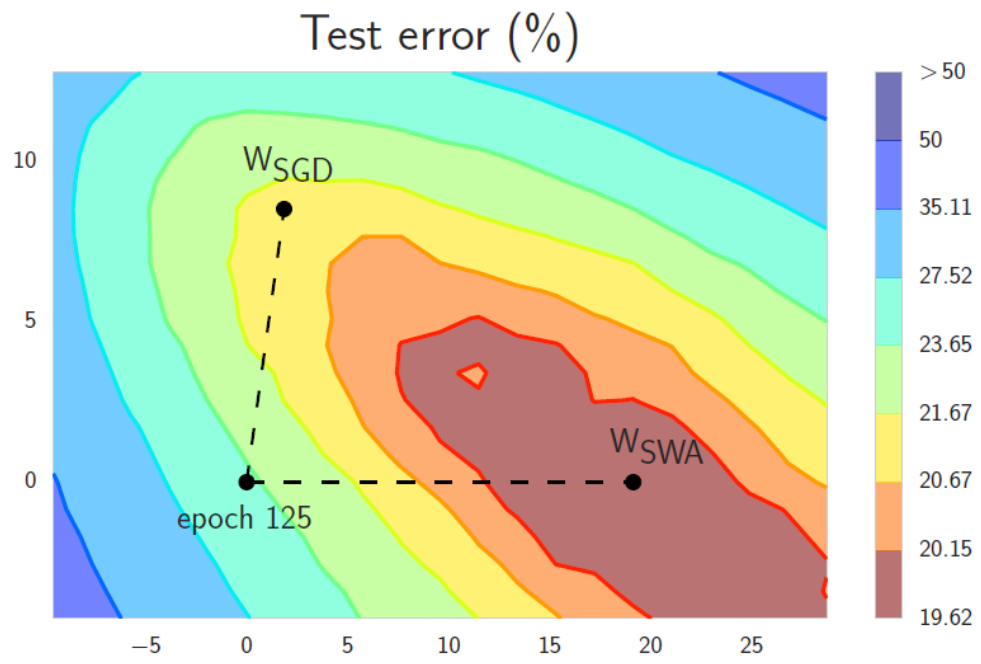


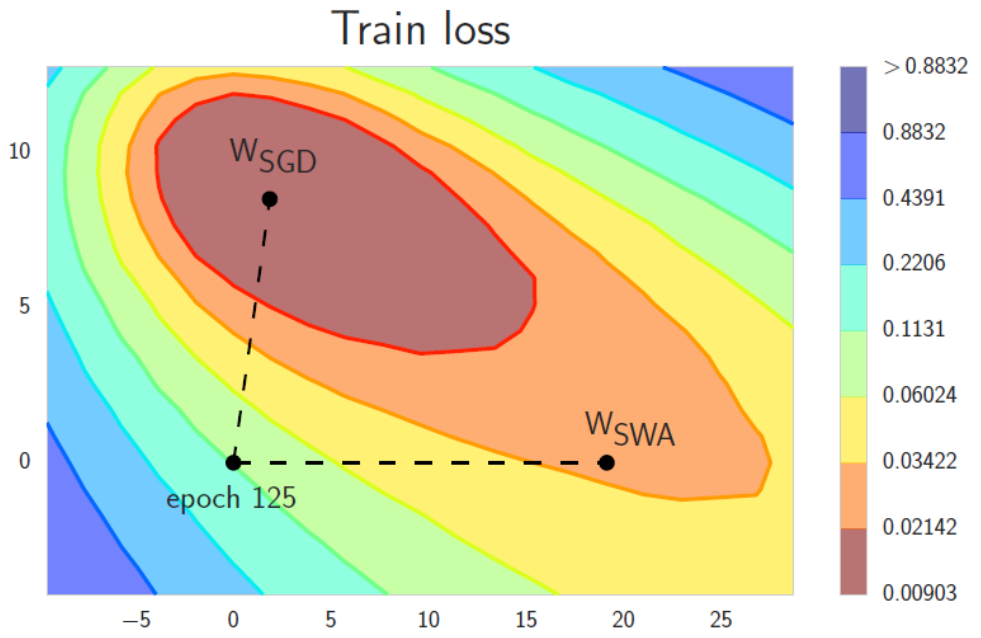
Methods

1. Stochastic Weight Averaging

❖ Stochastic Weight Averaging (SWA)

- SGD로 학습되는 모델 weight를 시간 축으로 누적 평균하여 하나의 모델을 구축
- 높은 Train loss를 갖게 되지만, 더 낮은 Test error을 갖음



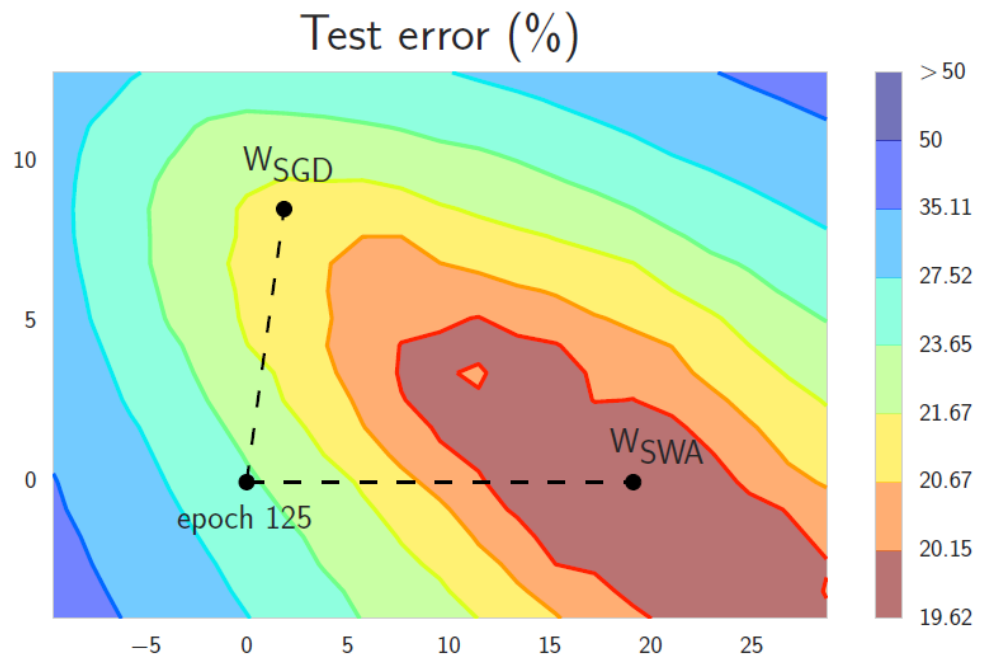


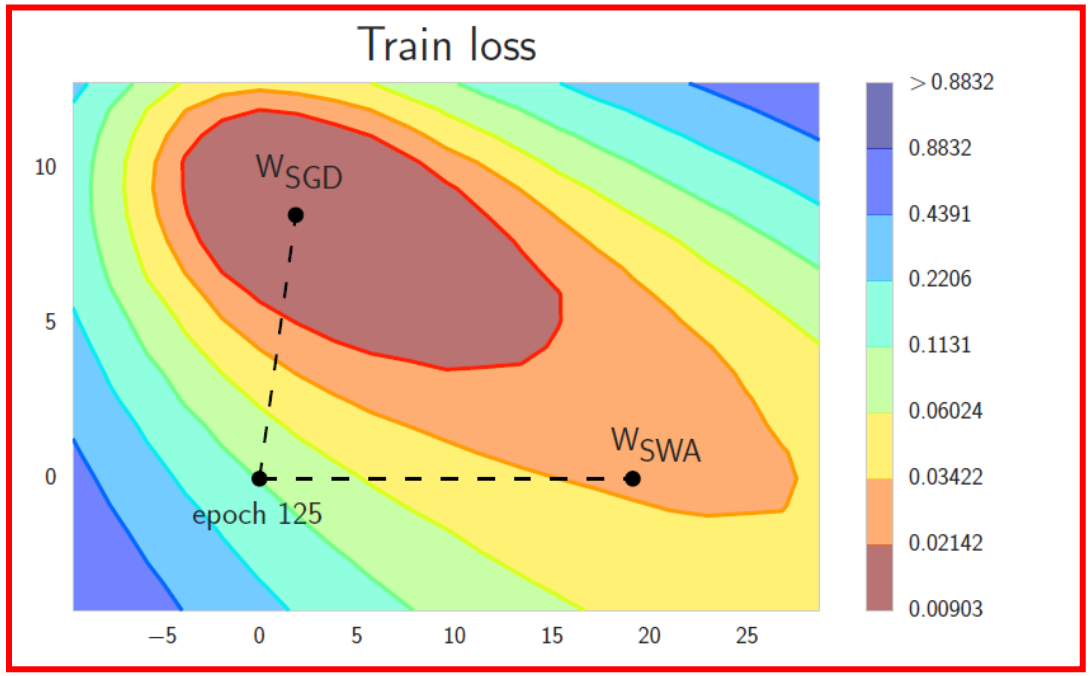
Methods

1. Stochastic Weight Averaging

❖ Stochastic Weight Averaging (SWA)

- SGD로 학습되는 모델 weight를 시간 축으로 누적 평균하여 하나의 모델을 구축
- 높은 Train loss를 갖게 되지만, 더 낮은 Test error을 갖음
- 학습 시 SGD보다 **flat한 local minima를 갖음** → loss의 등고선의 간극이 먼 곳으로 수렴





Methods

1. Stochastic Weight Averaging

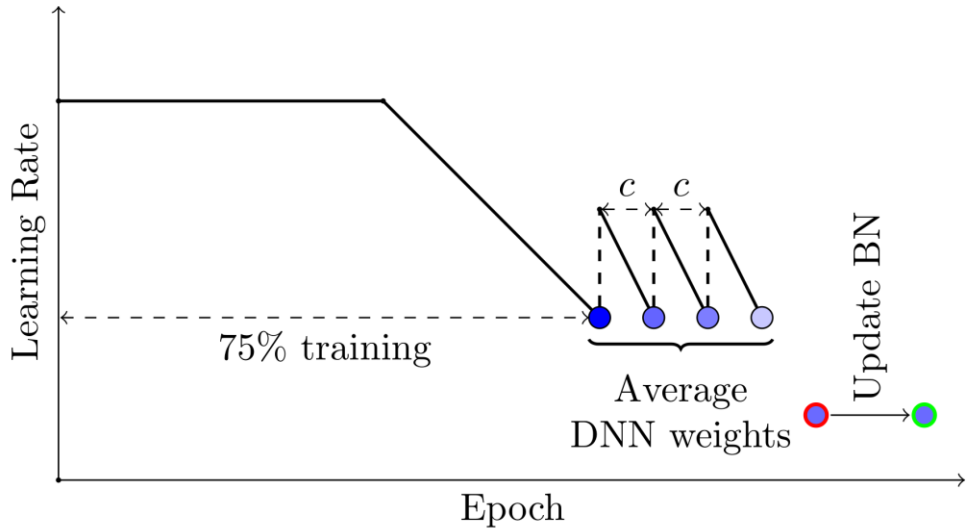
❖ Pseudo-code for Stochastic Weight Averaging (SWA) algorithm

- SWA는 일정 주기마다 모델에 가중치를 누적 평균하는 방식으로 학습
- 추가되는 메모리 및 연산 증가 거의 없음

Algorithm 1 Stochastic Weight Averaging

Require:
weights $\hat{w}$, LR bounds α_1, α_2,
cycle length c (for constant learning rate $c = 1$), number of iterations n
Ensure: w_{SWA}
$w \leftarrow \hat{w}$ {Initialize weights with $\hat{w}$}
$w_{\text{SWA}} \leftarrow w$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $\alpha \leftarrow \alpha(i)$ {Calculate LR for the iteration}
  $w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$ {Stochastic gradient update}
  if $\text{mod}(i, c) = 0$ **then**
    $n_{\text{models}} \leftarrow i/c$ {Number of models}
    $w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1}$ {Update average}
  end if
end for
{Compute BatchNorm statistics for w_{SWA} weights}

$$w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1},$$



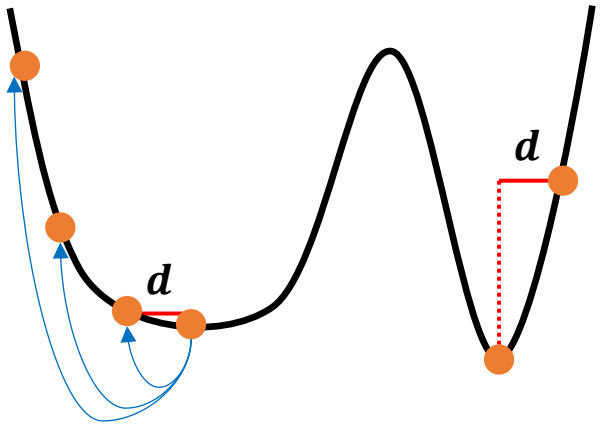
Methods

1. Stochastic Weight Averaging

❖ Stochastic Weight Averaging (SWA) Result

- 정말 Flat한 결과를 얻게 되는지 확인하기 위한 실험 진행

$$w_{\mathrm{SWA}}(t, d) = w_{\mathrm{SWA}} + t \cdot d,$$
$$w_{\mathrm{SGD}}(t, d) = w_{\mathrm{SGD}} + t \cdot d,$$



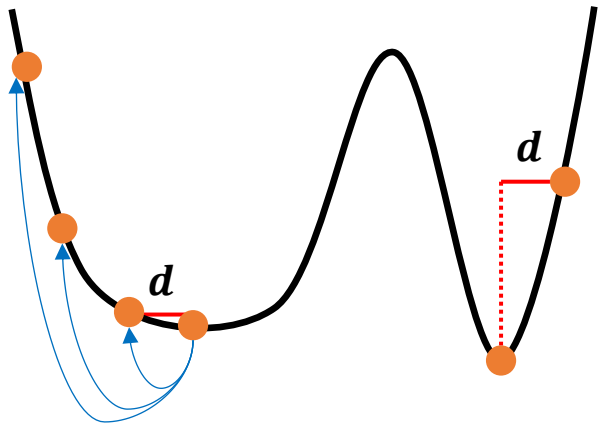
Methods

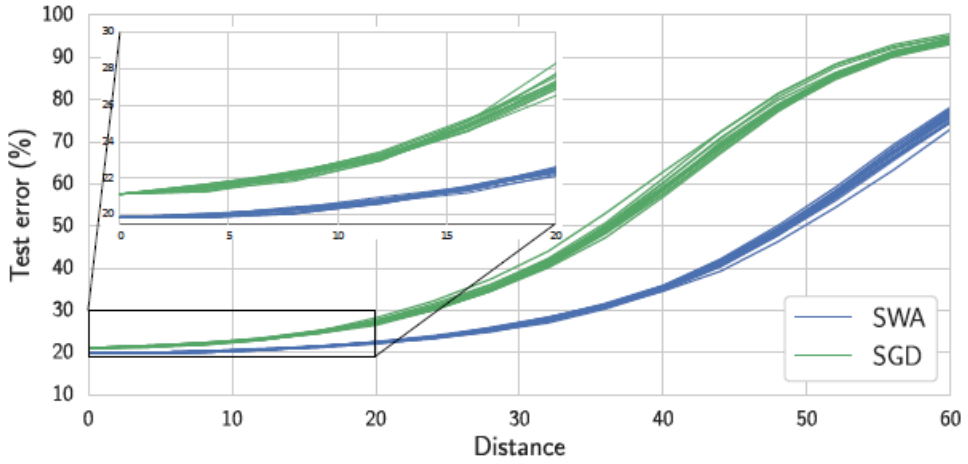
1. Stochastic Weight Averaging

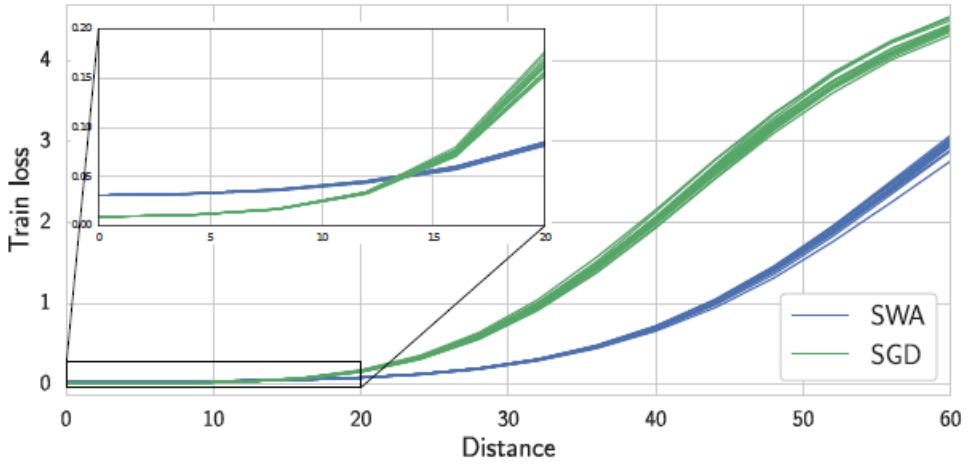
❖ Stochastic Weight Averaging (SWA) Result

- 정말 Flat한 결과를 얻게 되는지 확인하기 위한 실험 진행
- SWA가 모델 변동에 따른 loss변화가 작기 때문에 Flat한 minima

$$w_{\mathrm{SWA}}(t,d) = w_{\mathrm{SWA}} + t \cdot d,$$
$$w_{\mathrm{SGD}}(t,d) = w_{\mathrm{SGD}} + t \cdot d,$$







Methods

1. Stochastic Weight Averaging

❖ Stochastic Weight Averaging (SWA) Result

- CIFAR-10/100에서 SGD보다 높은 성능을 보이며, 선행연구인 FGE 보다도 높은 성능을 보임

Table 1: Accuracies (%) of SWA, SGD and FGE methods on CIFAR-100 and CIFAR-10 datasets for different training budgets. Accuracies for the FGE ensemble are from Garipov et al. [2018].

DNN (Budget)	SGD	FGE (1 Budget)	SWA		
			1 Budget	1.25 Budgets	1.5 Budgets
		CIFAR-100			
VGG-16 (200)	72.55 ± 0.10	74.26	73.91 ± 0.12	74.17 ± 0.15	74.27 ± 0.25
ResNet-164 (150)	78.49 ± 0.36	79.84	79.77 ± 0.17	80.18 ± 0.23	80.35 ± 0.16
WRN-28-10 (200)	80.82 ± 0.23	82.27	81.46 ± 0.23	81.91 ± 0.27	82.15 ± 0.27
PyramidNet-272 (300)	83.41 ± 0.21	–	–	83.93 ± 0.18	84.16 ± 0.15
		CIFAR-10			
VGG-16 (200)	93.25 ± 0.16	93.52	93.59 ± 0.16	93.70 ± 0.22	93.64 ± 0.18
ResNet-164 (150)	95.28 ± 0.10	95.45	95.56 ± 0.11	95.77 ± 0.04	95.83 ± 0.03
WRN-28-10 (200)	96.18 ± 0.11	96.36	96.45 ± 0.11	96.64 ± 0.08	96.79 ± 0.05
ShakeShake-2x64d (1800)	96.93 ± 0.10	–	–	97.16 ± 0.10	97.12 ± 0.06

Methods

1. Stochastic Weight Averaging

❖ Stochastic Weight Averaging (SWA) Result

- ImageNet data에서도 SGD보다 높은 성능을 보임

Table 2: Top-1 accuracies (%) on ImageNet for SWA and SGD with different architectures.

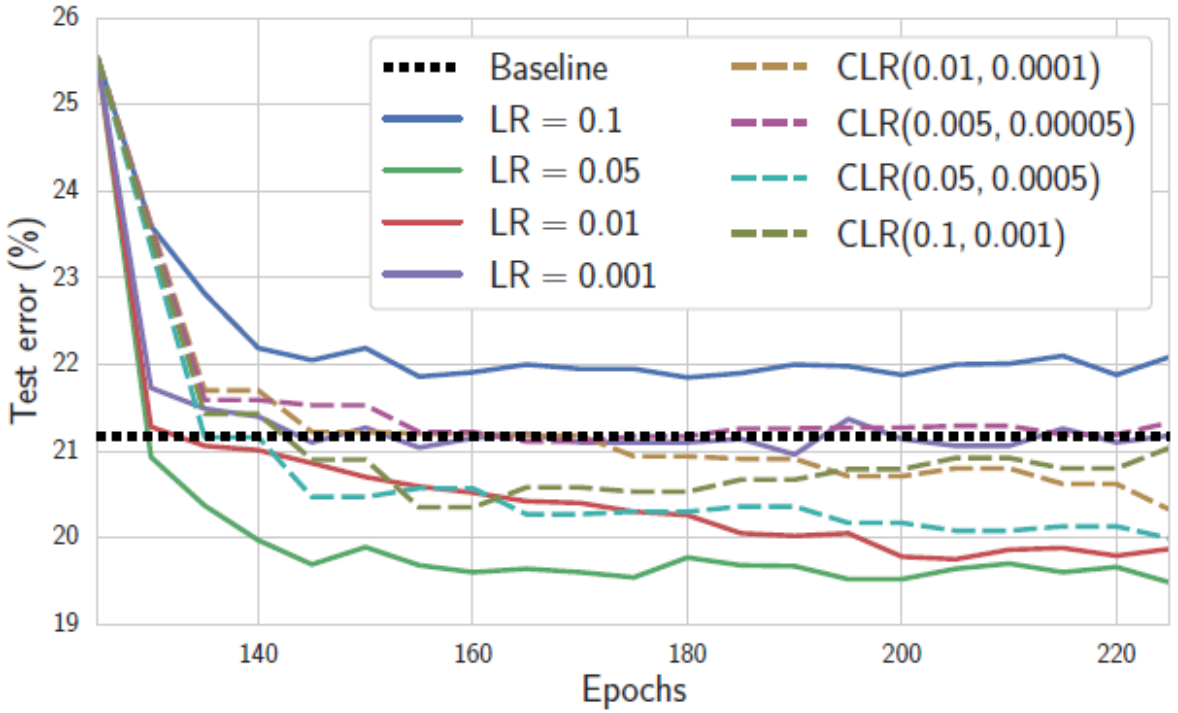
		SWA	
DNN	SGD	5 epochs	10 epochs
ResNet-50	76.15	76.83 ± 0.01	76.97 ± 0.05
ResNet-152	78.31	78.82 ± 0.01	78.94 ± 0.07
DenseNet-161	77.65	78.26 ± 0.09	78.44 ± 0.06

Methods

1. Stochastic Weight Averaging

❖ Stochastic Weight Averaging (SWA) Result

- Constant learning rate를 사용했을 때 높은 성능을 보임을 확인



Sharpness-aware minimization for efficiently improving generalization

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization for efficiently improving generalization[3]

- Loss의 Sharpness를 고려하여 Flat Minima를 찾는 방법론 (ICLR 2021, 24년 12월 기준 1409회 인용)

SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION

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

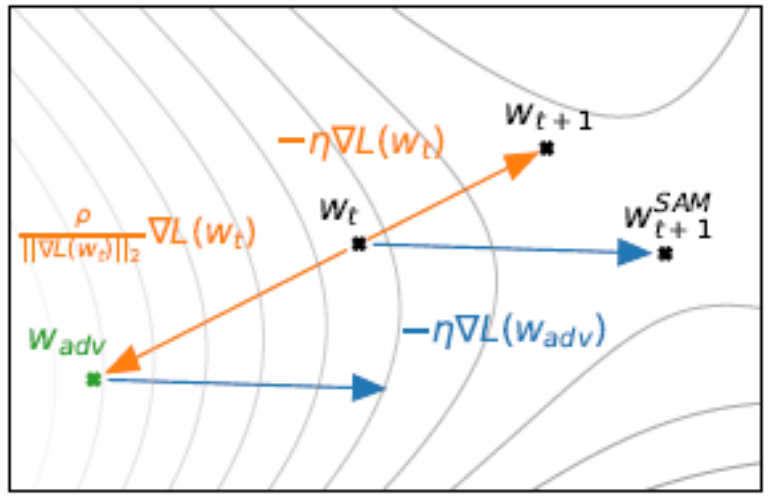
Ariel Kleiner
Google Research
akleiner@gmail.com

Hossein Mobahi
Google Research
hmobahi@google.com

Behnam Neyshabur
Blueshift, Alphabet
neyshabur@google.com

ABSTRACT

In today's heavily overparameterized models, the value of the training loss provides few guarantees on model generalization ability. Indeed, optimizing only the training loss value, as is commonly done, can easily lead to suboptimal model quality. Motivated by prior work connecting the geometry of the loss landscape and generalization, we introduce a novel, effective procedure for instead simultaneously minimizing loss value and loss sharpness. In particular, our procedure, Sharpness-Aware Minimization (SAM), seeks parameters that lie in neighborhoods having uniformly low loss; this formulation results in a min-max optimization problem on which gradient descent can be performed efficiently. We present empirical results showing that SAM improves model generalization across a variety of benchmark datasets (e.g., CIFAR-{10, 100}, ImageNet, finetuning tasks) and models, yielding novel state-of-the-art performance for several. Additionally, we find that SAM natively provides robustness to label noise on par with that provided by state-of-the-art procedures that specifically target learning with noisy labels. We open source our code at https://github.com/google-research/sam.



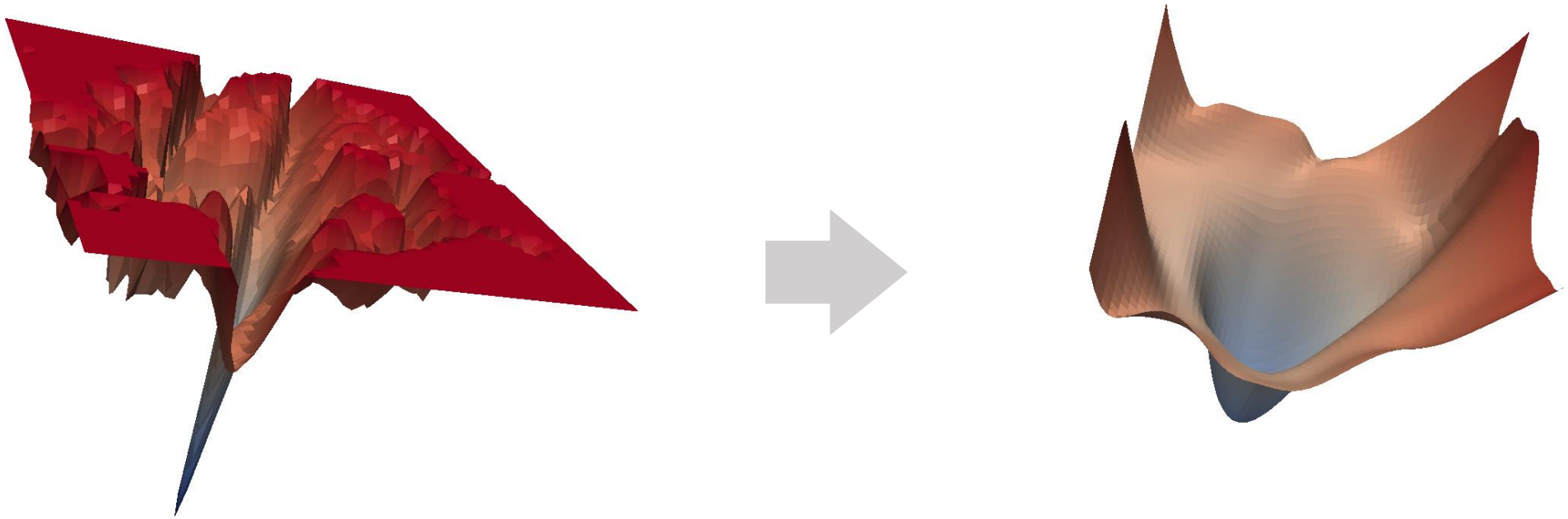
[3] Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization. arXiv preprint arXiv:2010.01412.

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM)

- SAM은 training loss 감소와 **동시에 loss의 sharpness 감소를 목적함수로 함**
- 목적함수를 최소화 하는 과정에서 loss landscape가 기존의 SGD보다 평탄화 됨



ResNet **+ SGD**

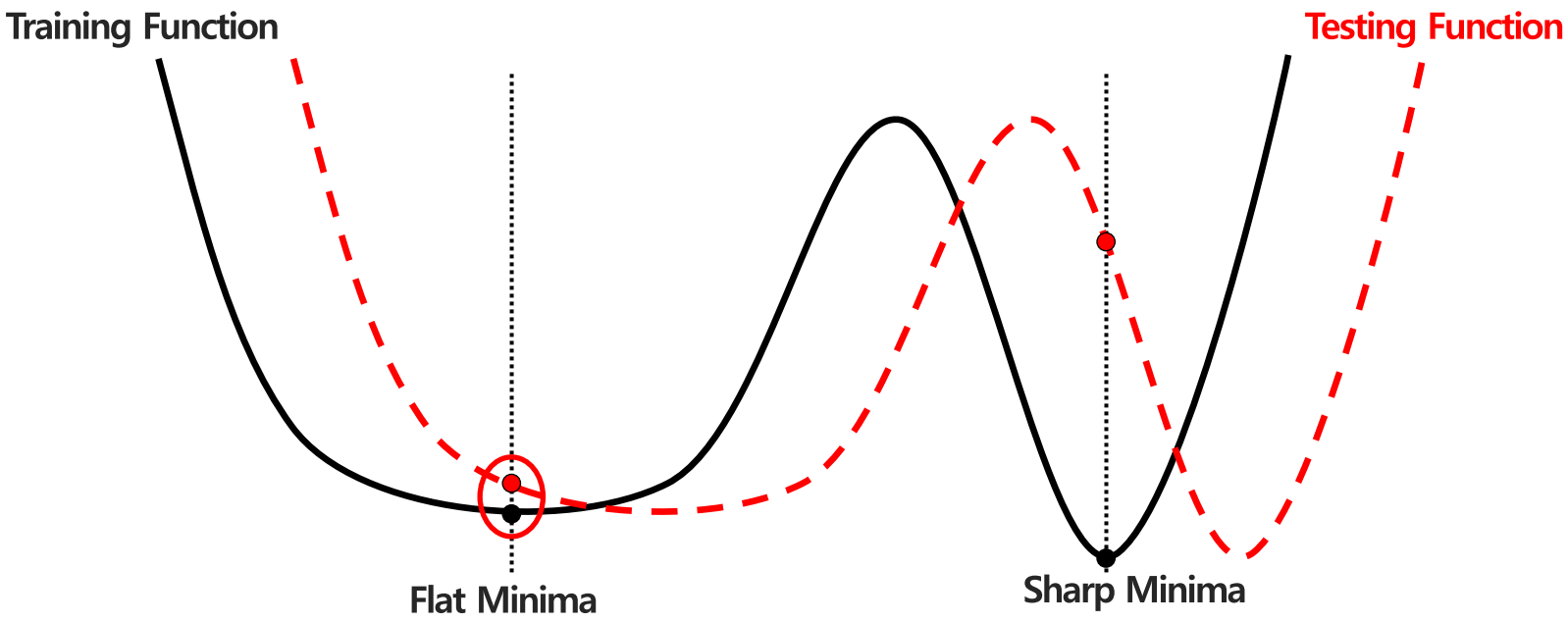
ResNet **+ SAM**

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) problem

- Training set loss: $L_s(w) \triangleq \frac{1}{n}\sum_{i=1}^{n} l(w, x_i, y_i)$
- Population loss: $L_D(w) \triangleq E_{(x,y)\sim D}[l(w, x, y)] \rightarrow L_D(w)$를 최소화를 통한 **Generalization**
- **SAM은 loss의 sharpness를 고려하여 Flat Minima를 찾는 것**을 목표로 함



Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) problem

- PAC (probably approximately correct) Bayesian generalization bound를 통해 loss sharpness를 고려하는 것이 가능

PAC Bayesian generalization bound: $L_D(w) \leq \max_{\|\epsilon\| \leq \rho} L_s(w+\epsilon) + h(\|w\|_2^2/\rho^2)$

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) problem

- PAC (probably approximately correct) Bayesian generalization bound를 통해 loss sharpness를 고려하는 것이 가능

PAC Bayesian generalization bound: $L_D(w) \leq \max_{\|\epsilon\| \leq \rho} L_s(w+\epsilon) + h(\|w\|_2^2/\rho^2)$

$$L_D(w) \leq [\max_{\|\epsilon\| \leq \rho} L_s(w+\epsilon) - L_s(w)] + L_s(w) + h(\|w\|_2^2/\rho^2)$$

Methods

2. Sharpness-Aware Minimization

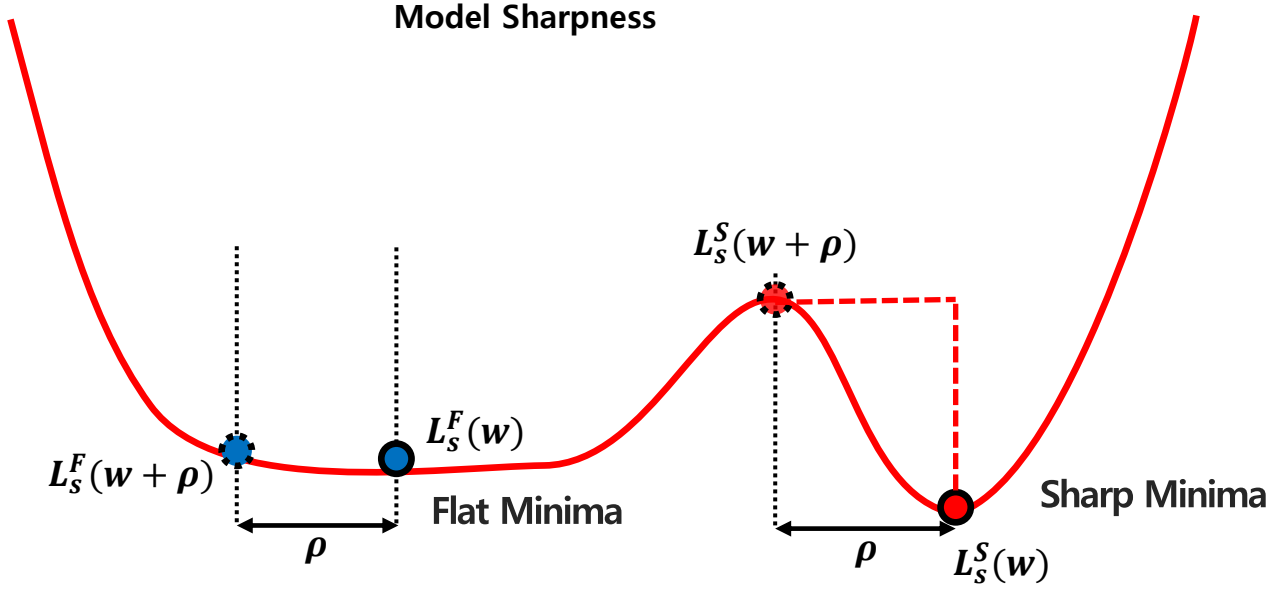
❖ Sharpness-aware minimization (SAM) problem

- PAC (probably approximately correct) Bayesian generalization bound를 통해 loss sharpness를 고려하는 것이 가능

PAC Bayesian generalization bound: $L_D(w) \leq \max_{\|\epsilon\| \leq \rho} L_s(w+\epsilon) + h(\|w\|_2^2/\rho^2)$

$$L_D(w) \leq \boxed{[\max_{\|\epsilon\| \leq \rho} L_s(w+\epsilon) - L_s(w)]} + L_s(w) + h(\|w\|_2^2/\rho^2)$$

Model Sharpness



Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) objective

$$L_D(w) \leq [\max_{\|\epsilon\| \leq \rho} L_s(w+\epsilon) - L_s(w)] + L_s(w) + \underbrace{h(\|w\|_2^2/\rho^2)}_{\text{Weight Decay}}$$

$$\min_w L_s^{SAM}(w) + \underbrace{\lambda\|w\|_2^2}_{\text{Weight Decay}}, \qquad \text{where } \mathrm{L_s^{SAM}(w)} \triangleq \max_{\|\epsilon\|_p \leq \rho} L_s(w+\epsilon)$$

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) objective

$$L_D(w) \leq [\max_{\|\epsilon\| \leq \rho} L_s(w+\epsilon) - L_s(w)] + L_s(w) + \underbrace{h(\|w\|_2^2/\rho^2)}_{\textbf{Weight Decay}}$$

$$\min_w L_s^{SAM}(w) + \underbrace{\lambda\|w\|_2^2}_{\textbf{Weight Decay}}, \qquad \text{where } \mathrm{L_s^{SAM}(w)} \triangleq \max_{\|\epsilon\|_p \leq \rho} L_s(w+\epsilon)$$

$$\min_w L_s^{SAM}(w) + \lambda\|w\|_2^2 \;\rightarrow\; \nabla_w L_s^{SAM}(w) = \underbrace{\nabla_w \max_{\|\epsilon\|_p \leq \rho} L_s(w+\epsilon)}_{\textbf{Approximation 필요}}$$

수학적으로 정확한 미분 계산이 어려움

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) objective

$$L_D(w) \leq [\max_{\|\epsilon\| \leq \rho} L_s(w+\epsilon) - L_s(w)] + L_s(w) + \underbrace{h(\|w\|_2^2/\rho^2)}_{\textbf{Weight Decay}}$$

$$\min_w L_s^{SAM}(w) + \underbrace{\lambda\|w\|_2^2}_{\textbf{Weight Decay}}, \qquad \text{where } \mathrm{L_s^{SAM}(w)} \triangleq \max_{\|\epsilon\|_p \leq \rho} L_s(w+\epsilon)$$

$$\min_w L_s^{SAM}(w) + \lambda\|w\|_2^2 \;\rightarrow\; \nabla_w L_s^{SAM}(w) = \underbrace{\nabla_w \max_{\|\epsilon\|_p \leq \rho} L_s(w+\epsilon)}_{\textbf{Approximation 필요}}$$

수학적으로 정확한 미분 계산이 어려움

First-order Taylor expansion 적용: $\underset{\|\epsilon\|_p \leq \rho}{\operatorname{argmax}} L_s(w+\epsilon) \approx \underset{\|\epsilon\|_p \leq \rho}{\operatorname{argmax}} L_s(w) + \epsilon^T \nabla_w L_s(w) = \underset{\|\epsilon\|_p \leq \rho}{arg\max}\, \epsilon^T \nabla_w L_s(w)$

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) objective

First-order Taylor expansion 적용: $\underset{\|\epsilon\|_p \le \rho}{\operatorname{argmax}} L_s(w+\epsilon) \approx \underset{\|\epsilon\|_p \le \rho}{\operatorname{argmax}} L_s(w) + \epsilon^T \nabla_w L_s(w) = \underset{\|\epsilon\|_p \le \rho}{arg\max} \epsilon^T \nabla_w L_s(w)$

Dual norm problem 적용: $\underset{\|\epsilon\|_p \le \rho}{arg\max} \epsilon^T \nabla_w L_s(w) = \|\rho \nabla_w L_s(w)\|_{p^*} = \|\rho \nabla_w L_s(w)\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1$

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) objective

First-order Taylor expansion 적용: $\underset{\|\epsilon\|_p \le \rho}{\operatorname{argmax}} L_s(w+\epsilon) \approx \underset{\|\epsilon\|_p \le \rho}{\operatorname{argmax}} L_s(w) + \epsilon^T \nabla_w L_s(w) = \underset{\|\epsilon\|_p \le \rho}{arg\max}\, \epsilon^T \nabla_w L_s(w)$

Dual norm problem 적용: $\underset{\|\epsilon\|_p \le \rho}{arg\max}\, \epsilon^T \nabla_w L_s(w) = \|\rho \nabla_w L_s(w)\|_{p^*} = \|\rho \nabla_w L_s(w)\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1$

$$\hat{\epsilon}(w) = \rho sign(\nabla_w L_s(w)) \frac{|\nabla_w L_s(w)|^{q-1}}{\left(\|\nabla_w L_s(w)\|_q^q\right)^{\frac{1}{p}}}$$

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) objective

First-order Taylor expansion 적용: $\underset{\|\epsilon\|_p \le \rho}{\operatorname{argmax}} L_s(w+\epsilon) \approx \underset{\|\epsilon\|_p \le \rho}{\operatorname{argmax}} L_s(w) + \epsilon^T \nabla_w L_s(w) = \underset{\|\epsilon\|_p \le \rho}{arg\max}\, \epsilon^T \nabla_w L_s(w)$

Dual norm problem 적용: $\underset{\|\epsilon\|_p \le \rho}{arg\max}\, \epsilon^T \nabla_w L_s(w) = \|\rho \nabla_w L_s(w)\|_{p^*} = \|\rho \nabla_w L_s(w)\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1$

$$\hat{\epsilon}(w) = \rho sign(\nabla_w L_s(w)) \frac{|\nabla_w L_s(w)|^{q-1}}{\left(\|\nabla_w L_s(w)\|_q^q\right)^{\frac{1}{p}}}$$

$$\nabla_w L_s^{SAM}(w) = \nabla_w \max_{\|\epsilon\|_p \le \rho} L_s(w+\epsilon) \approx \nabla_w L_s\big(w + \hat{\epsilon}(w)\big) = \frac{d\big(w + \hat{\epsilon}(w)\big)}{dw} \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}(w)} = \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}(w)} + \frac{d\hat{\epsilon}(w)}{dw} \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}(w)}$$

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) objective

First-order Taylor expansion 적용: $\underset{\|\epsilon\|_p \le \rho}{\operatorname{argmax}} L_s(w+\epsilon) \approx \underset{\|\epsilon\|_p \le \rho}{\operatorname{argmax}} L_s(w) + \epsilon^T \nabla_w L_s(w) = \underset{\|\epsilon\|_p \le \rho}{arg\max}\, \epsilon^T \nabla_w L_s(w)$

Dual norm problem 적용: $\underset{\|\epsilon\|_p \le \rho}{arg\max}\, \epsilon^T \nabla_w L_s(w) = \|\rho \nabla_w L_s(w)\|_{p^*} = \|\rho \nabla_w L_s(w)\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1$

$$\hat{\epsilon}(w) = \rho sign(\nabla_w L_s(w)) \frac{|\nabla_w L_s(w)|^{q-1}}{\left(\|\nabla_w L_s(w)\|_q^q\right)^{\frac{1}{p}}}$$

$$\nabla_w L_s^{SAM}(w) = \nabla_w \max_{\|\epsilon\|_p \le \rho} L_s(w+\epsilon) \approx \nabla_w L_s\big(w + \hat{\epsilon}(w)\big) = \frac{d\big(w+\hat{\epsilon}(w)\big)}{dw} \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}(w)} = \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}(w)} + \boxed{\frac{d\hat{\epsilon}(w)}{dw} \nabla_w L_s(w)\Big|_{w+\hat{\epsilon}(w)}}$$

Hessian 제거

∵ computation cost 감소

Final objective function: $\nabla_w L_s^{SAM}(w) \approx \nabla_w L_s(w)|_{w+\hat{\epsilon}(w)}$

Methods

2. Sharpness-Aware Minimization

❖ Pseudo-code for Sharpness-aware minimization (SAM) algorithm

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(x_i, y_i)\}$, 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 w_0, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(x_1, y_1), ...(x_b, y_b)\}$;
  Compute gradient $\nabla_w L_{\mathcal{B}}(w)$ of the batch's training loss;
  Compute $\hat{\epsilon}(w)$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $g = \nabla_w L_{\mathcal{B}}(w)|_{w+\hat{\epsilon}(w)}$;
  Update weights: $w_{t+1} = w_t - \eta g$;
  $t = t + 1$;
end
return w_t

Algorithm 1: SAM algorithm

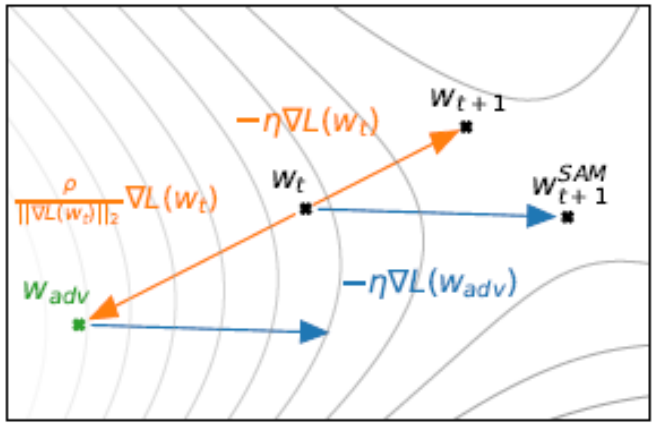


Figure 2: Schematic of the SAM parameter update.

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) Result

- SGD보다 CIFAR-10/100에 대해서 가장 높은 성능을 보임
- 적용하는 모델에 관계 없이 높은 성능을 보일 수 있는 방법론 **(model-agnostic)**

		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$_{\pm 0.1}$

Table 1: Results for SAM on state-of-the-art models on CIFAR-{10, 100} (WRN = WideResNet; AA = AutoAugment; SGD is the standard non-SAM procedure used to train these models).

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) Result

- ImageNet 벤치마크 데이터에 대해서도 가장 높은 성능을 보임

Model	Epoch	SAM		Standard Training (No SAM)	
		Top-1	Top-5	Top-1	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.

Methods

2. Sharpness-Aware Minimization

❖ Sharpness-aware minimization (SAM) Result

- Hessian term 사용 시 computational cost가 높음과 동시에 성능 저하가 발생

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

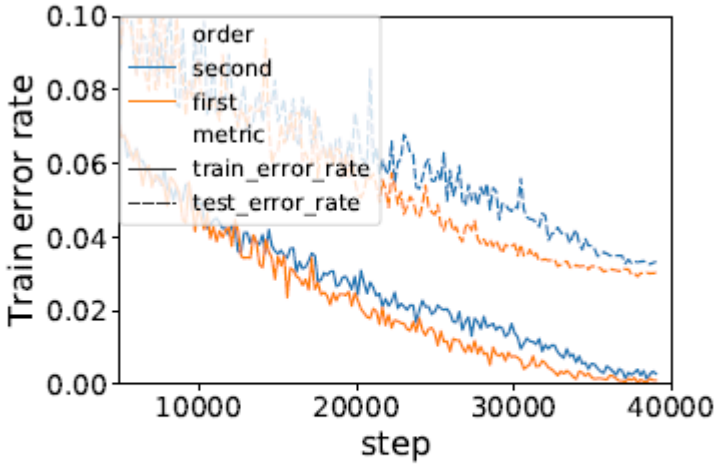


Figure 4: Training and test error for the first and second order version of the algorithm.

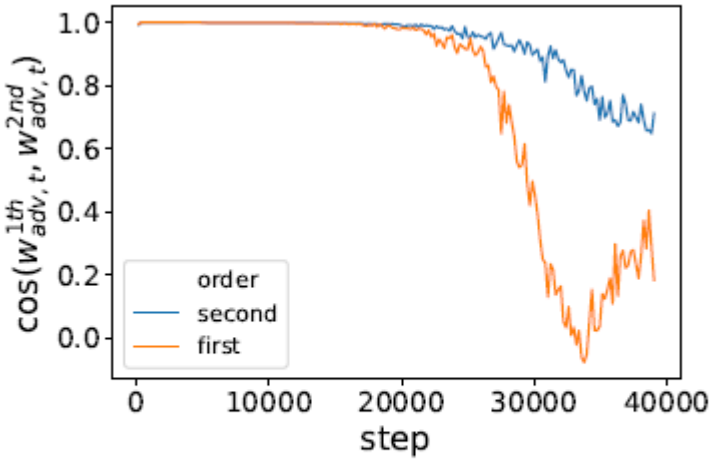


Figure 5: Cosine similarity between the first and second order updates.

When Do Flat Minima Optimizers Work?

Methods

3. When Do Flat Minima Optimizers Work?

❖ When Do Flat Minima Optimizers Work?[4]

- SWA와 SAM 방법론을 소개하고, 다양한 벤치마크 적용 인사이트 제공 (NeurIPS 2022, 24년 12월 기준 75회 인용)
- 두 방법론을 적절히 조합한 WASAM 방법론 제안

When Do Flat Minima Optimizers Work?

Jean Kaddour*
Centre for Artificial Intelligence
University College London

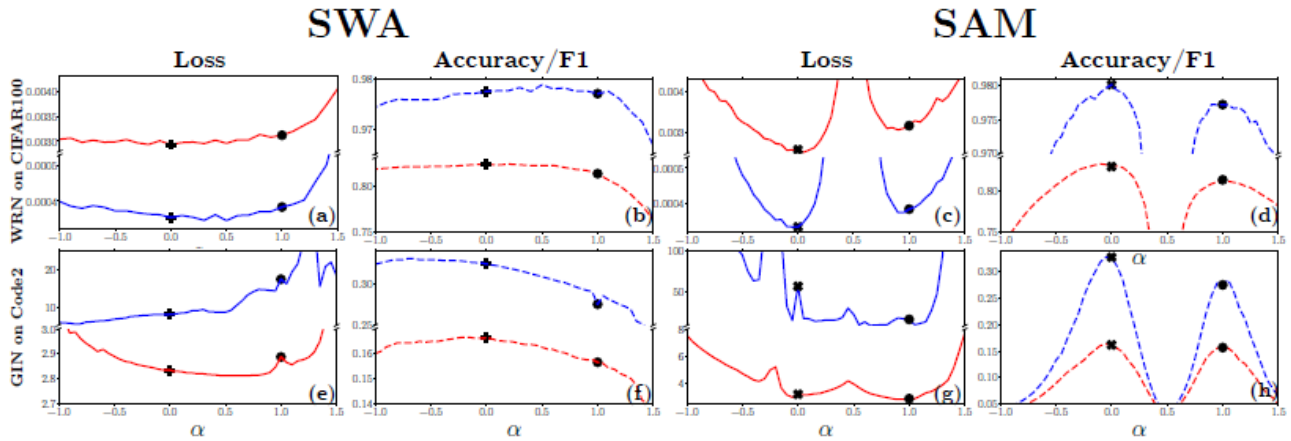
Linqing Liu*
Centre for Artificial Intelligence
University College London

Ricardo Silva
Department of Statistical Science
University College London

Matt J. Kusner
Centre for Artificial Intelligence
University College London

Abstract

Recently, *flat-minima optimizers*, which seek to find parameters in low-loss neighborhoods, have been shown to improve a neural network's generalization performance over stochastic and adaptive gradient-based optimizers. Two methods have received significant attention due to their scalability: 1. Stochastic Weight Averaging (SWA), and 2. Sharpness-Aware Minimization (SAM). However, there has been limited investigation into their properties and no systematic benchmarking of them across different domains. We fill this gap here by comparing the loss surfaces of the models trained with each method and through broad benchmarking across computer vision, natural language processing, and graph representation learning tasks. We discover several surprising findings from these results, which we hope will help researchers further improve deep learning optimizers, and practitioners identify the right optimizer for their problem.



[4] Kaddour, J., Liu, L., Silva, R., & Kusner, M. J. (2022). When do flat minima optimizers work?. Advances in Neural Information Processing Systems, 35, 16577-16595.

Methods

3. When Do Flat Minima Optimizers Work?

❖ Stochastic Weight Averaging (SWA) & Sharpness-aware minimization (SAM)

- **SWA:** 모델 가중치 평균화를 통해 Flat minima를 찾음으로써 일반화 성능 향상
- **SAM:** Loss의 Sharpness를 고려한 목적함수를 통해 Flat Minima를 찾음으로써 일반화 성능 향상

Algorithm 1 Stochastic Weight Averaging [48]

Input: Loss function $\mathcal{L}$, training budget in number of iterations b, training dataset $\mathcal{D} := \cup_{i=1}^{n}\{x_i\}$, mini-batch size $|\mathcal{B}|$, averaging start epoch E, averaging frequency ν, (scheduled) learning rate η, initial weights θ_0.

for $k \leftarrow 1, \ldots, b$ **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$
  Compute gradient $g \leftarrow \nabla \mathcal{L}\left(\theta_t\right)$
  Update parameters $\theta_{t+1} \leftarrow \theta_t - \eta g$
  if $k \geq E$ and $\mathrm{mod}(k, \nu) = 0$ **then**
    $\theta_{t+1}^{\mathrm{SWA}} = \left(\theta_t^{\mathrm{SWA}} \cdot l + \theta_{t+1}^{\mathrm{SWA}}\right) / (l+1)$
  end if
end for
return θ^{SWA}

Algorithm 2 Sharpness-Aware Minimization [22]

Input: Loss function $\mathcal{L}$, training budget in number of iterations b, training dataset $\mathcal{D} := \cup_{i=1}^{n}\{x_i\}$, mini-batch size $|\mathcal{B}|$, neighborhood radius ρ, (scheduled) learning rate η, initial weights θ_0.

for $k \leftarrow 1, \ldots, b$ **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$
  Compute worst-case perturbation $\widehat{\epsilon} \leftarrow \rho \frac{\nabla \mathcal{L}(\theta)}{\|\nabla \mathcal{L}(\theta)\|_2}$
  Compute gradient $g \leftarrow \nabla \mathcal{L}\left(\theta_t^{\mathrm{SAM}} + \widehat{\epsilon}\right)$
  Update parameters $\theta_{t+1}^{\mathrm{SAM}} \leftarrow \theta_t^{\mathrm{SAM}} - \eta g$
end for
return θ^{SAM}

Methods

3. When Do Flat Minima Optimizers Work?

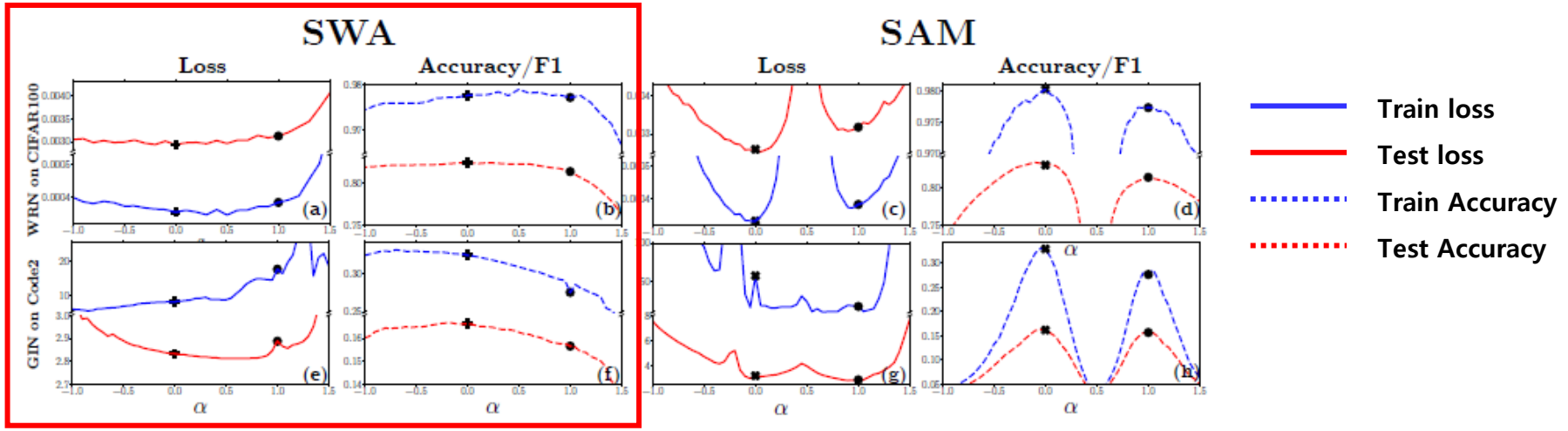
❖ **How do minima found by SWA and SAM differ?**

- 두 방법론에 대한 다양한 실험을 진행하고 차이점을 찾음

Obs.1: θ_{SWA}와 θ_{NF} 는 동일한 basin에 속함 + θ_{SWA}는 Flat minima를 찾음

Obs.2: θ_{SAM}와 θ_{NF} 는 동일한 basin에 속하지 않음 + θ_{SAM}는 더 높은 일반화 성능을 갖음

Obs.3: θ_{SAM}은 θ_{SWA}보다 sharp한 minima를 갖음 ← $L(\theta_{SAM}(0.1)) \approx 2 \cdot L(\theta_{SAM}(-0.1))$ / $L(\theta_{SWA}(0.1)) \approx L(\theta_{SWA}(-0.1))$



$\theta(\alpha) = (1-\alpha)\theta + \alpha\theta'$ $\{\theta, \theta'\} \leftarrow \{\theta_{SWA}, \theta_{NF}\}, \{\theta_{SAM}, \theta_{NF}\}$ **non-flat baseline (•), SWA (+), SAM (×)**

Methods

3. When Do Flat Minima Optimizers Work?

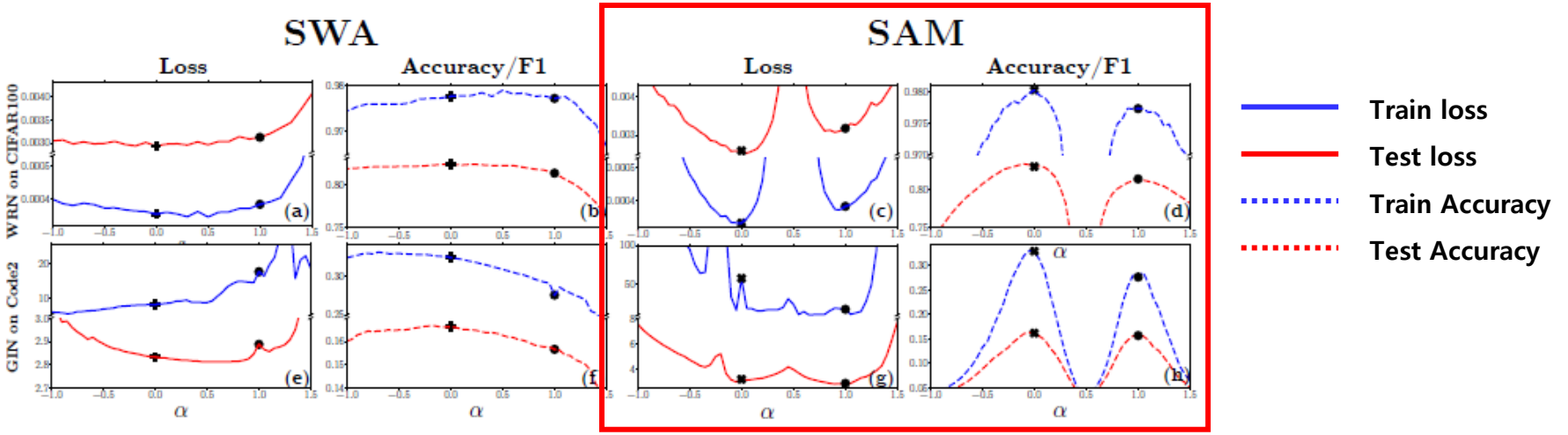
❖ How do minima found by SWA and SAM differ?

- 두 방법론에 대한 다양한 실험을 진행하고 차이점을 찾음

Obs.1: θ_{SWA}와 θ_{NF} 는 동일한 basin에 속함 + θ_{SWA}는 Flat minima를 찾음

Obs.2: θ_{SAM}와 θ_{NF} 는 동일한 basin에 속하지 않음 + θ_{SAM}는 더 높은 일반화 성능을 갖음

Obs.3: θ_{SAM}은 θ_{SWA}보다 sharp한 minima를 갖음 ← $L(\theta_{SAM}(0.1)) \approx 2 \cdot L(\theta_{SAM}(-0.1))$ / $L(\theta_{SWA}(0.1)) \approx L(\theta_{SWA}(-0.1))$



$\theta(\alpha) = (1-\alpha)\theta + \alpha\theta'$ $\{\theta, \theta'\} \leftarrow \{\theta_{SWA}, \theta_{NF}\}, \{\theta_{SAM}, \theta_{NF}\}$ **non-flat baseline (•), SWA (+), SAM (×)**

Methods

3. When Do Flat Minima Optimizers Work?

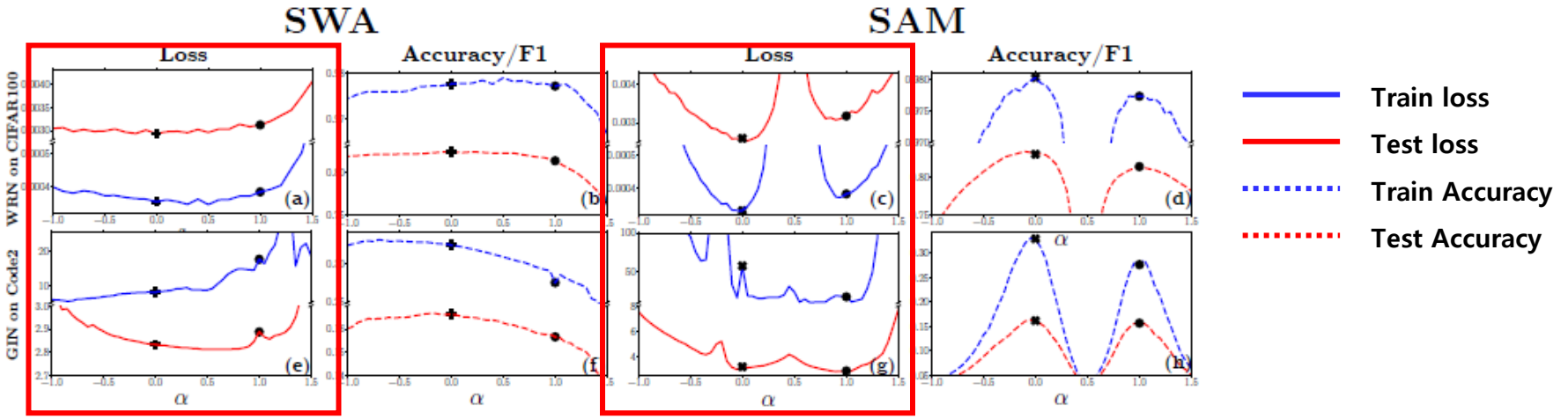
❖ **How do minima found by SWA and SAM differ?**

- 두 방법론에 대한 다양한 실험을 진행하고 차이점을 찾음

Obs.1: θ_{SWA}와 θ_{NF} 는 동일한 basin에 속함 + θ_{SWA}는 Flat minima를 찾음

Obs.2: θ_{SAM}와 θ_{NF} 는 동일한 basin에 속하지 않음 + θ_{SAM}는 더 높은 일반화 성능을 갖음

Obs.3: θ_{SAM}은 θ_{SWA}보다 sharp한 minima를 갖음 ← $L(\theta_{SAM}(0.1)) \approx 2 \cdot L(\theta_{SAM}(-0.1))$ / $L(\theta_{SWA}(0.1)) \approx L(\theta_{SWA}(-0.1))$



$\theta(\alpha) = (1-\alpha)\theta + \alpha\theta'$ $\quad \{\theta, \theta'\} \leftarrow \{\theta_{SWA}, \theta_{NF}\}, \{\theta_{SAM}, \theta_{NF}\}$ **non-flat baseline (•), SWA (+), SAM (×)**

Methods

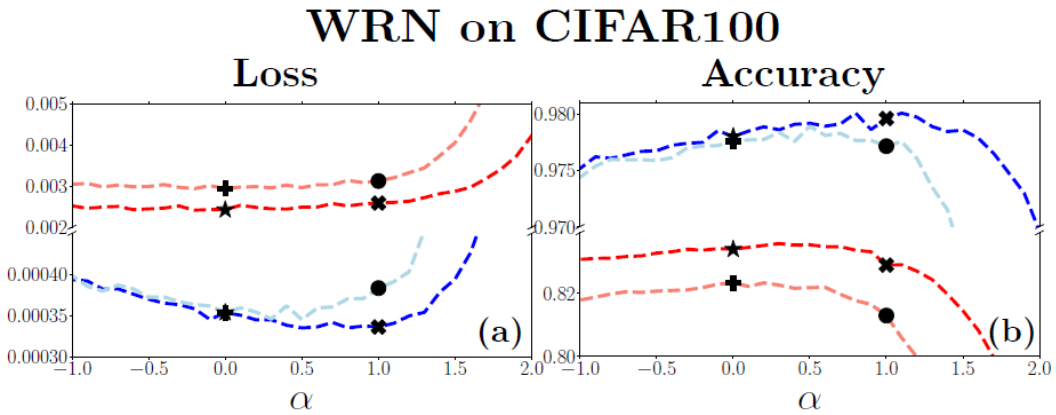
3. When Do Flat Minima Optimizers Work?

❖ What happens if we average SAM iterates

- θ_{SWA}보다 sharp한 minima를 갖는 θ_{SAM}을 평균화 하여 일반화 성능 향상

Obs.3: θ_{SAM}은 θ_{SWA}보다 sharp한 minima를 갖음 ← $L(\theta_{SAM}(0.1)) \approx 2 \cdot L(\theta_{SAM}(-0.1))$ / $L(\theta_{SWA}(0.1)) \approx L(\theta_{SWA}(-0.1))$

→ **Weight-Averaged Sharpness-Aware Minimization (WASAM)**



non-flat baseline (•), SWA (+), SAM (×) ,WASAM (⋆)

$$\theta(\alpha) = (1-\alpha)\theta + \alpha\theta' \qquad \{\theta, \theta'\} \leftarrow \{\theta_{SWA}, \theta_{NF}\}, \{\theta_{SAM}, \theta_{NF}\}, \{\theta_{WASAM}, \theta_{NF}\}$$

Methods

3. When Do Flat Minima Optimizers Work?

❖ 다양한 벤치마크에서의 성능 비교

- 컴퓨터 비전(CV) Supervised Classification과 Self-Supervised Learning에서 높은 성능을 보임

Table 2: CV test results: Supervised Classification (SC), and Self-Supervised Learning (SSL) tasks.

Task	Model	Baseline	SWA	SAM	WASAM
SC: CIFAR10	WRN-28-10	$96.78_{\pm 0.03}$	$-0.05_{\pm 0.04}$	$+\mathbf{0.34}_{\pm 0.09}$	$+0.25_{\pm 0.05}$
	PN-272	$96.73_{\pm 0.14}$	$+0.22_{\pm 0.14}$	$+\mathbf{0.42}_{\pm 0.06}$	$+\mathbf{0.41}_{\pm 0.02}$
	ViT-B-16	$98.95_{\pm 0.02}$	$-0.04_{\pm 0.04}$	$+0.07_{\pm 0.01}$	$+\mathbf{0.10}_{\pm 0.01}$
	Mixer-B-16	$96.65_{\pm 0.03}$	$+0.02_{\pm 0.03}$	$+\mathbf{0.19}_{\pm 0.05}$	$+\mathbf{0.22}_{\pm 0.06}$
SC: CIFAR100	WRN-28-10	$80.93_{\pm 0.19}$	$+1.62_{\pm 0.06}$	$+1.82_{\pm 0.14}$	$+\mathbf{2.24}_{\pm 0.14}$
	PN-272	$80.86_{\pm 0.12}$	$+1.88_{\pm 0.04}$	$+2.33_{\pm 0.08}$	$+\mathbf{2.60}_{\pm 0.09}$
	ViT-B-16	$92.77_{\pm 0.07}$	$-0.12_{\pm 0.05}$	$+\mathbf{0.19}_{\pm 0.09}$	$+\mathbf{0.13}_{\pm 0.07}$
	Mixer-B-16	$83.77_{\pm 0.08}$	$+0.45_{\pm 0.06}$	$+0.52_{\pm 0.15}$	$+\mathbf{0.97}_{\pm 0.12}$
SSL: CIFAR10	MoCo	$\mathbf{89.25}_{\pm 0.07}$	$-0.03_{\pm 0.10}$	$-0.25_{\pm 0.06}$	$-0.17_{\pm 0.10}$
	SimCLR	$\mathbf{88.66}_{\pm 0.08}$	$-0.05_{\pm 0.06}$	$+\mathbf{0.05}_{\pm 0.04}$	$-0.13_{\pm 0.06}$
	SimSiam	$\mathbf{89.86}_{\pm 0.22}$	$+\mathbf{0.12}_{\pm 0.26}$	$+\mathbf{0.07}_{\pm 0.10}$	$+\mathbf{0.11}_{\pm 0.10}$
	BarlowTwins	$\mathbf{86.34}_{\pm 0.24}$	$-0.09_{\pm 0.19}$	$+\mathbf{0.09}_{\pm 0.15}$	$+\mathbf{0.14}_{\pm 0.05}$
	BYOL	$90.32_{\pm 0.14}$	$+\mathbf{0.70}_{\pm 0.05}$	$+0.14_{\pm 0.03}$	$+0.21_{\pm 0.07}$
	SwaV	$87.28_{\pm 0.05}$	$+\mathbf{0.09}_{\pm 0.06}$	$+\mathbf{0.07}_{\pm 0.12}$	$+0.02_{\pm 0.06}$
SSL: ImageNette	MoCo	$81.74_{\pm 0.18}$	$+0.97_{\pm 0.10}$	$+0.91_{\pm 0.32}$	$+\mathbf{1.40}_{\pm 0.10}$
	SimCLR	$83.28_{\pm 0.22}$	$+\mathbf{0.95}_{\pm 0.25}$	$+0.18_{\pm 0.24}$	$+\mathbf{1.07}_{\pm 0.13}$
	SimSiam	$81.77_{\pm 0.14}$	$+\mathbf{0.20}_{\pm 0.37}$	$+\mathbf{0.33}_{\pm 0.28}$	$+\mathbf{0.18}_{\pm 0.26}$
	BarlowTwins	$77.49_{\pm 0.36}$	$+0.20_{\pm 0.16}$	$+\mathbf{0.47}_{\pm 0.27}$	$+\mathbf{0.66}_{\pm 0.57}$
	BYOL	$84.16_{\pm 0.14}$	$+\mathbf{0.76}_{\pm 0.08}$	$+0.15_{\pm 0.25}$	$+0.31_{\pm 0.19}$
	SwaV	$88.16_{\pm 0.31}$	$+\mathbf{1.04}_{\pm 0.27}$	$+0.03_{\pm 0.10}$	$+\mathbf{1.03}_{\pm 0.09}$

Methods

3. When Do Flat Minima Optimizers Work?

❖ 다양한 벤치마크에서의 성능 비교

- Nature Language Processing (NLP)와 Graph Representation Learning (GRL)에서도 높은 성능을 보임

Figure 4: (a) NLP test results: Open-Domain Question Answering and Natural Language Understanding (GLUE) including paraphrase, sentiment analysis, and textual entailment. (b) GRL test results: Node Property Prediction (NPP), Graph Property Prediction (GPP), Link Property Prediction (LPP).

Task	Model	Baseline	SWA	SAM	WASAM
NQ	FiD	$49.35_{\pm 0.44}$	$-0.20_{\pm 0.33}$	$+0.33_{\pm 0.19}$	$+0.48_{\pm 0.21}$
TriviaQA	FiD	$67.74_{\pm 0.29}$	$+0.40_{\pm 0.24}$	$+0.89_{\pm 0.03}$	$+0.92_{\pm 0.10}$
COLA	RoBERTa	$60.41_{\pm 0.22}$	$+0.09_{\pm 0.08}$	$+1.57_{\pm 1.20}$	$+1.41_{\pm 1.14}$
SST	RoBERTa	$94.95_{\pm 0.13}$	$-0.30_{\pm 0.27}$	$-0.23_{\pm 0.40}$	$+0.19_{\pm 0.14}$
MRPC	RoBERTa	$89.14_{\pm 0.57}$	$+0.08_{\pm 0.49}$	$+0.73_{\pm 0.43}$	$+0.81_{\pm 0.38}$
STSB	RoBERTa	$90.40_{\pm 0.02}$	$+0.00_{\pm 0.05}$	$+0.38_{\pm 0.17}$	$+0.35_{\pm 0.16}$
QQP	RoBERTa	$91.36_{\pm 0.07}$	$+0.01_{\pm 0.06}$	$+0.08_{\pm 0.07}$	$+0.06_{\pm 0.08}$
MNLI	RoBERTa	$87.41_{\pm 0.09}$	$+0.08_{\pm 0.11}$	$+0.39_{\pm 0.02}$	$+0.35_{\pm 0.03}$
QNLI	RoBERTa	$92.96_{\pm 0.06}$	$-0.08_{\pm 0.11}$	$+0.09_{\pm 0.01}$	$+0.11_{\pm 0.06}$
RTE	RoBERTa	$80.09_{\pm 0.23}$	$-0.23_{\pm 0.20}$	$+0.70_{\pm 0.65}$	$-0.46_{\pm 0.12}$

(a)

Task	Model	Baseline	SWA	SAM	WASAM
NPP: Proteins	SAGE	$77.79_{\pm 0.18}$	$-0.17_{\pm 0.22}$	$-0.02_{\pm 0.13}$	$-0.11_{\pm 0.15}$
	DGCN	$85.42_{\pm 0.17}$	$+0.11_{\pm 0.08}$	$-0.14_{\pm 0.05}$	$-0.08_{\pm 0.07}$
NPP: Products	SAGE	$78.92_{\pm 0.08}$	$+0.39_{\pm 0.10}$	$+0.13_{\pm 0.08}$	$+0.57_{\pm 0.03}$
	DGCN	$73.88_{\pm 0.13}$	$+0.44_{\pm 0.14}$	$+0.08_{\pm 0.09}$	$+0.53_{\pm 0.05}$
GPP: Code2	GCN	$16.04_{\pm 0.09}$	$+0.73_{\pm 0.11}$	$+0.36_{\pm 0.08}$	$+0.93_{\pm 0.15}$
	GIN	$15.73_{\pm 0.11}$	$+0.83_{\pm 0.11}$	$+0.57_{\pm 0.09}$	$+1.10_{\pm 0.09}$
GPP: Molpcba	GIN	$28.10_{\pm 0.11}$	$+0.40_{\pm 0.18}$	$-0.33_{\pm 0.14}$	$+0.33_{\pm 0.16}$
	DGCN	$25.65_{\pm 0.13}$	$+1.90_{\pm 0.20}$	$-0.13_{\pm 0.18}$	$+1.34_{\pm 0.12}$
LPP: Biokg	CP	$84.06_{\pm 0.00}$	$+0.07_{\pm 0.01}$	$0.00_{\pm 0.03}$	$+0.08_{\pm 0.02}$
	ComplEx	$84.94_{\pm 0.01}$	$+0.14_{\pm 0.01}$	$-0.02_{\pm 0.01}$	$+0.12_{\pm 0.02}$
LPP: Citation2	GCN	$79.52_{\pm 0.41}$	$-0.05_{\pm 0.52}$	$+1.32_{\pm 0.06}$	$+1.50_{\pm 0.13}$
	SAGE	$81.95_{\pm 0.02}$	$+1.15_{\pm 0.02}$	$-0.31_{\pm 0.07}$	$+0.86_{\pm 0.04}$

(b)

Methods

3. When Do Flat Minima Optimizers Work?

❖ 9개의 발견

1. **데이터**에 따라서 Flat-minima 최적화 방법의 **성능 차이 발생**
2. **모델 구조**에 따라 Flat-minima 최적화 방법의 **성능 차이 발생**
3. **NLP**에서는 **SAM**이 SWA보다 높은 성능을 보이며, SWA는 성능 저하의 원인이 되기도 함
4. **GRL**에서는 **SWA**가 SAM보다 높은 성능을 보임
5. **Transformer** 기반의 모델(ViT, T5)에서는 **SWA**가 성능 저하의 원인이 됨
6. **Self Supervised Learning**에서는 **SWA**와 **SAM** 모두 성능 개선에 도움이 됨
7. 일부 Task에서는 Non-flat 최적화 기법이 여전히 높은 성능을 보임
8. SAM/SWA는 성능 개선 폭은 크지만, 성능 저하 폭은 적음
9. **WASAM**은 대부분의 Task에서 가장 높은 성능을 보임

Conclusion

Conclusion

❖ **Generalization performance에서 Flat Minima의 중요성**

- 딥러닝 모델을 통한 문제 해결에서 일반화 성능이 높은 local minima를 찾는 것이 중요함
- Flat한 minima를 찾는 방법론을 적용함으로써 높은 일반화 성능 달성 가능

1. **Averaging weights leads to wider optima and better generalization (UAI, 2018)**
   - ➢ 모델 가중치 평균화를 통해 Flat minima를 찾음으로써 일반화 성능 향상

2. **Sharpness-aware minimization for efficiently improving generalization (ICLR, 2021)**
   - ➢ Loss의 Sharpness를 고려한 목적함수를 통해 Flat Minima를 찾음으로써 일반화 성능 향상

3. **When Do Flat Minima Optimizers Work? (NeurIPS, 2022)**
   - ➢ 다양한 벤치마크 적용 인사이트 제공 및 WASAM 방법론 제안

❖ Flat Minima Optimizer은 다양한 문제의 **일반화 성능 향상에 효과적**일 것으로 판단됨

고맙습니다.