

# Slowing Down the Weight Norm Increase in Momentum-based Optimizers

Clova AI Research, NAVER Corp.

Sein Jang

tpdls24@gmail.com

July 13, 2020

# Contents

---

- Problem
- Method
- Experiment
- Conclusion

# Problem

모델 훈련 과정에서 **momentum-based optimizer**를 사용하면 **weight norm**이 급격하게 증가한다. **Optimization step size**는 **weight norm**과 반비례하기 때문에 step size의 감소로 인해서 최종적으로 sub-optimal한 모델 성능을 얻을 수 있다.

## 3 key point

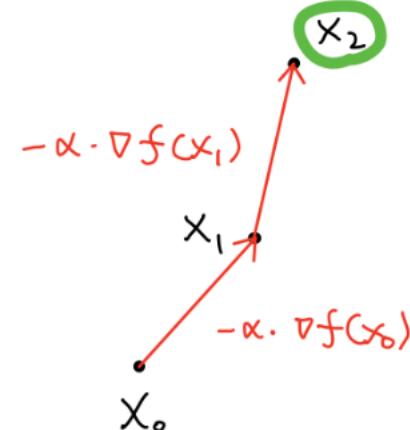
- Momentum-based optimizer
- Normalization
- Optimization step size

# Problem

## Momentum-based optimizer

기존 gradient descent optimizer의 한계

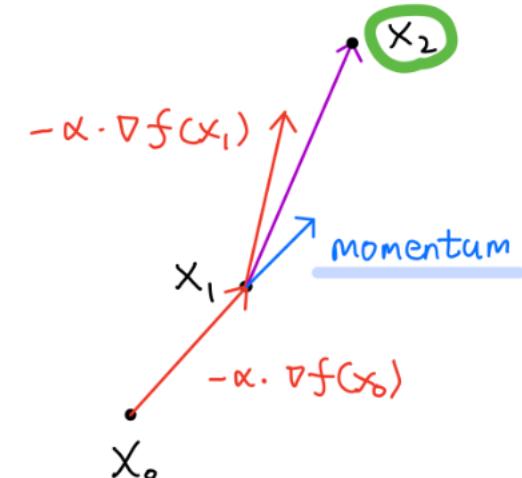
- 미분계수가 0인 지점에서 더 이상 이동하지 않는다.
- Global minimum이 아닌 local minimum으로 수렴한다.



Momentum-based optimizer

- Weight 업데이트에서 momentum이 추가되어 학습을 가속 시키고 기울기가 0인 지점에서도 업데이트가 된다.
- 초기값  $w_0$ , learning rate  $\eta$ , momentum  $\beta$ 에 대해서,
- $w_{t+1} \leftarrow w_t - \eta \cdot p_t$
- $p_t = \beta \cdot p_{t-1} + \nabla f(w_{t-1})$

$$x_1 = x_0 - \alpha \cdot \nabla f(x_0)$$
$$x_2 = x_1 - \alpha \cdot \nabla f(x_1)$$

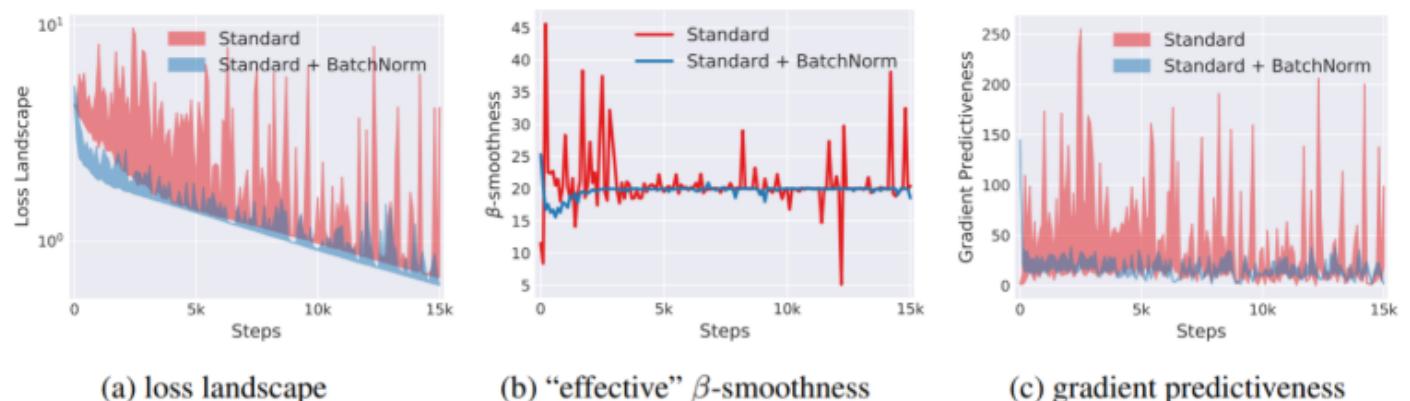
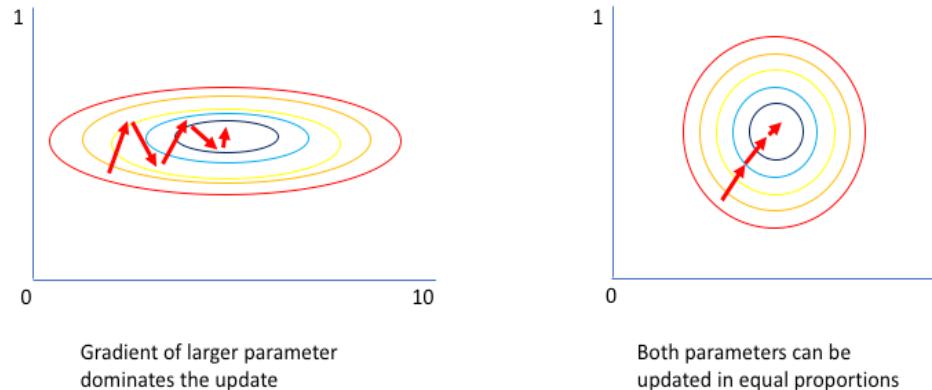


$$x_1 = x_0 - \alpha \cdot \nabla f(x_0)$$
$$x_2 = x_1 - \alpha \cdot \nabla f(x_1) + \text{momentum}$$

## Normalization

- Batch normalization
- Layer normalization
- Instance normalization
- Group normalization

Batch normalization을 사용하면 학습이 안정적으로 잘 되는 이유?



Santurkar, Shibani, et al. "How does batch normalization help optimization?." *Advances in Neural Information Processing Systems*. 2018.

## Problem - Normalization layer and scale invariance

- tensor  $x \in \mathbb{R}^{n_1 \times \dots \times n_r}$  에 대해서 normalization은 다음과 같이 정의된다
- $$Norm_k(x) = \frac{x - \mu_k(x)}{\sigma_k(x)}$$
       $\mu_k = mean, \sigma_k = standard deviation$
- 함수  $f(u)$ 에 대해서 만약  $c > 0$ 에 대해서  $f(cu) = f(u)$ 를 만족하면 scale invariant라고 한다.
- neural networks에서 만약  $Norm_k(w^T x) = Norm_k((cw)^T x)$ 라면  $Norm(\cdot)$ 은 scale invariant하다고 할 수 있다.
- weights의 norm  $\|w\|_2$ 는 neural network의 계산과정에서 forward  $f(w)$ 와 backward  $\nabla_w f(w)$ 에 아무런 영향을 주지 않기 때문에  $l_2$  normalized vector  $\hat{w} := \frac{w}{\|w\|_2}$ 를 scale-invariant weights로 나타낼 수 있다.

# Problem - Notations for the optimization steps

일반적인 gradient descent (GD) 알고리즘

$$w_{t+1} \leftarrow w_t - \eta p_t \quad (\eta = \text{learning rate})$$

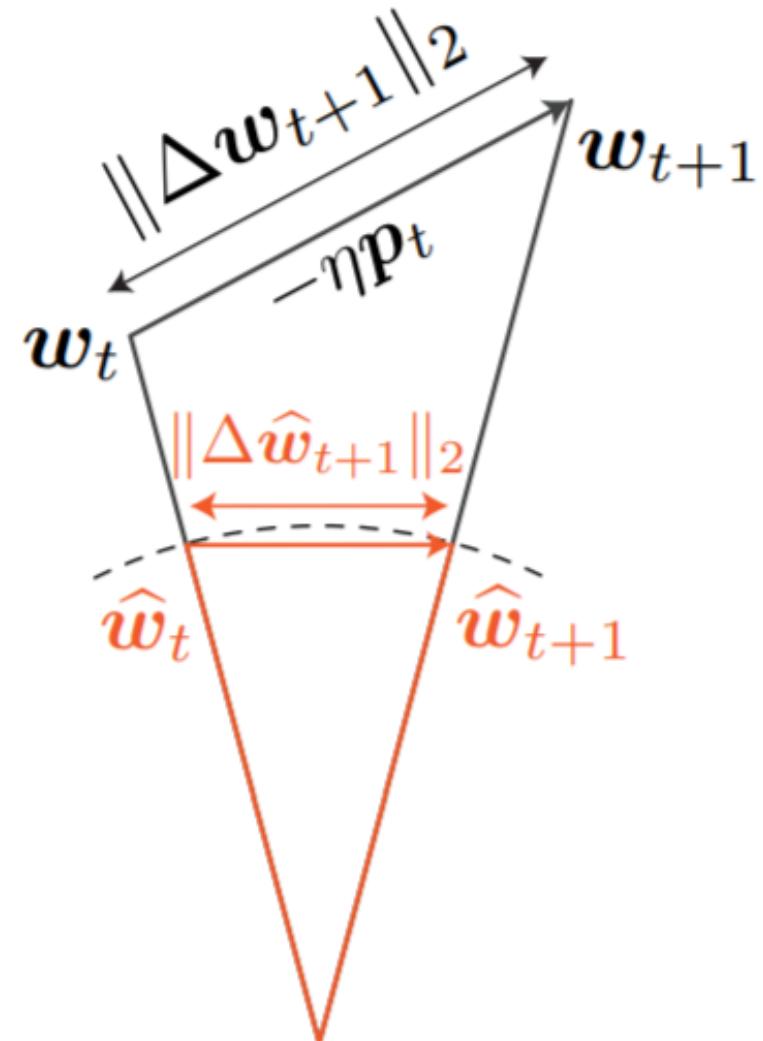
$$p = \nabla_w f(w)$$

- *step size*

$$\|\Delta w_{t+1}\|_2 := \|w_{t+1} - w_t\|_2 = \eta \|p_t\|_2$$

- *effective step size*

$$\|\Delta \hat{w}_{t+1}\|_2 := \|\hat{w}_{t+1} - \hat{w}_t\|_2$$



## Problem - Effective step sizes for vanilla gradient descent (GD)

scale invariance  $f(c\mathbf{w}) \equiv f(\mathbf{w})$  는 직교성(orthogonality)을 이끈다. 즉, 다음과 같은 수식을 만족하게 된다.

$$0 = \frac{\partial f(c\mathbf{w})}{\partial c} = \mathbf{w}^\top \nabla_{\mathbf{w}} f(\mathbf{w})$$

따라서, 일반적인 gradient descent 단계에서  $\mathbf{p} = \nabla_{\mathbf{w}} f(\mathbf{w})$  는 항상  $\mathbf{w}$ 에 수직이다.

이를 기반으로 *effective step size*를 계산하면 다음과 같이 나타낼 수 있다.

$$\|\widehat{\Delta \mathbf{w}}_{t+1}\|_2 := \left\| \frac{\mathbf{w}_{t+1}}{\|\mathbf{w}_{t+1}\|_2} - \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|_2} \right\|_2 \approx \left\| \frac{\mathbf{w}_{t+1}}{\|\mathbf{w}_t\|_2} - \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|_2} \right\|_2 = \frac{\|\Delta \mathbf{w}_t\|_2}{\|\mathbf{w}_t\|_2}$$

위 수식을 보면 *effective step size*는 *weight norm*에 반비례한다는 것을 확인할 수 있다.

**Lemma 2.1** (Norm growth by vanilla GD). *For a scale-invariant parameter  $\mathbf{w}$  and the vanilla GD, where  $\mathbf{p}_t = \nabla_{\mathbf{w}} f(\mathbf{w}_t)$ ,*

$$\|\mathbf{w}_{t+1}\|_2^2 = \|\mathbf{w}_t\|_2^2 + \eta^2 \|\mathbf{p}_t\|_2^2. \quad (6)$$

## Problem - Rapid norm growth for the momentum-based GD

momentum은 gradient-based optimization에서 수렴을 가속화하기 위해 사용되는 방법으로 momentum을 사용하면 weights는 다음과 같이 update 된다. 여기서  $\mathbf{p}_t$ 와  $\mathbf{w}_t$ 는 더 이상 수직관계가 아닐 수도 있다.

$$\begin{aligned}\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta \mathbf{p}_t \\ \mathbf{p}_t &\leftarrow \beta \mathbf{p}_{t-1} + \nabla_{\mathbf{w}_t} f(\mathbf{w}_t)\end{aligned}$$

아래 정리를 보면 momentum에 의해서 weight norm이 기존의 GD보다 더욱 증가하는 것을 확인할 수 있다.

**Lemma 2.2** (Norm growth by momentum). *For a scale-invariant parameter  $\mathbf{w}$  updated via equation 7, we have*

$$\|\mathbf{w}_{t+1}\|_2^2 = \|\mathbf{w}_t\|_2^2 + \eta^2 \|\mathbf{p}_t\|_2^2 + 2\eta^2 \sum_{k=0}^{t-1} \beta^{t-k} \|\mathbf{p}_k\|_2^2. \quad (8)$$

Momentum에 의해 추가된 부분이 weight norm의 증가를 더욱 가속 시킨다.

# Method – projected updates

## Projected updates

- Projection onto the tangent space of  $w$

$$\Pi_w(x) := x - (\hat{w} \cdot x)\hat{w}.$$

- Modified update rule

$$w_{t+1} = w_t - \eta q_t,$$

$$q_t = \begin{cases} \Pi_{w_t}(p_t) & \text{if } w^\top \nabla_w f(w) < \delta \\ p_t & \text{otherwise.} \end{cases}$$

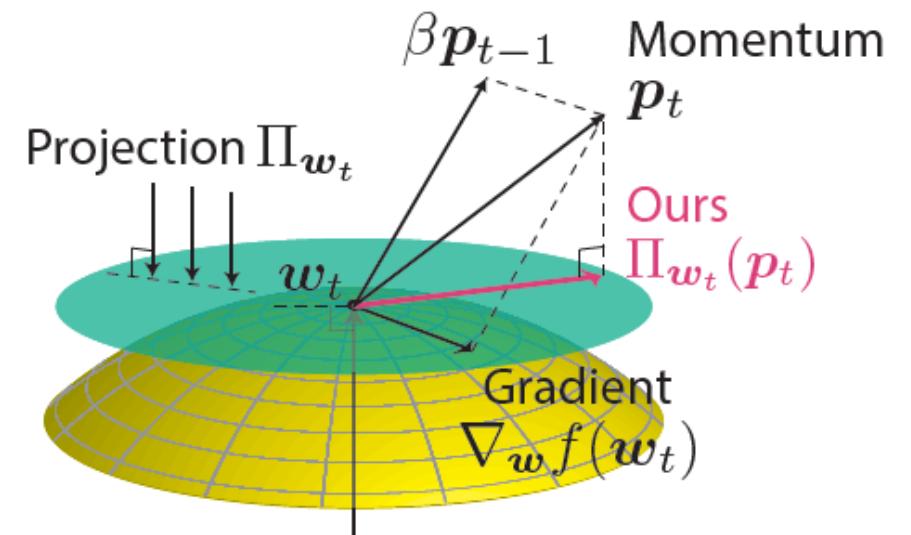


Figure 2: **Method.** Vector directions of the gradient, momentum, and ours.

---

## Algorithm 1: SGDP

---

**Require:** Learning rate  $\eta > 0$ , momentum  $\beta > 0$ , thresholds  $\delta, \varepsilon > 0$ .

```
1: while  $w_t$  not converged do
2:    $p_t \leftarrow \beta p_{t-1} + \nabla_w f_t(w_t)$ 
3:   if  $w_t \cdot \nabla_w f(w_t) < \delta$  then
4:      $w_{t+1} \leftarrow w_t - \eta \Pi_{w_t}(p_t)$ 
5:   else
6:      $w_{t+1} \leftarrow w_t - \eta p_t$ 
7:   end if
8: end while
```

---

---

## Algorithm 2: AdamP

---

**Require:** Learning rate  $\eta > 0$ , momentum  $0 < \beta_1, \beta_2 < 1$ , thresholds  $\delta, \varepsilon > 0$ .

```
1: while  $w_t$  not converged do
2:    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) \nabla_w f_t(w_t)$ 
3:    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_w f_t(w_t))^2$ 
4:    $p_t \leftarrow m_t / (\sqrt{v_t} + \varepsilon)$ 
5:   if  $w_t \cdot \nabla_w f(w_t) < \delta$  then
6:      $w_{t+1} \leftarrow w_t - \eta \Pi_{w_t}(p_t)$ 
7:   else
8:      $w_{t+1} \leftarrow w_t - \eta p_t$ 
9:   end if
10: end while
```

---

# Method - Empirical analysis of effective step sizes

Train ResNet18 models on ImageNet for 100 epochs with

- **vanilla SGD**
- **momentum SGD**
- **SGDP**

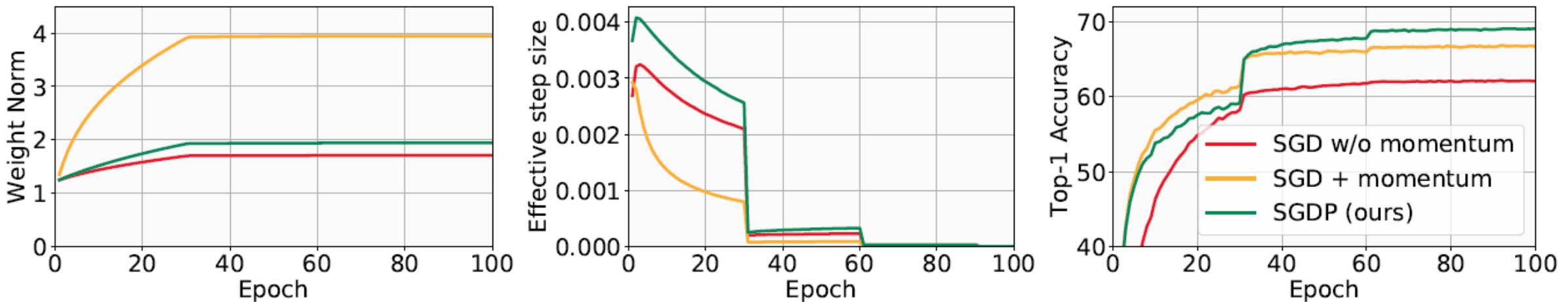


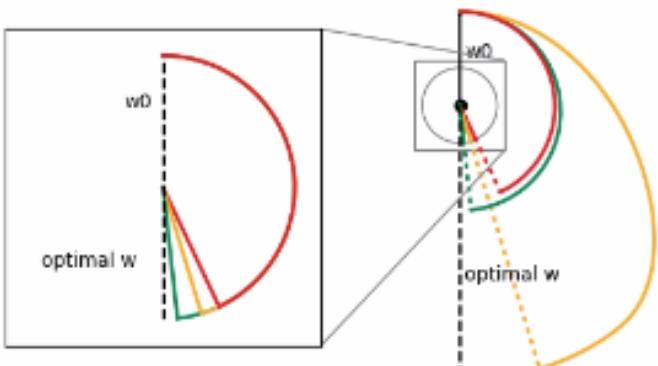
Figure 3: **Empirical analysis of optimizers.** Weight norms  $\|\mathbf{w}\|_2$  (left), effective step sizes  $\|\Delta\hat{\mathbf{w}}_t\|_2$  (center), and accuracies (right) for ResNet18 trained on ImageNet with variants of SGD.

# Experiments

## 2D Toy example

Momentum = 0.9

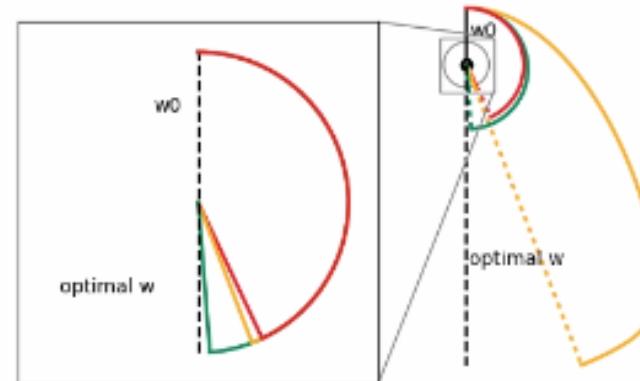
Iteration 491  
[Loss] GD: 0.0090 / GD + momentum: 0.0019 / ours: 0.0000  
[Norm] GD: 1.02 / GD + momentum: 2.93 / ours: 1.12



— Gradient descent (GD)  
— GD + momentum  
— GD + momentum + projection (ours)

Momentum = 0.95

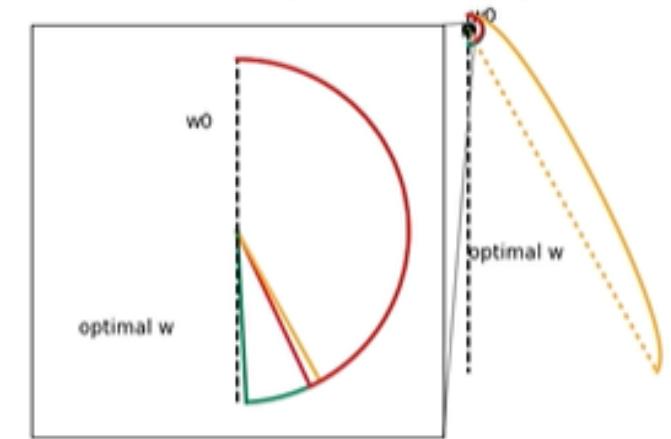
Iteration 491  
[Loss] GD: 0.0090 / GD + momentum: 0.0044 / ours: 0.0000  
[Norm] GD: 1.02 / GD + momentum: 5.67 / ours: 1.14



— Gradient descent (GD)  
— GD + momentum  
— GD + momentum + projection (ours)

Momentum = 0.99

Iteration 491  
[Loss] GD: 0.0090 / GD + momentum: 0.0155 / ours: 0.0000  
[Norm] GD: 1.02 / GD + momentum: 27.85 / ours: 1.15



— Gradient descent (GD)  
— GD + momentum  
— GD + momentum + projection (ours)

$$\min_w \frac{w}{\|w\|_2} \cdot \frac{w^*}{\|w^*\|_2}$$

두 벡터  $(w, w^*)$ 의 코사인 유사도를 최대화하는 문제

## Image classification

Table 1: **ImageNet classification.** Accuracies of state-of-the-art networks trained with SGDP and AdamP.

| Architecture                | # params | SGD [20] | SGDP (ours)          | Adam [9] | AdamP (ours)         |
|-----------------------------|----------|----------|----------------------|----------|----------------------|
| MobileNetV2 [22]            | 3.5M     | 71.61    | <b>72.18 (+0.57)</b> | 72.12    | <b>72.57 (+0.45)</b> |
| ResNet18 [11]               | 11.7M    | 70.28    | <b>70.73 (+0.45)</b> | 70.41    | <b>70.81 (+0.40)</b> |
| ResNet50 [11]               | 25.6M    | 76.64    | <b>76.71 (+0.07)</b> | 76.65    | <b>76.94 (+0.29)</b> |
| ResNet50 [11] + CutMix [23] | 25.6M    | 77.61    | <b>77.72 (+0.11)</b> | 78.00    | <b>78.31 (+0.31)</b> |

ImageNet-1k 데이터셋을 사용하여 Image classification에 대한 제안 기법의 성능 비교 결과  
위 Table 1에서와 같이 SGD, Adam optimizer에 비해 제안하는 SGDP, AdamP가 더 나은 정확  
도를 보여주고 있습니다.

## Image classification

Table 2: **Weight decay.** Optimizer performances for training ResNet18 on ImageNet.

|              | Baseline | w/o weight decay |
|--------------|----------|------------------|
| SGD          | 70.28    | 67.94 (-2.38)    |
| SGDP (ours)  | 70.73    | 70.21 (-0.52)    |
| Adam         | 70.41    | 68.52 (-1.89)    |
| AdamP (ours) | 70.81    | 70.50 (-0.31)    |

Norm의 증가를 방지하기 위한 기법으로 weight decay가 사용되기도 합니다. 하지만 논문에서 제안하는 Projection으로 이미 norm의 증가를 방지하기 때문에 weight decay를 적용하지 않아도 기존의 방식보다는 성능의 차이가 작은 것을 확인할 수 있습니다. 따라서 weight decay에 대한 부담을 줄일 수 있습니다.

## Object detection

Table 3: **MS-COCO object detection.** Average precision (AP) scores of CenterNet and SSD trained with Adam and AdamP optimizers.

| Model          | Initialize | Adam  | AdamP (ours)         |
|----------------|------------|-------|----------------------|
| CenterNet [24] | Scratch    | 26.57 | <b>27.11</b> (+0.54) |
| CenterNet [24] | ImageNet   | 28.29 | <b>29.05</b> (+0.76) |
| SSD [25]       | Scratch    | 27.10 | <b>27.97</b> (+0.87) |
| SSD [25]       | ImageNet   | 28.39 | <b>28.67</b> (+0.28) |

MS-COCO 데이터셋을 사용한 object detection에서도 제안 기법이 더 나은 성능을 보이는 것을 확인 할 수 있습니다. 각 모델에 대해서 ResNet18과 VGG16을 backbone으로 사용하였습니다.

# Experiments

## Robustness

PGD attacks을 활용하여 Wide-ResNet 모델을 Adversarial training 기법으로 학습시킨 결과 PGD L2 공격과 PGD Linf 공격 모든 경우에 대해서 제안한 AdamP이 기존의 Adam에 비해 약 9.3% 높은 정확도와 더 빠른 학습 수렴을 보여줬습니다.

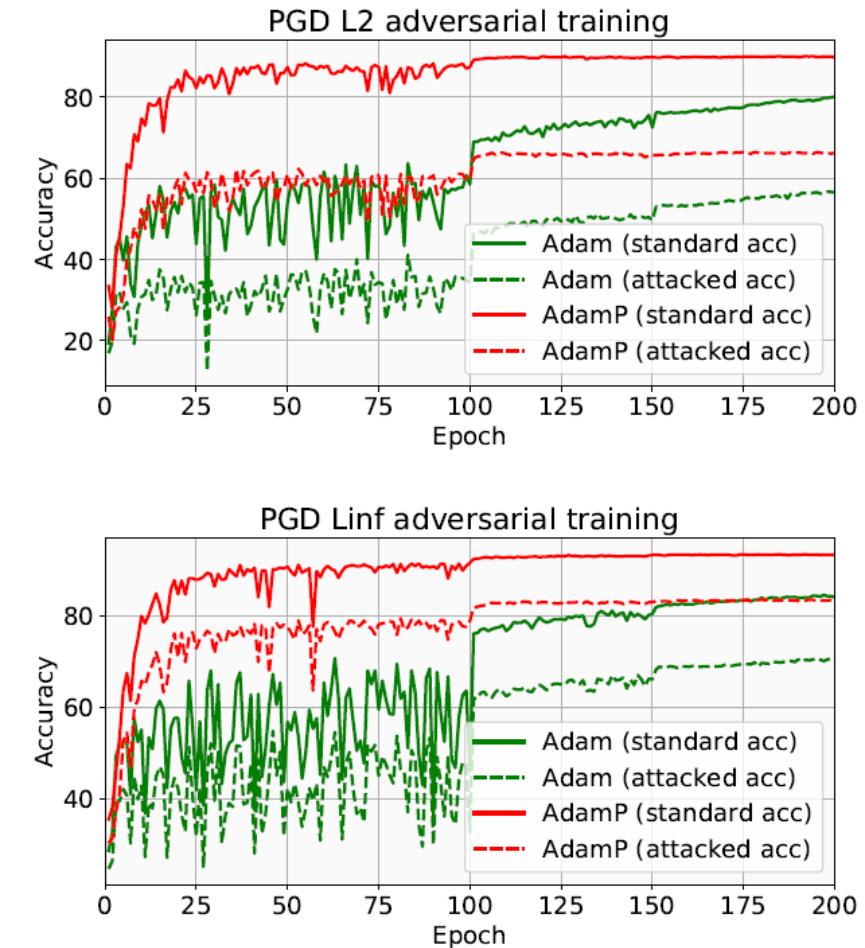


Figure 4: **Adversarial training.** Learning curves by Adam and AdamP.

# Experiments

## Robustness

Table 4: **Real-world bias robustness.** Biased MNIST and 9-Class ImageNet benchmarks with ReBias [32].

| Optimizer    | Biased MNIST Unbiased acc. at $\rho$ |                    |                    |                    |                    | 9-Class ImageNet   |                    |                    |
|--------------|--------------------------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
|              | .999                                 | .997               | .995               | .990               | avg.               | Biased             | UnBiased           | IN-A [34]          |
| Adam         | 22.9                                 | 63.0               | 74.9               | 87.0               | 61.9               | 93.8               | 92.6               | 31.2               |
| AdamP (ours) | <b>30.5 (+7.5)</b>                   | <b>70.9 (+7.9)</b> | <b>80.9 (+6.0)</b> | <b>89.6 (+2.6)</b> | <b>68.0 (+6.0)</b> | <b>95.2 (+1.4)</b> | <b>94.5 (+1.8)</b> | <b>32.9 (+1.7)</b> |

## Audio classification

Table 5: **Audio classification.** Results on three audio classification tasks with Harmonic CNN [38].

| Optimizer       | Music Tagging [35]   |                      | Keyword Spotting [36] | Sound Event Tagging [37] |
|-----------------|----------------------|----------------------|-----------------------|--------------------------|
|                 | ROC-AUC              | PR-AUC               | Accuracy              | F1 score                 |
| Adam + SGD [39] | 91.27                | 45.67                | 96.08                 | 54.60                    |
| Adam            | 91.12                | 45.61                | 96.47                 | 55.24                    |
| AdamP (ours)    | <b>91.35 (+0.23)</b> | <b>45.79 (+0.18)</b> | <b>96.89 (+0.42)</b>  | <b>56.04 (+0.80)</b>     |

제안한 AdamP는 audio classification의 총 3가지 task에 대해서 모두 기존의 방법론보다 좋은 성능을 보여줍니다. 해당 실험을 통해서 non-image 도메인에서도 제안하는 방법이 좋은 성능을 보인다는 것을 증명했습니다.

# Experiments

## Retrieval

Table 6: **Image retrieval.** Recall@1 on CUB, Cars-196, InShop, and SOP datasets. ImageNet-pretrained ResNet50 networks are fine-tuned by the triplet (semi-hard mining) [44] or the ProxyAnchor (PA) [45] loss.

| Optimizer    | CUB                |                    | Cars-196           |                    | InShop             |                    | SOP                |                    |
|--------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
|              | Triplet            | PA                 | Triplet            | PA                 | Triplet            | PA                 | Triplet            | PA                 |
| Adam         | 57.9               | 69.3               | 59.8               | 86.7               | 62.7               | 85.2               | 62.0               | 76.5               |
| AdamP (ours) | <b>58.2 (+0.3)</b> | <b>69.5 (+0.2)</b> | <b>59.9 (+0.2)</b> | <b>86.9 (+0.2)</b> | <b>62.8 (+0.0)</b> | <b>87.4 (+2.2)</b> | <b>62.6 (+0.6)</b> | <b>78.0 (+1.5)</b> |

## Conclusion

Momentum-based optimizers induce an excessive growth of the scale-invariant weight norms

The growth of weight norms slow down the progress of effective optimization steps, potentially leading to sub-optimal performances

Propose a simple and effective solution

- Projecting out the radial component from the optimization update at every iteration

# Q & A