Slowing Down the Weight Norm Increase in Momentum-based Optimizers

Byeongho Heo*, Sanghyuk Chun Chun*, Seong Joon Oh, Dongyoon Han,
Sangdoo Yun, Youngjung Uh, Jung-Woo Ha

Clova AI Research, NAVER Corp

Gio Paik
giopaik@naver.com

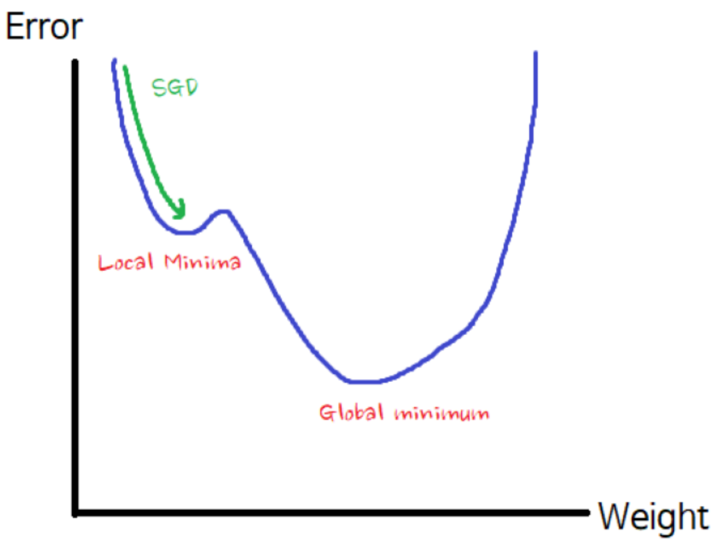
✱ indicates equal contribution

Two major breakthrough of DL

- Momentum-Based Weight Optimizers
  - Add "Momentum" to Optimizers
  - Give us more stable, fast convergence
  - Can go through the small Local Minima
- Normalization
  - Solve Internal Covariate Shift Problem
  - Makes scale-invariant weights.

Momentum-Based Weight Optimizer

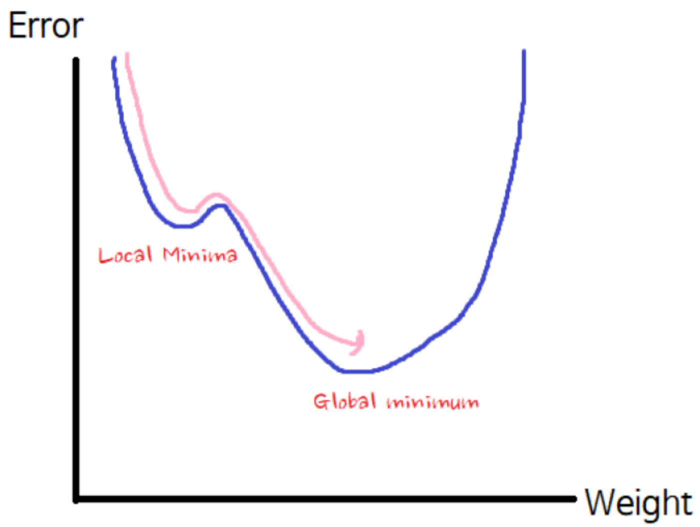
- Problem of previous (without momentum) optimizers.
  - Slow optimization
  - Prone to local minima



Let's add a bit of momentum!

Momentum-Based Weight Optimizer

- By adding Momentum to optimizer, we get..
  - More Stable and Fast Convergence
  - Immunity to local minima



$$W_1 = W_0 - \eta \frac{\partial cost}{\partial W} + \alpha v$$

Normalization

- To prevent overfitting, we should regularize model complexity
  - Batch Normalization
  - Weight Normalization
  - Other Normalization Techniques…

Example of L2 Weight Normalization, Gives penalty to Complex Weights of model.
λ is regularization strength

$$Final\ Cost = Cost(T, Y) + \frac{\lambda}{2}\|w\|_2$$

$$\|w\|_2 = w_1^2 + w_2^2 + w_3^2 + \cdots + w_n^2$$

Problem

- Normalization techniques in DNN result in the scale invariance for weights*
- Momentum-based optimizers with these scale-invariance abnormally increase the weight norms.
- Highly increased weight norms regularize the step size too earlier then we need.

*Elad Hoffer et al. Norm matters: efficient and accurate normalization schemes in deep networks., NIPS 2018.

Exploding Weight norm

- Figure on the right side,
  You can find out how weight norm drastically increased with momentum.

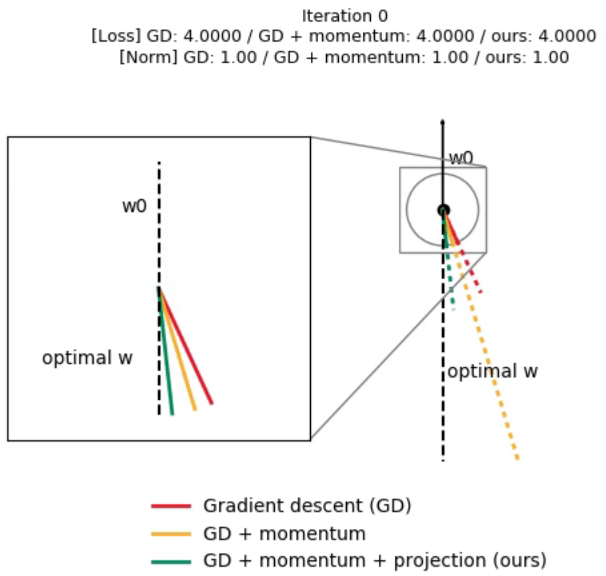


Figure from https://clovaai.github.io/AdamP/

Normalization and Scale invariance

- When if $f(cu) = f(u)$, we say that function f is *scale invariant*.
- When we normalize weight in neural nets, we observe
$$Norm(w^\intercal x) = Norm((cw)^\intercal x)$$
- For any $c > 0$, normalization layer makes w scale−invariant.
- In the paper, we represent the scale−invariant weight via their L2 normalized vectors $\hat{w} \coloneqq \frac{w}{\|w\|_2} \in \mathbb{S}^{d-1}$ $(\text{i.e. } \mathrm{c} = \frac{1}{\|w\|_2})$.

Notations for the optimization steps

- In this paper, we write a GD algorithm as below:
  $$w_{t+1} \leftarrow w_t - \eta p_t$$
  Where η is *learning rate* and p is $\nabla_w f(w)$.
- Thus, $\|\Delta w_{t+1}\|_2 \; := \; \|w_{t+1} - w_t\|_2 = \eta \|p_t\|_2$ is *step size*.
- Momentum−Based variants have more complex forms for p.

Notations for the optimization steps

- We will study the optimization problem in terms of the *L2 normalized weights* in $\mathbb{S}^{d-1}$, as opposed to the nominal space $\mathbb{R}^d$.
- As the result of GD algorithm, a virtual step takes place in $\mathbb{S}^{d-1}$.
- We refer to the length of the step on $\mathbb{S}^{d-1}$ as the *effective step size*.

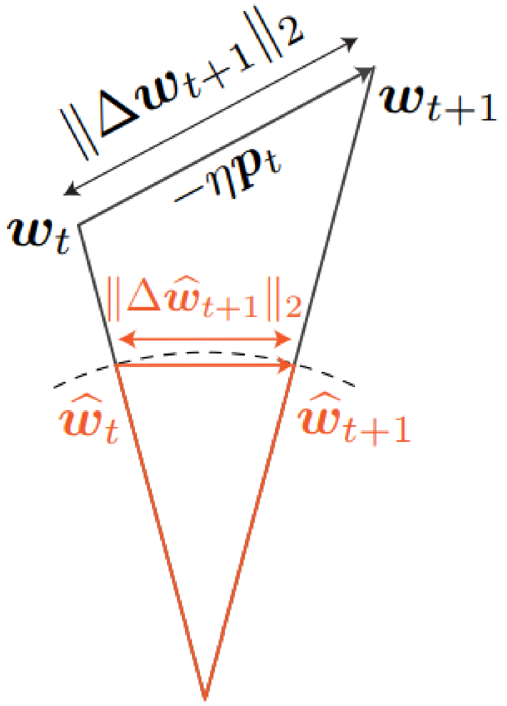


Illustration from the paper.

Effective Step Sizes for Vanilla GD

- Effective step sizes for vanilla GD is as follows.

$$\|\Delta\widehat{w}_{t+1}\|_2 \coloneqq \frac{\|\Delta w_t\|_2}{\|w_t\|_2}$$

- Following **Lemma 2.1** shows relationship between the effective step sizes and the weight norm.

$$\|w_{t+1}\|_2^2 = \|w_t\|_2^2 + \eta^2\|p_t\|_2^2.$$

- This Lemma indicates Norm growth by vanilla GD, where $p = \nabla_w f(w)$.
- The Lemma 2.1 follows from the orthogonality in equation below.

$$0 = \frac{\partial f(cw)}{\partial c} = w^\intercal \nabla_w f(w).$$

Rapid norm growth for the momentum-based GD

- Momentum is one of keystone for training modern DNNs.

$$w_{t+1} \leftarrow w_t - \eta p_t$$
$$p_t \leftarrow \beta p_{t-1} + \nabla_{w_t} f(w_t)$$

- Momentum−based GD uses previous gradient p_{t-1}, to avoid local minima and keep going to “Good Direction”.
- But what if the p_{t-1} is not a proper direction anymore?

Norm growth by Momentum

$$\|w_{t+1}\|_2^2 = \|w_t\|_2^2 + \eta^2\|p_t\|_2^2 + 2\eta^2 \sum_{k=0}^{t-1} \beta^{t-k}\|p_k\|_2^2$$

- Comparing this Lemma 2.2 with Lemma 2.1 in slide 11, you can notice the last term is added in Lemma 2.2.
- This term keep accumulating the past updates.
- It leads to **super−highly accelerated** increase of weight norms.
- We should remove the accumulated error term in Lemma 2.2, while retaining the benefits of momentum.

Lemma 2.2 from the paper.

Solution: Projection

- Let $\Pi_w(x)$ be a projection onto the tangent space of w:

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

- Let's apply $\Pi_w(x)$ to the momentum update p to remove unnecessary component.

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

$$q_t = \begin{cases} \Pi_w(p_t) & \text{if } w^\intercal \nabla_w f(w) < \delta \\ p_t & otherwise \end{cases}$$

- If $w^\intercal \nabla_w f(w) < \delta$ detects scale invariances, Algorithm takes projection of p_t.

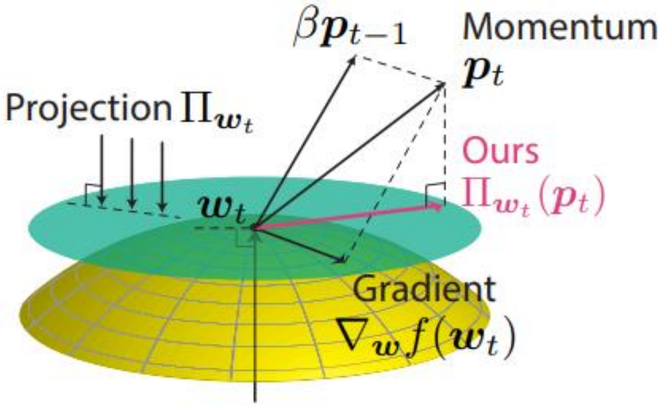


Figure 2: **Method.** Vector directions of the gradient, momentum, and ours.

Figure from the paper.

Solution: Projection

- It alleviate norm accumulation shown in Lemma 2.2 back to the vanilla GD growth rate in Lemma 2.1 due to the orthogonality.

$$\|w_{t+1}\|_2^2 = \|w_t\|_2^2 + \eta^2\|q_t\|_2^2.$$

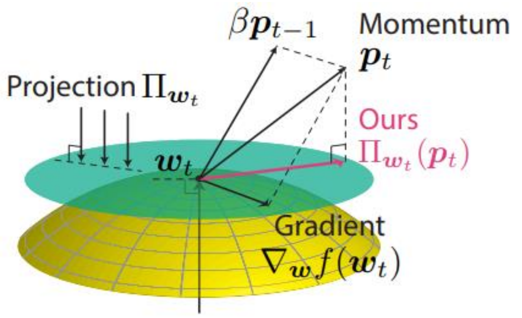


Figure 2: **Method.** Vector directions of the gradient, momentum, and ours.

Now, We can apply momentum without exploding weight norm.

Figure from the paper.

Algorithm Application: SGDP/AdamP

- Apply projection method to ADAM optimizer.

Algorithm 1: SGDP

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

while $\boldsymbol{w}_t$ not converged **do**
  $\boldsymbol{p}_t \leftarrow \beta \boldsymbol{p}_{t-1} + \nabla_{\boldsymbol{w}} f_t(\boldsymbol{w}_t)$
  if $\boldsymbol{w}_t \cdot \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_t) < \delta$ **then**
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \, \Pi_{\boldsymbol{w}_t}(\boldsymbol{p}_t)$
  else
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \, \boldsymbol{p}_t$
  end if
end while

Algorithm 2: AdamP

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

while $\boldsymbol{w}_t$ not converged **do**
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1) \nabla_{\boldsymbol{w}} f_t(\boldsymbol{w}_t)$
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2) (\nabla_{\boldsymbol{w}} f_t(\boldsymbol{w}_t))^2$
  $\boldsymbol{p}_t \leftarrow \boldsymbol{m}_t / (\sqrt{\boldsymbol{v}_t} + \varepsilon)$
  if $\boldsymbol{w}_t \cdot \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_t) < \delta$ **then**
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \, \Pi_{\boldsymbol{w}_t}(\boldsymbol{p}_t)$
  else
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \, \boldsymbol{p}_t$
  end if
end while

Algorithm from the paper.

Experiment

- Following shows the trend of three optimizers. Vanilla SGD, momentum SGD and finally Lemma 2.2 SGDP.
- Trained ResNet18 Models on ImageNet for 100 epochs.

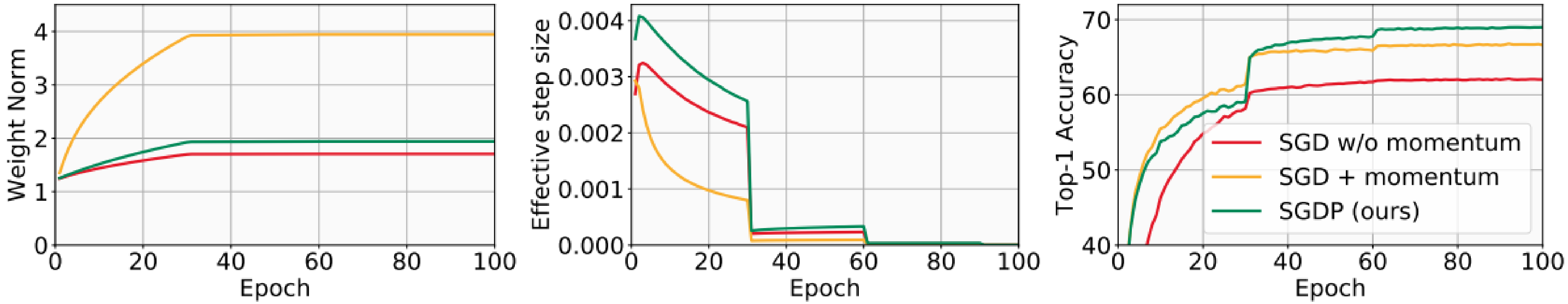


Figure 3: **Empirical analysis of optimizers.** Weight norms $\|\boldsymbol{w}\|_2$ (left), effective step sizes $\|\Delta\widehat{\boldsymbol{w}}_t\|_2$ (center), and accuracies (right) for ResNet18 trained on ImageNet with variants of SGD.

Figure from the paper.

Results

- SGDP and AdamP have shown better results on various DL tasks like Image Classification Object Detection, etc.

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	**72.18** (+0.57)	72.12	**72.57** (+0.45)
ResNet18 [11]	11.7M	70.28	**70.73** (+0.45)	70.41	**70.81** (+0.40)
ResNet50 [11]	25.6M	76.64	**76.71** (+0.07)	76.65	**76.94** (+0.29)
ResNet50 [11] + CutMix [23]	25.6M	77.61	**77.72** (+0.11)	78.00	**78.31** (+0.31)

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)	**91.35** (+0.23)	**45.79** (+0.18)	**96.89** (+0.42)	**56.04** (+0.80)

Tables from the paper.

Summary

- They found a huge problem: accumulated weight norms are slowing down the convergence.
- Problem came from Normalization and Momentum-based GD.
- Solved problem using projection to get rid of accumulated norms.
- And It works well!

Slowing Down the Weight Norm Increase in Momentum-based Optimizers

Official Project Page: https://clovaai.github.io/AdamP/

Paper: https://arxiv.org/abs/2006.08217

Official PyTorch Implementation: https://github.com/clovaai/adamp

Thank you for watching!

Gio Paik
giopaik@naver.com