Slowing Down the Weight Norm Increase in Momentum-based Optimizers

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

Clova AI Research, NAVER Corp.

Abstract

Normalization techniques, such as batch normalization (BN), have led to significant improvements in deep neural network performances. Prior studies have analyzed the benefits of the resulting scale invariance of the weights for the gradient descent (GD) optimizers: it leads to a stabilized training due to the auto-tuning of step sizes. However, we show that, combined with the momentum-based algorithms, the scale invariance tends to induce an excessive growth of the weight norms. This in turn overly suppresses the effective step sizes during training, potentially leading to sub-optimal performances in deep neural networks. We analyze this phenomenon both theoretically and empirically. We propose a simple and effective solution: at each iteration of momentum-based GD optimizers (*e.g.* SGD or Adam) applied on scale-invariant weights (*e.g.* Conv weights preceding a BN layer), we remove the radial component (*i.e.* parallel to the weight vector) from the update vector. Intuitively, this operation prevents the unnecessary update along the radial direction that only increases the weight norm without contributing to the loss minimization. We verify that the modified optimizers SGDP and AdamP successfully regularize the norm growth and improve the performance of a broad set of models. Our experiments cover tasks including image classification and retrieval, object detection, robustness benchmarks, and audio classification. Source code is available at https://github.com/clovaai/AdamP.

1 Introduction

Normalization techniques, such as batch normalization (BN) [1], layer normalization (LN) [2], instance normalization (IN) [3], and group normalization (GN) [4], have become standard tools for training deep neural network models. Originally proposed to reduce the internal covariate shift [1], normalization methods have proven to encourage several desirable properties in deep neural networks, such as better generalization [1, 5] and the scale invariance [6].

Prior studies have observed that the normalization-induced scale invariance of weights stabilizes the convergence for the neural network training [6, 7]. We provide a sketch of the argument here. Given weights $\boldsymbol{w}$ and an input $\boldsymbol{x}$, we observe that the normalization makes the weights scale-invariant:

$$\mathrm{Norm}(\boldsymbol{w}^\top \boldsymbol{x}) = \mathrm{Norm}(c\boldsymbol{w}^\top \boldsymbol{x}) \quad \forall c > 0. \tag{1}$$

The resulting equivalence relation among the weights lets us consider the weights only in terms of their ℓ_2-normalized vectors $\widehat{\boldsymbol{w}} := \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$ on the sphere $\mathbb{S}^{d-1} = \{\boldsymbol{v} \in \mathbb{R}^d : \|v\|_2 = 1\}$. We refer to $\mathbb{S}^{d-1}$ as the *effective space*, as opposed to the nominal space $\mathbb{R}^d$ where the actual optimization algorithms operate. The mismatch between these spaces results in the discrepancy between the gradient descent steps on $\mathbb{R}^d$ and their effective steps on $\mathbb{S}^{d-1}$. Specifically, for the gradient descent

*Equal contribution.

Preprint. Under review.

updates, the *effective step sizes* $\|\Delta\widehat{\boldsymbol{w}}_{t+1}\|_2 := \|\widehat{\boldsymbol{w}}_{t+1} - \widehat{\boldsymbol{w}}_t\|_2$ are the scaled versions of the nominal step sizes $\|\Delta\boldsymbol{w}_{t+1}\|_2 := \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|_2$ by the factor $\frac{1}{\|\boldsymbol{w}_t\|_2}$ [6]. Since the weight norm $\|\boldsymbol{w}_t\|_2$ generally increases during training [8, 7], the effective step sizes $\|\Delta\widehat{\boldsymbol{w}}_t\|_2$ decrease as the optimization progresses. The automatic decrease in step sizes stabilizes the convergence of gradient descent algorithms applied on models with normalization layers: even if the nominal learning rate is set to a constant, the theoretically optimal convergence rate is guaranteed [7].

In this work, we show that the widely used *momentum*-based gradient descent optimizers (*e.g.* SGD and Adam [9]) drastically boost up the increase in the weight norms, compared to the momentum-less counterparts considered in [7], and in turn prematurely reduce the effective step sizes $\Delta\widehat{\boldsymbol{w}}_t$. This leads to a slower effective convergence for $\widehat{\boldsymbol{w}}_t$ and potentially sub-optimal model performances. We illustrate this effect on a 2D toy optimization problem in Figure 1. Compared to "GD", "GD+momentum" speeds up the norm growth, resulting in a slower effective convergence in $\mathbb{S}^1$, though being faster in the nominal space $\mathbb{R}^2$.

We propose a simple solution to slow down the step size decay while maintaining the benefits of momentum. At each iteration of a momentum-based gradient descent optimizer, we propose to project out the radial component (*i.e.* component parallel to $\boldsymbol{w}$) from the update, thereby reducing the increase in the weight norm over time. The procedure does not hinder the loss minimization, since the loss is constant along the radial direction due to the scale invariance. We can immediately observe the benefit of our optimizer in the toy setting in Figure 1. "Ours" suppresses the norm growth, allowing the momentum-accelerated convergence in $\mathbb{R}^2$ to be transferred to the actual space $\mathbb{S}^1$. "Ours" converges most quickly and achieves the best terminal objective value.

The projection algorithm is simple and readily applicable to various optimizers for deep neural networks. We apply this technique on SGD and Adam (SGDP and AdamP, respectively) and verify the performance boosts over a diverse set of practical machine learning tasks including image classification, image retrieval, object detection, robustness benchmarks, and audio classification.

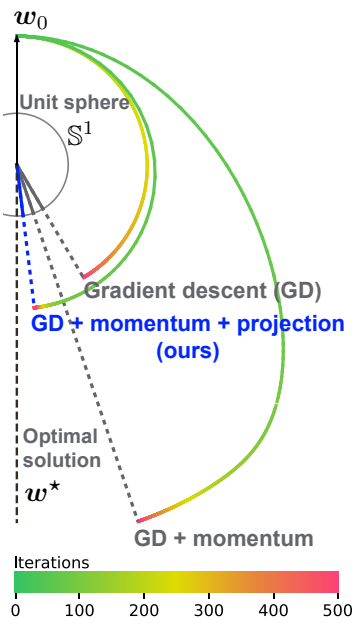


Figure 1: **Optimizer trajectories.** Shown is the $\boldsymbol{w}_t$ for the optimization problem $\min_{\boldsymbol{w}} -\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2} \cdot \frac{\boldsymbol{w}^\star}{\|\boldsymbol{w}^\star\|_2}$. Trajectories start from $\boldsymbol{w}_0$ towards the optimal solution $\boldsymbol{w}^\star$. The problem is invariant to the scale of $\boldsymbol{w}$. Video version in our project page.

Our contributions are: conceptual findings that the momentum induces rapid growth of weight norms (§2); a novel optimization module applicable to general gradient descent algorithms on scale-invariant weights (§3); a wide set of experiments to show the versatility and effectiveness of our method (§5).

2 Problem

Widely-used normalization techniques in deep networks result in the scale invariance for weights. We show that momentum-based optimizers, when applied on such scale-invariant parameters, result in an excessive growth of weight norms during training. This is problematic because the effective optimization step sizes are inversely proportional to the weight norm; the premature decay of effective step sizes may lead to sub-optimal model performances. In this section, we provide an analysis of the weight norm growths and effective step sizes. The analysis motivates our optimizer in §3.

2.1 Normalization layer and scale invariance

For a tensor $x \in \mathbb{R}^{n_1 \times \cdots \times n_r}$ of rank r, we define the normalization operation along the axes $\boldsymbol{k} \in \{0, 1\}^{\{1,\cdots,r\}}$ as $\mathrm{Norm}_{\boldsymbol{k}}(x) = \frac{x - \mu_{\boldsymbol{k}}(x)}{\sigma_{\boldsymbol{k}}(x)}$ where $\mu_{\boldsymbol{k}}, \sigma_{\boldsymbol{k}}$ are the mean and standard deviation functions along the axes $\boldsymbol{k}$, without axes reduction (to allow broadcasted operations with x). Depending on $\boldsymbol{k}$, $\mathrm{Norm}_{\boldsymbol{k}}$ includes special cases like batch normalization (BN) [1].

For a function $f(\boldsymbol{u})$, we say that f is *scale invariant* if $f(c\boldsymbol{u}) = f(\boldsymbol{u})$ for any $c > 0$. We then observe that $\mathrm{Norm}(\cdot)$ is scale invariant. In particular, under the context of neural networks,

$$\mathrm{Norm}(\boldsymbol{w}^\top \boldsymbol{x}) = \mathrm{Norm}((c\boldsymbol{w})^\top \boldsymbol{x}) \tag{2}$$

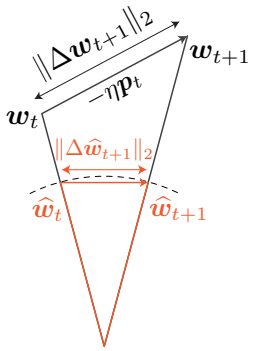
for any $c > 0$, leading to the scale invariance against the weights $\boldsymbol{w}$ preceding the normalization layer. The norm of such weights $\|\boldsymbol{w}\|_2$ does not affect the forward $f(\boldsymbol{w})$ or the backward $\nabla_{\boldsymbol{w}} f(\boldsymbol{w})$ computations of a neural network, where f is the loss function. We may represent the scale-invariant weights via their ℓ_2-normalized vectors $\widehat{\boldsymbol{w}} := \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2} \in \mathbb{S}^{d-1}$ (*i.e.* $c = \frac{1}{\|\boldsymbol{w}\|_2}$).

2.2 Notations for the optimization steps

See the illustration on the right for the summary of notations describing an optimization step. We write a general gradient descent (GD) algorithm as:

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \boldsymbol{p}_t \tag{3}$$

where $\eta > 0$ is the user-defined *learning rate*. The norm of the difference $\|\Delta \boldsymbol{w}_{t+1}\|_2 := \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|_2 = \eta \|\boldsymbol{p}_t\|_2$ is referred to as the *step size*. When $\boldsymbol{p} = \nabla_{\boldsymbol{w}} f(\boldsymbol{w})$, equation 3 is the vanilla GD algorithm. Momentum-based variants have more complex forms for $\boldsymbol{p}$.



In this work, we study the optimization problem in terms of the ℓ_2 normalized weights in $\mathbb{S}^{d-1}$, as opposed to the nominal space $\mathbb{R}^d$. As the result of equation 3, a virtual step takes place in $\mathbb{S}^{d-1}$. We refer to the length of the step $\|\Delta \widehat{\boldsymbol{w}}_{t+1}\|_2 = \|\widehat{\boldsymbol{w}}_{t+1} - \widehat{\boldsymbol{w}}_t\|_2$ as the *effective step size*.

2.3 Effective step sizes for vanilla gradient descent (GD)

We approximate the effective step sizes for the scale-invariant $\boldsymbol{w}$ under the vanilla GD algorithm. We observe that the scale invariance $f(c\boldsymbol{w}) \equiv f(\boldsymbol{w})$ leads to the orthogonality:

$$0 = \frac{\partial f(c\boldsymbol{w})}{\partial c} = \boldsymbol{w}^\top \nabla_{\boldsymbol{w}} f(\boldsymbol{w}). \tag{4}$$

For example, the vanilla GD update step $\boldsymbol{p} = \nabla_{\boldsymbol{w}} f(\boldsymbol{w})$ is always perpendicular to $\boldsymbol{w}$. Based on this, we establish the effective step size for $\boldsymbol{w}$ on $\mathbb{S}^{d-1}$:

$$\|\Delta \widehat{\boldsymbol{w}}_{t+1}\|_2 := \left\| \frac{\boldsymbol{w}_{t+1}}{\|\boldsymbol{w}_{t+1}\|_2} - \frac{\boldsymbol{w}_t}{\|\boldsymbol{w}_t\|_2} \right\|_2 \approx \left\| \frac{\boldsymbol{w}_{t+1}}{\|\boldsymbol{w}_t\|_2} - \frac{\boldsymbol{w}_t}{\|\boldsymbol{w}_t\|_2} \right\|_2 = \frac{\|\Delta \boldsymbol{w}_t\|_2}{\|\boldsymbol{w}_t\|_2} \tag{5}$$

where the approximation assumes $\frac{1}{\|\boldsymbol{w}_{t+1}\|_2} - \frac{1}{\|\boldsymbol{w}_t\|_2} = o(\eta)$, which holds when $\boldsymbol{p}_t \perp \boldsymbol{w}_t$ as in the vanilla GD. We have thus derived that the effective step size on $\mathbb{S}^{d-1}$ is inversely proportional to the weight norm, in line with the results in [6].

Having established the relationship between the effective step sizes and the weight norm, we derive the formula for its growth under the vanilla GD optimization.

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

$$\|\boldsymbol{w}_{t+1}\|_2^2 = \|\boldsymbol{w}_t\|_2^2 + \eta^2 \|\boldsymbol{p}_t\|_2^2. \tag{6}$$

The lemma follows from the orthogonality in equation 4 and the Pythagorean theorem. It follows that the norm of a scale-invariant parameter $\|\boldsymbol{w}\|_2$ is monotonically increasing and consequently decreases the effective step size for $\boldsymbol{w}$, as shown in Figure 1. [7] has further shown that GD with the above adaptive step sizes converges to a stationary point at the theoretically optimal convergence rate $O(T^{-1/2})$ under a fixed learning rate $\eta = C$.

2.4 Rapid norm growth for the momentum-based GD

Momentum is designed to accelerate the convergence of gradient-based optimization by letting $\boldsymbol{w}$ escape high-curvature regions and cope with small and noisy gradients [10]. It has become an indispensable ingredient for training modern deep neural networks. A momentum update follows:

$$\begin{aligned} \boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t - \eta \boldsymbol{p}_t \\ \boldsymbol{p}_t &\leftarrow \beta \boldsymbol{p}_{t-1} + \nabla_{\boldsymbol{w}_t} f(\boldsymbol{w}_t) \end{aligned} \tag{7}$$

for steps $t \geq 0$, where $\beta \in (0, 1)$ and $\boldsymbol{p}_{-1}$ is initialized at $\mathbf{0}$. Note that the step direction $\boldsymbol{p}_t$ and the parameter $\boldsymbol{w}_t$ may not be perpendicular anymore. We show below that momentum increases the weight norm under the scale invariance, even more so than does the vanilla GD.

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

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

Proof is in the Appendix A. Comparing Lemma 2.1 and 2.2, we notice that the formulation is identical, except for the last term on the right hand side of Lemma 2.2. This term is not only non-negative, but also is an accumulation of the past updates. This additional term results in the significantly accelerated increase of weight norms when the momentum is used. We derive a more precise asymptotic ratio of the weight norms for the GD with and without momentum below.

Corollary 2.3 (Asymptotic norm growth comparison)**.** *Let $\|\boldsymbol{w}_t^{\mathrm{GD}}\|_2$ and $\|\boldsymbol{w}_t^{\mathrm{GDM}}\|_2$ be the weight norms at step $t \geq 0$, following the recursive formula in Lemma 2.1 and 2.2, respectively. We assume that the norms of the updates $\|\boldsymbol{p}_t\|_2$ for GD with and without momentum are identical for every $t \geq 0$. We further assume that the sum of the update norms is non-zero and bounded: $0 < \sum_{t\geq 0} \|\boldsymbol{p}_t\|_2^2 < \infty$. Then, the asymptotic ratio between the two norms is given by:*

$$\frac{\|\boldsymbol{w}_t^{\mathrm{GDM}}\|_2^2 - \|\boldsymbol{w}_0\|_2^2}{\|\boldsymbol{w}_t^{\mathrm{GD}}\|_2^2 - \|\boldsymbol{w}_0\|_2^2} \longrightarrow 1 + \frac{2\beta}{1-\beta} \quad \text{as} \quad t \to \infty. \tag{9}$$

Proof in the Appendix A. While the identity assumption for $\|\boldsymbol{p}_t\|_2$ between GD with and without momentum is strong, the theory is designed to illustrate an approximate norm growth ratios between the algorithms. For a popular choice of $\beta = 0.9$, the factor is as high as $1 + 2\beta/(1-\beta) = 19$. In §3.2, we empirically demonstrate the momentum-induced norm increase.

3 Method

We have studied the accelerated norm growth for scale-invariant weights (*e.g.* those preceding a normalization layer) under the momentum. In this section, we propose a projection-based solution that regularizes the momentum-induced norm growth and improves model performances.

3.1 Our method: Projected updates

We remove the accumulated error term in Lemma 2.2, while retaining the benefits of momentum, through a simple modification. Let $\Pi_{\boldsymbol{w}}(\cdot)$ be a projection onto the tangent space of $\boldsymbol{w}$:

$$\Pi_{\boldsymbol{w}}(\boldsymbol{x}) := \boldsymbol{x} - (\widehat{\boldsymbol{w}} \cdot \boldsymbol{x})\widehat{\boldsymbol{w}}. \tag{10}$$

We apply $\Pi_{\boldsymbol{w}}(\cdot)$ to the momentum update $\boldsymbol{p}$ (*e.g.* equation 7) to remove the radial component, which only accumulates the weight norms without contributing to the loss minimization. Our modified update rule is:

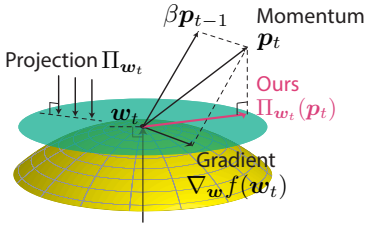


Figure 2: **Method.** Vector directions of the gradient, momentum, and ours.

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \eta \boldsymbol{q}_t, \\ \boldsymbol{q}_t &= \begin{cases} \Pi_{\boldsymbol{w}_t}(\boldsymbol{p}_t) & \text{if } \boldsymbol{w}^\top \nabla_{\boldsymbol{w}} f(\boldsymbol{w}) < \delta \\ \boldsymbol{p}_t & \text{otherwise.} \end{cases} \end{aligned} \tag{11}$$

Instead of manually registering weights preceding normalization layers, our algorithm automatically detects scale invariances with the criterion $\boldsymbol{w}^\top \nabla_{\boldsymbol{w}} f(\boldsymbol{w}) < \delta$ for user convenience. The proposed update rule makes a scale-invariant parameter $\boldsymbol{w}$ perpendicular to its update step $\boldsymbol{q}$. It follows then that the rapid weight norm accumulation shown in Lemma 2.2 is alleviated back to the vanilla gradient descent growth rate in Lemma 2.1 due to the orthogonality:

$$\|\boldsymbol{w}_{t+1}\|_2^2 = \|\boldsymbol{w}_t\|_2^2 + \eta^2\|\boldsymbol{q}_t\|_2^2. \tag{12}$$

The proposed method is readily adaptable to existing gradient-based optimization algorithms like SGD and Adam. Their modifications, SGDP and AdamP are shown in Algorithms 1 and 2, respectively (Modifications are colorized). Note that for Adam the projection is applied on the Adam momentum.

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

3.2 Empirical analysis of effective step sizes

To verify that the proposed projection reduces the excessive norm growth and slows down the decay of effective step sizes, we design a set of controlled experiments. We train ResNet18 [11] models on ImageNet [12] for 100 epochs with the vanilla SGD, momentum SGD, and SGDP (ours). We measure the average ℓ_2 norm of the scale-invariant parameters $\|\boldsymbol{w}_t^l\|_2$ across iterations t in each epoch and across scale-invariant layers l in the model[2]. The averaged effective step sizes $\|\Delta \widehat{\boldsymbol{w}}_t\|_2$ are also reported. We use the step decay schedule for η_t: multiply with factor 0.1 at every 30 epochs. The step decay scheduling is a practical and effective scheme used by many applications.

Figure 3 shows the trends for the three optimizers. Compared to vanilla SGD, momentum SGD exhibits a steep increase in $\|\boldsymbol{w}\|_2$, resulting in a quick drop in the effective step sizes, validating Lemma 2.2. SGDP, on the other hand, does not allow the norm to increase far beyond the level of vanilla SGD. It maintains the effective step size at a comparable magnitude as the vanilla SGD does.

Final performances reflect the benefit of the regularized norm growths. While momentum itself is a crucial ingredient for improved model performances [10], further gain is possible by regularizing the norm growth. Compared to the momentum SGD performance (66.6% accuracy), SGDP achieves 69.0% accuracy. SGDP fully realizes the performance gain from the momentum by not overly suppressing the effective step sizes. We have reached the same observations and conclusions with Adam and AdamP (ours).

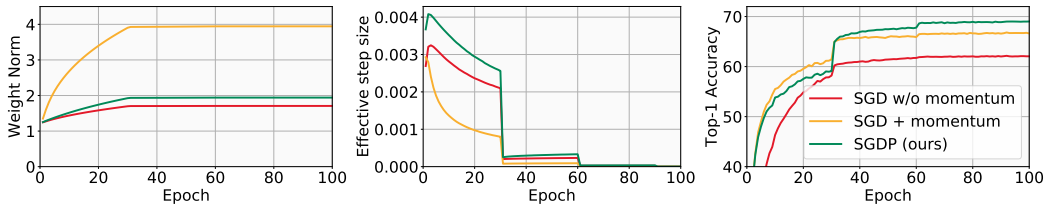


Figure 3: **Empirical analysis of optimizers.** Weight norms $\|\boldsymbol{w}\|_2$ (left), effective step sizes $\|\Delta \widehat{\boldsymbol{w}}_t\|_2$ (center), and accuracies (right) for ResNet18 trained on ImageNet with variants of SGD.

4 Related work

We provide a brief overview of related prior work. A line of work is dedicated to the development general and effective optimizers, such as Adagrad [13], Adam [9], and RMSprop. Researchers have sought strategies to improve Adam through *e.g.* improved convergence [14], warmup learning rate [15], moving average [16], Nesterov momentum [17], and rectified weight decay [18]. Another line of researches studies existing optimization algorithms in greater depth. For example, [6, 7, 19] have delved into the effective learning rates on scale-invariant weights. This paper at the intersection between the two. We study the issues when momentum-based optimizers are applied on scale-invariant weights. We then propose a new optimization method to address the problem.

[2] ℓ_2 norm is taken channel-wise because BN computes the mean and variance per channel.

5 Experiments

In this section, we demonstrate the effectiveness of our projection module for training scale-invariant weights with momentum-based optimizers. We experiment over various real-world tasks and datasets. From the image domain, we show results on ImageNet classification (§5.1), object detection (§5.2), and robustness benchmarks (§5.3). From the audio domain, we study music tagging, speech recognition, and sound event detection (§5.4). Finally, the metric learning experiments with ℓ_2 normalized embeddings (§5.5) show that our method works also on the scale invariances that do not originate from the statistical normalization. In the above set of experiments, we show that the proposed modifications (SGDP and AdamP) bring consistent performance gains against the baselines (SGD [20] and Adam [9]). We provide the implementation details in the Appendix C.

5.1 Image classification

Batch normalization (BN) and momentum-based optimizer are standard techniques to train state-of-the-art image classification models [11, 21, 22]. We evaluate the proposed method with ResNet [11], one of the most popular and powerful architectures on ImageNet, and MobileNetV2 [22], a relatively lightweight model with ReLU6 and depthwise convolutions, on the ImageNet-1K benchmark [12]. For ResNet, we employ the training hyperparameters in [11]. For MobileNetV2, we have searched for the best setting, as it is generally difficult to train it with the usual settings. We use the cosine annealed learning rates for all experiments.

Our optimizers SGDP and AdamP are compared against their corresponding baselines in Table 1. Across the spectrum of network sizes from MobileNetV2 to ResNet50, our optimizers outperform the baselines. Even when the state-of-the-art CutMix [23] regularization is applied, our optimizers bring further gains.

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)

Impact of weight decay. Projection is not the only way to suppress the norm increases. Weight decay (WD) is another simple way to regularize them. Note that WD is used throughout our experiments. We study the effects of removing WD for each optimizer considered. Table 2 shows the results. Removing WD results in performance drops, signifying the importance of the weight norm regularization. For SGDP and AdamP, however, the drops are less dramatic. As our optimizers already regularize the norm growth, the models rely less on WD for suppressing the norms. Our method thus lifts the burden for practitioners to choose WD carefully. We report the weight norms and effective step sizes for various WD values in the Appendix D.

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)

5.2 Object detection

Object detection is another widely-used real-world task where the models often include normalization layers and are trained with momentum-based optimizers. We study the two detectors CenterNet [24] and SSD [25] to verify that the proposed optimizers are also applicable to various objective functions beyond the classification task. The detectors are either initialized with the ImageNet-pretrained network (official PyTorch models) or trained from scratch,

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	**27.11** (+0.54)
CenterNet [24]	ImageNet	28.29	**29.05** (+0.76)
SSD [25]	Scratch	27.10	**27.97** (+0.87)
SSD [25]	ImageNet	28.39	**28.67** (+0.28)

in order to separate the effect of our method from that of the pretraining. ResNet18 [11] and VGG16 BN [26] are used for the CenterNet and SSD backbones, respectively. In Table 3, we report average precision performances based on the MS-COCO [27] evaluation protocol. We observe that AdamP boosts the performance against the baselines. It demonstrates the versatility of our optimizers.

5.3 Robustness

Model robustness is an emerging problem in real-world applications of machine learning. Due to the inherent difficulty of the problem, methods typically involve complex optimization problems. We examine how our optimizers stabilize the complex optimization problems.

Adversarial robustness. Adversarial training alternatively optimizes a minimax problem where the inner optimization is an adversarial attack and the outer optimization is the standard classification problem. Adam is commonly used for the adversarial training in order to handle the complexity of the optimization. The adversarial robustness remains a difficult problem; even the current best solutions have large generalization gaps between the standard accuracies and attacked accuracies [28, 29].

In our experiments, we train Wide-ResNet [30] with the projected gradient descent (PGD) [31] attacks. We use 10 inner PGD iterations and $\varepsilon = 80/255$ for the L_2 PGD attack and $\varepsilon = 4/255$ for the L_∞ PGD attack. Figure 4 shows the effect of AdamP. By handling the effective step sizes, AdamP achieves a faster convergence than Adam (less than half the epochs required); loss plots in the Appendix. AdamP brings more than $+9.3$ pp performance gap in all settings.

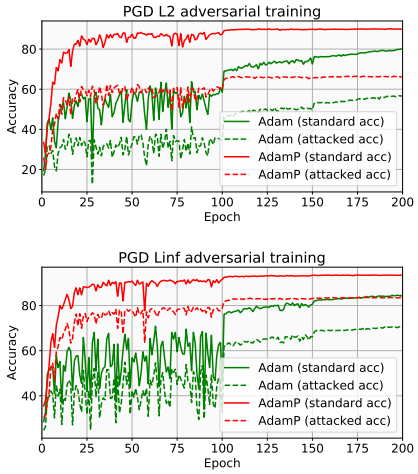


Figure 4: **Adversarial training.** Learning curves by Adam and AdamP.

Robustness against real-world biases. Cross-bias generalization problem [32] tackles the scenario where the training and test distributions have different real-world biases. This often occurs when the training-set biases provide an easy shortcut to solve the problem: *e.g.*, using the snow cues for recognizing snowmobiles. [32] has proposed the ReBias scheme based on the minimax optimization, where the inner problem maximizes the independence between an intentionally biased representation and the target model of interest and the outer optimization solves the standard classification problem. As for the adversarial training, [32] has employed the Adam to handle the complex optimization.

We follow the two cross-bias generalization benchmarks proposed by [32]. The first benchmark is the Biased MNIST, the dataset synthesized by injecting colors on the MNIST background pixels. Each sample is colored according to a pre-defined class-color mapping with probability ρ. The color is selected at random with $1 - \rho$ chance. For example, $\rho = 1.0$ leads a completely biased dataset and $\rho = 0.1$ leads to an unbiased dataset. Each model is trained on the ρ-biased MNIST and tested on the unbiased MNIST. We train a stacked convolutional network with BN and ReLU. The second benchmark is the 9-Class ImageNet representing the real-world biases, such as textures [33]. The unbiased accuracy is measured by pseudo-labels generated by the texture clustering. We also report the performance on ImageNet-A [34], the collection of failure samples of existing CNNs.

In Table 4, we observe that AdamP consistently outperforms Adam in all the benchmarks. AdamP is a good alternative to Adam for difficult optimization problems applied on scale-invariant parameters.

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

Optimizer	Biased MNIST Unbiased acc. at ρ					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)	**30.5** (+7.5)	**70.9** (+7.9)	**80.9** (+6.0)	**89.6** (+2.6)	**68.0** (+6.0)	**95.2** (+1.4)	**94.5** (+1.8)	**32.9** (+1.7)

5.4 Audio classification

We evaluate the proposed optimizer on three different audio classification tasks with different physical properties: music clips, verbal audios, and acoustic signals. For automatic music tagging, we use the MagnaTagATune (MTAT) benchmark [35] with 21k samples and 50 tags. Each clip contains multiple tags. We also use the Speech Commands dataset [36] for the keyword spotting task (106k samples, 35 classes, one-hot label). For acoustic signals, we use the sound event detection dataset from the DCASE 2017 challenge [37] (53k samples, 17 tags), where each audio has multiple tags.

We trained the Harmonic CNN [38] on the three benchmarks. Harmonic CNN consists of data-driven harmonic filters, stacked convolutional filters with BN. Audio datasets are usually smaller than the image datasets and are multi-labeled, posing another set of difficulty for the optimization problem. [38] has employed the mixed optimizer with Adam and SGD proposed by [39]. Instead of the mixture solution, we have searched the best hyperparameters for the Adam baseline and our AdamP on a validation set. The results are given in Table 5. AdamP shows better performance than the baselines, without having to adopt the complex mixture of Adam and SGD. The audio experiments signify the superiority of AdamP for training scale-invariant weights on a non-image domain.

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)

5.5 Retrieval

In the previous experiments, we have examined the scale invariances induced by the batch normalization (BN). Here, we consider another source of scale invariance, ℓ_2 normalization. Like BN, ℓ_2 normalization induces the scale invariance in the preceding weights. It is widely used in retrieval tasks for more efficient distance computations and better performances. We fine-tune the ImageNet-pretrained ResNet-50 network on CUB [40], Cars-196 [41], In-Shop Clothes [42] and Stanford Online Products (SOP) [43] benchmarks with the semi-hard mined triplet margin loss [44] and the ProxyAnchor loss [45]. We follow the official implementation of ProxyAnchor [45]. In Table 6, we observe that AdamP outperforms Adam over all four image retrieval datasets. The results support the superiority of AdamP for networks with ℓ_2 normalized embeddings.

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)	**58.2** (+0.3)	**69.5** (+0.2)	**59.9** (+0.2)	**86.9** (+0.2)	**62.8** (+0.0)	**87.4** (+2.2)	**62.6** (+0.6)	**78.0** (+1.5)

6 Conclusion

We have demonstrated that the 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. The problem is prevalent in the training of modern deep neural networks: momentum-based optimizers such as SGD and Adam are standard techniques and often a large proportion of weights are scale-invariant due to the omnipresence of batch normalization in widely-used models. We propose a simple and effective solution: projecting out the radial component from the optimization update at every iteration. We have demonstrated that the resulting SGDP and AdamP successfully suppress the weight norm growth and train a model at an unobstructed speed. Empirically, our optimizers have demonstrated their superiority over the baselines on various real-world data.

Acknowledgement

We thank Clova AI Research team for discussion and advice, especially Junsuk Choe for the internal review. Naver Smart Machine Learning (NSML) platform [46] has been used in the experiments.

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[4] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[5] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*. 2018.

[6] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *Advances in Neural Information Processing Systems*, 2018.

[7] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. In *International Conference on Learning Representations (ICLR)*, 2019.

[8] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1), 2018.

[9] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, pages 288–291. MIT press, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[13] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[14] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018.

[15] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations (ICLR)*, 2020.

[16] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, 2019.

[17] Timothy Dozat. Incorporating nesterov momentum into adam. In *International Conference on Learning Representations (ICLR) workshop*, 2016.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[19] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[20] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning (ICML)*, 2013.

[21] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*, 2019.

[24] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[28] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[29] Sanghyuk Chun, Seong Joon Oh, Sangdoo Yun, Dongyoon Han, Junsuk Choe, and Youngjoon Yoo. An empirical evaluation on robustness and uncertainty of regularization methods. *Uncertainty and Robustness in Deep Learning. International Conference on Machine Learning (ICML) Workshop*, 2019.

[30] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *The British Machine Vision Conference (BMVC)*, 2016.

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[32] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning*, 2020.

[33] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019.

[34] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *arXiv preprint arXiv:1907.07174*, 2019.

[35] Edith Law, Kris West, Michael I Mandel, Mert Bay, and J Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2009.

[36] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[37] Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Benjamin Elizalde, Ankit Shah, Emmanuel Vincent, Bhiksha Raj, and Tuomas Virtanen. DCASE 2017 challenge setup: Tasks, datasets and baseline system. In *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[38] Minz Won, Sanghyuk Chun, Oriol Nieto, and Xavier Serra. Data-driven harmonic filters for audio representation learning. In *International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[39] Minz Won, Sanghyuk Chun, and Xavier Serra. Toward interpretable music tagging with self-attention. *arXiv preprint arXiv:1906.04972*, 2019.

[40] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[41] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision workshops*, 2013.

[42] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[43] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[44] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[45] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[46] Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. NSML: Meet the MLaaS platform with a real-world case study. *arXiv preprint arXiv:1810.09957*, 2018.

[47] A. Krizhevsky. Learning multiple layers of features from tiny images. In *Tech Report*, 2009.

[48] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[49] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Appendix

This document provides additional materials for the main paper. Content includes the proofs (§A), detailed experimental setups (§B and §C), and the additional analysis on the learning rate scheduling and weight decay (§D).

A Proofs for the claims

We provide proofs for Lemma 2.2 and Corollary 2.3 in the main paper.

Lemma A.1 (Monotonic norm growth by the momentum)**.** *For a scale-invariant parameter $\boldsymbol{w}$ updated via equation 7, we have*

$$\|\boldsymbol{w}_{t+1}\|_2^2 = \|\boldsymbol{w}_t\|_2^2 + \eta^2\|\boldsymbol{p}_t\|_2^2 + 2\eta^2 \sum_{k=0}^{t-1} \beta^{t-k}\|\boldsymbol{p}_k\|_2^2. \tag{A.1}$$

Proof. From equation 7, we have

$$\|\boldsymbol{w}_{t+1}\|_2^2 = \|\boldsymbol{w}_t\|_2^2 + \eta^2\|\boldsymbol{p}_t\|_2^2 - 2\eta \boldsymbol{w}_t \cdot \boldsymbol{p}_t \tag{A.2}$$

It remains to prove that $\boldsymbol{w}_t \cdot \boldsymbol{p}_t = -\eta \sum_{k=0}^{t-1} \beta^{t-k}\|\boldsymbol{p}_k\|_2^2$. We prove by induction on $t \geq 0$.

First, when $t = 0$, we have $\boldsymbol{w}_0 \cdot \boldsymbol{p}_0 = \boldsymbol{w}_0 \cdot \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_0) = 0$ because of equation 4.

Now, assuming that $\boldsymbol{w}_\tau \cdot \boldsymbol{p}_\tau = -\eta \sum_{k=0}^{\tau-1} \beta^{\tau-k}\|\boldsymbol{p}_k\|_2^2$, we have

$$\boldsymbol{w}_{\tau+1} \cdot \boldsymbol{p}_{\tau+1} = \boldsymbol{w}_{\tau+1} \cdot (\beta \boldsymbol{p}_\tau + \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_{\tau+1})) = \beta \boldsymbol{w}_{\tau+1} \cdot \boldsymbol{p}_\tau = \beta(\boldsymbol{w}_\tau - \eta \boldsymbol{p}_\tau) \cdot \boldsymbol{p}_\tau \tag{A.3}$$

$$= -\beta\eta \sum_{k=0}^{\tau-1} \beta^{\tau-k}\|\boldsymbol{p}_k\|_2^2 - \beta\eta\|\boldsymbol{p}_\tau\|_2^2 = -\eta \sum_{k=0}^{\tau} \beta^{\tau-k+1}\|\boldsymbol{p}_k\|_2^2 \tag{A.4}$$

which completes the proof. □

Corollary A.2 (Asymptotic norm growth comparison)**.** *Let $\|\boldsymbol{w}_t^{\mathrm{GD}}\|_2$ and $\|\boldsymbol{w}_t^{\mathrm{GDM}}\|_2$ be the weight norms at step $t \geq 0$, following the vanilla gradient descent growth (Lemma 2.1) and momentum-based gradient descent growth (Lemma 2.2), respectively. We assume that the norms of the updates $\|\boldsymbol{p}_t\|_2$ for GD with and without momentum are identical for every $t \geq 0$. We further assume that the sum of the update norms is non-zero and bounded: $0 < \sum_{t\geq 0} \|\boldsymbol{p}_t\|_2^2 < \infty$. Then, the asymptotic ratio between the two norms is given by:*

$$\frac{\|\boldsymbol{w}_t^{\mathrm{GDM}}\|_2^2 - \|\boldsymbol{w}_0\|_2^2}{\|\boldsymbol{w}_t^{\mathrm{GD}}\|_2^2 - \|\boldsymbol{w}_0\|_2^2} \longrightarrow 1 + \frac{2\beta}{1-\beta} \quad \text{as} \quad t \to \infty. \tag{A.5}$$

Proof. From Lemma 2.1 and Lemma 2.2, we obtain

$$\|\boldsymbol{w}_t^{\mathrm{GD}}\|_2^2 - \|\boldsymbol{w}_0\|_2^2 = \eta^2 \sum_{k=0}^{t-1} \|\boldsymbol{p}_k\|_2^2 \tag{A.6}$$

$$\|\boldsymbol{w}_t^{\mathrm{GDM}}\|_2^2 - \|\boldsymbol{w}_0\|_2^2 = \eta^2 \sum_{k=0}^{t-1} \|\boldsymbol{p}_k\|_2^2 + 2\eta^2 \sum_{k=0}^{t-1} \left(\sum_{l=1}^{t-1-k} \beta^l \right) \|\boldsymbol{p}_k\|_2^2. \tag{A.7}$$

Thus, the corollary boils down to the claim that

$$F_t := \frac{\sum_{k=0}^{t} \left(\sum_{l=1}^{t-k} \beta^l \right) A_k}{\sum_{k=0}^{t} A_k} \longrightarrow \frac{\beta}{1-\beta} \quad \text{as} \quad t \to \infty \tag{A.8}$$

where $A_k := \|\boldsymbol{p}_k\|_2^2$.

Let $\epsilon > 0$. We will find a large-enough t that bounds F_t around $\frac{\beta}{1-\beta}$ by a constant multiple of ϵ.

We first let T be large enough such that

$$\sum_{k \geq T+1} A_k \leq \epsilon \tag{A.9}$$

which is possible because $\sum_{t \geq 0} A_t < \infty$. We then let T' be large enough such that

$$\frac{\beta}{1-\beta} - \sum_{l=1}^{T'} \beta^l \leq \frac{\epsilon}{T \max_k A_k} \tag{A.10}$$

which is possible due to the convergence of the geometric sum and the boundedness of A_k (because its infinite sum is bounded).

We then define $t = T + T'$ and break down the sums in F_t as follows:

$$F_t = \frac{\sum_{k=0}^{T} \left(\sum_{l=1}^{T+T'-k} \beta^l \right) A_k + \sum_{k=T+1}^{T+T'} \left(\sum_{l=1}^{T+T'-k} \beta^l \right) A_k}{\sum_{k=0}^{T} A_k + \sum_{k=T+1}^{T+T'} A_k} \tag{A.11}$$

$$= \frac{\sum_{k=0}^{T} \left(\frac{\beta}{1-\beta} + r_1(\epsilon) \right) A_k + r_2(\epsilon)}{\sum_{k=0}^{T} A_k + r_3(\epsilon)} \tag{A.12}$$

$$\leq \frac{\frac{\beta}{1-\beta} \sum_{k=0}^{T} A_k + T \max_k A_k r_1(\epsilon) + r_2(\epsilon)}{\sum_{k=0}^{T} A_k + r_3(\epsilon)} \tag{A.13}$$

where r_1, r_2, and r_3 are the residual terms that are bounded as follows:

$$|r_1(\epsilon)| \leq \frac{\epsilon}{T \max_k A_k} \tag{A.14}$$

by equation A.10 and

$$|r_2(\epsilon)| \leq \frac{(1-\beta)\epsilon}{\beta} \quad \text{and} \quad |r_3(\epsilon)| \leq \epsilon \tag{A.15}$$

by equation A.9.

It follows that

$$\left| F_t - \frac{\beta}{1-\beta} \right| \leq \left| \frac{-\frac{\beta}{1-\beta} r_3(\epsilon) + T \max_k A_k r_1(\epsilon) + r_2(\epsilon)}{\sum_{k=0}^{T} A_k + r_3(\epsilon)} \right| \tag{A.16}$$

$$\leq \frac{1}{\sum_{k=0}^{T} A_k} \left(\frac{\beta}{1-\beta} + \frac{1-\beta}{\beta} + 1 \right) \epsilon \tag{A.17}$$

$$\leq \frac{1}{M} \left(\frac{\beta}{1-\beta} + \frac{1-\beta}{\beta} + 1 \right) \epsilon \tag{A.18}$$

due to the triangular inequality and the positivity of r_3. $M > 0$ is a suitable constant independent of T. □

B Toy example details

We describe the details of the toy example in Figure 1. We solve the following optimization problem:

$$\min_{\boldsymbol{w}} \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2} \cdot \frac{\boldsymbol{w}^\star}{\|\boldsymbol{w}^\star\|_2} \tag{B.1}$$

where $\boldsymbol{w}$ and $\boldsymbol{w}^\star$ are 2-dimensional vectors. The problem is identical to the maximization of the cosine similarity between $\boldsymbol{w}$ and $\boldsymbol{w}^\star$. We set the $\boldsymbol{w}^\star$ to $(0, -1)$ and the initial $\boldsymbol{w}$ to $(0.001, 1)$.

Table C.1: **Dataset statistics.** Summary of the dataset specs used in the experiments.

Task	Dataset	#classes	#samples	Note
Image classification	ImageNet-1k [12]	1,000	$\approx$ 1.33M	
Object detection	MS-COCO [27]	80	$\approx$ 123k	
Robustness	CIFAR-10 [47]	10	$\approx$ 60k	
	Biased-MNIST [48, 32]	10	$\approx$ 60k	colors are injected to be biased
	9-Class ImageNet [12, 32]	9	$\approx$ 57k	a subset of ImageNet-1k [12]
	9-Class ImageNet-A [34, 32]	9	617	a subset of ImageNet-A [34]
Audio classification	MagnaTagATune [35]	50	$\approx$ 21k	mutl-labeled dataset
	Speech Commands [36]	35	$\approx$ 106k	
	DCASE 2017 task 4 [37]	17	$\approx$ 53k	mutl-labeled dataset
Image retrieval	CUB [40]	200	$\approx$ 12k	tr classes (100), te classes (100)
	Cars-196 [41]	196	$\approx$ 16k	tr classes (98), te classes (98)
	In-Shop Clothes [42]	7,982	$\approx$ 53k	tr classes (3,997), te classes (3985)
	SOP [43]	22,634	$\approx$ 120k	tr classes (11,318), te classes (11,316)

This toy example has two interesting properties. First, the normalization term makes the optimal $\boldsymbol{w}$ for the problem not unique: if $\boldsymbol{w}^\star$ is optimal, then $c\boldsymbol{w}^\star$ is optimal for any $c > 0$. In fact, the cost function is scale-invariant. Second, the cost function is not convex.

As demonstrated in Figure 1 and videos attached in our project page[3], the momentum gradient method fails to optimize equation B.1 because of the excessive norm increases. In particular, our simulation results show that a larger momentum induces a larger norm increase (maximum norm 2.93 when momentum is 0.9, and 27.87 when momentum is 0.99), as we shown in the main paper § 2.4. On the other hand, our method converges most quickly, among the compared methods, by taking advantage of the momentum-induced accelerated convergence, while avoiding the excessive norm increase.

C Experiments settings

We describe the experimental settings in full detail for reproducibility.

C.1 Common settings

All experiments are conducted based on PyTorch. SGDP and AdamP are implemented to handle channel-wise (*e.g.* batch normalization [1] and instance normalization [3]) and layer-wise normalization (layer normalization [2]). Based on the empirical measurement of the inner product between the weight vector and the corresponding gradient vector for scale-invariant parameters (they are supposed to be orthogonal), we set the δ in Algorithms 1 and 2 to 0.1. We use the decoupled weight decay [18] for SGDP and AdamP in order to separate the gradient due to the weight decay from the gradient due to the loss function. Please refer to the attached codes: `sgdp.py` and `adamp.py` for further details.

C.2 Image classification

Experiments involving ResNet [11] are conducted based on the standard settings : learning rate 0.1, weight decay 10^{-4}, batch-size 256, momentum 0.9 with Nesterov [20] for SGD and SGDP. For Adam and AdamP, we use the learning rate 0.001, weight decay 10^{-4}, batch-size 256, β_1 0.9, β_2 0.999, ϵ 10^{-8}. We use decoupled weight decay [18] for all experiments in image classification.

For training MobileNetV2 [22], we have additionally used label-smoothing and large batch size 1024, and have searched the best learning rates and weight decay values for each optimizer.

The training sessions are run for 100 epochs (ResNet18, ResNet50) or 150 epochs (MobileNetV2, ResNet50 + CutMix) with the cosine learning rate schedule [49] on a machine with four NVIDIA V100 GPUs.

[3] https://clovaai.github.io/AdamP/

C.3 Object detection

Object detection performances have been measured on the MS-COCO dataset [27] with two popular object detectors: CenterNet [24] and SSD [25]. We adopt the CenterNet with ResNet18 [11] backbone and the SSD with VGG16 BN [26] backbone as baseline detectors. CenterNet has been trained for 140 epochs with learning rate 2.5×10^{-4}, weight decay 10^{-5}, batch size 64, and the cosine learning rate schedule. SSD has been trained for 110 epochs with learning rate 10^{-4}, weight decay 10^{-5}, batch size 64, and the step learning rate schedule which decays learning rates by $1/10$ at 70% and 90% of training.

C.4 Robustness

C.4.1 Adversarial training

Adversarial robustness benchmark results have been reproduced using the unofficial PyTorch implementation of the adversarial training of Wide-ResNet [30][4] for the CIFAR-10 attack challenge[5]. Projected gradient descent (PGD) attack variants [31] have been used as the threat model for the all the experiments. We employed 10 inner PGD iterations and $\varepsilon = 80/255$ for the L_2 PGD attack and $\varepsilon = 4/255$ for the L_∞ PGD attack. In all the experiments, Wide-ResNet-34-10 have been trained with the PGD threat model. The models have been trained for 200 epochs with learning rate 0.01, weight decay 0.0002, batch size 128, and the step learning rate schedule which decays learning rates by $1/10$ at epochs 100 and 150. Table C.2 shows the detailed results.

Table C.2: **Adversarial training.** Standard and attacked accuracies of PGD-adversarially trained Wide-ResNet on CIFAR-10.

Attack Method	Optimizer	Standard Acc	Attacked Acc
ℓ_∞ ($\varepsilon = 4/255$)	Adam	80.12	56.58
	AdamP	**89.85** (+9.73)	**66.28** (+9.70)
ℓ_2 ($\varepsilon = 80/255$)	Adam	84.14	70.33
	AdamP	**93.46** (+9.32)	**83.59** (+13.26)

C.4.2 Robustness against real-world biases

We follow the two cross-bias generalization benchmarks proposed by [32]. We refer [32] for interested readers. For all experiments, the batch size is 256 and 128 for Biased MNIST and 9-Class ImageNet, respectively. For Biased MNIST, the initial learning rate is 0.001, decayed by factor 0.1 every 20 epochs. For 9-Class ImageNet, the learning rate is 0.001, decayed by cosine annealing. We train the fully convolutional network and ResNet18 for 80 and 120 epochs, respectively. The weight decay is 10^{-4} for all experiments.

C.5 Audio classification

Dataset. Three datasets with different physical properties are employed as the audio benchmarks. We illustrate the statistics in Table C.1. The **music tagging** is a multi-label classification task for the prediction of user-generated tags, *e.g.*, genres, moods, and instruments. We use a subset of MagnaTagATune (MTAT) dataset [35] which contains ≈21k audio clips and 50 tags. The average of tag-wise Area Under Receiver Operating Characteristic Curve (ROC-AUC) and Area Under Precision-Recall Curve (PR-AUC) are used as the evaluation metrics. **Keyword spotting** is a primitive speech recognition task where an audio clip containing a keyword is categorized among a list of limited vocabulary. We use the Speech Commands dataset [36] which contains ≈ 106k samples and 35 command classes such as "yes", "no", "left", "right". The accuracy metric is used for the evaluation. **Acoustic sound detection** is a multi-label classification task with non-music and non-verbal audios. We use the "large-scale weakly supervised sound event detection for smart cars" dataset used for the DCASE 2017 challenge [37]. It has ≈53k audio clips with 17 events such as "Car", "Fire truck", and "Train horn". For evaluation, we use the F1-score by setting the prediction threshold as 0.1.

[4] https://github.com/louis2889184/pytorch-adversarial-training
[5] https://github.com/MadryLab/cifar10_challenge

Training setting. We use the 16kHz sampling rate for the all experiments, and all hyperparameters, *e.g.*, the number of harmonics, trainable parameters, are set to the same as in [38]. The official implementation by [38][6] is used for all the experiments. We compare three different optimizers, Adam, AdamP (ours), and the complex mixture of Adam and SGD proposed by [39].

For the mixture of Adam and SGD, we adopt the same hyperparameters as in the previous papers [39, 38]. The mixed optimization algorithm first runs Adam for 60 epochs with learning rate 10^{-4}. After 60 epochs, the model with the best validation performance is selected as the initialization for the second phase. During the second phase, the model is trained using SGD for 140 epochs with the learning rate 10^{-4}, decayed by 1/10 at epochs 20 and 40. We use the weight decay 10^{-4} for the optimizers.

To show the effectiveness of our method, we have searched the best hyperparameters for the Adam optimizer on the MTAT validation dataset and have transferred them to AdamP experiments. As the result of our search, we set the weight decay as 0 and the initial learning rate as 0.0001 decayed by the cosine annealing scheduler. The number of training epochs are set to 100 for MTAT dataset and 30 for SpeechCommand and DCASE dataset. As a result, we observe that AdamP shows superior performances compared to the complex mixture, with a fewer number of training epochs ($200 \rightarrow 30$).

C.6 Retrieval

Dataset. We use four retrieval benchmark datasets. For the CUB [40] dataset which contains bird images with 200 classes, we use 100 classes for training and the rest for evaluation. For evaluation, we query every test image to the test dataset, and measure the recall@1 metric. The same protocol is applied to Cars-196 [41] (196 classes) and SOP [43] (22,634 classes) datasets. For InShop [42] experiments, we follow the official benchmark setting proposed by [42]. We summarize the dataset statistics in Table C.1

Training setting. For the all experiments, we use the same backbone network and the same training setting excepting the optimizer and the loss function. The official implementation by [45][7] is used for the all experiments.

We use the Pytorch official ImageNet-pretrained ResNet50 model as the initialization. During the training, we freeze the BN statistics as the ImageNet statistics (`eval` mode in PyTorch). We replace the global average pooling (GAP) layer of ResNet with the summation of GAP and global max pooling layer as in the implementation provided by [45]. Pooled features are linearly mapped to the 512 dimension embedding space and ℓ_2-normalized.

We set the initial learning rate 10^{-4}, decayed by the factor 0.5 for every 5 epochs. Every mini-batch contains 120 randomly chosen samples. For the better stability, we train only the last linear layer for the first 5 epochs, and update all the parameters for the remaining steps. The weight decay is set to 0.

D Analysis with learning rate schedule and weight decay

In Figure 3, we analyze the norm growth of scale-invariant parameters and the corresponding change in effective step-size. We provide extended results of this experiment by measuring norm growth and effective step-size for SGD, SGDP, Adam and AdamP under various weight decay values. The experiment is based on ResNet18 trained on the ImageNet dataset, and the network was trained for 100 epoch in the standard setting as in C.2. We have analyzed the impact of learning rate schedule and weight decay for the scale-invariant parameters. Figure D.1 and Figure D.2 show the results of SGD and SGDP under the step-decay and cosine-annealing learning rate schedules, respectively. The same results for Adam and AdamP are shown in Figures D.3 and Figure D.4. We have used the optimal weight decay value of the baseline as the reference point and changed the weight decay values. We write the weight decay in each experiment in relative values with respect to the corresponding optimal values.

In all considered settings, SGDP and AdamP effectively prevent the norm growth, which prevents the rapid decrease of the effective step sizes. SGDP and AdamP shows better performances than the

[6] https://github.com/minzwon/data-driven-harmonic-filters
[7] https://github.com/tjddus9597/Proxy-Anchor-CVPR2020

baselines. Another way to prevent the norm growth is to control the weight decay. However, this way of norm adjustment is sensitive to the weight decay value and results in poor performances as soon as non-optimal weight decay values are used. Figure D.1 and D.3 shows that the learning curves are generally sensitive to the weight decay values even showing abnormalities such as the gradual increase of the effective step sizes. On the other hand, SGDP and AdamP prevent rapid norm growth without weight decay, leading to smooth effective step size reduction. SGDP and AdamP are not sensitive to the weight decay values.

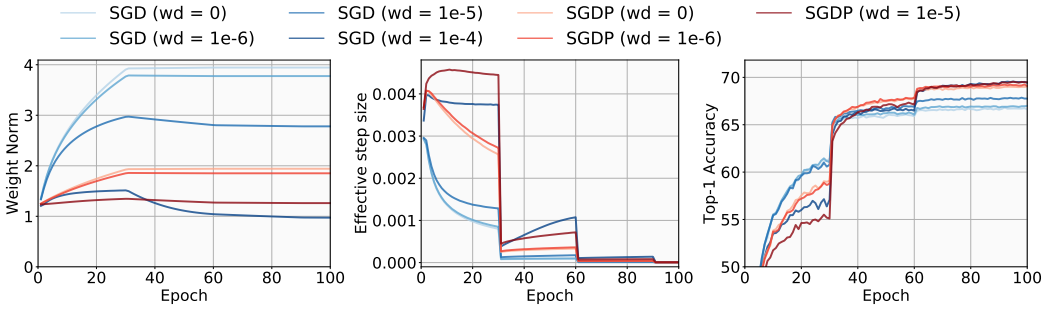


Figure D.1: Norm value analysis: SGD + step learning rate decay.

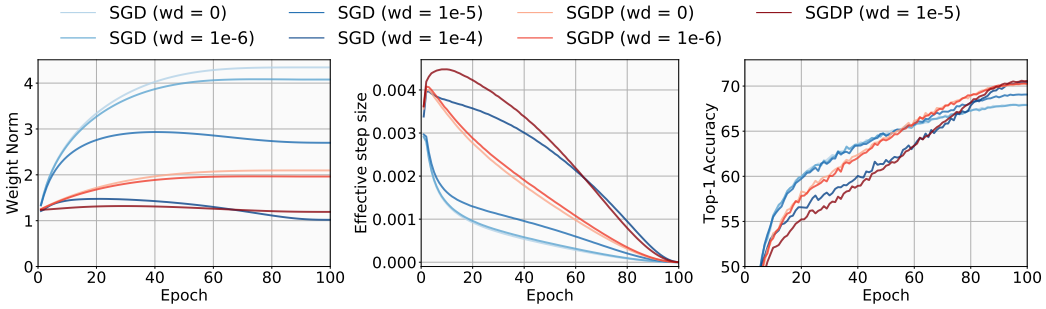


Figure D.2: Norm value analysis: SGD + cosine learning rate decay. SGD (wd=10^{-4}) and SGDP (wd=10^{-5}) are the same setting as the reported numbers in Table 1

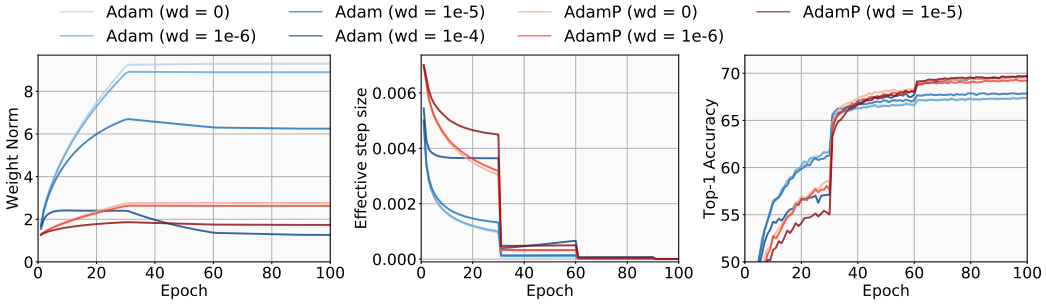


Figure D.3: Norm value analysis: Adam + step learning rate decay.

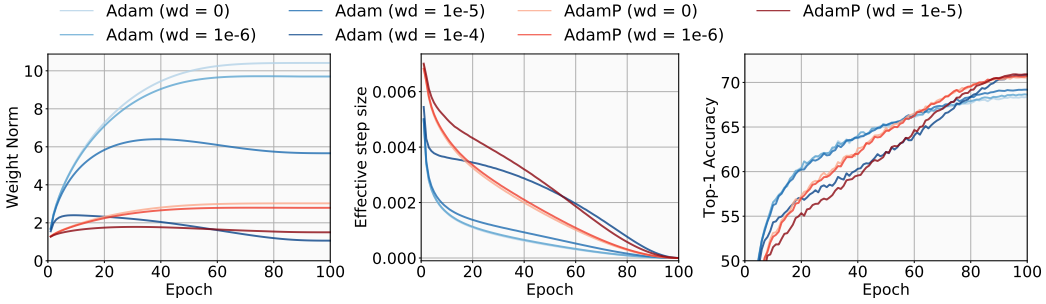


Figure D.4: Norm value analysis: Adam + cosine learning rate decay. Adam (wd=10^{-4}) and AdamP (wd=10^{-6}) are the same setting as the reported numbers in Table 1