ADAMP: SLOWING DOWN THE SLOWDOWN FOR MOMENTUM OPTIMIZERS ON SCALE-INVARIANT WEIGHTS

Byeongho Heo, [1] Sanghyuk Chun, [1] Seong Joon Oh, [1] Dongyoon Han, [1] Sangdoo Yun, [1] Gyuwan Kim, [1] Youngjung Uh, [2] and Jung-Woo Ha [1]

1) *Naver AI Lab*

2) *Applied Information Engineering, Yonsei University (Works done at Naver AI Lab)*

Byeonho Heo and Sanghyuk Chun contribute equally.

ABSTRACT

Normalization techniques, such as batch normalization (BN), are a boon for modern deep learning. They let weights converge more quickly with often better generalization performances. It has been argued that the normalization-induced scale invariance among the weights provides an advantageous ground for gradient descent (GD) optimizers: the effective step sizes are automatically reduced over time, stabilizing the overall training procedure. It is often overlooked, however, that the additional introduction of *momentum* in GD optimizers results in a far more rapid reduction in effective step sizes for scale-invariant weights, a phenomenon that has not yet been studied and may have caused unwanted side effects in the current practice. This is a crucial issue because arguably the vast majority of modern deep neural networks consist of (1) momentum-based GD (*e.g.* SGD or Adam) and (2) scale-invariant parameters (*e.g.* more than 90% of the weights in ResNet are scale-invariant due to BN). In this paper, we verify that the widely-adopted combination of the two ingredients lead to the premature decay of effective step sizes and sub-optimal model performances. We propose a simple and effective remedy, SGDP and AdamP: get rid of the radial component, or the norm-increasing direction, at each optimizer step. Because of the scale invariance, this modification only alters the effective step sizes without changing the effective update directions, thus enjoying the original convergence properties of GD optimizers. Given the ubiquity of momentum GD and scale invariance in machine learning, we have evaluated our methods against the baselines on 13 benchmarks. They range from vision tasks like classification (*e.g.* ImageNet), retrieval (*e.g.* CUB and SOP), and detection (*e.g.* COCO) to language modelling (*e.g.* WikiText) and audio classification (*e.g.* DCASE) tasks. We verify that our solution brings about uniform gains in performances in those benchmarks. The full study is presented at International Conference on Learning Representations (ICLR) [1].

PRIMARY HEADING

Normalization techniques, such as batch normalization (BN) [2], have become standard tools for training deep neural network models. Originally proposed to reduce the internal covariate shift [2], normalization methods have proven to encourage several desirable properties in deep neural networks, such as better generalization and the scale invariance [3]. Prior studies have observed that the normalization-induced scale invariance of weights stabilizes the convergence for the neural network training [3,4]. 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 become 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 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 $\frac{1}{\|\boldsymbol{w}_t\|_2}$ [3]. Since $\|\boldsymbol{w}_t\|_2$ increases during training [4], 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 [4].

In this work, we show that the widely used *momentum*-based gradient descent optimizers decreases the effective step size $\Delta\widehat{\boldsymbol{w}}_t$ even more rapidly than the momentum-less counterparts considered in [4]. This leads to a slower convergence for $\widehat{\boldsymbol{w}}_t$ and potentially sub-optimal model performances. This phenomenon is not confined to the toy setup, for example, 95.5% and 91.8% of the parameters of the widely-used ResNet18 and ResNet50 [5] are scale-invariant due to BN.

We propose a simple solution to slow down the decay of effective step sizes while maintaining the step directions of the original optimizer in the effective space. 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. Because of the scale invariance, the procedure does not alter the update direction in the effective space; it only changes the effective step sizes. We apply this technique on SGD and Adam [6] (SGDP and AdamP, respectively) and verify the resulting performance boosts over a diverse set of practical machine learning tasks.

REFERENCES

1. B. Heo, S. Chun, S. J. Oh, D. Han, S. Yun, G. Kim, Y. Uh, and J.-W. Ha, "Adamp: Slowing down the slowdown for momentum optimizers on scale-invariant weights," in *International Conference on Learning Representations*, 2021.
2. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015.
3. E. Hoffer, R. Banner, I. Golan, and D. Soudry, "Norm matters: efficient and accurate normalization schemes in deep networks," in *Advances in Neural Information Processing Systems*, 2018.
4. S. Arora, Z. Li, and K. Lyu, "Theoretical analysis of auto rate-tuning by batch normalization," in *International Conference on Learning Representations*, 2019.
5. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
6. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.