Win: Weight-Decay-Integrated Nesterov Acceleration for Adaptive Gradient Algorithms

Pan Zhou[1] **Xingyu Xie**[1,2] **Shuicheng Yan**[1]
[1]Sea AI Lab [2]Peking University
{zhoupan,xyxie,yansc}@sea.com xyxie@pku.cn

Abstract

Training deep networks on increasingly large-scale datasets is computationally challenging. In this work, we explore the problem of "*how to accelerate the convergence of adaptive gradient algorithms in a general manner*", and aim at providing practical insights to boost the training efficiency. To this end, we propose an effective Weight-decay-Integrated Nesterov acceleration (Win) for adaptive algorithms to enhance their convergence speed. Taking AdamW and Adam as examples, we minimize a dynamical loss per iteration which combines the vanilla training loss and a dynamic regularizer inspired by proximal point method (PPM) to improve the convexity of the problem. Then we respectively use the first- and second-order Taylor approximations of vanilla loss to update the variable twice while fixing the above dynamic regularization brought by PPM. In this way, we arrive at our Win acceleration (like Nesterov acceleration) for AdamW and Adam that uses a conservative step and a reckless step to update twice and then linearly combines these two updates for acceleration. Next, we extend this Win acceleration to LAMB and SGD. Our transparent acceleration derivation could provide insights for other accelerated methods and their integration into adaptive algorithms. Besides, we prove the convergence of Win-accelerated adaptive algorithms and justify their convergence superiority over their non-accelerated counterparts by taking AdamW and Adam as examples. Experimental results testify the faster convergence speed and superior performance of our Win-accelerated AdamW, Adam, LAMB and SGD over their vanilla counterparts on vision classification tasks and language modeling tasks with CNN and Transformer backbones.

1 Introduction

Deep neural networks (DNNs) are effective to model realistic data and have been successfully applied to various applications, *e.g.* image classification [1, 2] and speech recognition [3, 4]. Typically, their training models can be formulated as the following nonconvex optimization problem:

$$\min_{\boldsymbol{z}\in\mathbb{R}^d} F(\boldsymbol{z}) := \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}}[f(\boldsymbol{z},\boldsymbol{\zeta})] + \frac{\lambda}{2}\|\boldsymbol{z}\|_2^2, \tag{1}$$

where $\boldsymbol{z}$ is the model parameters; sample $\boldsymbol{\zeta}$ is drawn from a data distribution $\mathcal{D}$; the loss f is differentiable; λ is a constant. To solve problem (1), SGD [5] uses its compositional structure to efficiently estimate gradient via minibatch data, and has become a dominant algorithm to train DNNs. However, on sparse data or ill-conditioned problems, SGD suffers from slow convergence speed [6], as it scales the gradient uniformly in all parameter coordinate and ignores the data or problem properties on each coordinate. To resolve this issue, recent work has proposed a variety of adaptive methods, *e.g.* Adam [6] and AdamW [7], that scale each gradient coordinate according to the current geometry curvature of the loss $F(\boldsymbol{z})$. This coordinate-wise scaling greatly accelerates

Has it Trained Yet? Workshop at the Conference on Neural Information Processing Systems (NeurIPS 2022).

the optimization convergence and helps them, *e.g.* Adam and AdamW, become default optimizers to train DNNs.

Unfortunately, along with the increasing scale of both datasets and models, efficient DNN training even with SGD or adaptive algorithms has become very challenging. In this work, we are particularly interested in the problem of "*how to accelerate the convergence of adaptive algorithms in a general manner*" because of their dominant popularity across many DNNs. Heavy ball acceleration [8] and Nesterov acceleration [9] are widely used in SGD but are rarely studied in adaptive algorithms.

Contributions: In this work, based on a recent Nesterov-type acceleration formulation [10] and proximal point method (PPM) [11], we propose a new *Weight-decay-Integrated Nesterov acceleration* (Win) to accelerate adaptive algorithms. By taking AdamW and Adam as examples, we follow PPM spirit and minimize a dynamically regularized loss which combines vanillas loss and a dynamical regularization, and independently approximate the vanilla loss by its first- and second-order Taylor expansions to update the variable twice while fixing the above dynamic regularization. As a result, we achieve at our Win acceleration, a Nesterov-alike acceleration, for AdamW and Adam that uses a conservative step and a reckless step to update twice and then linearly combines these two updates for acceleration. Then we extend Win acceleration to LAMB [12] and SGD. This transparent acceleration derivation may motivate other accelerations and provide examples to introduce other accelerations into adaptive algorithms. Moreover, we analyze the convergence of Win-accelerated adaptive algorithms to justify their convergence superiority by using AdamW & Adam as examples.

Finally, experimental results on both vision classification tasks and language modeling tasks show that our Win-accelerated algorithms, i.e. accelerated AdamW, Adam, LAMB and SGD, can accelerate the convergence speed and also improve the performance of their corresponding non-accelerated counterparts by a remarkable margin on both CNN and transformer architectures.

2 Weight-decay-Integrated Nesterov Acceleration

In deterministic optimization, one widely used optimization-stabilizing and acceleration approach is proximal point method (PPM) [11, 13]. At the k-th iteration, PPM optimizes an ℓ_2-regularized loss $F(\boldsymbol{z})+\frac{1}{2\eta_k}\|\boldsymbol{z}-\boldsymbol{z}_{k-1}\|_2^2$ instead of the vanilla loss $F(\boldsymbol{z})$. This change enhances the convexity of the problem, accelerating and also stabilizing optimization [14, 15]. To make the ℓ_2-regularized problem solvable iteratively, PPM approximates $F(\boldsymbol{z})$ by its first- or second-order Taylor expansion to get a close-form solution. At below, we borrow the idea in PPM to induce a *Weight-decay-Integrated Nesterov acceleration* (Win) for adaptive algorithms by using AdamW and Adam as examples.

Win-Accelerated AdamW and Adam. To begin with, following most adaptive gradient algorithms, *e.g.* Adam and AdamW, we estimate the first- and second-order moments $\boldsymbol{m}_k$ and $\boldsymbol{v}_k$ of gradient as

$$\boldsymbol{g}_k = \frac{1}{b}\sum\nolimits_{i=1}^{b} \nabla f(\boldsymbol{z}_k;\boldsymbol{\zeta}_i), \quad \boldsymbol{m}_k = (1-\beta_1)\boldsymbol{m}_{k-1}+\beta_1\boldsymbol{g}_k, \quad \boldsymbol{v}_k = (1-\beta_2)\boldsymbol{v}_{k-1}+\beta_2\boldsymbol{g}_k^2, \quad (2)$$

where $\boldsymbol{m}_0 = \boldsymbol{g}_0$, $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$, $\beta_1 \in [0,1]$ and $\beta_2 \in [0,1]$. For brevity, with a small scaler $\nu > 0$, we define

$$\boldsymbol{s}_k = \sqrt{\boldsymbol{v}_k+\nu}, \qquad \boldsymbol{u}_k = \boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k+\nu}. \quad (3)$$

Then following PPM spirit, at the k-th iteration, we minimize a regularized loss $F(\boldsymbol{x})+\frac{1}{2\eta}\|\boldsymbol{x}-\boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2$. Here we use the regularizer $\|\boldsymbol{x}-\boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2$ instead of the ℓ_2-regularization $\|\boldsymbol{x}-\boldsymbol{x}_k\|_2^2$, since 1) this new regularization allows us to handle adaptive algorithms as shown below Eqn. (4), and 2) it also helps increase the problem convexity to speed up the convergence. To make problem solvable iteratively, we approximate $F(\boldsymbol{z})$ by its first-order Taylor expansion at the point $\boldsymbol{z}_k$ and update $\boldsymbol{x}_{k+1}$ as

$$\boldsymbol{x}_{k+1} = \operatorname{argmin}_{\boldsymbol{x}} F(\boldsymbol{z}_k)+\langle \boldsymbol{m}_k, \boldsymbol{x}-\boldsymbol{z}_k\rangle + \frac{1}{2\eta_k}\|\boldsymbol{x}-\boldsymbol{x}_t\|_{\boldsymbol{s}_k}^2 + \frac{\lambda}{2}\|\boldsymbol{x}\|_{\boldsymbol{s}_k}^2 = \frac{1}{1+\lambda\eta_k}(\boldsymbol{x}_k-\eta_k\boldsymbol{u}_k), \quad (4)$$

where $\|\boldsymbol{x}\|_{\boldsymbol{s}_k} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{s}_k * \boldsymbol{x}\rangle}$ with element-wise product $*$, $\boldsymbol{m}_k$ is used to approximate the full gradient $\nabla F(\boldsymbol{z}_k)$ for Taylor expansion. We add a small regularization $\frac{\lambda}{2}\|\boldsymbol{x}\|_{\boldsymbol{s}_k}^2$, as 1) it can improve the generalization in practice [7, 16]; 2) it allows us to derive Adam ($\lambda = 0$) and AdamW ($\lambda > 0$). If $\lambda = 0$, the updating (4) becomes the exact Adam algorithm. If $\lambda > 0$, the updating (4) can approximate the updating rule $\boldsymbol{x}_{k+1} = (1-\lambda\eta_k)\boldsymbol{x}_k - \eta_k\boldsymbol{u}_k$ of AdamW. This is because

Algorithm 1: Win-Accelerated AdamW, Adam and LAMB

Input: initialization $\boldsymbol{x}_0 = \boldsymbol{z}_0 = \mathbf{0}$, step size $\{(\eta_k, \bar{\eta}_k)\}_{k=0}^{T}$, moment parameters $\{\beta_1, \beta_2\}$.

1 **while** $k < T$ **do**

2 $\quad \boldsymbol{g}_k = \frac{1}{b}\sum_{i=1}^{b} \nabla f(\boldsymbol{z}_k; \boldsymbol{\zeta}_i)$

3 $\quad \boldsymbol{m}_k = (1-\beta_1)\boldsymbol{m}_{k-1} + \beta_1 \boldsymbol{g}_k$ /* $\boldsymbol{m}_0 = \boldsymbol{g}_0$ */

4 $\quad \boldsymbol{v}_k = (1-\beta_2)\boldsymbol{v}_{k-1} + \beta_2 \boldsymbol{g}_k^2$ /* $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$ */

5 $\quad \boldsymbol{u}_k = \frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}}$ for AdamW and Adam, $\boldsymbol{u}_k = \frac{\|\boldsymbol{x}_k\|_2}{\|\boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k+\nu}\|_2}\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}}$ for LAMB

6 $\quad \boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta}\left(\boldsymbol{x}_k - \eta_k \boldsymbol{u}_k\right)$

7 $\quad \boldsymbol{z}_{k+1} = \bar{\eta}_k \tau_k \boldsymbol{x}_{k+1} + \eta_k \tau_k \left(\boldsymbol{z}_k - \bar{\eta}_k \boldsymbol{u}_k\right)$ with $\tau_k = \frac{1}{\eta_k + \bar{\eta}_k + \lambda\eta_k\bar{\eta}_k}$

8 **end while**

consider $\lambda\eta_k$ is small in practice, we approximate $(1+\lambda\eta_k)^{-1} = 1 - \lambda\eta_k + \mathcal{O}(\lambda^2\eta_k^2)$ and thus $\frac{1}{1+\lambda\eta_k}(\boldsymbol{x}_k - \eta_k\boldsymbol{u}_k) = [1-\lambda\eta_k + \mathcal{O}(\lambda^2\eta_k^2)]\boldsymbol{x}_k - [\eta_k - \mathcal{O}(\lambda\eta_k^2) + \mathcal{O}(\lambda^3\eta_k^3)]\boldsymbol{u}_k$ which becomes AdamW by ignoring $\mathcal{O}(\eta_k^2)$ and $\mathcal{O}(\eta_k^3)$. This is one reason that we adopt the regularizer $\|\boldsymbol{x}-\boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2$ in (4) instead of the ℓ_2-regularization in PPM, as we can flexibly derive Adam and AdamW.

Similarly, we minimize a regularized loss $F(\boldsymbol{z}) + \frac{1}{2\eta_k}\|\boldsymbol{z} - \boldsymbol{x}_{t+1}\|_{\boldsymbol{s}_k}^2$, and further approximate $F(\boldsymbol{z})$ by its second-order approximation $F(\boldsymbol{z}_k) + \langle \boldsymbol{m}_k, \boldsymbol{z} - \boldsymbol{z}_k\rangle + \frac{1}{2\bar{\eta}_k}\|\boldsymbol{z} - \boldsymbol{z}_k\|_{\boldsymbol{s}_k}^2$:

$$\begin{aligned}\boldsymbol{z}_{k+1} &= \operatorname{argmin}_{\boldsymbol{z}} F(\boldsymbol{z}_k) + \langle \boldsymbol{m}_k, \boldsymbol{z} - \boldsymbol{z}_k\rangle + \frac{1}{2\bar{\eta}_k}\|\boldsymbol{z} - \boldsymbol{z}_k\|_{\boldsymbol{s}_k}^2 + \frac{1}{2\eta_k}\|\boldsymbol{z} - \boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_k}^2 + \frac{\lambda}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2 \\ &= \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\big(\boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{u}_k\big),\end{aligned} \tag{5}$$

where $\tau_k = \frac{1}{\eta_k + \bar{\eta}_k + \lambda\eta_k\bar{\eta}_k}$, $\boldsymbol{m}_k$ can approximate $\nabla F(\boldsymbol{x}_k)$ as shown in Theorem 1 in Appendix B.

For more clear, we introduce a virtual sequence $\boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{u}_k$ in Win, and rewrite (5) as

$$\boldsymbol{x}_{k+1} = (1+\lambda\eta_k)^{-1}\left(\boldsymbol{x}_k - \eta_k\boldsymbol{u}_k\right), \quad \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{u}_k, \quad \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\boldsymbol{y}_{k+1}. \tag{6}$$

See detailed steps in Algorithm 1. Interestingly, Win acceleration is similar to Nesterov-type acceleration, since they both use a conservative step η_k and a reckless step $\bar{\eta}_k$ to update $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_{k+1}$ respectively, and then linearly combine $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_{k+1}$ to obtain $\boldsymbol{z}_{k+1}$.

Our Win-acceleration is quit simple and efficient, as our accelerated AdamW/Adam only adds an extra simple algorithmic step, i.e. the 7th step in Algorithm 1, on vanilla AdamW/Adam. Moreover, for the only extra hyper-parameter, the reckless step $\bar{\eta}_k$, in Algorithm 1 over AdamW/Adam, we always set it as $\bar{\eta}_k = 2\eta_k$, which works well in our all experiments.

Extension to LAMB and SGD. Here we generalize Win acceleration to LAMB [12] and SGD [5]. For LAMB, it scales the update $\boldsymbol{u}_k$ of AdamW in Eqn. (3) so that $\boldsymbol{u}_k$ is at the same magnitude of the network weight $\boldsymbol{x}_k$. That is, it changes the update rule $\boldsymbol{x}_{k+1} = (1-\lambda\eta_k)\boldsymbol{x}_k - \eta_k\boldsymbol{m}_k/\boldsymbol{s}_k$ in AdamW to $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k\frac{\|\boldsymbol{x}_k\|_2}{\|\boldsymbol{r}_k+\lambda\boldsymbol{x}_k\|_2}(\boldsymbol{r}_k + \lambda\boldsymbol{x}_k)$ where $\boldsymbol{r}_k = \boldsymbol{m}_k/\boldsymbol{s}_k$. This modification is to avoid too large or small update, improving optimization efficiency. To extend Win acceleration to LAMB, we inherit this scaling spirit, and scale the update $\boldsymbol{u}_k$ in (3) to the following one:

$$\boldsymbol{u}_k = (\|\boldsymbol{x}_k\|_2/\|\boldsymbol{m}_k/\boldsymbol{s}_k\|_2)\cdot(\boldsymbol{m}_k/\boldsymbol{s}_k). \tag{7}$$

Next, we can follow Eqn. (4) and (5) to update, and summarize detailed steps in Algorithm 1.

For SGD, applying Win acceleration to it is quite direct. Specifically, the only algorithmic difference between SGD and AdamW on the ℓ_2-regularized problems is that SGD has no second-order moment $\boldsymbol{v}_k$ while AdamW has. So we can borrow the acceleration framework of AdamW to accelerate SGD by setting $\boldsymbol{s}_k = \mathbf{1} \in \mathbb{R}^d$ in Eqn. (3), (4) and (5), and obtain WIN-accelerated SGD:

$$\boldsymbol{m}_k = \beta_1\boldsymbol{m}_{k-1} + \beta_1'\boldsymbol{g}_k, \ \boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}(\boldsymbol{x}_k - \eta_k\boldsymbol{m}_k), \ \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\big(\boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{m}_k\big), \tag{8}$$

where $\beta_1' \in [0,1]$ is dampening parameter. Here we slightly modify the moment $\boldsymbol{m}_k$ to accord with the one used in Nesterov-accelerated SGD (*e.g.* SGD-M in Pytorch).

Convergence Analysis. Theorem 1 in Appendix B analyzes the convergence of Win-accelerated adaptive algorithms to justify their convergence superiority by using AdamW & Adam as examples.

Table 1: ImageNet top-1 accuracy (%) of ResNet50&101 whose official optimizer is LAMB due to the stronger data augmentation for better performance. $*$ is reported in [17].

	ResNet50				ResNet101			
Epoch	100	200	300	avg.	100	200	300	avg.
SAM	77.3	78.7	79.4	78.5	79.5	81.1	81.6	80.7
SGD-H	75.3	76.9	77.2	76.5	77.7	78.6	78.8	78.4
SGD-M	77.0	78.6	79.3	78.3	79.3	81.0	81.4	80.6
SGD-Win	78.0	79.2	79.7	$79.0_{+0.7}$	80.1	81.2	81.6	$81.0_{+0.4}$
Adam	76.9	78.4	78.8	78.1	78.4	80.2	80.6	79.7
Adam-Win	77.8	78.8	79.3	$78.7_{+0.6}$	79.2	80.6	81.0	$80.3_{+0.6}$
AdamW	77.0	78.9	79.3	78.4	78.9	79.9	80.4	79.7
AdamW-Win	78.0	79.3	79.9	$79.1_{+0.7}$	80.2	81.1	81.3	$80.9_{+1.2}$
LAMB	77.0	79.2	79.8^*	78.7	79.4	81.1	81.3^*	80.6
LAMB-Win	78.4	79.7	80.1	$79.4_{+0.7}$	80.6	81.5	81.7	$81.3_{+0.7}$

Table 2: ImageNet top-1 accuracy (%) of ViT and PoolFormer whose default optimizers are both AdamW. $*$ and $\diamond$ are respectively reported in [16] and [18].

	ViT-S			ViT-B			PoolFormer-S12		
Epoch	150	300	avg.	150	300	avg.	150	300	avg.
SGD-M	77.4	79.4	78.4	79.6	80.0	79.8	69.7	74.3	72.0
SGD-Win	78.1	80.1	$79.1_{+0.7}$	80.4	80.8	$80.6_{+0.8}$	71.1	74.5	$72.8_{+0.8}$
Adam	77.3	79.3	78.3	79.0	79.7	79.4	74.3	76.3	75.3
Adam-Win	78.6	80.2	$79.4_{+1.1}$	80	80.5	$80.3_{+0.9}$	75.6	77.1	$76.4_{+1.1}$
AdamW	78.3	79.8^*	79.1	79.5	81.8^*	80.7	75.2	77.1^*	76.2
AdamW-Win	79.3	80.8	$80.1_{+1.0}$	81.0	82.2	$81.6_{+0.9}$	76.7	77.6	$77.2_{+1.0}$
LAMB	78.0	79.6	78.8	80.3	80.8	80.6	75.4	77.4	76.4
LAMB-Win	79.3	80.6	$80.0_{+1.2}$	81.0	81.4	$81.2_{+0.6}$	76.7	78.0	$77.4_{+1.0}$

3 Experiments

For clarity, we call our accelerated algorithm "X-Win", where "X" denotes vanilla optimizers. In all experiments, our accelerated algorithms, *e.g.* AdamW-Win, always use the default hyper-parameters of vanilla optimizers, *e.g.* moment parameters β_1 and β_2 in AdamW; and set $\bar{\eta}_k = 2\eta_k$.

Results on ResNets and ViTs. Table 1 reports accuracy of ResNets under the setting in [17], and Table 2 gives the performance of ViT [2] and PoolFormer [18]. Our accelerated algorithms always outperform their corresponding non-accelerated version. On ResNet, LAMB-Win achieves remarkable improvement over the official optimizer LAMB for this setting; SGD-Win also surpasses heavy-ball accelerated SGD (SGD-H) and Nesterov accelerated SGD (SGD-M). On ViTs, our accelerated algorithms consistently outperform the corresponding non-accelerated counterparts. Fig. 1 shows the faster faster convergence behaviors of our accelerated algorithms over non-accelerated counterparts which could benefit their better performance under the same computational cost.

Results on Transformer-XL. Table 3 shows that under different training steps on WikiText-103 dataset, our accelerated Adam-Win always achieves lower test PPL than the official Adam optimizer of Transformer-XL-base, and improves 1.5 average test PPL over Adam.

Table 3: Test PPL of Transformer-XL-B. * is officially reported.

Transformer-XL	Training Steps			
	50k	100k	200k	avg.
Adam	28.5	25.5	24.2^*	26.7
Adam-Win	26.7	25.0	24.0	$25.2_{+1.5}$

4 Conclusion

In this work, we adopt proximal point method to derive a weight-decay-integrated Nesterov acceleration for AdamW and Adam, and extend it to LAMB and SGD. Moreover, we prove the convergence of our accelerated algorithms, i.e. accelerated AdamW, Adam and SGD, and observe the superiority of the accelerated Adam-type algorithm over the vanilla ones in terms of stochastic gradient complexity. Finally, experimental results validate the advantages of our accelerated algorithms.

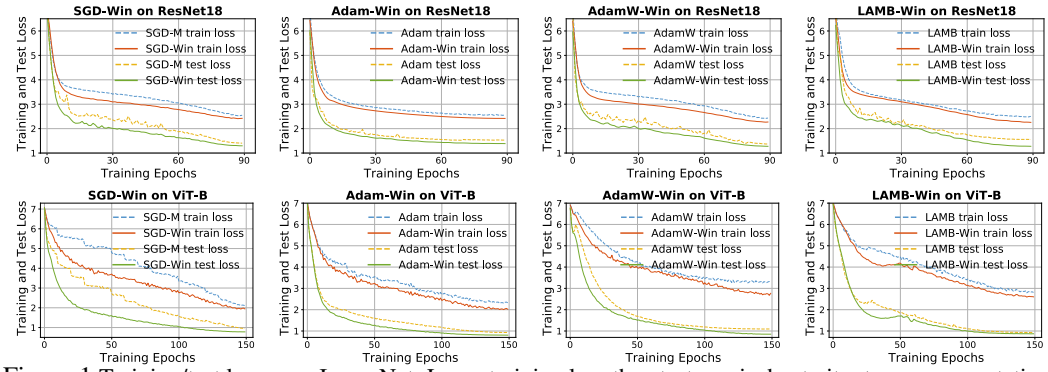

Figure 1:Training/test losses on ImageNet. Lager training loss than test one is due to its strong augmentation.

References

[1] 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*, pages 770–778, 2016.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[3] T. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for LVCSR. In *ICASSP*, pages 8614–8618. IEEE, 2013.

[4] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 2014.

[5] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Int'l Conf. Learning Representations*, 2014.

[7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[8] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[9] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[10] Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

[11] Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

[12] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2019.

[13] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[14] Junhyung Lyle Kim, Panos Toulis, and Anastasios Kyrillidis. Convergence and stability of the stochastic proximal point algorithm with momentum. In *Learning for Dynamics and Control Conference*, pages 1034–1047. PMLR, 2022.

[15] Pan Zhou, Hanshu Yan, Xiaotong Yuan, Jiashi Feng, and Shuicheng Yan. Towards understanding why lookahead generalizes better than sgd and beyond. *Advances in Neural Information Processing Systems*, 34:27290–27304, 2021.

[16] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[17] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

[18] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. *arXiv preprint arXiv:2111.11418*, 2021.

[19] Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3267–3275, 2021.

[20] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in Neural Information Processing Systems*, 33:18795–18806, 2020.

[21] 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*, 2019.

[22] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[23] Jürgen Schmidhuber, Sepp Hochreiter, et al. Long short-term memory. *Neural Comput*, 9(8):1735–1780, 1997.

[24] Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, 273, 1994.

[25] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[26] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

[27] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[28] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[29] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

[30] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

[31] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing both cnns and vits. *Axriv*, 2022.

[32] Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

[33] Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Ayush Sekhari, and Karthik Sridharan. Second-order information in non-convex stochastic optimization: Power and limitations. In *Conference on Learning Theory*, pages 242–299. PMLR, 2020.

[34] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2018.

[35] Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

[36] Zhishuai Guo, Yi Xu, Wotao Yin, Rong Jin, and Tianbao Yang. A novel convergence analysis for algorithms of the adam family and beyond. *arXiv e-prints*, pages arXiv–2104, 2021.

[37] Tieleman Tijmen and Hinton Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a run- ning average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.