Adan: Adaptive Nesterov Momentum Algorithm for Faster Optimizing Deep Models

Xingyu Xie[1,2*] **Pan Zhou**[1*] **Huan Li**[3] **Zhouchen Lin**[2⋆] **Shuicheng Yan**[1⋆]
[1]Sea AI Lab [2]Peking University [3]Nankai University
{xyxie,zhoupan,yansc}@sea.com {xyxie,zlin}@pku.cn lihuanss@nankai.edu.cn

Abstract

Adaptive gradient algorithms [1–4] combine the moving average idea with heavy ball acceleration to estimate accurate first- and second-order moments of gradient for accelerating convergence. But Nesterov acceleration which converges faster than heavy ball acceleration in theory [5] and also in many empirical cases [6] is much less investigated under the adaptive gradient setting. In this work, we propose the ADAptive Nesterov momentum algorithm (Adan) to speed up the training of deep neural networks. Adan first reformulates the vanilla Nesterov acceleration to develop a new Nesterov momentum estimation (NME) method that avoids the extra computation and memory overhead of computing gradient at the extrapolation point. Then Adan adopts NME to estimate the first- and second-order gradient moments in adaptive gradient algorithms for convergence acceleration. Besides, we prove that Adan finds an ϵ-approximate stationary point within $\mathcal{O}\big(\epsilon^{-4}\big)$ stochastic gradient complexity on the non-convex stochastic problems, matching the best-known lower bound. Extensive experimental results show that Adan surpasses the corresponding SoTA optimizers for vision, language, and RL tasks and sets new SoTAs for many popular networks and frameworks, *e.g.* ResNet [7], ConvNext [8], ViT [9], Swin [10], MAE [11], Transformer-XL [12] and BERT [13]. More surprisingly, Adan can use half of the training cost (epochs) of SoTA optimizers to achieve higher or comparable performance on ViT, ResNet, MAE, *etc*, and also shows great tolerance to a large range of minibatch size, *e.g.* from 1k to 32k. Code is released at https://github.com/sail-sg/Adan.

1 Introduction

Deep neural networks (DNNs) have made remarkable success in many fields, *e.g.* computer vision [7, 8, 14–16] and natural language processing [17, 18]. A noticeable part of such success is contributed by the stochastic gradient based optimizers which find satisfactory solutions with high efficiency. Starting from AdaGrad [19] and RMSProp [20], adaptive gradient algorithms [1–3, 19–25] have gained wide attention with faster convergence speed. They adjust the learning rate for each gradient coordinate according to the current geometry curvature of the loss objective. Indeed, Adam [1] and AdamW [3], as two representatives which often offer fast convergence speed across many DNN frameworks, have become the default choice to train CNNs and ViTs [9], respectively.

However, none of the above optimizers can always stay undefeated among all its competitors across different network architectures and application settings. For instance, for vanilla ResNet, SGD often achieves better generalization performance than adaptive gradient algorithms such as Adam, whereas on vision transformers (ViTs) [9, 10, 26], SGD often fails and AdamW is the dominant optimizer with higher and more stable performance. Moreover, these commonly used optimizers usually fail for

*Equal contribution. Xingyu did this work during an internship at Sea AI Lab.
⋆Co-corresponding authors.

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

large-batch training, but which is a default setting of the prevalent distributed training. Although there is some performance degradation, we still tend to choose the large-batch setting for large-scale deep learning training tasks due to the unaffordable training time. Though some methods, *e.g.* LARS [27] and LAMB [4], have been proposed to handle large batch sizes, their performance often varies significantly across batch sizes. This performance inconsistency increases the training cost and engineering burden, since one usually has to try various optimizers for different architectures.

When we rethink the current adaptive gradient algorithms, we find that they mainly combine the moving average idea with the heavy ball acceleration technique to estimate the first- and second-order moments of the gradient [1–4]. However, previous studies [5, 28, 29] have revealed that Nesterov acceleration can theoretically achieve a faster convergence speed than heavy ball acceleration, as it uses gradient at an extrapolation point of the current solution and sees a slight "future". Moreover, a recent work [6] has shown the potential of Nesterov acceleration for large-batch training [30]. Thus we are inspired to consider efficiently integrating Nesterov acceleration with adaptive algorithms.

Contributions: 1) We propose an efficient dnn optimizer, named Adan, to train DNNs. Adan develops a Nesterov momentum estimation method to estimate stable and accurate first- and second-order gradient moments in adaptive algorithms for acceleration. **2)** Adan enjoys provably faster convergence speed than previous adaptive algorithms, *e.g.* Adam. **3)** Empirically, Adan shows superior performance over the SoTA deep optimizers across vision, language, and RL tasks.

2 Methodology

In this work, we study the following regularized nonconvex optimization problem:

$$\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}}\left[f(\boldsymbol{\theta},\boldsymbol{\zeta})\right] + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2, \tag{1}$$

where loss $f(\cdot,\cdot)$ is differentiable and possibly nonconvex, data $\boldsymbol{\zeta}$ is drawn from an unknown distribution $\mathcal{D}$, $\boldsymbol{\theta}$ is learnable parameters, and $\|\cdot\|$ is the classical ℓ_2 norm. Here we consider the square ℓ_2 regularizer as it can improve generalization performance and is widely used in practice [3].

2.1 Preliminaries

Taking a deeper step into Adam, one can easily observe that the key moving average idea in Adam is similar to the classical (stochastic) heavy-ball acceleration (HBA) technique [31]:

$$\text{HBA:} \quad \mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k) + \boldsymbol{\xi}_k, \qquad \mathbf{m}_k = (1-\beta_1)\mathbf{m}_{k-1} + \mathbf{g}_k, \qquad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\mathbf{m}_k,$$

where $\mathbf{g}_k$ is the minibatch gradient $\mathbf{g}_k \coloneqq \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}}[\nabla f(\boldsymbol{\theta}_k,\boldsymbol{\zeta})] + \boldsymbol{\xi}_k$, $\boldsymbol{\xi}_k$ is the gradient noise, and the scalar constant η is the base learning rate.

In addition to HBA, Nesterov's accelerated (stochastic) gradient descent (AGD) [5, 28, 29] is another popular acceleration technique in the optimization community:

$$\text{AGD:} \ \mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k - \eta(1-\beta_1)\mathbf{m}_{k-1}) + \boldsymbol{\xi}_k, \ \ \mathbf{m}_k = (1-\beta_1)\mathbf{m}_{k-1} + \mathbf{g}_k, \ \ \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\mathbf{m}_k. \tag{2}$$

Unlike HBA, AGD uses the gradient at the extrapolation point $\boldsymbol{\theta}'_k = \boldsymbol{\theta}_k - (1-\beta_1)(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$. Hence AGD sees a slight "future" to converge faster. Indeed, AGD theoretically converges faster than HBA and achieves optimal convergence rate on the general smooth convex problems [5]. Meanwhile, since the over-parameterized DNNs have been observed/proved to have many convex-alike local basins [32–39], AGD seems more suitable than HBA for DNNs.

For large-batch training, the recent work [6] shows that AGD has the potential to achieve comparable performance to some specifically designed optimizers, *e.g.* LARS and LAMB. With its advantage in convergence and large-batch training, we consider applying AGD to improve adaptive algorithms.

2.2 Adaptive Nesterov Momentum Algorithm

Main iteration. We temporarily set $\lambda = 0$ in Eqn. (1). As aforementioned, AGD computes gradient at an extrapolation point $\boldsymbol{\theta}'_k$ instead of the current iterate $\boldsymbol{\theta}_k$, which however brings extra computation and memory overhead for computing $\boldsymbol{\theta}'_k$ and preserving both $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}'_k$. To solve the issue, we reformulate AGD (2) into its equivalent (See Lemma 1 in Appendix) but more DNN-efficient version:

$$\text{Reformulated AGD:} \ \ \mathbf{m}_k = (1-\beta_1)\mathbf{m}_{k-1} + [\mathbf{g}_k + (1-\beta_1)(\mathbf{g}_k - \mathbf{g}_{k-1})], \quad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\mathbf{m}_k.$$

Algorithm 1: Adan (Adaptive Nesterov Momentum Algorithm)

Input: initialization $\boldsymbol{\theta}_0$, step size η, momentum $(\beta_1, \beta_2, \beta_3) \in [0,1]^3$, weight decay $\lambda > 0$.
Output: some average of $\{\boldsymbol{\theta}_k\}_{k=1}^K$.

while $k < K$ **do**
  compute the stochastic gradient estimator $\mathbf{g}_k$ at $\boldsymbol{\theta}_k$;
  $\mathbf{m}_k = (1-\beta_1)\mathbf{m}_{k-1} + \beta_1 \mathbf{g}_k$ /* set $\mathbf{m}_0 = \mathbf{g}_0$ */;
  $\mathbf{v}_k = (1-\beta_2)\mathbf{v}_{k-1} + \beta_2(\mathbf{g}_k - \mathbf{g}_{k-1})$ /* set $\mathbf{v}_1 = \mathbf{g}_1 - \mathbf{g}_0$ */;
  $\mathbf{n}_k = (1-\beta_3)\mathbf{n}_{k-1} + \beta_3[\mathbf{g}_k + (1-\beta_2)(\mathbf{g}_k - \mathbf{g}_{k-1})]^2$ /* set $\mathbf{n}_0 = \mathbf{g}_0^2$ */;
  $\boldsymbol{\eta}_k = \eta/(\sqrt{\mathbf{n}_k + \varepsilon})$ /* $\varepsilon > 0$ is for stabilize training */;
  $\boldsymbol{\theta}_{k+1} = (1+\lambda\eta)^{-1}[\boldsymbol{\theta}_k - \boldsymbol{\eta}_k \circ (\mathbf{m}_k + (1-\beta_2)\mathbf{v}_k)]$;
end while

where $\mathbf{g}_k = \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})] + \boldsymbol{\xi}_k$. The main idea here is that we maintain $(\boldsymbol{\theta}_k - \eta(1-\beta_1)\mathbf{m}_{k-1})$ rather than $\boldsymbol{\theta}_k$ in vanilla AGD per iteration, as there is no difference between them when the algorithm converges. Like Adam, by regarding $\mathbf{g}'_k = \mathbf{g}_k + (1-\beta_1)(\mathbf{g}_k - \mathbf{g}_{k-1})$ as the current stochastic gradient and movingly averaging $\mathbf{g}'_k$ to estimate the first- and second-moments of gradient, we obtain

$$\text{Vanilla Adan:} \begin{cases} \mathbf{m}_k = (1-\beta_1)\mathbf{m}_{k-1} + \beta_1[\mathbf{g}_k + (1-\beta_1)(\mathbf{g}_k - \mathbf{g}_{k-1})] \\ \mathbf{n}_k = (1-\beta_3)\mathbf{n}_{k-1} + \beta_3(\mathbf{g}_k + (1-\beta_1)[\mathbf{g}_k - \mathbf{g}_{k-1})]^2 \\ \boldsymbol{\eta}_k = \eta/\big(\sqrt{\mathbf{n}_k + \varepsilon}\big), \quad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\eta}_k \circ \mathbf{m}_k. \end{cases}$$

The main difference of Adan with Adam-type methods is that as compared in Eqn. (3), the moment $\mathbf{m}_k$ of Adan is the average of $\{\mathbf{g}_t + (1-\beta_1)(\mathbf{g}_t - \mathbf{g}_{t-1})\}_{t=1}^k$ while those of Adam-type are the average of $\{\mathbf{g}_t\}_{t=1}^k$. So is their second-order term $\mathbf{n}_k$.

$$\mathbf{m}_k = \begin{cases} \sum_{t=0}^k c_{k,t}[\mathbf{g}_t + (1-\beta_1)(\mathbf{g}_t - \mathbf{g}_{t-1})], & \text{Adan}, \\ \sum_{t=0}^k c_{k,t}\mathbf{g}_t, & \text{Adam}, \end{cases} \quad c_{k,t} = \begin{cases} \beta_1(1-\beta_1)^{(k-t)} & t > 0, \\ (1-\beta_1)^k & t = 0, \end{cases} \tag{3}$$

The first-order moment $\mathbf{m}_k = \sum_{t=0}^k c_{k,t}[\mathbf{g}_t + (1-\beta_1)(\mathbf{g}_t - \mathbf{g}_{t-1})]$ consists of two terms, i.e., gradient term $\mathbf{g}_t$ and gradient difference term $(\mathbf{g}_t - \mathbf{g}_{t-1})$, which actually have different physic meanings. So we further decouple them for greater flexibility and also better trade-off between them:

$$(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)/\boldsymbol{\eta}_k = \sum\nolimits_{t=0}^k \big[c_{k,t}\mathbf{g}_t + (1-\beta_2)c'_{k,t}(\mathbf{g}_t - \mathbf{g}_{t-1})\big] = \mathbf{m}_k + (1-\beta_2)\mathbf{v}_k,$$

where $c'_{k,t} = \beta_2(1-\beta_2)^{(k-t)}$ for $t > 0$, $c'_{k,t} = (1-\beta_2)^k$ for $t = 0$, and $\mathbf{m}_k$ and $\mathbf{v}_k$ are defined as

$$\mathbf{m}_k = (1-\beta_1)\mathbf{m}_{k-1} + \beta_1\mathbf{g}_k, \qquad \mathbf{v}_k = (1-\beta_2)\mathbf{v}_{k-1} + \beta_2(\mathbf{g}_k - \mathbf{g}_{k-1}).$$

This change for a flexible estimation does not impair convergence speed. As we show in Theorem 1 (see Sec. C in Appendix), the complexity of Adan under this change matches the lower complexity bound. We do not separate the gradients and their difference in the second-order moment $\mathbf{n}_k$, since $\mathbb{E}(\mathbf{n}_k)$ contains the correlation term $\mathrm{Cov}(\mathbf{g}_k, \mathbf{g}_{k-1}) \neq 0$ which may have statistical significance.

Decay Weight by Proximation. As observed in AdamW, decoupling the optimization objective and ℓ_2 regularizer can largely improve the generalization performance. Intuitively, at each iteration, we minimize a first-order approximation of $F(\cdot)$ at the point $\boldsymbol{\theta}_k$:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\eta}_k \circ \bar{\mathbf{m}}_k = \operatorname{argmin}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}_k) + \langle \bar{\mathbf{m}}_k, \boldsymbol{\theta} - \boldsymbol{\theta}_k \rangle + \frac{1}{2\eta}\|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}},$$

where $\|\mathbf{x}\|^2_{\sqrt{\mathbf{n}_k}} := \langle \mathbf{x}, \sqrt{\mathbf{n}_k + \varepsilon} \circ \mathbf{x} \rangle$ and $\bar{\mathbf{m}}_k := \mathbf{m}_k + (1-\beta_2)\mathbf{v}_k$ is the first-order derivative of $F(\cdot)$ in some sense. Interestingly, follow the idea of proximal gradient descent [40, 41], we decouple the ℓ_2 regularizer from $F(\cdot)$ and only linearize the loss function $f(\cdot)$:

$$\boldsymbol{\theta}_{k+1} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left(\frac{\lambda}{2}\|\boldsymbol{\theta}\|^2_{\sqrt{\mathbf{n}_k}} + \langle \bar{\mathbf{m}}_k, \boldsymbol{\theta} - \boldsymbol{\theta}_k \rangle + \frac{1}{2\eta}\|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} \right) = \frac{\boldsymbol{\theta}_k - \boldsymbol{\eta}_k \circ \bar{\mathbf{m}}_k}{1 + \lambda\eta}.$$

From the perspective of the proximal operator, we find that the optimization objective of Decoupled Regularization methods (*e.g.* Adan and AdamW) at the k-th iteration is changed from the

Table 1: Top-1 Acc. (%) of ResNet and ConvNext on ImageNet. $*$ and $\diamond$ are reported in [42], [8].

Epoch	ResNet-50 100	ResNet-50 200	ResNet-50 300	ResNet-101 100	ResNet-101 200	ResNet-101 300
SAM	77.3	78.7	79.4	79.5	81.1	81.6
SGD-M	77.0	78.6	79.3	79.3	81.0	81.4
Adam	76.9	78.4	78.8	78.4	80.2	80.6
AdamW	77.0	78.9	79.3	78.9	79.9	80.4
LAMB	77.0	79.2	79.8*	79.4	81.1	81.3*
Adan (ours)	**78.1**	**79.7**	**80.2**	**79.9**	**81.6**	**81.8**

Epoch	ConvNext Tiny 150	ConvNext Tiny 300
AdamW [3, 8]	81.2	82.1$^\diamond$
Adan (ours)	**81.7**	**82.4**
Epoch	ConvNext Small 150	ConvNext Small 300
AdamW [3, 8]	82.2	83.1$^\diamond$
Adan (ours)	**82.5**	**83.3**

Table 2: Top-1 Acc. (%) of ViT and Swin on ImageNet. $*$ and $\diamond$ are respectively reported in [9], [10].

Epoch	ViT Small 150	ViT Small 300	ViT Base 150	ViT Base 300	Swin Tiny 150	Swin Tiny 300	Swin small 150	Swin small 300	Swin Base 150	Swin Base 300
AdamW [3, 9, 10]	78.3	79.9*	79.5	81.8*	79.9	81.2$^\diamond$	82.1	83.2$^\diamond$	82.6	83.5$^\diamond$
Adan (ours)	**79.6**	**80.9**	**81.7**	**82.3**	**81.3**	**81.6**	**82.9**	**83.7**	**83.3**	**83.8**

vanilla "static" function $F(\cdot)$ in (1) to a "dynamic" function $F_k(\cdot)$, which adaptively regularizes the coordinates with larger gradient square terms more. We summarize our Adan in Algorithm 1.

Convergence Analysis: As shown in Theorems 1 in Appendix. C, the convergence speed of Adan matches the best-known theoretical lower bound for non-convex stochastic optimization problems. This conclusion is still valid when it also uses the decoupled weight decay.

3 Experimental Results

For all the tested vision tasks, NLP, and RL tasks, we only replace the default optimizer with our Adan, and do not make other changes, *e.g.* network architectures and data augmentation.

Vision Results. 1) supervised settings: we report the results on CNN-type architectures and ViTs in Tables 1 and 2, respectively. 2) self-supervised settings: we follow the MAE training framework to pretrain and fine-tune ViT-B and ViT-L, and report results in Table 3. All these results show that *in most cases, Adan can use half of the training cost (epochs) of SoTA optimizers to achieve higher or comparable performance on ViT, ResNet, MAE,* etc.

Table 3: Top-1 Acc. (%) of ViT-B and ViT-L trained by self-supervised MAE on ImageNet.

Epoch	MAE-ViT-B 300	MAE-ViT-B 800	MAE-ViT-B 1600	MAE-ViT-L 800	MAE-ViT-L 1600
AdamW	82.9	—	83.6	85.4	85.9
Adan	**83.4**	**83.8**	—	**85.9**	—

NLP Results. 1) supervised settings: we investigate the performance of Adan on Transformer-XL, and report the results in Table 4. 2) self-supervised settings: we use Adan to train BERT from scratch, and report the results in Table 5. *For all NLP tasks, Adan achieves higher performance than the default SoTAs, and suppress Adam within half training steps on Transformer-XL.*

Table 4: Test PPL for Transformer-XL-base model on WikiText-103.

Transformer-XL	Training Steps 50k	Training Steps 100k	Training Steps 200k
Adam [1]	28.5	25.5	24.2
Adan (ours)	**26.2**	**24.2**	**23.5**

Table 5: Results (the higher, the better) of BERT-base model on the development set of GLUE.

BERT-base	MNLI	QNLI	QQP	RTE	SST-2	CoLA	STS-B	**Average**
Adam [1] (from [43])	83.7/84.8	89.3	90.8	71.4	91.7	48.9	91.3	81.5
Adam [1] (reproduced)	84.9/84.9	90.8	90.9	69.3	92.6	58.5	88.7	82.5
Adan (ours)	**85.7/85.6**	**91.3**	**91.2**	**73.3**	**93.2**	**64.6**	**89.3**	**84.3 (+1.8)**

RL Results. We replace the default Adam optimizer in PPO [44] (one of the most popular policy gradient method), and do not make other change in PPO. Fig. 1 shows that *on representative MuJoCo games, PPO-Adan achieves much higher rewards than PPO with Adam as its optimizer.*

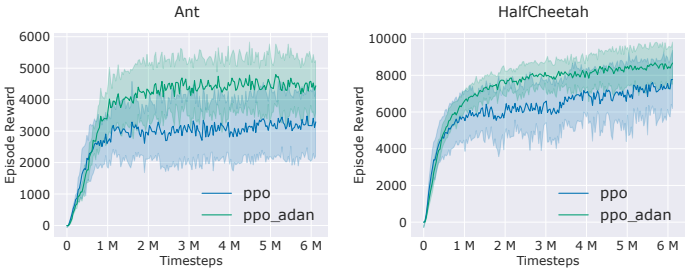

Figure 1: PPO with Adam and Adan as its optimizer.

More Extra Results. Due to space limitation, we defer more extra results on vision, NLP and RL tasks (*e.g.* results with large batch size, loss curve, ablation study, *etc.*) to Appendix.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[2] 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.

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

[4] 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.

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

[6] Zachary Nado, Justin M Gilmer, Christopher J Shallue, Rohan Anil, and George E Dahl. A large batch optimizer reality check: Traditional, generic optimizers suffice across batch sizes. *arXiv preprint arXiv:2102.06356*, 2021.

[7] 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.

[8] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[9] 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.

[10] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022.

[12] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[15] 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. In *International Conference on Learning Representations*, 2020.

[16] Pan Zhou, Yichen Zhou, Chenyang Si, Weihao Yu, Teck Khim Ng, and Shuicheng Yan. Mugs: A multi-granular self-supervised learning framework. In *arXiv preprint arXiv:2203.14415*, 2022.

[17] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George E Dahl, George Saon, Hagen Soltau, Tomas Beran, Aleksandr Y Aravkin, and Bhuvana Ramabhadran. Improvements to deep convolutional neural networks for LVCSR. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 315–320. IEEE, 2013.

[18] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE Trans. on audio, speech, and language processing*, 22(10):1533–1545, 2014.

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

[20] 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.

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

[22] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2018.

[23] 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.

[24] 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.

[25] Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. Adamp: Slowing down the slowdown for momentum optimizers on scale-invariant weights. In *International Conference on Learning Representations*, 2020.

[26] 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.

[27] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[28] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o $(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[29] Yurii Nesterov. On an approach to the construction of optimal methods of minimization of smooth convex functions. *Ekonomika i Mateaticheskie Metody*, 24(3):509–517, 1988.

[30] Xiaoxin He, Fuzhao Xue, Xiaozhe Ren, and Yang You. Large-scale deep learning optimizations: A comprehensive survey. *arXiv preprint arXiv:2111.00856*, 2021.

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

[32] M. Hardt and T. Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

[33] Bo Xie, Yingyu Liang, and Le Song. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics*, pages 1216–1224. PMLR, 2017.

[34] Z. Li and Y. Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, 2017.

[35] Z. Charles and D. Papailiopoulos. Stability and generalization of learning algorithms that converge to global optima. In *International Conference on Machine Learning*, pages 745–754. PMLR, 2018.

[36] Y. Zhou and Y. Liang. Characterization of gradient dominance and regularity conditions for neural networks. In *International Conference on Learning Representations*, 2018.

[37] Pan Zhou, Hanshu Yan, Xiaotong Yuan, Jiashi Feng, and Shuicheng Yan. Towards understanding why lookahead generalizes better than sgd and beyond. In *Neural Information Processing Systems*, 2021.

[38] Quynh Nguyen, Marco Mondelli, and Guido F Montufar. Tight bounds on the smallest eigenvalue of the neural tangent kernel for deep relu networks. In *International Conference on Machine Learning*, pages 8119–8129, 2021.

[39] Quynh N Nguyen and Marco Mondelli. Global convergence of deep networks with one wide layer followed by pyramidal topology. *Advances in Neural Information Processing Systems*, 33:11961–11972, 2020.

[40] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[41] Zhenxun Zhuang, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona. Understanding adamw through proximal methods and scale-freeness. *arXiv preprint arXiv:2202.00089*, 2022.

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

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[44] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

[45] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[46] Zhishuai Guo, Yi Xu, Wotao Yin, Rong Jin, and Tianbao Yang. A novel convergence analysis for algorithms of the adam family. *arXiv preprint arXiv:2112.03459*, 2021.

[47] 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.

[48] 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.

[49] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[50] Ashok Cutkosky and Harsh Mehta. Momentum improves normalized sgd. In *International Conference on Machine Learning*, pages 2260–2268. PMLR, 2020.

[51] Mingrui Liu, Wei Zhang, Francesco Orabona, and Tianbao Yang. Adam +: A stochastic method with adaptive variance reduction. *arXiv preprint arXiv:2011.11985*, 2020.

[52] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pages 5905–5914. PMLR, 2021.

[53] Yizhou Wang, Yue Kang, Can Qin, Huan Wang, Yi Xu, Yulun Zhang, and Yun Fu. Adapting stepsizes by momentumized gradients improves optimization and generalization. *arXiv preprint arXiv:2106.11514*, 2021.

[54] 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.

[55] 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.

[56] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.

[57] Naichen Shi, Dawei Li, Mingyi Hong, and Ruoyu Sun. Rmsprop converges with proper hyper-parameter. In *International Conference on Learning Representations*, 2020.

[58] Cong Fang, Zhouchen Lin, and Tong Zhang. Sharp analysis for nonconvex sgd escaping from saddle points. In *Conference on Learning Theory*, pages 1192–1234. PMLR, 2019.

[59] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[60] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

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

[62] 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.

[63] 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.

[64] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661, 2016.

[65] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. *arXiv preprint arXiv:2106.01548*, 2021.

[66] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *Advances in Neural Information Processing Systems*, 34:30392–30400, 2021.

[67] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[68] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[69] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *arXiv preprint arXiv:2107.14171*, 2021.