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:

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

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 $\mathbf{g}_k := \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}}[\nabla f(\boldsymbol{\theta}_k,\boldsymbol{\zeta})] + \boldsymbol{\xi}_k$, $\boldsymbol{\xi}_k$ 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 $\lambda_k > 0$.
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 */;
  $\boldsymbol{\theta}_{k+1} = (1+\lambda_k\eta)^{-1}[\boldsymbol{\theta}_k - \boldsymbol{\eta}_k \circ (\mathbf{m}_k + (1-\beta_2)\mathbf{v}_k)]$;
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:

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

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 simple-type regularization (*e.g.* ℓ_2 regularizer) can largely improve the generalization performance. Here we follow this idea but from a rigorous optimization perspective. Intuitively, at each iteration, we minimize the first-order approximation of $F(\cdot)$ at the point $\boldsymbol{\theta}_k$:

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

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. Follow the idea of proximal gradient descent [40, 41], we decouple the ℓ_2 regularizer from $F(\cdot)$ and only linearize the loss function $f(\cdot)$:

$$\boldsymbol{\theta}_{k+1} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left(\frac{\lambda_k}{2}\|\boldsymbol{\theta}\|^2_{\sqrt{\mathbf{n}_k}} + \langle \bar{\mathbf{m}}_k, \boldsymbol{\theta} - \boldsymbol{\theta}_k \rangle + \frac{1}{2\eta}\|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} \right) = \frac{\boldsymbol{\theta}_k - \boldsymbol{\eta}_k \circ \bar{\mathbf{m}}_k}{1 + \lambda_k\eta}, \tag{4}$$

where $\lambda_k > 0$ is the weight decay constant at the k-th iteration. One can find that the optimization objective of Separated Regularization at the k-th iteration is changed from the vanilla "static" function

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**

$F(\cdot)$ 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.*

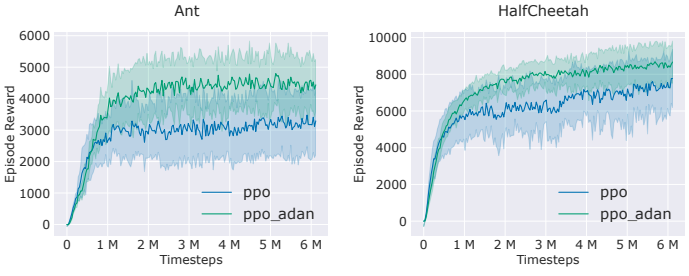


Figure 1: PPO with Adam and Adan as its optimizer.

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.

Appendix

The appendix contains some additional experimental results, detailed derivation of Adan, and the technical proofs of convergence results of the paper entitled "Adan: Adaptive Nesterov Momentum Algorithm for Faster Optimizing Deep Models". It is structured as follows. Sec. A summarizes the notations throughout this document. Sec. B provides the motivation and derivation of Adan in great details. Sec. C offers the convergence analysis for Adan and compares several prevalent deep learning optimizers from the theoretical perspective. We provide additional details, result, and discussion for experiments in Sec. D.

Moreover, Sec. E provides the proof of the equivalence between AGD and reformulated AGD, i.e., the proof of Lemma 1. And then, we provide the proof of Theorem 1 in Sec. F. Finally, we present some auxiliary lemmas in Sec. G.

A Notation

We provide some notations that are frequently used throughout the paper. The scale c is in normal font. And the vector is in bold lowercase. Give two vectors $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x} \geq \mathbf{y}$ means that $(\mathbf{x}-\mathbf{y})$ is a non-negative vector. $\mathbf{x}/\mathbf{y}$ or $\frac{\mathbf{x}}{\mathbf{y}}$ represents the element-wise vector division. $\mathbf{x} \circ \mathbf{y}$ means the element-wise multiplication, and $(\mathbf{x})^2 = \mathbf{x} \circ \mathbf{x}$. $\langle \cdot, \cdot \rangle$ is the inner product. Given a non-negative vector $\mathbf{n} \geq 0$, we let $\|\mathbf{x}\|^2_{\sqrt{\mathbf{n}}} \coloneqq \langle \mathbf{x}, \sqrt{\mathbf{n}+\varepsilon} \circ \mathbf{x} \rangle$. Unless otherwise specified, $\|\mathbf{x}\|$ is the vector ℓ_2 norm. Note that $\mathbb{E}(\mathbf{x})$ is the expectation of random random vector $\mathbf{x}$.

B Detailed Methodology

At below, we elaborate on detailed algorithmic steps in Sec. B.1.

B.1 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, Lemma 1 with proof in Appendix E reformulates AGD (2) into its equivalent but more DNN-efficient version.

Lemma 1. *Assume $\mathbb{E}(\boldsymbol{\xi}_k) = \mathbf{0}$, $\mathrm{Cov}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j) = 0$ for any $k, i, j > 0$, $\bar{\boldsymbol{\theta}}_k$ and $\bar{\mathbf{m}}_k$ be the iterate and momentum of the vanilla AGD in Eqn. (2), respectively. Let $\boldsymbol{\theta}_{k+1} \coloneqq \bar{\boldsymbol{\theta}}_{k+1} - \eta(1-\beta_1)\bar{\mathbf{m}}_k$ and $\mathbf{m}_k \coloneqq (1-\beta_1)^2\bar{\mathbf{m}}_{k-1} + (2-\beta_1)(\nabla f(\boldsymbol{\theta}_k) + \boldsymbol{\xi}_k)$. The vanilla AGD in Eqn. (2) becomes*

$$\textit{Reformulated AGD:} \begin{cases} \mathbf{g}_k = \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})] + \boldsymbol{\xi}_k \\ \mathbf{m}_k = (1-\beta_1)\mathbf{m}_{k-1} + [\mathbf{g}_k + (1-\beta_1)(\mathbf{g}_k - \mathbf{g}_{k-1})] \\ \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\mathbf{m}_k \end{cases}.$$

Moreover, if vanilla AGD in Eqn. (2) converges, so does AGD-II, and $\mathbb{E}(\boldsymbol{\theta}_\infty) = \mathbb{E}(\bar{\boldsymbol{\theta}}_\infty)$.

The main idea in Lemma 1 is that we maintain $(\boldsymbol{\theta}_k - \eta(1-\beta_1)\mathbf{m}_{k-1})$ rather than $\boldsymbol{\theta}_k$ in vanilla AGD at each iteration, since there is no difference between them when the algorithm converges. Like other adaptive optimizers, 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

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

The main difference of Adan with Adam-type methods and Nadam [45] is that as compared in Eqn. (5), the momentom $\mathbf{m}_k$ 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 and Nadam are the average of $\{\mathbf{g}_t\}_{t=1}^k$. So is their second-order term $\mathbf{n}_k$.

$$\mathbf{m}_k = \begin{cases} \sum_{t=0}^k c_{k,t}[\mathbf{g}_t + (1-\beta_1)(\mathbf{g}_t - \mathbf{g}_{t-1})], & \text{Adan}, \\ \sum_{t=0}^k c_{k,t}\mathbf{g}_t, & \text{Adam}, \\ \frac{\mu_{k+1}}{\mu'_{k+1}}\left(\sum_{t=0}^k c_{k,t}\mathbf{g}_t\right) + \frac{1-\mu_k}{\mu'_k}\mathbf{g}_k, & \text{Nadam}, \end{cases} \quad c_{k,t} = \begin{cases} \beta_1(1-\beta_1)^{(k-t)} & t > 0, \\ (1-\beta_1)^k & t = 0, \end{cases} \tag{5}$$

where $\{\mu_t\}_{t=1}^\infty$ is a predefined exponentially decaying sequence, $\mu'_k = 1 - \prod_{t=1}^k \mu_t$. So Nadam is more like Adam than Adan, as their $\mathbf{m}_k$ movingly averages the historical gradients instead of gradient differences in Adan. For a large k (i.e. small μ_k), $\mathbf{m}_k$ in Nadam and Adam are almost the same.

As shown in Eqn. (5), 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 here we further decouple them for greater flexibility and also better trade-off between them. Specifically, we estimate

$$(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)/\boldsymbol{\eta}_k = \sum_{t=0}^k \left[c_{k,t}\mathbf{g}_t + (1-\beta_2)c'_{k,t}(\mathbf{g}_t - \mathbf{g}_{t-1})\right] = \mathbf{m}_k + (1-\beta_2)\mathbf{v}_k := \bar{\mathbf{m}}_k, \tag{6}$$

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, 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 simple-type regularization (*e.g.* ℓ_2 regularizer) can largely improve the generalization performance. Here we follow this idea but from a rigorous optimization perspective. Intuitively, at each iteration, we minimize the first-order approximation of $F(\cdot)$ at the point $\boldsymbol{\theta}_k$:

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

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. Follow the idea of proximal gradient descent [40, 41], we decouple the ℓ_2 regularizer from $F(\cdot)$ and only linearize the loss function $f(\cdot)$:

$$\boldsymbol{\theta}_{k+1} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left(\frac{\lambda_k}{2}\|\boldsymbol{\theta}\|^2_{\sqrt{\mathbf{n}_k}} + \langle \bar{\mathbf{m}}_k, \boldsymbol{\theta} - \boldsymbol{\theta}_k\rangle + \frac{1}{2\eta}\|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}}\right) = \frac{\boldsymbol{\theta}_k - \boldsymbol{\eta}_k \circ \bar{\mathbf{m}}_k}{1 + \lambda_k\eta}, \tag{7}$$

where $\lambda_k > 0$ is the weight decay constant at the k-th iteration. Interestingly, we can easily reveal the updating rule $\boldsymbol{\theta}_{k+1} = (1-\lambda_k\eta)\boldsymbol{\theta}_k - \boldsymbol{\eta}_k \circ \bar{\mathbf{m}}_k$ of AdamW by using the first-order approximation of Eqn. (7) around $\eta = 0$: 1) $(1+\lambda_k\eta)^{-1} = (1-\lambda_k\eta) + \mathcal{O}(\eta^2)$; 2) $\lambda_k\eta\boldsymbol{\eta}_k = \mathcal{O}(\eta^2)$. One can find that the optimization objective of Separated Regularization at the k-th iteration is changed from the vanilla "static" function $F(\cdot)$ in (1) to a "dynamic" function $F_k(\cdot)$, shown in Eqn. (8), adaptively regularizes the coordinates with larger gradient square terms more:

$$F_k(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}}\left[f(\boldsymbol{\theta}, \boldsymbol{\zeta})\right] + \frac{\lambda_k}{2}\|\boldsymbol{\theta}\|^2_{\sqrt{\mathbf{n}_k}}. \tag{8}$$

We summarize our Adan in Algorithm 1.

C Convergence Analysis

For non-convex optimization, we follow [2, 46–53] to make several mild assumptions.

Assumption 1 (L-smoothness)**.** *The function $f(\cdot,\cdot)$ is L-smooth w.r.t. the parameter, if $\exists L > 0$,*

$$\|\nabla\, \mathbb{E}_{\boldsymbol{\zeta}}\left[f(\mathbf{x}, \boldsymbol{\zeta})\right] - \nabla\, \mathbb{E}_{\boldsymbol{\zeta}}\left[f(\mathbf{y}, \boldsymbol{\zeta})\right]\| \leq L\|\mathbf{x} - \mathbf{y}\|, \qquad \forall \mathbf{x},\ \mathbf{y}.$$

Assumption 2 (Unbiased and bounded gradient oracle)**.** *The stochastic gradient oracle $\mathbf{g}_k = \mathbb{E}_{\boldsymbol{\zeta}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})] + \boldsymbol{\xi}_k$ is unbiased, and its magnitude and variance are bounded with probability 1:*

$$\mathbb{E}\left(\boldsymbol{\xi}_k\right) = \mathbf{0}, \quad \|\mathbf{g}_k\|_\infty \leq c_\infty/3, \quad \mathbb{E}\Big(\|\boldsymbol{\xi}_k\|^2\Big) = \mathbb{E}\Big(\|\nabla\, \mathbb{E}_{\boldsymbol{\zeta}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})] - \mathbf{g}_k\|^2\Big) \leq \sigma^2, \quad \forall k \in [T].$$

For a general nonconvex problem, if Assumptions 1 and 2 hold, the lower bound of the stochastic gradient complexity (a.k.a. IFO complexity) to find an ϵ-approximate first-order stationary point (ϵ-ASP) is $\Omega(\epsilon^{-4})$ [54, 55].

Table 6: Comparison of different adaptive gradient algorithms on nonconvex stochastic problems. "Separated Reg." refers to whether the ℓ_2 regularizer (weight decay) can be separated from the loss objective like AdamW [3]. "Complexity" denotes stochastic gradient complexity to find an ϵ-approximate first-order stationary point. Adam-type methods [46] includes Adam, AdaMomentum [53], AdaGrad [19], AdaBound [23] and AMSGrad [21], *etc*. AdamW has no available convergence result. For SAM [49], we compare their adaptive versions. d is the variable dimension.

Optimizer	Separated Reg.	Batch Size Condition	Grad Bound	Complexity	Lower Bound [55]
Adam-type [46]	✗	✗	$\ell_\infty \le c_\infty$	$\mathcal{O}(c_\infty^2 d\epsilon^{-4})$	$\Omega(\epsilon^{-4})$
RMSProp [20, 47]	✗	✗	$\ell_\infty \le c_\infty$	$\mathcal{O}(\sqrt{c_\infty} d\epsilon^{-4})$	$\Omega(\epsilon^{-4})$
AdamW [3]	✓	—	—	—	—
Adabelief [2]	✗	✗	$\ell_2 \le c_2$	$\mathcal{O}(c_2^6\epsilon^{-4})$	$\Omega(\epsilon^{-4})$
Padam [48]	✗	✗	$\ell_\infty \le c_\infty$	$\mathcal{O}(\sqrt{c_\infty} d\epsilon^{-4})$	$\Omega(\epsilon^{-4})$
LAMB [4]	✗	$\mathcal{O}(\epsilon^{-4})$	$\ell_2 \le c_2$	$\mathcal{O}(c_2^2 d\epsilon^{-4})$	$\Omega(\epsilon^{-4})$
Adan (ours)	✓	✗	$\ell_\infty \le c_\infty$	$\mathcal{O}(c_\infty^{2.5}\epsilon^{-4})$	$\Omega(\epsilon^{-4})$

Lipschitz Gradient Theorem 1 with proof in Appendix F proves the convergence of Adan on problem (8) with lipschitz gradient condition.

Theorem 1. *Suppose Assumptions 1 and 2 hold. Let* $\max\{\beta_1, \beta_2\} = \mathcal{O}(\epsilon^2)$, $\mu := \beta_3 c_\infty^2/\varepsilon \ll 1$, $\eta = \mathcal{O}(\epsilon^2)$, *and* $\lambda_k = \lambda(1-\mu)^k$. *Algorithm 1 runs at most* $K = \Omega(c_\infty^{2.5}\epsilon^{-4})$ *iterations to achieve*

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\Big(\|\nabla F_k(\boldsymbol{\theta}_k)\|^2\Big) \le 4\epsilon^2.$$

That is, to find an ϵ-ASP, the stochastic gradient complexity of Adan on problem (8) is $\mathcal{O}(c_\infty^{2.5}\epsilon^{-4})$.

Theorem 1 shows that under Assumptions 1 and 2, Adan can converge to an ϵ-ASP of a nonconvex stochastic problem with stochastic gradient complexity $\mathcal{O}(c_\infty^{2.5}\epsilon^{-4})$ which accords with the lower bound $\Omega(\epsilon^{-4})$ in [54]. For this convergence, Adan has no requirement on minibatch size and only assumes gradient estimation to be unbiased and bounded. Moreover, as shown in Table 6 in Sec. 1, the complexity of Adan is superior to those of previous adaptive gradient algorithms. For Adabelief and LAMB, Adan always has lower complexity, and respectively enjoys $d^3\times$ and $d^2\times$ lower complexity for the worst case. See detailed discussion in Sec. 1. Adam-type optimizers (*e.g.* Adam and AMSGrad) enjoy the same complexity with Adan. But they cannot separate the ℓ_2 regularizer with the objective like AdamW and our Adan. In other words, they always solve a static loss $F(\cdot)$ rather than a dynamic loss $F_k(\cdot)$ in Adan or AdamW. The regularizer separation can boost generalization performance [9, 10] and already helps AdamW dominate training of ViT-alike architectures.

Besides, some previous analyses [23, 24, 56, 57] need the moving average hyper-parameters (i.e. βs) to be close or increased to one, which contradicts with the practice that βs are close to zero. In contrast, Theorem 1 assumes that all βs are very small, which is more consistent with the practice. Note that when $\mu = c/T$, we have $\lambda_k/\lambda \in [(1-c), 1]$ during training. We may let $(1-c)$ close to 1, and hence we could choose the λ_k as a fixed constant in the experiment for convenience.

C.1 Comparison between DNN Optimizers in Theory

As shown in Table 6, we theoretically justify the advantages of Adan over previous SoTA adaptive gradient algorithms on nonconvex stochastic problems, *e.g.* deep learning problems.

Given Lipschitz Gradient condition, to find an ϵ-approximate first-order stationary point, Adan has the stochastic gradient complexity $\mathcal{O}(c_\infty^{2.5}\epsilon^{-4})$ which accords with the lower bound $\Omega(\epsilon^{-4})$ in [58] (up to a constant factor). This complexity is lower than $\mathcal{O}(c_2^6\epsilon^{-4})$ of Adabelief [2] and $\mathcal{O}(c_2\sqrt{d}\epsilon^{-4})$ of LAMB, especially on over-parameterized networks. Specifically, for the d-dimensional gradient, compared with its ℓ_2 norm c_2, its ℓ_∞ norm c_∞ is usually much smaller, and can be $\sqrt{d}\times$ smaller for the best case. Moreover, different from Adam-type optimizers (*e.g.* Adam), Adan can separate the ℓ_2 regularizer with the loss objective like AdamW whose generalization benefits have been validated in many works [9, 10]. For AdamW, its convergence has not been proved yet.

D Detailed Experimental Results

We evaluate Adan on vision tasks, natural language processing (NLP) tasks and reinforcement learning (RL) tasks. For vision tasks, we test Adan on several representative SoTA backbones under the conventional supervised settings, including 1) CNN-type architectures (ResNets [7] and ConvNexts [8]) and 2) ViTs (ViTs [9, 15] and Swins [10]). Moreover, we also investigate Adan via the self-supervised pretraining by using it to train MAE ViT [11]. For NLP tasks, we train Transformer-XL [12] and BERT [13] for sequence modeling. On RL tasks, we evaluate Adan on four games in MuJoCo [59]. For fairness, in all experiments, we only replace the optimizer with Adan and tune the base step size, warmup epochs and weight decay while fixing the optimizer-independent hyper-parameters, *e.g.* data augmentation and model parameters.

D.1 Experiments for Vision Tasks

Besides the vanilla supervised training setting used in ResNets [7], we further consider the following two prevalent training settings on ImageNet [60].

Training Setting I. The recently proposed "A2 training recipe" in [42] has pushed the performance limits of many SoTA CNN-type architectures by using stronger data augmentation and more training iterations. For example, on ResNet50 it sets new SoTA 80.2%, and improves the accuracy 76.1% under vanilla setting in [7]. Specifically, for data augmentation, this setting uses random crop, horizontal flipping, Mixup (0.1) [61]/CutMix (1.0) [62] with probability 0.5, and RandAugment [63] with $M = 7, N = 2$ and MSTD $= 0.5$. It sets stochastic depth (0.05) [64], and adopts cosine learning rate decay and binary cross-entropy (BCE) loss. For Adan, we use batch size 4096 for ViT and 2048 for ResNet.

Training Setting II. We follow the same official training procedure of ViT/Swin/ConvNext. For this setting, data augmentation includes random crop, horizontal flipping, Mixup (0.8), CutMix (1.0), RandAugment ($M = 9$, MSTD $= 0.5$) and Random Erasing ($p = 0.25$). We use CE loss, the cosine decay for base learning rate, the stochastic depth (with official parameters), and weight decay. For Adan, we set batch size 2048 for Swin and 4096 for ViT/ConvNext/MAE. We follow MAE and tune β_3 as 0.1.

Implementation Details of Adan. For the large-batch training experiment, we use the sqrt rule to scale the learning rate: lr $= \sqrt{\frac{\text{batch size}}{256}} \times$ 6.25e-3, and respectively set warmup epochs $\{20, 40, 60, 100, 160, 200\}$ for batch size bs $= \{1k, 2k, 4k, 8k, 16k, 32k\}$. For other remaining experiments, we use the hyper-parameters: learning rate 1.5e-2 for ViT/Swin/ResNet/ConvNext and MAE fine-tuning, and 2.0e-3 for MAE pre-training according to the official settings. We set $\beta_1 = 0.02, \beta_2 = 0.08$ and $\beta_3 = 0.01$, and let weight decay be 0.02 unless noted otherwise. We clip the global gradient norm to 5 for ResNet training and do not clip gradient for ViT, Swin, ConvNext, and MAE.

D.1.1 Results on CNN-type Architectures

To train ResNet and ConvNext, we respectively use their official Training Setting I and II. For ResNet/ConvNext, its default official optimizer is LAMB/AdamW. From Table 1, one can observe that on ResNet, 1) in most cases, Adan only running 200 epochs can achieve higher or comparable top-1 accuracy on ImageNet [60] compared with the official SoTA result trained by LAMB with 300 epochs; 2) Adan gets more improvements over other optimizers, when training is insufficient, *e.g.* 100 epochs. The possible reason for observation 1) is the regularizer separation, which can dynamically adjust the weight decay for each coordinate instead of sharing a common one like LAMB. For observation 2), this can be explained by the faster convergence speed of Adan than other optimizers. As shown in Figure 2, Adan converges faster than many adaptive gradient optimizers. This faster speed partially comes from its large learning rate guaranteed by Theorem 1, almost $3\times$ larger than that of LAMB. The same as Nesterov acceleration, Adan could look ahead for possible corrections. Note, we have tried to adjust learning rate and warmup-epoch for Adam and LAMB, but observed unstable training behaviors. On ConvNext (tiny and small), one can observe similar comparison results on ResNet.

Table 7: Top-1 accuracy (%) of ViT-S on ImageNet under the Training Setting I.

Batch Size	1k	2k	4k	8k	16k	32k
LAMB [4, 30]	78.9	79.2	79.8	79.7	79.5	78.4
Adan (ours)	**80.9**	**81.1**	**81.1**	**80.8**	**80.5**	**80.2**

Table 8: Top-1 accuracy (%) of different optimizers when training ViT-S on ImageNet trained under training setting II. * is reported in [9].

Epoch	100	150	200	300
AdamW (default)	76.1	78.9	79.2	79.9*
Adam	62.0	64.0	64.5	66.7
SGD-M (AGD)	64.3	68.7	71.4	73.9
LAMB	69.4	73.8	75.9	77.7
Adan (ours)	**77.5**	**79.6**	**80.0**	**80.9**

D.1.2 Results on ViTs

Supervised Training. For this setting, we train ViT and Swin under their official training setting, i.e. Training Setting II. Table 2 shows that across different model sizes of ViT and Swin, Adan outperforms the official AdamW optimizer by a large margin. For ViTs, their gradient per iteration differs much from the previous one due to the much sharper loss landscape than CNNs [65] and strong random augmentations for training. So it is hard to train ViTs to converge within a few epochs. Thanks to its faster convergence, as shown in Figure 2, Adan is very suitable for this situation. Moreover, the direction correction term from the gradient difference $\mathbf{v}_k$ of Adan can also better correct the first- and second-order moments. One piece of evidence is that the first-order moment decay coefficient $\beta_1 = 0.02$ of Adan is much smaller than 0.1 used in other deep optimizers. Note, we have also tried to increase the decay coefficient for AdamW but observed performance degradation.

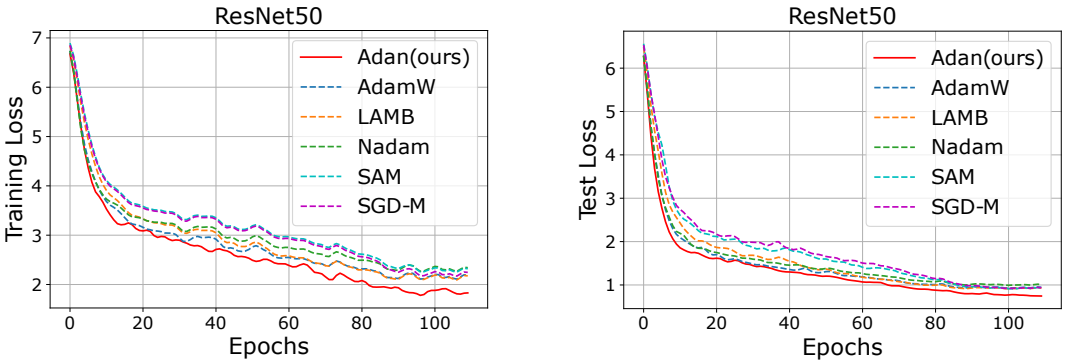
Besides AdamW, we also compare Adan with several other popular optimizers, including Adam, SGD-M, and LAMB, on ViT-S. Table 8 shows that SGD, Adam, and LAMB perform poorly on ViT-S, which is also observed in the works [6, 66]. These results demonstrate that the decoupled weight decay in Adan and AdamW is much more effective than 1) the vanilla weight decay, namely the commonly used ℓ_2 regularization in SGD, and 2) the one without any weight decay, since as shown in Eqn. (8), the decoupled weight decay is a dynamic regularization along the training trajectory and could better regularize the loss. Compared with AdamW, the advantages of Adan mainly come from its faster convergence shown in Figure 2 (b). We will discuss this below.

Self-supervised MAE Training (pretraining + finetuning). We follow the MAE training framework to pretrain and fine tune ViT-B on ImageNet, i.e. 300/800 pretraining epochs and 100 fine-tuning epochs. Table 3 shows that 1) with 300 pretraining epochs, Adan makes 0.5% improvement over AdamW; 2) Adan pretrained 800 epochs surpasses AdamW pretrained 1,600 epochs by non-trial 0.2%. All these results show the superior convergence and generalization performance of Adan.

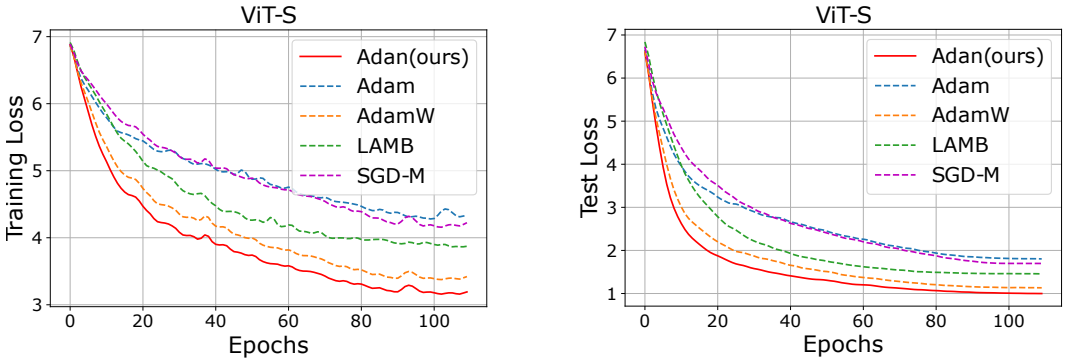
Large-Batch Training. Although large batch size can increase computation parallelism to reduce training time and is heavily desired, optimizers often suffer performance degradation, or even fail. For instance, AdamW fails to train ViTs when batch size is beyond 4,096. How to solve the problem remains open [30]. At present, LAMB is the most effective optimizer for large batch size. Table 7 reveals that Adan is robust to batch sizes from 2k to 32k, and shows higher performance and robustness than LAMB.

D.1.3 Detailed Comparison and Convergence Curve

Besides AdamW, we also compare Adan with several other popular optimizers, including Adam, SGD-M, and LAMB, on ViT-S. Table 8 shows that SGD, Adam, and LAMB perform poorly on ViT-S, which is also observed in the works [6, 66]. These results demonstrate that the decoupled weight decay in Adan and AdamW is much more effective than 1) the vanilla weight decay, namely the commonly used ℓ_2 regularization in SGD, and 2) the one without any weight decay, since as shown in Eqn. (8), the decoupled weight decay is a dynamic regularization along the training trajectory and could better regularize the loss. Compared with AdamW, the advantages of Adan mainly come from its faster convergence shown in Figure 2 (b). We will discuss this below.



(a) Training and test curves on ResNet-50 under Training Setting I.



(b) Training and test curves on ViT-S under Training Setting II.

Figure 2: Training and test curves of various optimizers on ImageNet dataset. Training loss is larger due to its stronger data argumentation.

In Figure 2 (a), we plot the curve of training and test loss along with the training epochs on ResNet50. One can observe that Adan converges faster than the compared baselines and enjoys the smallest training and test losses. This demonstrates its fast convergence property and good generalization ability. To sufficiently investigate the fast convergence of Adan, we further plot the curve of training and test loss on the ViT-Small in Figure 2 (b). From the results, we can see that Adan consistently shows faster convergence behaviors than other baselines in terms of both training loss and test loss. This also partly explains the good performance of Adan over other optimizers.

D.2 Experiments for Natural Language Processing Tasks

D.2.1 Results on Transformer-XL

Here we investigate the performance of Adan on Transformer-XL [12] which is often used to model long sequences. We follow the exact official setting [1] to train Transformer-XL-base on the WikiText-103 dataset that is the largest available word-level language modeling benchmark with long-term dependency. We only replace the default Adam optimizer of Transformer-XL-base by our Adan, and do not make other changes for the hyper-parameter. For Adan, we set $\beta_1 = 0.1$, $\beta_2 = 0.1$, and $\beta_3 = 0.001$, and choose learning rate as 0.001. We test Adan and Adam with several training steps, including 50k, 100k, and 200k (official), and report the results in Table 4.

From Table 4, one can observe that on Transformer-XL-base, Adan surpasses its default Adam optimizer in terms of test PPL (the lower, the better) under all training steps. Surprisingly, Adan using 100k training steps can even achieve comparable results to Adam with 200k training steps. All these results demonstrate the superiority of Adan over the default SoTA Adam optimizer in Transformer-XL.

[1] https://github.com/kimiyoung/transformer-xl

D.2.2 Results on BERT

Similar to the pretraining experiments of MAE which is also a self-supervised learning framework on vision tasks, we utilize Adan to train BERT [13] from scratch, which is one of the most widely used pretraining models/frameworks for NLP tasks. We employ the exact BERT training setting in the widely used codebase—Fairseq [67]. We replace the default Adam optimizer in BERT with our Adan for both pretraining and fune-tuning. Specifically, we first pretrain BERT-base on the Bookcorpus and Wikipedia datasets, and then finetune BERT-base separately for each GLUE task on the corresponding training data. Note, GLUE is a collection of 9 tasks/datasets to evaluate natural language understanding systems, in which the tasks are organized as either single-sentence classification or sentence-pair classification.

Here we simply replace the Adam optimizer in BERT with our Adan and does not make other changes, *e.g.* random seed, warmup steps and learning rate decay strategy, dropout probability, *etc*. For pretraining, we use Adan with its default weight decay (0.02) and βs ($\beta_1 = 0.02, \beta_2 = 0.08$, and $\beta_3 = 0.01$), and choose learning rate as 0.001. For fine-tuning, we consider a limited hyper-parameter sweep for each task, with a batch size of 16, and learning rates $\in \{2e-5, 4e-5\}$ and use Adan with $\beta_1 = 0.02, \beta_2 = 0.01$, and $\beta_3 = 0.01$ and weight decay 0.01. Following the conventional setting, we run each fine-tuning experiment three times and report the median performance in Table 5. On MNLI, we report the mismatched and matched accuracy. The performance of our reproduced one (second row) is slightly better than the vanilla results of BERT reported in Huggingface-transformer [43] (widely used codebase for transformers in NLP), since the vanilla Bookcorpus data in [43] is not available and thus we train on the latest Bookcorpus data version.

From Table 5, one can see that in the most commonly used BERT training experiment, Adan reveals much better advantage over Adam. Specifically, in all GLUE tasks, on BERT-base model, Adan achieves higher performance than Adam, and makes 1.8 average improvements on all tasks. In addition, on some tasks of Adan, BERT-base trained by Adan can outperform some large models. e.g., BERT-large which achieves 70.4% on RTE, 93.2% on SST-2 and 60.6 correlation on CoLA, and XLNet-large which has 63.6 correlation on CoLA. See [68] for more results.

D.3 Results on Reinforcement Learning Tasks

Here we evaluate Adan on reinforcement learning tasks. Specifically, we replace the default Adam optimizer in PPO [44] which is one of the most popular policy gradient method, and do not many any other change in PPO. For brevity, we call this new PPO version "PPO-Adan". Then we test PPO and PPO-Adan on several games which are actually continuous control environments simulated by the standard and widely-used engine, MuJoCo [59]. For these test games, their agents receive a reward at each step. Following standard evaluation, we run each game under 10 different and independent random seeds (i.e. $1 \sim 10$), and test the performance for 10 episodes every 30,000 steps. All these experiments are based on the widely used codebase Tianshou[2] [69]. For fairness, we use the default hyper-parameters in Tianshou, *e.g.* batch size, discount, and GAE parameter. We use Adan with its default βs ($\beta_1 = 0.02, \beta_2 = 0.08$, and $\beta_3 = 0.01$). Following the default setting, we do not adopt the weight decay and choose learning rate as 3e-4.

We report the results on four test games in Figure 3, in which the solid line denotes the averaged episodes rewards in evaluation and the shaded region is its 75% confidence intervals. From Figure 3, one can observe that on the four test games, PPO-Adan achieves much higher rewards than vanilla PPO which uses Adam as its optimizer. These results demonstrate the advantages of Adan over Adam, since PPO-Adan simply replaces the Adam optimizer in PPO with our Adan and does not make other changes.

D.4 Ablation Study

D.4.1 Robustness to in momentum coefficients

Here we choose MAE to investigate the effects of the momentum coefficients (βs) to Adan, since as shown in MAE, its pre-training is actually sensitive to momentum coefficients of AdamW. To this end, following MAE, we pretrain and fine tune ViT-B on ImageNet for 800 pretraining and 100 fine-tuning

[2] https://github.com/thu-ml/tianshou

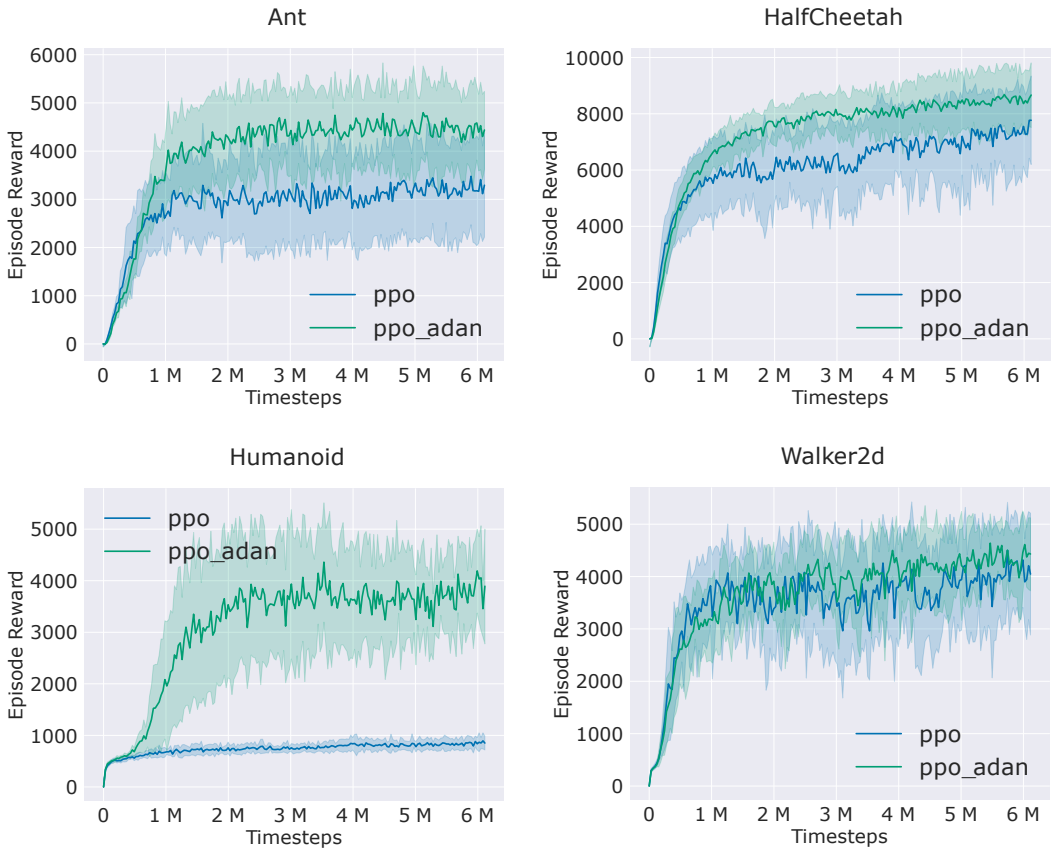


Figure 3: Comparison of PPO and our PPO-Adan on several RL games simulated by MuJoCo. Here PPO-Adan simply replaces the Adam optimizer in PPO with our Adan and does not change others.

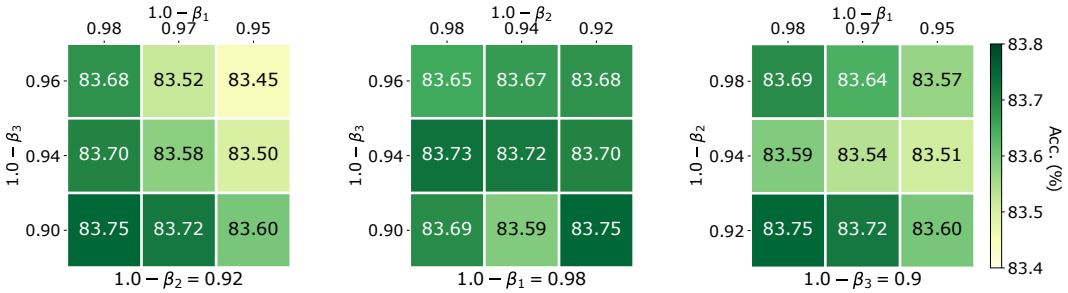


Figure 4: Effects of momentum coefficients $(\beta_1, \beta_2, \beta_3)$ to top-1 accuracy (%) of Adan on ViT-B under MAE training framework (800 pretraining and 100 fine-tuning epochs on ImageNet).

epochs. We also fix one of $(\beta_1, \beta_2, \beta_3)$ and tune others. Figure 4 shows that by only pretraining 800 epochs, Adan achieves $83.7\%+$ in most cases and outperforms the official accuracy 83.6% obtained by AdamW with 1600 pretraining epochs, indicating the robustness of Adan to βs. We also observe 1) Adan is not sensitive to β_2; 2) β_1 has a certain impact on Adan, namely the smaller the $(1.0-\beta_1)$, the worse the accuracy; 3) similar to findings of MAE, a small second-order coefficient $(1.0-\beta_3)$ can improve the accuracy. The smaller the $(1.0-\beta_3)$, the more current landscape information the optimizer would utilize to adjust the coordinate-wise learning rate. Maybe the complex pre-training task of MAE is more preferred to the local geometric information.

E Proof of Lemma 1: equivalence between the AGD and AGD II

In this section, we show how to get AGD II from AGD. For convenience, we omit the noise term $\boldsymbol{\zeta}_k$. Note that, let $\alpha \coloneqq 1-\beta_1$:

$$\text{AGD:} \begin{cases} \mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k - \eta\alpha\mathbf{m}_{k-1}) \\ \mathbf{m}_k = \alpha\mathbf{m}_{k-1} + \mathbf{g}_k \\ \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\mathbf{m}_k \end{cases}.$$

We can get:

$$\begin{aligned} &\boldsymbol{\theta}_{k+1} - \eta\alpha\mathbf{m}_k = \boldsymbol{\theta}_k - \eta\mathbf{m}_k - \eta\alpha\mathbf{m}_k \\ =&\boldsymbol{\theta}_k - \eta(1+\alpha)(\alpha\mathbf{m}_{k-1} + \nabla f(\boldsymbol{\theta}_k - \eta\alpha\mathbf{m}_{k-1})) \\ =&\boldsymbol{\theta}_k - \eta\alpha\mathbf{m}_{k-1} - \eta\alpha^2\mathbf{m}_{k-1} - \eta(1+\alpha)(\nabla f(\boldsymbol{\theta}_k - \eta\alpha\mathbf{m}_{k-1})). \end{aligned} \tag{9}$$

Let

$$\begin{cases} \bar{\boldsymbol{\theta}}_{k+1} \coloneqq \boldsymbol{\theta}_{k+1} - \eta\alpha\mathbf{m}_k, \\ \bar{\mathbf{m}}_k \coloneqq \alpha^2\mathbf{m}_{k-1} + (1+\alpha)\nabla f(\boldsymbol{\theta}_k - \eta\alpha\mathbf{m}_{k-1}) = \alpha^2\mathbf{m}_{k-1} + (1+\alpha)\nabla f(\bar{\boldsymbol{\theta}}_k) \end{cases}$$

Then, by Eq.(9), we have:

$$\bar{\boldsymbol{\theta}}_{k+1} = \bar{\boldsymbol{\theta}}_k - \eta\bar{\mathbf{m}}_k. \tag{10}$$

On the other hand, we have $\bar{\mathbf{m}}_{k-1} = \alpha^2\mathbf{m}_{k-2} + (1+\alpha)\nabla f(\bar{\boldsymbol{\theta}}_{k-1})$ and :

$$\begin{aligned} \bar{\mathbf{m}}_k - \alpha\bar{\mathbf{m}}_{k-1} &= \alpha^2\mathbf{m}_{k-1} + (1+\alpha)\nabla f(\bar{\boldsymbol{\theta}}_k) - \alpha\bar{\mathbf{m}}_{k-1} \\ &= (1+\alpha)\nabla f(\bar{\boldsymbol{\theta}}_k) + \alpha^2\big(\alpha\mathbf{m}_{k-2} + \nabla f(\bar{\boldsymbol{\theta}}_{k-1})\big) - \alpha\bar{\mathbf{m}}_{k-1} \\ &= (1+\alpha)\nabla f(\bar{\boldsymbol{\theta}}_k) + \alpha\big(\alpha^2\mathbf{m}_{k-2} + \alpha\nabla f(\bar{\boldsymbol{\theta}}_{k-1}) - \bar{\mathbf{m}}_{k-1}\big) \\ &= (1+\alpha)\nabla f(\bar{\boldsymbol{\theta}}_k) + \alpha\big(\alpha^2\mathbf{m}_{k-2} + \alpha\nabla f(\bar{\boldsymbol{\theta}}_{k-1})\big) - \alpha\bar{\mathbf{m}}_{k-1} \\ &= (1+\alpha)\nabla f(\bar{\boldsymbol{\theta}}_k) - \alpha\nabla f(\bar{\boldsymbol{\theta}}_{k-1}) \\ &= \nabla f(\bar{\boldsymbol{\theta}}_k) + \alpha\big(\nabla f(\bar{\boldsymbol{\theta}}_k) - \nabla f(\bar{\boldsymbol{\theta}}_{k-1})\big). \end{aligned} \tag{11}$$

Finally, due to Eq.(10) and Eq.11, we have:

$$\begin{cases} \bar{\mathbf{m}}_k = \alpha\bar{\mathbf{m}}_{k-1} + \Big(\nabla f(\bar{\boldsymbol{\theta}}_k) + \alpha\big(\nabla f(\bar{\boldsymbol{\theta}}_k) - \nabla f(\bar{\boldsymbol{\theta}}_{k-1})\big)\Big) \\ \bar{\boldsymbol{\theta}}_{k+1} = \bar{\boldsymbol{\theta}}_k - \eta\bar{\mathbf{m}}_k \end{cases}$$

F Proof of Theorem 1

Before starting the proof, we first provide several notations. Let $F_k(\boldsymbol{\theta}) \coloneqq E_{\boldsymbol{\zeta}}[f(\boldsymbol{\theta}, \boldsymbol{\zeta})] + \frac{\lambda_k}{2}\|\boldsymbol{\theta}\|^2_{\sqrt{\mathbf{n}_k}}$ and $\mu \coloneqq \beta_3 c_\infty^2/\varepsilon = \mathcal{O}\big(\epsilon^4\big)$,

$$\|\mathbf{x}\|^2_{\sqrt{\mathbf{n}_k}} \coloneqq \big\langle \mathbf{x}, \sqrt{\mathbf{n}_k + \varepsilon} \circ \mathbf{x} \big\rangle, \quad \lambda_k = \lambda(1-\mu)^k.$$

Moreover, we let

$$\tilde{\boldsymbol{\theta}}_k \coloneqq \sqrt{\mathbf{n}_k + \varepsilon} \circ \boldsymbol{\theta}_k.$$

Lemma 2. *Assume $f(\cdot)$ is L-smooth. For*

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

With $\eta \le \min\{\frac{1}{3L/\sqrt{\varepsilon}}, \frac{1}{10\lambda}\}$, *then we have:*

$$F_{k+1}(\boldsymbol{\theta}_{k+1}) \le F_k(\boldsymbol{\theta}_k) - \frac{\eta}{4c_\infty}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2 + \frac{\eta}{2\sqrt{\varepsilon}}\|\mathbf{g}_k - \mathbf{u}_k\|^2,$$

where $\mathbf{g}_k \coloneqq \nabla f(\boldsymbol{\theta}_k)$.

Proof. We denote $\mathbf{p}_k \coloneqq \mathbf{u}_k/\sqrt{\mathbf{n}_k + \varepsilon}$. By the optimality condition of $\boldsymbol{\theta}_{k+1}$, we have

$$\lambda_k\boldsymbol{\theta}_k + \mathbf{p}_k = \frac{\lambda_k\tilde{\boldsymbol{\theta}}_k + \mathbf{u}_k}{\sqrt{\mathbf{n}_k + \varepsilon}} = \frac{1 + \eta\lambda_k}{\eta}(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k+1}). \tag{12}$$

Then for $\eta \le \frac{1}{3L/\sqrt{\varepsilon}}$, we have:

$$\begin{aligned}
&F_{k+1}(\boldsymbol{\theta}_{k+1}) \le f(\boldsymbol{\theta}_k) + \langle\nabla f(\boldsymbol{\theta}_k), \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\rangle + \frac{L}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 + \frac{\lambda_{k+1}}{2}\|\boldsymbol{\theta}_{k+1}\|^2_{\sqrt{\mathbf{n}_{k+1}}} \\
\overset{(a)}{\le}& f(\boldsymbol{\theta}_k) + \langle\nabla f(\boldsymbol{\theta}_k), \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\rangle + \frac{L}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 + \frac{\lambda_k}{2}\|\boldsymbol{\theta}_{k+1}\|^2_{\sqrt{\mathbf{n}_k}} \\
\overset{(b)}{\le}& F_k(\boldsymbol{\theta}_k) + \left\langle\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k, \lambda_k\boldsymbol{\theta}_k + \frac{\mathbf{g}_k}{\sqrt{\mathbf{n}_k + \varepsilon}}\right\rangle_{\sqrt{\mathbf{n}_k}} + \frac{L/\sqrt{\varepsilon} + \lambda_k}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} \\
=& F_k(\boldsymbol{\theta}_k) + \frac{L/\sqrt{\varepsilon} + \lambda_k}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} + \left\langle\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k, \lambda_k\boldsymbol{\theta}_k + \mathbf{p}_k + \frac{\mathbf{g}_k - \mathbf{u}_k}{\sqrt{\mathbf{n}_k + \varepsilon}}\right\rangle_{\sqrt{\mathbf{n}_k}} \\
\overset{(c)}{=}& F_k(\boldsymbol{\theta}_k) + \left(\frac{L/\sqrt{\varepsilon} + \lambda_k}{2} - \frac{1 + \eta\lambda_k}{\eta}\right)\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} + \left\langle\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k, \frac{\mathbf{g}_k - \mathbf{u}_k}{\sqrt{\mathbf{n}_k + \varepsilon}}\right\rangle_{\sqrt{\mathbf{n}_k}} \\
\overset{(d)}{\le}& F_k(\boldsymbol{\theta}_k) + \left(\frac{L/\sqrt{\varepsilon}}{2} - \frac{1}{\eta}\right)\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} + \frac{1}{2\eta}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} + \frac{\eta}{2\sqrt{\varepsilon}}\|\mathbf{g}_k - \mathbf{u}_k\|^2 \\
\le& F_k(\boldsymbol{\theta}_k) - \frac{1}{3\eta}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} + \frac{\eta}{2\sqrt{\varepsilon}}\|\mathbf{g}_k - \mathbf{u}_k\|^2 \\
\le& F_k(\boldsymbol{\theta}_k) - \frac{\eta}{4c_\infty}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2 + \frac{\eta}{2\sqrt{\varepsilon}}\|\mathbf{g}_k - \mathbf{u}_k\|^2,
\end{aligned}$$

where (a) comes from the fact $\lambda_{k+1}(1-\mu)^{-1} = \lambda_k$ and Proposition 1:

$$\left(\frac{\sqrt{\mathbf{n}_k + \varepsilon}}{\sqrt{\mathbf{n}_{k+1} + \varepsilon}}\right)_i \ge 1 - \mu,$$

which implies:

$$\lambda_{k+1}\|\boldsymbol{\theta}_{k+1}\|^2_{\sqrt{\mathbf{n}_{k+1}}} \le \frac{\lambda_{k+1}}{1-\mu}\|\boldsymbol{\theta}_{k+1}\|^2_{\sqrt{\mathbf{n}_k}} = \lambda_k\|\boldsymbol{\theta}_{k+1}\|^2_{\sqrt{\mathbf{n}_k}},$$

and (b) is from:

$$\|\boldsymbol{\theta}_{k+1}\|^2_{\sqrt{\mathbf{n}_k}} = \left(\|\boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} + 2\left\langle\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k, \boldsymbol{\theta}_k\right\rangle_{\sqrt{\mathbf{n}_k}} + \|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}}\right),$$

(c) is due to Eqn. (12), and for (d), we utilize:

$$\left\langle\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k, \frac{\mathbf{g}_k - \mathbf{u}_k}{\sqrt{\mathbf{n}_k + \varepsilon}}\right\rangle_{\sqrt{\mathbf{n}_k}} \le \frac{1}{2\eta}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2_{\sqrt{\mathbf{n}_k}} + \frac{\eta}{2\sqrt{\varepsilon}}\|\mathbf{g}_k - \mathbf{u}_k\|^2,$$

the last inequality comes from the fact in Eqn. (12) and $\eta \le \frac{1}{10\lambda}$, such that:

$$\frac{1}{3\eta}\|(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)\|^2_{\sqrt{\mathbf{n}_k}} = \frac{\eta}{3\sqrt{\mathbf{n}_k + \varepsilon}(1 + \eta\lambda_k)}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2 \ge \frac{\eta}{4c_\infty}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2.$$

□

Theorem 1. *Suppose Assumptions 1 and 2 hold. Let* $c_l \coloneqq \frac{1}{c_\infty}$ *and* $c_u \coloneqq \frac{1}{\sqrt{\varepsilon}}$. *With* $\beta_3 c_\infty^2/\varepsilon \ll 1$,

$$\eta^2 \leq \frac{c_l \beta_1^2}{8 c_u^3 L^2}, \quad \max\{\beta_1, \beta_2\} \leq \frac{c_l \epsilon^2}{96 c_u \sigma^2}, \quad T \geq \max\left\{\frac{24\Delta_0}{\eta c_l \epsilon^2}, \frac{24 c_u \sigma^2}{\beta_1 c_l \epsilon^2}\right\},$$

where $\Delta_0 \coloneqq F(\boldsymbol{\theta}_0) - f^*$ *and* $f^* \coloneqq \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\zeta}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})]$, *then we let* $\mathbf{u}_k \coloneqq \mathbf{m}_k + (1-\beta_1)\mathbf{v}_k$ *and have:*

$$\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}\left(\left\|\mathbf{u}_k + \lambda_k \tilde{\boldsymbol{\theta}}_k\right\|^2\right) \leq \epsilon^2,$$

and

$$\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}\left(\left\|\mathbf{m}_k - \mathbf{g}_k^{full}\right\|^2\right) \leq \frac{\epsilon^2}{4}, \quad \frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}\left(\|\mathbf{v}_k\|^2\right) \leq \frac{\epsilon^2}{4}.$$

Hence, we have:

$$\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}\left(\left\|\nabla_{\boldsymbol{\theta}_k}\left(\frac{\lambda_k}{2}\|\boldsymbol{\theta}\|^2_{\sqrt{\mathbf{n}_k}} + \mathbb{E}_{\boldsymbol{\zeta}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})]\right)\right\|^2\right) \leq 4\epsilon^2.$$

Proof. For convince, we let $\mathbf{u}_k \coloneqq \mathbf{m}_k + (1-\beta_1)\mathbf{v}_k$ and $\mathbf{g}_k^{full} \coloneqq \mathbb{E}_{\boldsymbol{\zeta}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})]$. We have:

$$\left\|\mathbf{u}_k - \mathbf{g}_k^{full}\right\|^2 \leq 2\left\|\mathbf{m}_k - \mathbf{g}_k^{full}\right\|^2 + 2(1-\beta_1)^2\|\mathbf{v}_k\|^2.$$

By Lemma 2, Lemma 3, and Lemma 4, we already have:

$$F_{k+1}(\boldsymbol{\theta}_{k+1}) \leq F_k(\boldsymbol{\theta}_k) - \frac{\eta c_l}{4}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2 + \eta c_u\left\|\mathbf{g}_k^{full} - \mathbf{m}_k\right\|^2 + \eta c_u(1-\beta_1)^2\|\mathbf{v}_k\|^2, \tag{13}$$

$$\mathbb{E}\left(\left\|\mathbf{m}_{k+1} - \mathbf{g}_{k+1}^{full}\right\|^2\right) \leq (1-\beta_1)\mathbb{E}\left(\left\|\mathbf{m}_k - \mathbf{g}_k^{full}\right\|^2\right) + \frac{(1-\beta_1)^2 L^2}{\beta_1}\mathbb{E}\left(\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2\right) + \beta_1^2\sigma^2 \tag{14}$$

$$\mathbb{E}\left(\|\mathbf{v}_{k+1}\|^2\right) \leq (1-\beta_2)\mathbb{E}\left(\|\mathbf{v}_k\|^2\right) + 2\beta_2\mathbb{E}\left(\left\|\mathbf{g}_{k+1}^{full} - \mathbf{g}_k^{full}\right\|^2\right) + 3\beta_2^2\sigma^2 \tag{15}$$

Then by adding Eq.(13) with $\frac{\eta c_u}{\beta_1}\times$ Eq.(14) and $\frac{\eta c_u(1-\beta_1)^2}{\beta_2}\times$ Eq.(15), we can get:

$$\begin{aligned}
\mathbb{E}(\Phi_{k+1}) \leq& \mathbb{E}\left(\Phi_k - \frac{\eta c_l}{4}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2 + \frac{\eta c_u}{\beta_1}\left(\frac{(1-\beta_1)^2 L^2}{\beta_1}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 + \beta_1^2\sigma^2\right)\right)\\
&+ \frac{\eta c_u(1-\beta_1)^2}{\beta_2}\mathbb{E}\left(2\beta_2 L^2\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 + 3\beta_2^2\sigma^2\right)\\
\leq& \mathbb{E}\left(\Phi_k - \frac{\eta c_l}{4}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2 + \eta c_u L^2\left(\frac{(1-\beta_1)^2}{\beta_1^2} + 2(1-\beta_1)^2\right)\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2\right)\\
&+ (\beta_1 + 3\beta_2)\eta c_u\sigma^2\\
\overset{(a)}{\leq}& \mathbb{E}\left(\Phi_k - \frac{\eta c_l}{4}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2 + \frac{\eta c_u L^2}{\beta_1^2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2\right) + 4\beta_m\eta c_u\sigma^2\\
\overset{(b)}{\leq}& \mathbb{E}\left(\Phi_k + \left(\frac{(\eta c_u)^3 L^2}{\beta_1^2} - \frac{\eta c_l}{4}\right)\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2\right) + 4\beta_m\eta c_u\sigma^2\\
\leq& \mathbb{E}\left(\Phi_k - \frac{\eta c_l}{8}\left\|\mathbf{u}_k + \lambda_k\tilde{\boldsymbol{\theta}}_k\right\|^2\right) + 4\beta_m\eta c_u\sigma^2,
\end{aligned}$$

where we let:

$$\Phi_k \coloneqq F_k(\boldsymbol{\theta}_k) - f^* + \frac{\eta c_u}{\beta_1}\left\|\mathbf{m}_k - \mathbf{g}_k^{full}\right\|^2 + \frac{\eta c_u(1-\beta_1)^2}{\beta_2}\|\mathbf{v}_k\|^2,$$

$$\beta_m = \max\{\beta_1, \beta_2\} \leq \frac{2}{3}, \quad \eta \leq \frac{c_l\beta_1^2}{8c_u^3 L^2},$$

and for (a), when $\beta_1 \leq \frac{2}{3}$, we have:

$$\frac{(1-\beta_1)^2}{\beta_1^2} + 2(1-\beta_1)^2 < \frac{1}{\beta_1^2},$$

and (b) is due to Eq.(12) from Lemma 2. And hence, we have:

$$\sum_{k=0}^{T} \mathbb{E}(\Phi_{k+1}) \leq \sum_{k=0}^{T} \mathbb{E}(\Phi_k) - \frac{\eta c_l}{8} \sum_{k=0}^{T} \left\| \mathbf{u}_k + \lambda_k \tilde{\boldsymbol{\theta}}_k \right\|^2 + (T+1) 4\eta c_u \beta_m \sigma^2.$$

Hence, we can get:

$$\frac{1}{T+1} \sum_{k=0}^{T} \mathbb{E}\left(\left\| \mathbf{u}_k + \lambda_k \tilde{\boldsymbol{\theta}}_k \right\|^2 \right) \leq \frac{8\Phi_0}{\eta c_l T} + \frac{32 c_u \beta \sigma^2}{c_l} = \frac{8\Delta_0}{\eta c_l T} + \frac{8 c_u \sigma^2}{\beta_1 c_l T} + \frac{32 c_u \beta_m \sigma^2}{c_l} \leq \epsilon^2,$$

where

$$\Delta_0 := F(\boldsymbol{\theta}_0) - f^*, \quad \beta_m \leq \frac{c_l \epsilon^2}{96 c_u \sigma^2}, \quad T \geq \max\left\{ \frac{24\Delta_0}{\eta c_l \epsilon^2}, \frac{24 c_u \sigma^2}{\beta_1 c_l \epsilon^2} \right\}.$$

We finish the first part of the theorem. From Eq.(14), we can conclude that:

$$\frac{1}{T+1} \sum_{k=0}^{T} \mathbb{E}\left(\left\| \mathbf{m}_k - \mathbf{g}_k^{full} \right\|^2 \right) \leq \frac{\sigma^2}{\beta T} + \frac{L^2 \eta^2 c_u^2 \epsilon^2}{\beta_1^2} + \beta_1 \sigma^2 < \frac{\epsilon^2}{4}.$$

From Eq.(15), we can conclude that:

$$\frac{1}{T+1} \sum_{k=0}^{T} \mathbb{E}\left(\left\| \mathbf{v}_k \right\|^2 \right) \leq 2L^2 \eta^2 c_u^2 \epsilon^2 + 3\beta_2 \sigma^2 < \frac{\epsilon^2}{4}.$$

Finally we have:

$$\begin{aligned}
&\frac{1}{T+1} \sum_{k=0}^{T} \mathbb{E}\left(\left\| \nabla_{\boldsymbol{\theta}_k} \left(\frac{\lambda_k}{2} \|\boldsymbol{\theta}\|^2_{\sqrt{\mathbf{n}_k}} + \mathbb{E}_{\boldsymbol{\zeta}}[f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})] \right) \right\|^2 \right) \\
\leq &\frac{1}{T+1} \left(\sum_{k=0}^{T} \mathbb{E}\left(2 \left\| \mathbf{u}_k + \lambda_k \tilde{\boldsymbol{\theta}}_k \right\|^2 + 4 \left\| \mathbf{m}_k - \mathbf{g}_k^{full} \right\|^2 + 4 \|\mathbf{v}_k\|^2 \right) \right) \leq 4\epsilon^2.
\end{aligned}$$

Now, we have finished the proof. □

G Auxiliary Lemmas

Proposition 1. *If Assumption 2 holds. Given $\beta_3 \leq \frac{\varepsilon}{c_\infty^2}$, we have:*

$$\left\| \frac{\boldsymbol{\eta}_k - \boldsymbol{\eta}_{k-1}}{\boldsymbol{\eta}_{k-1}} \right\|_\infty \leq \frac{\beta_3 c_\infty^2}{\varepsilon}.$$

Proof. Note that, give any index $i \in [d]$:

$$\left| \left(\frac{\boldsymbol{\eta}_k - \boldsymbol{\eta}_{k-1}}{\boldsymbol{\eta}_{k-1}} \right)_i \right| = \left| \left(\frac{\sqrt{\mathbf{n}_{k-1} + \varepsilon}}{\sqrt{\mathbf{n}_k + \varepsilon}} \right)_i - 1 \right|.$$

By the definition of $\mathbf{n}_k$, we have:

$$\begin{aligned}
&\left(\frac{\mathbf{n}_{k-1} + \varepsilon}{\mathbf{n}_k + \varepsilon} \right)_i = 1 + \left(\frac{\mathbf{n}_{k-1} - \mathbf{n}_k}{\mathbf{n}_k + \varepsilon} \right)_i \\
=&1 + \beta_3 \left(\frac{\mathbf{n}_{k-1} - (\mathbf{g}_k + (1-\beta_2)(\mathbf{g}_k - \mathbf{g}_{k-1}))^2}{\mathbf{n}_k + \varepsilon} \right)_i \in \left[1 - \frac{\beta_3 c_\infty^2}{\varepsilon}, 1 + \frac{\beta_3 c_\infty^2}{\varepsilon} \right],
\end{aligned}$$

hence, we have:

$$\left(\frac{\sqrt{\mathbf{n}_{k-1}+\varepsilon}}{\sqrt{\mathbf{n}_k+\varepsilon}}\right)_i \in \left[\sqrt{1-\frac{\beta_3 c_\infty^2}{\varepsilon}}, \sqrt{1+\frac{\beta_3 c_\infty^2}{\varepsilon}}\right] \subset \left[1-\frac{\beta_3 c_\infty^2}{\varepsilon}, 1+\frac{\beta_3 c_\infty^2}{\varepsilon}\right],$$

where we utilize the fact $\sqrt{1-\frac{\beta_3 c_\infty^2}{\varepsilon}} \geq 1-\frac{\beta_3 c_\infty^2}{\varepsilon}$ when $\beta_3 \leq \frac{\sqrt{\varepsilon}}{c_\infty}$. In conclusion, we get:

$$\left|\left(\frac{\boldsymbol{\eta}_k-\boldsymbol{\eta}_{k-1}}{\boldsymbol{\eta}_{k-1}}\right)_i\right| \in \left[0, \frac{\beta_3 c_\infty^2}{\varepsilon}\right].$$

We finish the proof. □

Lemma 3. *Consider a moving average sequence:*

$$\mathbf{m}_k = (1-\beta)\mathbf{m}_{k-1} + \beta \mathbf{g}_k,$$

where we note that:

$$\mathbf{g}_k = \mathbb{E}_{\boldsymbol{\zeta}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})] + \boldsymbol{\xi}_k,$$

and we denote $\mathbf{g}_k^{full} \coloneqq E_{\boldsymbol{\zeta}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})]$ *for convenience. Then we have:*

$$\mathbb{E}\left(\left\|\mathbf{m}_k - \mathbf{g}_k^{full}\right\|^2\right) \leq (1-\beta)\mathbb{E}\left(\left\|\mathbf{m}_{k-1} - \mathbf{g}_{k-1}^{full}\right\|^2\right) + \frac{(1-\beta)^2 L^2}{\beta}\mathbb{E}\left(\left\|\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k\right\|^2\right) + \beta^2\sigma^2.$$

Proof. Note that, we have:

$$\begin{aligned}
\mathbf{m}_k - \mathbf{g}_k^{full} =& (1-\beta)\left(\mathbf{m}_{k-1} - \mathbf{g}_{k-1}^{full}\right) + (1-\beta)\mathbf{g}_{k-1}^{full} - \mathbf{g}_k^{full} + \beta\mathbf{g}_k \\
=& (1-\beta)\left(\mathbf{m}_{k-1} - \mathbf{g}_{k-1}^{full}\right) + (1-\beta)\left(\mathbf{g}_{k-1}^{full} - \mathbf{g}_k^{full}\right) + \beta\left(\mathbf{g}_k - \mathbf{g}_k^{full}\right).
\end{aligned}$$

Then, take expectation on both sides:

$$\begin{aligned}
&\mathbb{E}\left(\left\|\mathbf{m}_k - \mathbf{g}_k^{full}\right\|^2\right) \\
=&(1-\beta)^2\mathbb{E}\left(\left\|\mathbf{m}_{k-1} - \mathbf{g}_{k-1}^{full}\right\|^2\right) + (1-\beta)^2\mathbb{E}\left(\left\|\mathbf{g}_{k-1}^{full} - \mathbf{g}_k^{full}\right\|^2\right) + \beta^2\sigma^2 + \\
&2(1-\beta)^2\mathbb{E}\left(\left\langle \mathbf{m}_{k-1} - \mathbf{g}_{k-1}^{full}, \mathbf{g}_{k-1}^{full} - \mathbf{g}_k^{full}\right\rangle\right) \\
\leq&\left((1-\beta)^2 + (1-\beta)^2 a\right)\mathbb{E}\left(\left\|\mathbf{m}_{k-1} - \mathbf{g}_{k-1}^{full}\right\|^2\right) + \\
&\left(1+\frac{1}{a}\right)(1-\beta)^2\mathbb{E}\left(\left\|\mathbf{g}_{k-1}^{full} - \mathbf{g}_k^{full}\right\|^2\right) + \beta^2\sigma^2 \\
\overset{(a)}{\leq}&(1-\beta)\mathbb{E}\left(\left\|\mathbf{m}_{k-1} - \mathbf{g}_{k-1}^{full}\right\|^2\right) + \frac{(1-\beta)^2}{\beta}\mathbb{E}\left(\left\|\mathbf{g}_{k-1}^{full} - \mathbf{g}_k^{full}\right\|^2\right) + \beta^2\sigma^2 \\
\leq&(1-\beta)\mathbb{E}\left(\left\|\mathbf{m}_{k-1} - \mathbf{g}_{k-1}^{full}\right\|^2\right) + \frac{(1-\beta)^2 L^2}{\beta}\mathbb{E}\left(\left\|\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k\right\|^2\right) + \beta^2\sigma^2,
\end{aligned}$$

where for (a), we set $a = \frac{\beta}{1-\beta}$. □

Lemma 4. *Consider a moving average sequence:*

$$\mathbf{v}_k = (1-\beta)\mathbf{v}_{k-1} + \beta(\mathbf{g}_k - \mathbf{g}_{k-1}),$$

where we note that:

$$\mathbf{g}_k = \mathbb{E}_{\boldsymbol{\zeta}}[\nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})] + \boldsymbol{\xi}_k,$$

and we denote $\mathbf{g}_k^{full} \coloneqq E_{\boldsymbol{\zeta}}[f(\boldsymbol{\theta}_k, \boldsymbol{\zeta})]$ *for convenience. Then we have:*

$$\mathbb{E}\left(\left\|\mathbf{v}_k\right\|^2\right) \leq (1-\beta)\mathbb{E}\left(\left\|\mathbf{v}_{k-1}\right\|^2\right) + 2\beta\mathbb{E}\left(\left\|\mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\right\|^2\right) + 3\beta^2\sigma^2.$$

Proof. Take expectation on both sides:

$$\begin{aligned}
&\mathbb{E}\Big(\|\mathbf{v}_k\|^2\Big) = (1-\beta)^2\mathbb{E}\Big(\|\mathbf{v}_{k-1}\|^2\Big) + \beta^2\mathbb{E}\Big(\|\mathbf{g}_k - \mathbf{g}_{k-1}\|^2\Big) + 2\beta(1-\beta)\mathbb{E}(\langle \mathbf{v}_{k-1}, \mathbf{g}_k - \mathbf{g}_{k-1}\rangle)\\
\overset{(a)}{=}&(1-\beta)^2\mathbb{E}\Big(\|\mathbf{v}_{k-1}\|^2\Big) + \beta^2\mathbb{E}\Big(\|\mathbf{g}_k - \mathbf{g}_{k-1}\|^2\Big) + 2\beta(1-\beta)\mathbb{E}\Big(\Big\langle \mathbf{v}_{k-1}, \mathbf{g}_k^{full} - \mathbf{g}_{k-1}\Big\rangle\Big)\\
\overset{(b)}{\leq}&(1-\beta)^2\mathbb{E}\Big(\|\mathbf{v}_{k-1}\|^2\Big) + 2\beta^2\mathbb{E}\Big(\Big\|\mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\Big\|^2\Big) + 2\beta(1-\beta)\mathbb{E}\Big(\Big\langle \mathbf{v}_{k-1}, \mathbf{g}_k^{full} - \mathbf{g}_{k-1}\Big\rangle\Big) + 3\beta^2\sigma^2\\
\overset{(c)}{\leq}&(1-\beta)^2\mathbb{E}\Big(\|\mathbf{v}_{k-1}\|^2\Big) + 2\beta^2\mathbb{E}\Big(\Big\|\mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\Big\|^2\Big) + 2\beta(1-\beta)\mathbb{E}\Big(\Big\langle \mathbf{v}_{k-1}, \mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\Big\rangle\Big) + 3\beta^2\sigma^2\\
\overset{(d)}{\leq}&(1-\beta)\mathbb{E}\Big(\|\mathbf{v}_{k-1}\|^2\Big) + 2\beta\mathbb{E}\Big(\Big\|\mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\Big\|^2\Big) + 3\beta^2\sigma^2,
\end{aligned}$$

where for (a), we utilize the independence between $\mathbf{g}_k$ and $\mathbf{v}_{k-1}$, while for (b):

$$\mathbb{E}\Big(\|\mathbf{g}_k - \mathbf{g}_{k-1}\|^2\Big) \leq \mathbb{E}\Big(\Big\|\mathbf{g}_k - \mathbf{g}_k^{full}\Big\|^2\Big) + 2\mathbb{E}\Big(\Big\|\mathbf{g}_{k-1}^{full} - \mathbf{g}_{k-1}\Big\|^2\Big) + 2\mathbb{E}\Big(\Big\|\mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\Big\|^2\Big),$$

for (c), we know:

$$\begin{aligned}
&\mathbb{E}\Big(\Big\langle \mathbf{v}_{k-1}, \mathbf{g}_{k-1}^{full} - \mathbf{g}_{k-1}\Big\rangle\Big) = \mathbb{E}\Big(\Big\langle (1-\beta)\mathbf{v}_{k-2} + \beta(\mathbf{g}_{k-1} - \mathbf{g}_{k-2}), \mathbf{g}_{k-1}^{full} - \mathbf{g}_{k-1}\Big\rangle\Big)\\
=&\mathbb{E}\Big(\Big\langle (1-\beta)\mathbf{v}_{k-2} - \beta\mathbf{g}_{k-2}, \mathbf{g}_{k-1}^{full} - \mathbf{g}_{k-1}\Big\rangle\Big) + \beta\mathbb{E}\Big(\Big\langle \mathbf{g}_{k-1} - \mathbf{g}_{k-1}^{full} + \mathbf{g}_{k-1}^{full}, \mathbf{g}_{k-1}^{full} - \mathbf{g}_{k-1}\Big\rangle\Big)\\
=&-\beta\mathbb{E}\Big(\Big\|\mathbf{g}_{k-1}^{full} - \mathbf{g}_{k-1}\Big\|^2\Big),
\end{aligned}$$

and thus $\mathbb{E}\Big(\Big\langle \mathbf{v}_{k-1}, \mathbf{g}_k^{full} - \mathbf{g}_{k-1}\Big\rangle\Big) = \mathbb{E}\Big(\Big\langle \mathbf{v}_{k-1}, \mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\Big\rangle\Big) - \beta\mathbb{E}\Big(\Big\|\mathbf{g}_{k-1}^{full} - \mathbf{g}_{k-1}\Big\|^2\Big)$. Finally, for (d), we use:

$$2\mathbb{E}\Big(\Big\langle \mathbf{v}_{k-1}, \mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\Big\rangle\Big) \leq \mathbb{E}\Big(\|\mathbf{v}_{k-1}\|^2\Big) + \mathbb{E}\Big(\Big\|\mathbf{g}_k^{full} - \mathbf{g}_{k-1}^{full}\Big\|^2\Big).$$

□