WIN: WEIGHT-DECAY-INTEGRATED NESTEROV ACCELERATION FOR ADAPTIVE GRADIENT ALGORITHMS

Anonymous authors
Paper under double-blind review

ABSTRACT

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

1 INTRODUCTION

Deep neural networks (DNNs) are effective to model realistic data and have been successfully applied to many applications, *e.g.* image classification (He et al., 2016) and speech recognition (Sainath et al., 2013). Typically, their training models can be formulated as a nonconvex problem:

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

where $\boldsymbol{z} \in \mathbb{R}^d$ is the model parameter; sample $\boldsymbol{\zeta}$ is drawn from a data distribution $\mathcal{D}$; the loss f is differentiable; λ is a constant. Though many algorithms, *e.g.* gradient descent (Cauchy et al., 1847) and variance-reduced algorithms (Johnson & Zhang, 2013), can solve problem (1), SGD (Robbins & Monro, 1951) uses the compositional structure in (1) to efficiently estimate gradient via minibatch data, and has become a dominant algorithm to train DNNs in practice because of its higher efficiency and effectiveness. However, on sparse data or ill-conditioned problems, SGD suffers from slow convergence speed (Kingma & Ba, 2014), as it scales the gradient uniformly in all parameter coordinate and ignores the problem properties on each coordinate. To resolve this issue, recent work has proposed a variety of adaptive methods, *e.g.* Adam (Kingma & Ba, 2014) and AdamW (Loshchilov & Hutter, 2018), that scale each gradient coordinate according to the current geometry curvature of the loss $F(\boldsymbol{z})$. This coordinate-wise scaling greatly accelerates the optimization convergence and helps them, *e.g.* Adam and AdamW, become more popular in DNN training, especially for transformers.

Unfortunately, along with the increasing scale of both datasets and models, efficient DNN training even with SGD or adaptive algorithms has become very challenging. In this work, we are particularly

interested in the problem of "*how to accelerate the convergence of adaptive algorithms in a general manner*" because of their dominant popularity across many DNNs. Heavy ball acceleration (Polyak, 1964) and Nesterov acceleration (Nesterov, 2003) are widely used in SGD but are rarely studied in adaptive algorithms. Among the very few, NAdam (Dozat, 2016) simplifies Nesterov acceleration to estimate the first moment of gradient in Adam while totally ignoring the second-order moments, which is not exact Nesterov acceleration and may not inherit its full acceleration merit.

Contributions: In this work, based on a recent Nesterov-type acceleration formulation (Nesterov et al., 2018) and proximal point method (PPM) (Moreau, 1965), we propose a new *Weight-decay-Integrated Nesterov acceleration* (Win for short) to accelerate adaptive algorithms, and also further analyze the convergence of Win-accelerated adaptive algorithms to justify their convergence superiority by taking AdamW and Adam as examples. Our main contributions are highlighted below.

Firstly, we use PPM to rigorously derive our Win acceleration for accelerating adaptive algorithms. By taking AdamW and Adam as examples, at the k-iteration, we follow PPM spirit and minimize a dynamically regularized loss $F(\boldsymbol{z})+\frac{1}{2\eta_k}\|\boldsymbol{z}-\boldsymbol{x}_k\|^2_{\sqrt{\boldsymbol{v}_k+\nu}}$ with the second-order gradient moment $\boldsymbol{v}_k$ and the stabilizing constant ν in AdamW and Adam. Then to introduce Nesterov-alike acceleration and also make the problem solvable iteratively, we respectively approximate $F(\boldsymbol{z})$ by its first- and second-order Taylor expansions to update the variable $\boldsymbol{z}$ twice while always fixing the above dynamic regularization and also an extra regularizer $\frac{1}{2\eta_k}\|\boldsymbol{z}\|^2_{\sqrt{\boldsymbol{v}_k+\nu}}$ induced by the weight decay in AdamW. As a result, we achieve at our Win acceleration, a Nesterov-alike acceleration, for AdamW and Adam that uses a conservative step and a reckless step to update twice and then linearly combines these two updates for acceleration. Then we extend this Win acceleration to LAMB (You et al., 2019) and SGD. The above acceleration derivation is transparent and general which could motivate other accelerations and provide examples to introduce other accelerations into adaptive algorithms.

Secondly, we prove the convergence of our Win-accelerated AdamW and Adam. For both, to find an ϵ-approximate first-order stationary point, their stochastic gradient complexity is $\mathcal{O}\big(\frac{c_\infty^{2.5}}{\nu^{0.5}\epsilon^4}\big)$ and matches the lower bound $\Omega(\frac{1}{\epsilon^4})$ in (Arjevani et al., 2019; 2020) (up to constant factors) under the same conditions, where c_∞ upper bounds the ℓ_∞ norm of stochastic gradient. Moreover, this complexity improves a factor $\mathcal{O}\left(\frac{1}{\nu^{0.75}}\right)$ over the complexity $\mathcal{O}(\frac{c_\infty^{2.5}}{\nu^{1.25}\epsilon^4})$ of Adam-type optimizers in (Zhou et al., 2018; Guo et al., 2021), *e.g.* Adam, AdaGrad (Duchi et al., 2011), AdaBound (Luo et al., 2018), since ν is often small, *e.g.* $\mathcal{O}(\nu)=10^{-4}$ in practice. Indeed, Win-accelerated Adam and AdamW also enjoy superior complexity than other Adam variants, *e.g.* Adabelief (Zhuang et al., 2020) with compelxity $\mathcal{O}(\frac{c_2^6}{\nu^2\epsilon^4})$ and RMSProp (Zhou et al., 2018) with comeplexity $\mathcal{O}(\frac{c_\infty^{0.5}d^{0.5}}{\nu\epsilon^4})$, especially on over-parameterized networks, where c_2 is the maximum ℓ_2-norm of stochastic gradient.

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

2 Related Work

In the context of deep learning, when considering efficiency and generalization, one often prefers to adopt SGD and adaptive gradient algorithms (Bernstein et al., 2020), *e.g.* Adam, instead of other algorithms, *e.g.* variance-reduced algorithms (Johnson & Zhang, 2013), to solve problem (1). But, in practice, adaptive algorithms often suffer from inferior generalization performance than SGD. To solve this issue, AdamW (Loshchilov & Hutter, 2018) proposes a decoupled weight decay which introduces an ℓ_2-alike regularization into Adam to decay network weight iteratively, and its effectiveness is widely validated on ViTs (Touvron et al., 2021) and CNNs (Touvron et al., 2021). Later, LAMB (You et al., 2019) scales the update in AdamW to the weight magnitude for getting rid of too large or small update, but suffers from unsatisfactory performance on small batch. In this work, we aim to design a general acceleration approach to accelerate these adaptive algorithms.

Heavy-ball acceleration (Polyak, 1964) and Nesterov acceleration (Nesterov, 2003) are two classical acceleration techniques, and their effectiveness in SGD is well testified. Later, NAdam (Dozat, 2016) integrates Nesterov acceleration into the first-order gradient moment estimation but ignores the second-order gradient moments which harms the acceleration effect. Recently, for full gra-

dient decent algorithm, a new general Nesterov-type acceleration (Nesterov et al., 2018) directly interpolates two variables to look ahead for correction, and is more flexible than vanilla Nesterov acceleration (Nesterov, 2003) which interpolates the variable and gradient. See more discussion in Sec. 3.2. In this work, we use proximal point method to introduce this new acceleration into adaptive algorithms by a rigorous and transparent derivation and also necessary adaptations (or tailors).

3 Weight-decay-Integrated Nesterov Acceleration

To accelerate full gradient descent algorithm, given a full gradient $\nabla F(\boldsymbol{z}_k)$ of problem (1) at the k-th iteration, Nesterov-type acceleration (Nesterov et al., 2018) generally uses a conservative step η_k and a reckless step $\bar{\eta}_k$ to update two sequences $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_{k+1}$ respectively, and then linearly combines them to update the variable $\boldsymbol{z}_{k+1}$ of the problem. Similar formulations are also observed and proved in many recent works, *e.g.* (Allen-Zhu & Orecchia, 2014; Bansal & Gupta, 2019; Ahn & Sra, 2022). In general, their acceleration formulation can be formally formulated as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k \nabla F(\boldsymbol{z}_k), \quad \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k \nabla F(\boldsymbol{z}_k), \quad \boldsymbol{z}_{k+1} = \rho_k \boldsymbol{x}_{k+1} + (1-\rho_k)\boldsymbol{y}_{k+1}. \quad (2)$$

This acceleration enjoys provably faster convergence rate for full gradient descent method on convex problems (Nesterov et al., 2018), and is then empirically validated in many convex and nonconvex cases, *e.g.* (Wilson et al., 2017; Nado et al., 2021). Despite its effectiveness, such acceleration is rarely explored in adaptive gradient algorithms, especially for network training. In deterministic optimization setting, another widely used optimization-stabilizing and acceleration approach is proximal point method (PPM) (Moreau, 1965; Rockafellar, 1976). At the k-th iteration, PPM optimizes an ℓ_2-regularized loss $F(\boldsymbol{z}) + \frac{1}{2\eta_k}\|\boldsymbol{z} - \boldsymbol{z}_{k-1}\|_2^2$ instead of the vanilla loss $F(\boldsymbol{z})$. This small change enhances the convexity of the problem, which accelerates and stabilizes optimization process (Kim et al., 2022; Zhou et al., 2021). To make the ℓ_2-regularized problem solvable iteratively, PPM approximates the loss $F(\boldsymbol{z})$ by its first- or second-order Taylor expansion so that each iteration has a close-form solution (see below). In the following, we borrow the PPM idea to induce a *Weight-decay-Integrated Nesterov acceleration* (Win) for adaptive algorithms by using AdamW and Adam as examples in Sec. 3.1, and extend this acceleration technique to LAMB and SGD in Sec. 3.2.

3.1 Win-Accelerated AdamW and Adam

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

$$\boldsymbol{g}_k = \frac{1}{b}\sum\nolimits_{i=1}^{b} \nabla f(\boldsymbol{z}_k; \boldsymbol{\zeta}_i), \quad \boldsymbol{m}_k = (1-\beta_1)\boldsymbol{m}_{k-1} + \beta_1 \boldsymbol{g}_k, \quad \boldsymbol{v}_k = (1-\beta_2)\boldsymbol{v}_{k-1} + \beta_2 \boldsymbol{g}_k^2, \quad (3)$$

where $\boldsymbol{g}_k$ is the average gradient on a minibatch data of size b, $\beta_1 \in [0,1]$ and $\beta_2 \in [0,1]$. For the initialization, we set $\boldsymbol{m}_0 = \boldsymbol{g}_0$, $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$. For brevity, with a small scaler $\nu > 0$, we define

$$\boldsymbol{s}_k = \sqrt{\boldsymbol{v}_k + \nu}, \qquad \boldsymbol{u}_k = \boldsymbol{m}_k / \sqrt{\boldsymbol{v}_k + \nu}. \quad (4)$$

Then following the spirit of PPM, at the k-th iteration, we minimize a regularized loss $F(\boldsymbol{x}) + \frac{1}{2\eta_k}\|\boldsymbol{x} - \boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2$. Here we use the regularizer $\|\boldsymbol{x} - \boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2$ instead of the ℓ_2-regularization $\|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2$, since 1) this new regularization allows us to handle adaptive algorithms as shown below Eqn. (5), and 2) it also helps increase the convexity of the problem and actually further considers the different sharpness properties of each coordinates of the loss because of different elements in $\boldsymbol{s}_k$, thus also speeding up the convergence. To make the problem solvable iteratively, we approximate the vanilla loss $F(\boldsymbol{z})$ by its first-order Taylor expansion at the point $\boldsymbol{z}_k$ and update $\boldsymbol{x}_{k+1}$ as

$$\boldsymbol{x}_{k+1} = \operatorname{argmin}_{\boldsymbol{x}} F(\boldsymbol{z}_k) + \langle \boldsymbol{m}_k, \boldsymbol{x} - \boldsymbol{z}_k \rangle + \frac{1}{2\eta_k}\|\boldsymbol{x} - \boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2 + \frac{\lambda}{2}\|\boldsymbol{x}\|_{\boldsymbol{s}_k}^2 = \frac{1}{1+\lambda\eta_k}(\boldsymbol{x}_k - \eta_k \boldsymbol{u}_k), \quad (5)$$

where $\|\boldsymbol{x}\|_{\boldsymbol{s}_k} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{s}_k * \boldsymbol{x} \rangle}$ with an element-wise product operation $*$, $\boldsymbol{m}_k$ is used to approximate the full gradient $\nabla F(\boldsymbol{z}_k)$ for Taylor expansion. We add a small regularization $\frac{\lambda}{2}\|\boldsymbol{x}\|_{\boldsymbol{s}_k}^2$, since 1) it can largely improve the generalization performance in practice (Loshchilov & Hutter, 2018; Touvron et al., 2021); 2) it allows us to derive Adam ($\lambda = 0$) and AdamW ($\lambda > 0$). Specifically, when $\lambda = 0$, the updating (5) becomes the exact Adam algorithm. If $\lambda > 0$, the updating (5) is an approximation to the updating rule $\boldsymbol{x}_{k+1} = (1 - \lambda\eta_k)\boldsymbol{x}_k - \eta_k \boldsymbol{u}_k$ of AdamW. This is because consider $\lambda\eta_k$ is small in practice, we can approximate $(1+\lambda\eta_k)^{-1} = 1 - \lambda\eta_k + \mathcal{O}(\lambda^2\eta_k^2)$ and thus $\frac{1}{1+\lambda\eta_k}(\boldsymbol{x}_k - \eta_k \boldsymbol{u}_k) = [1 - \lambda\eta_k + \mathcal{O}(\lambda^2\eta_k^2)]\boldsymbol{x}_k - [\eta_k - \mathcal{O}(\lambda\eta_k^2) + \mathcal{O}(\lambda^3\eta_k^3)]\boldsymbol{u}_k$ which becomes AdamW by ignoring the ignorable terms related to $\mathcal{O}(\eta_k^2)$ or $\mathcal{O}(\eta_k^3)$. This is also one reason that we adopt the regularizer $\|\boldsymbol{x} - \boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2$ in (5) instead of the ℓ_2-regularization in PPM, since we can flexibly derive Adam and AdamW.

Algorithm 1: Win-Accelerated AdamW, Adam and LAMB

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

1 **while** $k < T$ **do**
2 $\quad \boldsymbol{g}_k = \frac{1}{b}\sum_{i=1}^b \nabla f(\boldsymbol{z}_k; \boldsymbol{\zeta}_i)$
3 $\quad \boldsymbol{m}_k = (1-\beta_1)\boldsymbol{m}_{k-1} + \beta_1 \boldsymbol{g}_k$ $\qquad$ /* $\boldsymbol{m}_0 = \boldsymbol{g}_0$ */
4 $\quad \boldsymbol{v}_k = (1-\beta_2)\boldsymbol{v}_{k-1} + \beta_2 \boldsymbol{g}_k^2$ $\qquad$ /* $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$ */
5 $\quad \boldsymbol{u}_k = \frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}}$ for AdamW and Adam, $\boldsymbol{u}_k = \frac{\|\boldsymbol{x}_k\|_2}{\|\boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k+\nu}\|_2}\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}}$ for LAMB
6 $\quad \boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}(\boldsymbol{x}_k - \eta_k \boldsymbol{u}_k)$
7 $\quad \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k(\boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{u}_k)$ with $\tau_k = \frac{1}{\eta_k+\bar{\eta}_k+\lambda\eta_k\bar{\eta}_k}$
8 **end while**

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

$$\begin{aligned}\boldsymbol{z}_{k+1} &= \operatorname{argmin}_{\boldsymbol{z}} F(\boldsymbol{z}_k) + \langle \boldsymbol{m}_k, \boldsymbol{z} - \boldsymbol{z}_k\rangle + \frac{1}{2\bar{\eta}_k}\|\boldsymbol{z} - \boldsymbol{z}_k\|_{\boldsymbol{s}_k}^2 + \frac{1}{2\eta_k}\|\boldsymbol{z} - \boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_k}^2 + \frac{\lambda}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2 \\ &= \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\big(\boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{u}_k\big),\end{aligned} \tag{6}$$

where $\tau_k = \frac{1}{\eta_k+\bar{\eta}_k+\lambda\eta_k\bar{\eta}_k}$, $\boldsymbol{m}_k$ is used to approximate $\nabla F(\boldsymbol{x}_k)$ as guaranteed by Theorem 1 in Sec. 4, $\bar{\eta}_k$ approximates the inverse of the local smoothness parameter of $F(\boldsymbol{z})$ around $\boldsymbol{z}_k$. Here we use a regularizer $\|\boldsymbol{z} - \boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_k}^2$ with the latest update $\boldsymbol{x}_{k+1}$ instead of $\boldsymbol{x}_k$ as an anchor point, since the latest update $\boldsymbol{x}_{k+1}$ could often provide better regularization for the concurrent optimization.

Now we have used PPM to rigorously derive our Win-accelerated AdamW and Adam in Eqns. (3), (5) and (6). For more clarity, we summarize their algorithmic steps in Algorithm 1 in which we omit the bias-correction term for simplicity. When $\lambda = 0$, it is Win-accelerated Adam; if $\lambda > 0$, it gives Win-accelerated AdamW. Generally, AdamW can greatly improve the generalization performance of Adam by simply adding a weight decay (i.e. the regularizer $\frac{\lambda}{2}\|\cdot\|_{\boldsymbol{s}_k}^2$) into Adam as observed in many works, *e.g.* (Loshchilov & Hutter, 2018; Touvron et al., 2021). Our Win-acceleration is quit simple and efficient, since our accelerated AdamW/Adam only adds one extra simple algorithmic step, i.e. the seventh step in Algorithm 1, on vanilla AdamW/Adam, and brings negligible extra computational overhead into vanilla optimizer, e.g. about 2% extra training time per iteration on AdamW evaluated on ViT-small and ViT-base. Moreover, for the only extra hyper-parameter, the reckless step $\bar{\eta}_k$, in Algorithm 1 over AdamW/Adam, we always set it $2\times$ larger than the conservative step η_k for all iterations, i.e. $\bar{\eta}_k = 2\eta_k$, working well in our all experiments.

Now we discuss the relations between Nesterov-type acceleration (2) and our Win acceleration (6). For comparison, we introduce a virtual sequence $\boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{u}_k$ in Win, and rewrite (6) as

$$\boldsymbol{x}_{k+1} = (1+\lambda\eta_k)^{-1}(\boldsymbol{x}_k - \eta_k\boldsymbol{u}_k), \quad \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{u}_k, \quad \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\boldsymbol{y}_{k+1}, \tag{7}$$

where $\boldsymbol{u}_k$ is defined in (4). By comparing Nesterov-type acceleration (2) with our Win acceleration (7), one can observe some similarity and also differences as well. For similarity, both acceleration use a conservative step η_k and a reckless step $\bar{\eta}_k$ to update $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_{k+1}$ respectively, and then linearly combine $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_{k+1}$ to obtain $\boldsymbol{z}_{k+1}$. For the differences, the first one is that Win has a weight-decay-alike factor $\frac{1}{1+\lambda\eta_k}$ in (7) which slightly decays the variable $\boldsymbol{x}_k$ like AdamW and also the update $\boldsymbol{u}_k$, while Nesterov acceleration has no such important factor. Note, this weight decay can greatly benefit generalization in practice as shown in many works, *e.g.* (Loshchilov & Hutter, 2018; Touvron et al., 2021; Liu et al., 2021). Another difference is that for almost all acceleration techniques, including Nesterov-type acceleration in (2), the sum of the linear combination factors (*e.g.* ρ_k and $1-\rho_k$ in (2)) is always one. In contrast, in Eqn. (7), Win uses $\bar{\eta}_k\tau_k + \eta_k\tau_k = 1 - \frac{\lambda\eta_k\bar{\eta}_k}{\eta_k+\bar{\eta}_k+\lambda\eta_k\bar{\eta}_k} < 1$ when $\lambda > 0$, which indeed gives a second weigh decay. Since these two differences are caused by the weight decay, we call our acceleration "weight-decay-integrated Nesterov acceleration" (Win).

3.2 Extension to LAMB and SGD

Here we generalize Win acceleration to LAMB (You et al., 2019) and SGD (Robbins & Monro, 1951). For LAMB, it scales the update $\boldsymbol{u}_k$ of AdamW in Eqn. (4) so that $\boldsymbol{u}_k$ is at the same magnitude

of the network weight $\boldsymbol{x}_k$. That is, it changes the update rule $\boldsymbol{x}_{k+1} = (1-\lambda\eta_k)\boldsymbol{x}_k - \eta_k \boldsymbol{m}_k/\boldsymbol{s}_k$ in AdamW to $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k \frac{\|\boldsymbol{x}_k\|_2}{\|\boldsymbol{r}_k+\lambda\boldsymbol{x}_k\|_2}(\boldsymbol{r}_k+\lambda\boldsymbol{x}_k)$ where $\boldsymbol{r}_k = \boldsymbol{m}_k/\boldsymbol{s}_k$. This modification is to avoid too large or small update, improving optimization efficiency. To extend Win acceleration to LAMB, we inherit this scaling spirit, and scale the update $\boldsymbol{u}_k$ in (4) to the following one:

$$\boldsymbol{u}_k = (\|\boldsymbol{x}_k\|_2/\|\boldsymbol{m}_k/\boldsymbol{s}_k\|_2)\cdot(\boldsymbol{m}_k/\boldsymbol{s}_k). \tag{8}$$

We scale $\boldsymbol{m}_k/\boldsymbol{s}_k$ instead of $(\boldsymbol{m}_k/\boldsymbol{s}_k+\lambda\boldsymbol{x}_k)$ in LAMB, as our scaling can be repeatedly used to update our two sequences $\boldsymbol{x}_k$ and $\boldsymbol{z}_k$. Next, we can respectively follow Eqn. (5) and (6) to update the two sequences $\boldsymbol{x}_k$ and $\boldsymbol{z}_k$. See the detailed steps of Win-accelerated LAMB in Algorithm 1, and the detailed comparison between LAMB and Win-accelerated LAMB in Appendix A.

For SGD, applying Win acceleration to it is quite direct. Specifically, the only algorithmic difference between SGD and AdamW on the ℓ_2-regularized problems is that SGD has no second-order moment $\boldsymbol{v}_k$ while AdamW has. So we can borrow the acceleration framework of AdamW in Sec. 3.1 to accelerate SGD by setting $\boldsymbol{s}_k = \mathbf{1} \in \mathbb{R}^d$ in Eqn. (4), (5) and (6), and obtain WIN-accelerated SGD:

$$\boldsymbol{m}_k = \beta_1\boldsymbol{m}_{k-1} + \beta_1'\boldsymbol{g}_k,\ \boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}(\boldsymbol{x}_k - \eta_k\boldsymbol{m}_k),\ \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\big(\boldsymbol{z}_k - \bar{\eta}_k\boldsymbol{m}_k\big), \tag{9}$$

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

$$\boldsymbol{m}_k = \beta_1\boldsymbol{m}_{k-1} + \beta_1'(\boldsymbol{g}_k + \lambda\boldsymbol{x}_k), \quad \boldsymbol{x}_{k+1} = (1-\lambda\eta_k)\boldsymbol{x}_k - \eta_k(\boldsymbol{g}_k + \beta_2\boldsymbol{m}_k). \tag{10}$$

By comparing Win-accelerated SGD and SGD-M in (10), one can find their big differences mainly caused by their different acceleration strategies and ways to handle weight decay. Win-accelerated SGD is derived from PPM and a recently proposed acceleration (2), while SGD-M modifies another previous Nesterov-type acceleration (Nesterov, 2003) (with formulation $\boldsymbol{m}_k = \beta_1\boldsymbol{m}_{k-1} - \frac{\eta_k}{b}\sum_{i=1}^b \nabla f(\boldsymbol{x}_k + \eta_k\boldsymbol{m}_{k-1};\boldsymbol{\zeta}_i)$ and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{m}_k$) to better train networks. See more mechanisms of previous Nesterov acceleration and (10) in (Sutskever et al., 2013; Bengio et al., 2013).

4 Convergence Analysis

Here we investigate the convergence performance of Win-accelerated algorithms by taking accelerated AdamW, Adam and SGD as examples, as these algorithms are more preferably used in deep learning field. Moreover, since we aim to accelerate deep network training which is highly nonconvex problems, we focus on analyzing nonconvex problems to accord with the practical setting.

For analysis, we follow previous works on optimizer, *e.g.* (Kingma & Ba, 2014; Reddi et al., 2019; Luo et al., 2019; Duchi et al., 2011; Xie et al., 2022), and introduce necessary assumptions.

Assumption 1 (L-smoothness)**.** *We say a function $f(\boldsymbol{z},\cdot)$ to be L-smooth w.r.t. $\boldsymbol{z}$, if for $\forall\boldsymbol{z}_1,\boldsymbol{z}_2$ and $\forall\boldsymbol{\zeta}\sim\mathcal{D}$, we have $\|\nabla f(\boldsymbol{z}_1,\boldsymbol{\zeta}) - \nabla f(\boldsymbol{z}_2,\boldsymbol{\zeta})\|_2 \le L\|\boldsymbol{z}_1-\boldsymbol{z}_2\|_2$ with a universal constant L.*

Assumption 2 (Unbiased and bounded gradient estimation)**.** *The gradient estimation $\boldsymbol{g}_k$ is unbiased, i.e. for $\forall k$, $\mathbb{E}[\boldsymbol{g}_k] = \nabla F(\boldsymbol{z}_k)$, and its magnitude and variance are bounded, namely, for $\forall k$, $\|\boldsymbol{g}_k\|_\infty \le c_\infty$ and $\mathbb{E}[\|\nabla F(\boldsymbol{z}_k) - \boldsymbol{g}_k\|_2] \le \sigma$ with two universal constants c_∞ and σ.*

Next, we first define a dynamic function $F_k(\boldsymbol{z})$ at the k-th iteration which is real loss minimized by our algorithms. It combines the vanilla loss $F(\boldsymbol{z})$ in (1) and a dynamic regularization $\frac{\lambda}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2$:

$$F_k(\boldsymbol{z}) = F(\boldsymbol{z}) + \frac{\lambda}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2 = \mathbb{E}_{\boldsymbol{\zeta}}[f(\boldsymbol{z};\boldsymbol{\zeta})] + \frac{\lambda}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2, \tag{11}$$

where $\boldsymbol{s}_k$ is given in (4). To obtain (11), following PPM spirit and Eqn. (5), one can approximate $F(\boldsymbol{z})$ by its first-order Taylor expansion, and obtain Eqn. (5) with $\boldsymbol{x}$ replaced by $\boldsymbol{z}$ to update $\boldsymbol{z}_{k+1} = \frac{1}{1+\lambda\eta_k}(\boldsymbol{z}_k - \eta_k\boldsymbol{m}_k/\boldsymbol{s}_k)$. Since $\lambda\eta_k$ is very small, one can follow the discussion below Eqn. (5) and approximate $\boldsymbol{z}_{k+1}$ as $\boldsymbol{z}_{k+1} = (1-\lambda\eta_k)\boldsymbol{z}_k - \eta_k\boldsymbol{m}_k/\boldsymbol{s}_k$ which becomes the update rule of AdamW. This is the reason that our analysis on Win-accelerated AdamW involves a dynamic loss $F_k(\boldsymbol{z})$ in (11). Note, for Win-accelerated Adam ($\lambda=0$), $F_k(\boldsymbol{z})$ degenerates to the vanilla loss $F(\boldsymbol{z})$.

With these assumptions, we analyze the convergence behaviors of our accelerated algorithms on general nonconvex problems, and summarize our main results in Theorem 1 with proof in Appendix E.

Theorem 1. *Suppose Assumptions 1 and 2 hold, and $\boldsymbol{x}_\star \in \operatorname{argmin}_{\boldsymbol{x}} F(\boldsymbol{x})$. By setting $\bar{\eta}_k = \gamma\eta_k, \gamma > 1, \eta_k = \eta \le \frac{\nu^{0.5}b\epsilon^2}{36c^{1.5}(\gamma+1)^2L\sigma^2}$, $\beta_1 \le \frac{\nu^{0.5}b\epsilon^2}{6c\sigma^2}$, $\beta_2 \in (0,1)$, $c = (c_\infty^2+\nu)^{0.5}$ after $T = \mathcal{O}\big(\frac{c_\infty^{2.5}\Delta\sigma^2L}{\nu^{0.5}b\epsilon^4}\big)$ iterations with minibatch size b and $\Delta = F(\boldsymbol{x}_0) - F(\boldsymbol{x}_\star)$, the sequence $\{(\boldsymbol{x}_k,\boldsymbol{z}_k)\}_{k=0}^T$ generated*

by Win-accelerated AdamW and Adam in Algorithm 1 satisfies the following four properties.
a) *The gradient $\nabla F_k(\boldsymbol{x}_k)$ of the sequence $\{\boldsymbol{x}_k\}_{k=0}^T$ can be upper bounded by*

$$\frac{1}{T}\sum\nolimits_{k=0}^{T-1}\mathbb{E}\Big[\,\|\nabla F_k(\boldsymbol{x}_k)\|_2^2+\frac{1}{4}\,\|\boldsymbol{m}_k+\lambda\boldsymbol{x}_k*\boldsymbol{s}_k\|_2^2\Big]\leq\epsilon^2.$$

b) *The gradient moment $\boldsymbol{m}_k$ can well estimate the full gradient $\nabla F(\boldsymbol{x}_k)$ and $\nabla F(\boldsymbol{z}_k)$:*

$$\frac{1}{T}\sum\nolimits_{k=0}^{T-1}\max\Big\{\mathbb{E}\|\boldsymbol{m}_k-\nabla F(\boldsymbol{x}_k)\|_2^2\,,\mathbb{E}\|\boldsymbol{m}_k-\nabla F(\boldsymbol{z}_k)\|_2^2\Big\}\leq\Big(16+\frac{1}{2c}\nu^{0.5}L\Big)\epsilon^2.$$

c) *The sequence $\{\boldsymbol{x}_k,\boldsymbol{z}_k\}$ satisfies*

$$\frac{1}{T}\sum_{k=0}^{T-1}\Big\{\mathbb{E}\|\boldsymbol{x}_k-\boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_k}^2,\mathbb{E}\|\boldsymbol{z}_{k+1}-\boldsymbol{z}_k\|_2^2,\mathbb{E}\|\boldsymbol{z}_k-\boldsymbol{x}_k\|_2^2\Big\}\leq\Big\{\frac{\eta^2\epsilon^2}{4(1+\lambda\eta)^2},\frac{\nu^{1.5}\beta_1^2\epsilon^2}{4c(1-\beta_1)^3L^2},\frac{\nu^{0.5}\epsilon^2}{4cL}\Big\}.$$

d) *The total stochastic gradient complexity to achieve the above three properties is $\mathcal{O}\big(\frac{c_\infty^{2.5}\Delta\sigma^2L}{\nu^{0.5}\epsilon^4}\big)$.*

Theorem 1 guarantees the convergence of Win-accelerated AdamW and Adam in Algorithm 1 on nonconvex problems. When $\lambda>0$, Algorithm 1 corresponds to Win-accelerated AdamW; and if $\lambda=0$, it becomes Win-accelerated Adam. For both cases, Theorem 1 holds. Theorem 1 a) shows that by running at most $T=\mathcal{O}\big(\frac{c_\infty^{2.5}\Delta\sigma^2L}{\nu^{0.5}b\epsilon^4}\big)$ iterations, the average gradient $\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\big[\,\|\nabla F_k(\boldsymbol{x}_k)\|_2^2\,\big]$ is upper bounded by ϵ^2, guaranteeing the algorithmic convergence. Theorem 1 b) indicates the gradient moment $\boldsymbol{m}_k$ can well estimate the full gradient $\nabla F(\boldsymbol{z}_k)$ and also $\nabla F(\boldsymbol{x}_k)$ because of their small distances, guaranteeing the good Taylor approximation used in Eqn. (5) and (6). Moreover, in Theorem 1 c), one can find that although Algorithm 1 uses a conservative step η_k and a reckless step $\bar{\eta}_k=\gamma\eta_k$ $(\forall\gamma>1)$ to update, the two sequences $\boldsymbol{x}_{k+1}$ and $\boldsymbol{z}_{k+1}$ can converge to each other, which could be the key for the good convergence behavior of both Win-accelerated AdamW and Adam.

Now we discuss the stochastic gradient complexity of Win-accelerated Adam and AdamW. Theorem 1 d) shows that to find an ϵ-approximate first-order stationary point, both Win-accelerated Adam and AdamW have the complexity $\mathcal{O}\big(\frac{c_\infty^{2.5}\sigma^2L}{\nu^{0.5}\epsilon^4}\big)$ which matches the lower bound $\Omega(\frac{1}{\epsilon^4})$ in (Arjevani et al., 2019; 2020) (up to constant factors) under the same Assumptions 1 and 2. Our accelerated Adam and AdamW enjoy superior complexity over Adam-type optimizers, *e.g.* Adam, AdaGrad (Duchi et al., 2011), AdaBound (Luo et al., 2018), whose previously best known complexity under the same assumptions is $\mathcal{O}(\frac{c_\infty^{2.5}\sigma^2L}{\nu^{1.25}\epsilon^4})$ in (Zhou et al., 2018; Chen et al., 2021; Guo et al., 2021). By comparison, both accelerated Adam and AdamW improve their complexity by a factor $\mathcal{O}\left(\frac{1}{\nu^{0.75}}\right)$, where ν is often very small in practice, *e.g.* $\mathcal{O}(\nu)=10^{-4}$. Since the convergence of AdamW has not been proved yet in the literatures, here we cannot directly compare with it. Moreover, the complexity of Win-accelerated Adam and AdamW is also lower than $\mathcal{O}(\frac{c_2^6\sigma^2L}{\nu^2\epsilon^4})$ of Adabelief (Zhuang et al., 2020) and $\mathcal{O}(\frac{c_\infty^{0.5}d^{0.5}\sigma^2L}{\nu\epsilon^4})$ of RMSProp (Tijmen & Geoffrey, 2012; Zhou et al., 2018), especially on over-parameterized networks, since for a d-dimensional gradient, its ℓ_2-norm upper bound c_2 is often much larger than the ℓ_∞-norm c_∞ and can be $\sqrt{d}\times$ larger for worse case.

Now we discuss the convergence performance of Win-accelerated SGD in Theorem 2.

Theorem 2. *Suppose Assumptions 1 and 2 hold, and $\boldsymbol{x}_\star\in\operatorname{argmin}_{\boldsymbol{x}}F(\boldsymbol{x})$. By setting $\bar{\eta}_k=\gamma\eta_k,\gamma>1,\eta_k=\eta\leq\frac{b\epsilon^2}{36(\gamma+1)^2L\sigma^2}$, $\beta_1\leq\frac{b\epsilon^2}{6\sigma^2}$ and $\beta_1'=1-\beta_1$, after $T=\mathcal{O}\big(\frac{\Delta\sigma^2L}{\nu^{0.5}b\epsilon^4}\big)$ iterations with minibatch size b and $\Delta=F(\boldsymbol{x}_0)-F(\boldsymbol{x}_\star)$, the sequence $\{(\boldsymbol{x}_k,\boldsymbol{z}_k)\}_{k=0}^T$ generated by Win-accelerated SGD in (9) satisfies the four properties in Theorem 1 with $\nu=c_\infty=c=1$ and $\boldsymbol{s}_k=\mathbf{1}\in\mathbb{R}^d$.*

See its proof in Appendix F. Theorem 2 also guarantees the convergence of Win-accelerated SGD. By using the hyper-parameter settings in Theorem 2, the sequence $\{(\boldsymbol{x}_k,\boldsymbol{z}_k)\}_{k=0}^T$ generated by Win-accelerated SGD satisfies the four properties in Theorem 1 with $\nu=c_\infty=c=1$ and $\boldsymbol{s}_k=\mathbf{1}$. It shows the complexity $\mathcal{O}(\frac{L\sigma^2}{\epsilon^4})$ of Win-accelerated SGD which also matches the lower bound $\Omega(\frac{1}{\epsilon^4})$ in (Arjevani et al., 2019; 2020) (up to constant factors) under the same Assumptions 1 and 2.

5 Experiments

Here we investigate the performance of our accelerated algorithms on two representative tasks, including vision classification tasks and also natural language modeling tasks. For vision tasks, we test accelerated algorithms on both CNNs, *e.g.* ResNet (He et al., 2016), and vision transformers

Table 2: ImageNet top-1 accuracy (%) of ResNet50&101 whose official optimizer is LAMB due to the stronger data augmentation for better performance. $*$ is reported in (Wightman et al., 2021).

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

(ViTs), such as ViT (Dosovitskiy et al., 2020) and PoolFormer (Yu et al., 2021). For natural language modeling tasks, we use LSTM (Schmidhuber et al., 1997) and Transformer-XL (Dai et al., 2019) to evaluate the proposed accelerated algorithms for sequence modeling.

For clarity, we call our accelerated algorithm "X-Win", where "X" denotes vanilla optimizers, *e.g.* Adam. In all experiments, we do not change model architectures and data augmentations, and only replace the default optimizer with ours. Moreover, for all experiments, our accelerated algorithms, *e.g.* AdamW-Win, always use the default optimizer-inherent hyper-parameters of the vanilla optimizers, *e.g.* first- and second-order moment parameters β_1 and β_2 in AdamW; and their reckless step $\bar{\eta}_k$ always satisfies $\bar{\eta}_k = 2\eta_k$. These settings well reduce the parameter-tuning cost of our algorithms. In the experiments, same with other optimizers, we only slightly tune other widely tuned hyper-parameters around the vanilla ones, *e.g.* step size and warm-up epochs, *etc*, which is reasonable, as our accelerated algorithms have two step sizes and the vanilla ones are not very suitable.

5.1 Results on Vision Classification Tasks

Results on ResNet18. Here we follow the conventional supervised training setting used in ResNets (He et al., 2016) and evaluate our accelerated algorithms on ImageNet (Deng et al., 2009). Due to limited space, we defer the hyper-parameter settings of the four accelerated algorithms in Table 1 into Appendix B.

Table 1: ImageNet top-1 accuracy (%) of ResNet18. $*$, † and ‡ are respectively reported in (Chen et al., 2021), (Zhuang et al., 2020) and (Liu et al., 2019).

AdaBound	68.1*	Radam	67.7*
Nadam	68.8	Padam	70.1*
Yogi	68.2*	MSVAG	66.0*
SGD-H	67.3	AdaBelief	70.1†
SGD-M	70.2*	Adam	66.5‡
SGD-Win	$70.7_{+0.5}$	Adam-Win	$69.3_{+2.8}$
AdamW	67.9*	LAMB	68.5
AdamW-Win	$71.0_{+3.1}$	LAMB-Win	$71.1_{+2.6}$

Table 1 shows that our accelerated algorithms can improve the corresponding non-accelerated versions by a remarkable margin. For instance, AdamW-Win, Adam-Win and LAMB-Win respectively make 3.1%, 2.2% and 2.6% improvement over their corresponding non-accelerated counterparts, AdamW, Adam and LAMB. Moreover, SGD-Win improves SGD-H (i.e. SGD + heavy ball) by 3.4%, and also surpasses SGD-M (Nesterov-accelerated SGD in Sec. 3.2) by 0.5%, also validating the superiority of our Win acceleration. Besides, our accelerated algorithms, i.e. SGD-Win, AdamW-Win and LAMB-Win, beat several other optimizers, *e.g.* AdaBound, Radam (Liu et al., 2019), Nadam, Padam (Chen et al., 2021), AdaBelief, Yogi (Zaheer et al., 2018) and MSVAG (Balles & Hennig, 2018), in which Nadam uses Nesterov acceleration to estimate its first-order gradient moment. Actually, LAMB-Win sets a new SoTA top-1 accuracy on ResNet18. All these results show the strong compatibility and superiority of our Win-acceleration in adaptive algorithms.

Results on ResNet50&101. Here we adopt the training setting in (Wightman et al., 2021) to train ResNet50&101, as this setting uses stronger data augmentation and largely improves CNNs' performance. See augmentation details and our algorithmic hyper-parameter settings in Appendix B. Here LAMB is the default optimizer because of its higher performance than other optimizers caused by the stronger augmentations (Wightman et al., 2021). All optimizers in Table 2 are under this setting.

Table 2 reports the top-1 accuracy of the compared optimizers on ImageNet. By comparison, one can observe that our accelerated algorithms consistently outperform their corresponding non-accelerated version. For example, across the three training epoch settings on ResNet50 and ResNet101, LAMB-Win always achieves remarkable improvement over LAMB which is the official optimizer for this training recipe. Specifically, LAMB-Win makes 0.7% average improvement over LAMB on both

Table 3: ImageNet top-1 accuracy (%) of ViT and PoolFormer whose default optimizers are both AdamW. $*$ and $\diamond$ are respectively reported in (Touvron et al., 2021) and (Yu et al., 2021).

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

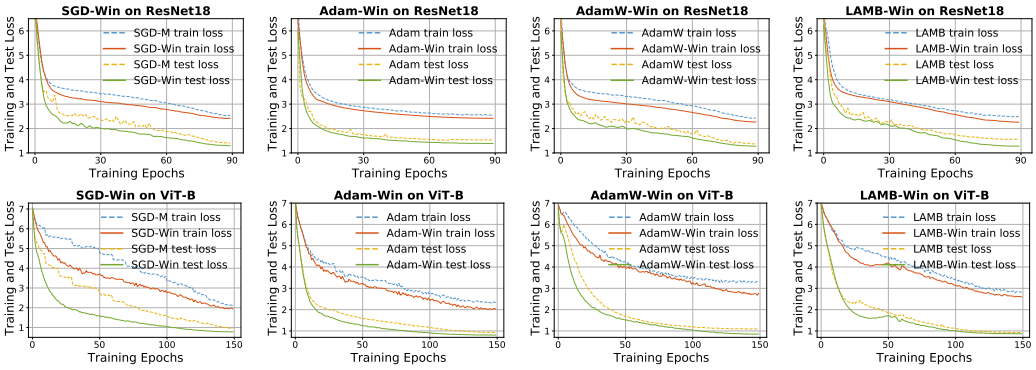

Figure 1: Visualization of training and test losses on ImageNet. In all figures, training loss is larger than test one, as training data use random augmentations, *e.g.* random crop and clip, while test data only adopt the centralization crop which eases the recognition difficulty and thus has small loss.

ResNet50 and ResNet101. For AdamW-Win and Adam-Win, they also respectively improve their counterparts by 0.7% and 0.6% on ResNet50, 1.2% and 0.6% on ResNet101. SGD-Win also makes 2.5% and 0.8% overall improvement over heavy-ball accelerated SGD (SGD-H) and Nesterov accelerated SGD (SGD-M) on ResNet50, and also has similar advantage on ResNet101. These improvements are not trivial because of the following two reasons. 1) Since the performance is already high and may approach the model limit, it is already very hard to make very large improvement. This can be testified by the fact that in (Wightman et al., 2021), using LAMB to train ResNet50 for even 600 epochs only gives 80.4% top-1 accuracy. In contrast, our accelerated LAMB-Win uses 300 epochs (only half computational cost) to achieve 80.2%. 2) By comparing the previous optimizers, including SAM, SGD-M, Adam, AdamW and LAMB, one can observe smaller accuracy gap ($\leq 0.2\%$) between the best optimizer and the runner-up. For example, on ResNet101, the SoTA optimizer, i.e. SAM, only makes 0.1% average improvement over the runner-up, namely LAMB. All these comparisons show the non-travail improvement of our accelerated algorithms over their counterparts, and also the superiority of our acceleration technique.

Results on ViTs. We follow the widely used official training setting of ViTs (Touvron et al., 2021; Yu et al., 2021). To evaluate the performance of our accelerated algorithms, we select two popular and representative ViT architures, including ViT (Dosovitskiy et al., 2020) and PoolFormer (Yu et al., 2021). See the training setting and our hyper-parameter settings in Appendix B.

We test our accelerated algorithms under different model sizes and different training epochs, and report the results in Table 3. One can find that since AdamW and LAMB use the decoupled weight decay, they enjoy better performance than SGD and Adam, which is also observed in other works, *e.g.* (Xiao et al., 2021; Nado et al., 2021). Moreover, under different training settings, our accelerated algorithms consistently outperform the corresponding non-accelerated counterparts. Specifically, compared the default AdamW optimizer on both ViT and PoolFormer, our accelerated AdamW-Win respectively makes about 1.0%, 0.9%, 1.0% average improvement under the two training epoch settings on ViT-S, ViT-B and PoolFormer-S12. For Adam-Win and LAMB-Win, one can also observe their remarkable improvements on the three ViT backbones. Moreover, our accelerated SGD-Win also outperforms the Nesterov-accelerated SGD denoted as "SGD-M" by non-trivial margins under all settings. All these results are consistent with the observations on ResNets, and they together demonstrate the advantage of our accelerated optimizers for deep network training.

Results Analysis. Here we investigate the convergence behaviors of our accelerated algorithms, and aim to explain their better test performance over their non-accelerated counterparts. In Fig. 1, we plot the curves of training and test losses along with the training epochs on ResNet18 and ViT-B. One can find that our accelerated algorithms, *e.g.* AdamW-Win, show much faster convergence behaviors than their non-accelerated counterparts, *e.g.* AdamW. Moreover, SGD-Win also converges faster than Nesterove-accelerated SGD, i.e. SGD-M. We also plot the curves of test accuracy in Fig. 2, showing the superior convergence speed of AdamW-Win and LAMB-Win over their non-accelerated versions. Fig. 3 in Appendix B also reveals SGD-Win and Adam-Win enjoy faster convergence than their non-accelerated counterparts in terms of test accuracy. So these faster convergence behaviors could contribute to our accelerated algorithms for their higher performance over non-accelerated counterparts under the same computational cost.

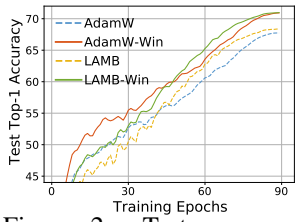

Figure 2: Test accuracy curves of AdamW-Win and LAMB-Win on ResNet18.

Robust Analysis. For the only extra hyper-parameter $\bar{\eta}_k$ in our accelerated algorithms over their non-accelerated counterparts, in experiments, we always set $\bar{\eta}_k = \gamma \eta_k$, where $\gamma = 2$ determines the relation between the reckless step $\bar{\eta}_k$ and the conservative step η_k. Here we investigate the effects of γ to the accelerated algorithms on ResNet50 by taking AdamW-Win and LAMB-Win as examples because of their superior performance. Table 4 shows the stable performance of AdamW-Win and LAMB-Win when tuning γ in a relatively large range, thus testifying the robustness of AdamW-Win and LAMB-Win to the hyper-parameter γ.

Table 4: Effects of γ to top-1 accuracy (%) of AdamW-Win and LAMB-Win on ResNet50.

γ	1.5	2	3	4	6	8
AdamW-Win	77.9	78.0	78.0	77.9	78.1	78.0
LAMB-Win	78.3	78.4	78.4	78.4	78.5	78.3

5.2 Results on Natural Language Modeling Tasks

Results on LSTM. We follow AdaBelief to test our accelerated algorithms via training three-layered LSTM (Schmidhuber et al., 1997) on the Penn TreeBank dataset (Marcinkiewicz, 1994) for 200 epochs. See optimization and training details in Appendix B.

From Table 7, one can observe that our Win-accelerated algorithms consistently surpass the corresponding non-accelerated counterparts, and actually bring 1.2 overall average perplexity improvement over the four non-accelerated counterparts.

Table 5: Test perplexity of LSTM on Penn Treebank. $*$ is reported by AdaBelief (Zhuang et al., 2020).

AdaBound	63.6*	Radam	70.0*
Yogi	67.5*	AdaBelief	61.2*
fromage	68.0*	MSVAG	65.3*
SGD-H	67.4	Padam	63.2*
SGD-M	63.8*	Adam	64.3*
SGD-Win	$61.6_{+2.2}$	Adam-Win	$62.7_{+1.6}$
AdamW	67.0*	LAMB	66.8
AdamW-Win	$66.5_{+0.5}$	LAMB-Win	$66.2_{+0.6}$

Results on Transformer-XL. We adopt a widely used language sequence model, i.e. Transformer-XL (Dai et al., 2019), to further evaluate the performance of our accelerated algorithms. Since 1) Adam is the most popular and used optimizer in NLP models, including Transformer-XL, and 2) our limited resource cannot well tune the hyper-parameters of other optimizers in Sec. 5.1, we take Adam as an example to show the superiority of our accelerated algorithms. Follow the official setting of Transformer-XL-base, we use Adam-Win with the default hyper-parameters of Adam on the WikiText-103 dataset. See more details in Appendix B.

Table 6: Test PPL of Transformer-XL-base on WikiText-103 where Adam is the official optimizer. * is reported in the official implementation.

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

Table 8 shows that under different training steps, our accelerated Adam-Win always achieves lower test PPL than the official Adam optimizer. Spefically, it improves 1.5 average test PPL over Adam on the three test cases. All these results are consistent with observations on vision tasks, and they together demonstrate the advantages of our accelerated algorithms.

6 Conclusion

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

REFERENCES

Kwangjun Ahn and Suvrit Sra. Understanding nesterov's acceleration via proximal point method. In *Symposium on Simplicity in Algorithms (SOSA)*, pp. 117–130. SIAM, 2022.

Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*, 2014.

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.

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*, pp. 242–299. PMLR, 2020.

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pp. 404–413. PMLR, 2018.

Nikhil Bansal and Anupam Gupta. Potential-function proofs for gradient methods. *Theory of Computing*, 15(1):1–32, 2019.

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in optimizing recurrent networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8624–8628. IEEE, 2013.

Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming-Yu Liu. On the distance between two neural networks and the stability of learning. *Advances in Neural Information Processing Systems*, 33: 21370–21381, 2020.

Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

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*, pp. 3267–3275, 2021.

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*, pp. 702–703, 2020.

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*, pp. 2978–2988, 2019.

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

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

Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

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

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

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*, pp. 770–778, 2016.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proc. Conf. Neural Information Processing Systems*, pp. 315–323, 2013.

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

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

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.

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*, pp. 10012–10022, 2021.

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

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.

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

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

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

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.

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

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

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

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

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

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

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

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

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

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.

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*, pp. 10347–10357. PMLR, 2021.

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

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

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.

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

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.

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.

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*, pp. 6023–6032, 2019.

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

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

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.

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

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.

Algorithm 2: LAMB in (You et al., 2019)

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

while $k < T$ **do**
- $\boldsymbol{g}_k = \frac{1}{b}\sum_{i=1}^b \nabla f(\boldsymbol{z}_k; \boldsymbol{\zeta}_i)$
- $\boldsymbol{m}_k = (1-\beta_1)\boldsymbol{m}_{k-1} + \beta_1 \boldsymbol{g}_k$ /* $\boldsymbol{m}_0 = \boldsymbol{g}_0$ */
- $\boldsymbol{v}_k = (1-\beta_2)\boldsymbol{v}_{k-1} + \beta_2 \boldsymbol{g}_k^2$ /* $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$ */
- $\boldsymbol{u}_k = \frac{\|\boldsymbol{x}_k\|_2}{\|\boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k+\nu}+\lambda \boldsymbol{x}_k\|_2}\left(\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}} + \lambda \boldsymbol{x}_k\right)$
- $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k \boldsymbol{u}_k$

end while

Algorithm 3: Win-Accelerated LAMB

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

while $k < T$ **do**
- $\boldsymbol{g}_k = \frac{1}{b}\sum_{i=1}^b \nabla f(\boldsymbol{z}_k; \boldsymbol{\zeta}_i)$
- $\boldsymbol{m}_k = (1-\beta_1)\boldsymbol{m}_{k-1} + \beta_1 \boldsymbol{g}_k$ /* $\boldsymbol{m}_0 = \boldsymbol{g}_0$ */
- $\boldsymbol{v}_k = (1-\beta_2)\boldsymbol{v}_{k-1} + \beta_2 \boldsymbol{g}_k^2$ /* $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$ */
- $\boldsymbol{u}_k = \frac{\|\boldsymbol{x}_k\|_2}{\|\boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k+\nu}\|_2}\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}}$
- $\boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}\left(\boldsymbol{x}_k - \eta_k \boldsymbol{u}_k\right)$
- $\boldsymbol{z}_{k+1} = \bar{\eta}_k \tau_k \boldsymbol{x}_{k+1} + \eta_k \tau_k \left(\boldsymbol{z}_k - \bar{\eta}_k \boldsymbol{u}_k\right)$ with $\tau_k = \frac{1}{\eta_k + \bar{\eta}_k + \lambda \eta_k \bar{\eta}_k}$

end while

APPENDIX

The appendix contains the technical proofs of convergence results and some additional experimental details of the paper entitled "Win: Weight-Decay-Integrated Nesterov Acceleration for Adaptive Gradient Algorithms". It is structured as follows. Appendix B provides more experimental details, such as hyper-parameter settings of the four accelerated algorithms and the official data augmentations. In Appendix C, we define some necessary notations for our analysis. Then Appendix D provides some auxiliary lemmas throughout this document. Then Appendix E presents the proof of the convergence results in Sec. 1, i.e., the proof of Theorems 1 and 2. Finally, Appendix G provides the proofs of some auxiliary lemmas in Appendix D.

A MORE DETAILS OF LAMB AND WIN-ACCELERATED LAMB

Here we introduce more details of vanilla LAMB (You et al., 2019) and our Win-accelerated LAMB. Specifically, Algorithm 2 and 3 respectively summarize the algorithmic steps of LAMB and Win-accelerated LAMB. By comparison, we can find two differences. The first one is that LAMB (You et al., 2019) scales $\left(\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}} + \lambda \boldsymbol{x}_k\right)$ via $\boldsymbol{u}_k = \frac{\|\boldsymbol{x}_k\|_2}{\|\boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k+\nu}+\lambda \boldsymbol{x}_k\|_2}\left(\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}} + \lambda \boldsymbol{x}_k\right)$, while Win-accelerated LAMB scales $\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}}$ through $\boldsymbol{u}_k = \frac{\|\boldsymbol{x}_k\|_2}{\|\boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k+\nu}\|_2}\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}}$. We scale $\boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k+\nu}$ instead of $\left(\frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k+\nu}} + \lambda \boldsymbol{x}_k\right)$ in LAMB, as our scaling can be repeatedly used to update our two sequences $\boldsymbol{x}_k$ and $\boldsymbol{z}_k$. The second difference is that Win-accelerated LAMB has one extra acceleration step, i.e. the 7-th step in Algorithm 3, while LAMB has not.

B MORE EXPERIMENTAL DETAILS

Due to space limitation, we defer the experimental details, such as hyper-parameter settings of the four accelerated algorithms, and their official augmentations in (He et al., 2016) and (Wightman et al., 2021), to this section.

For accelerated algorithms, including AdamW-Win, LAMB-Win, Adam-Win and SGD-Win, always share the default optimizer-inherent hyper-parameters of the vanilla optimizers and its reckless step $\bar{\eta}_k$ is always $2\times$ larger than its conservative step η_k for all iterations, i.e. $\bar{\eta}_k = 2\eta_k$. For AdamW-Win, LAMB-Win, Adam-Win, their first- and second-order moment parameters β_1 and β_2 are set to the default values $\beta_1 = 0.9$ and $\beta_2 = 0.999$ used in AdamW, LAMB and Adam. For LAMB-Win, its other key parameters, such as "grad averaging" and "trust clip", also adopt the default ones in vanilla LAMB. For SGD-Win, it uses the default momentum parameter 0.9 and set dampening parameter as 0.0 used in vanilla SGD.

Settings on ResNet18. Here we follow the conventional supervised training setting used in ResNets (He et al., 2016) and evaluate our accelerated algorithms on ImageNet (Deng et al., 2009). For data augmentation in (He et al., 2016), it uses random crop and horizontal flipping with probability 0.5. For warm-up epochs, for all four accelerated algorithms, we set it as 5.0. For base learning rate, we respectively set it as 3×10^{-3}, 5×10^{-3}, 3×10^{-3}, and 1.2 for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. Moreover, we follow the default setting and use cosine learning rate decay. For weight decay, we respectively set it as 5×10^{-2}, 5×10^{-2}, 10^{-6}, and 10^{-3} for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. On ResNet18, all algorithms are trained for 90 epochs with minibatch size 512 by following the conventional setting.

Settings on ResNet50&101. For these two networks, we use "A2 training recipe" in (Wightman et al., 2021) to train them, since this training setting uses stronger data augmentation and largely improves CNNs' performance. Specifically, the data augmentation in (Wightman et al., 2021) uses random crop, horizontal flipping with probability, Mixup with parameter 0.1 (Zhang et al., 2018), CutMix with parameter 1.0 and probability 0.5 (Yun et al., 2019), and RandAugment (Cubuk et al., 2020) with $M = 7, N = 2$ and MSTD $= 0.5$. Moreover, it often use binary cross-entropy (BCE) loss for training.

On both ResNet50 and ResNet101, for base learning rate, we respectively set it as 2×10^{-3}, 8×10^{-3}, 1×10^{-3}, and 0.8 for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. Moreover, we follow the default setting and use cosine learning rate decay. On both ResNet50 and ResNet101, for weight decay, we respectively set it as 5×10^{-2}, 2×10^{-2}, 10^{-5}, and 5×10^{-4} for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. On both ResNet50 and ResNet101, for warm-up epoch number, we respectively set it as 5, 5, 10, 5 for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win.

Settings on ViT and PoolFormer. We follow the widely used official training setting of ViTs (Touvron et al., 2021; Yu et al., 2021). For this setting, data augmentation includes random crop, horizontal flipping with probability, Mixup with parameter 0.8 (Zhang et al., 2018), CutMix with parameter 1.0 and probability 0.5 (Yun et al., 2019), RandAugment (Cubuk et al., 2020) with $M = 9, N = 2$ and MSTD $= 0.5$, and Random Erasing with parameter $p = 0.25$. For training loss, we use cross entropy loss.

On both ViT-S and ViT-B, for base learning rate, we respectively set it as 2×10^{-3}, 5×10^{-3}, 1×10^{-4}, and 0.8 for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. Moreover, we follow the default setting and use cosine learning rate decay. On both ResNet50 and ResNet101, for weight decay, we respectively set it as 5×10^{-2}, 2×10^{-2}, 10^{-5}, and 5×10^{-4} for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. On both ResNet50 and ResNet101, for warm-up epoch number, we respectively set it as 5, 60, 30, 5 for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. For AdamW-Win, following the default setting in AdamW, its minibatch size is 1024 for ViT-S and 512 for ViT-B. For all other accelerated optimizer, their minibatch sizes are always 1024.

Settings on LSTM. On LSTM, for base learning rate, we respectively set it as 1×10^{-3}, 1×10^{-2}, 1×10^{-2}, and 15.0 for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. Moreover, we follow the default setting and divide the learning rate by 10 at epoch 100 and 145. For weight decay, we respectively set it as 2×10^{-2}, 5×10^{-2}, 1.8×10^{-6}, and 2×10^{-5} for AdamW-Win, LAMB-Win, Adam-Win and SGD-Win. We do not utilize the warmup strategy in this experiment. Following the default setting, we set minibatch size as 20.

Settings on Transformer-XL. On Transformer-XL, for base learning rate, we set it as 4×10^{-4} for Adam-Win. Moreover, we follow the default setting and use cosine learning rate decay. For weight decay, we set it as 10^{-6} for Adam-Win. For warm-up steps, we set it as 2000. Following the default setting, we set minibatch size as 60×4.

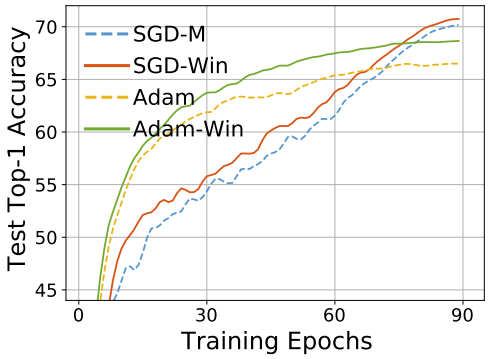

Figure 3: Test accuracy curve of SGD-Win and Adam-Win on ResNet18. See the curves of AdamW-Win and LAMB-Win in manuscript.

Test accuracy curves of SGD-Win and Adam-Win on ResNet18. Here we investigate the convergence behaviors of our accelerated algorithms and hope to explain their better test performance over their non-accelerated counterparts. In each sub-figure pair of Fig. 1, we plot the curves of training and test losses along with the training epochs on ResNet18 and ViT-B. One can find that our accelerated algorithms, *e.g.* AdamW-Win, show much faster convergence behaviors than their non-accelerated counterparts, *e.g.* AdamW. Moreover, SGD-Win also converges faster than Nesterove-accelerated SGD, i.e. SGD-M. We also plot the curves of test accuracy in Fig. 2, showing the superior convergence speed of AdamW-Win and LAMB-Win over their corresponding non-accelerated versions. Fig. 3 in Appendix B also reveals SGD-Win and Adam-Win enjoy faster convergence than their non-accelerated counterparts in terms of test accuracy. So these faster convergence behaviors could contribute to our accelerated algorithms for their higher performance over non-accelerated algorithms under the same computational cost.

C NOTATIONS

Here we first give some important notations used in this document. For brevity, we let

$$\boldsymbol{s}_k = \sqrt{\boldsymbol{v}_k + \nu}.$$

Since we have $\|\boldsymbol{m}_k\|_\infty \leq c_\infty$ and $\nu \leq \|\boldsymbol{v}_i + \nu\|_\infty \leq c_\infty^2 + \nu$ in Lemma 3 (see Appendix D), for brevity, let

$$c_1 := \nu^{0.5} \leq \|\boldsymbol{s}_k\|_\infty \leq c_2 := (c_\infty^2 + \nu)^{0.5}.$$

Also we define

$$\boldsymbol{w}_k := \boldsymbol{m}_k + \lambda \boldsymbol{x}_k * \boldsymbol{s}_k, \qquad \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = -\frac{\eta_k}{1+\lambda\eta_k}\frac{\boldsymbol{m}_k + \lambda \boldsymbol{x}_k * \boldsymbol{s}_k}{\boldsymbol{s}_k} = -\frac{\eta_k}{1+\lambda\eta_k}\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}.$$

Next, we introduce an virtual sequence $\{\boldsymbol{y}_k\}$ into the algorithm. In this way, we can rewrite the update steps in Algorithm 1 in the manuscript as its equivalent form (12):

$$\begin{cases} \boldsymbol{g}_k = \frac{1}{b}\sum_{i=1}^b \nabla f(\boldsymbol{z}_k; \boldsymbol{\zeta}_i); \\ \boldsymbol{m}_k = (1-\beta_1)\boldsymbol{m}_{k-1} + \beta_1 \boldsymbol{g}_k; \\ \boldsymbol{v}_k = (1-\beta_2)\boldsymbol{v}_{k-1} + \beta_2 \boldsymbol{g}_k^2; \\ \boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}\left(\boldsymbol{x}_k - \eta_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right) \\ \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k} \\ \boldsymbol{z}_{k+1} = \bar{\eta}_k \tau_k \boldsymbol{x}_{k+1} + \eta_k \tau_k \boldsymbol{y}_{k+1} \end{cases} \tag{12}$$

where $\boldsymbol{m}_0 = \boldsymbol{g}_0$ and $\boldsymbol{v}_0 = \boldsymbol{g}_0^2$.

For analysis, we further define

$$\widetilde{F}_k(\boldsymbol{z}_k) = F(\boldsymbol{z}) + \frac{\lambda_k}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2 = \mathbb{E}_{\boldsymbol{\zeta}}[f(\boldsymbol{z};\boldsymbol{\zeta})] + \frac{\lambda_k}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2, \tag{13}$$

where $\lambda_k = \frac{\lambda}{2}\sum_{i=1}^k \left(\frac{1-\mu}{2}\right)^i$ $(k > 0)$ and $\lambda_0 = 0$ in which $\mu = \frac{\beta_2 c_\infty^2}{\nu}$. In the following, we mainly use these notations to finish our proofs.

D AUXILIARY LEMMAS

Before giving our analysis, we first provide some important lemmas.

Lemma 3. *Suppose the sequence* $\{\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{z}_k\}$ *are updated by Eqn. (12). That is,* $\boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}\left(\boldsymbol{x}_k - \eta_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right), \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}, \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\boldsymbol{y}_{k+1}$, $\boldsymbol{s}_k = \sqrt{\boldsymbol{v}_k + \nu}$. *Then* $\{(\boldsymbol{m}_k, \boldsymbol{s}_k)\}$ *satisfies Assume* $c_{s,\infty} \leq \|g_k\|_\infty \leq c_\infty$, *then we have*

$$\|\boldsymbol{m}_k\|_\infty \leq c_\infty, \quad \|\boldsymbol{v}_i + \nu\|_\infty \leq c_\infty^2 + \nu, \quad \frac{\beta_2 c_\infty^2}{2(c_{s,\infty}^2 + \nu)} \leq \left\|\frac{\boldsymbol{s}_k}{\boldsymbol{s}_{k+1}}\right\|_\infty < 1 + \frac{\beta_2 c_\infty^2}{2(c_{s,\infty}^2 + \nu)}.$$

See its proof in Appendix G.1.

Lemma 4. *(Xie et al., 2022) Suppose the sequence* $\{\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{z}_k\}$ *are updated by Eqn. (12). That is,* $\boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}\left(\boldsymbol{x}_k - \eta_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right), \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}, \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\boldsymbol{y}_{k+1}$, $\boldsymbol{s}_k = \sqrt{\boldsymbol{v}_k + \nu}$. *Then* $\{\boldsymbol{x}_k\}$ *satisfies*

$$\frac{\lambda_{k+1}}{1-\mu}\|\boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_k}^2 \leq \lambda_k \|\boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2 + \lambda \langle \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \boldsymbol{x}_k\rangle_{\boldsymbol{s}_k} + \frac{\lambda}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2.$$

Lemma 5. *(Xie et al., 2022) Suppose the sequence* $\{\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{z}_k\}$ *are updated by Eqn. (12). That is,* $\boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}\left(\boldsymbol{x}_k - \eta_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right), \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}, \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\boldsymbol{y}_{k+1}$, $\boldsymbol{s}_k = \sqrt{\boldsymbol{v}_k + \nu}$. *Then* $\{\boldsymbol{z}_k\}$ *satisfies*

$$\begin{aligned}&\mathbb{E}\left[\|\boldsymbol{m}_k - \nabla F(\boldsymbol{z}_k)\|^2\right]\\ \leq&(1-\beta_1)\mathbb{E}\left[\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\|^2\right] + \frac{(1-\beta_1)^2 L^2}{\beta_1}\mathbb{E}\left[\|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\|^2\right] + \frac{\beta_1^2\sigma^2}{b}.\end{aligned}$$

Lemma 6. *Suppose the sequence* $\{\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{z}_k\}$ *are updated by Eqn. (12). That is,* $\boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}\left(\boldsymbol{x}_k - \eta_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right), \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}, \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\boldsymbol{y}_{k+1}$, $\boldsymbol{s}_k = \sqrt{\boldsymbol{v}_k + \nu}$. *By setting* $\eta_k = \eta$, $\bar{\eta}_k = \bar{\eta}$, $\beta_{1,k} = \beta_1$ *and* $\beta_{2,k} = \beta_2$, *then we have*

$$\begin{aligned}\boldsymbol{y}_{k+1} - (1+\lambda\bar{\eta})\boldsymbol{x}_{k+1} =& -\rho_{k+1}\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\frac{\bar{\eta}-\eta}{1+\lambda\eta}\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\\ \|\boldsymbol{y}_{k+1} - (1+\lambda\bar{\eta})\boldsymbol{x}_{k+1}\|^2 \leq& \frac{\rho_{k+1}(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2\\ \|\boldsymbol{z}_{k+1} - \boldsymbol{x}_{k+1}\|^2 \leq& \frac{\rho_{k+1}\eta\tau(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2,\\ \|\boldsymbol{z}_{k+1} - \boldsymbol{z}_k\|^2 \leq& \frac{2\bar{\eta}^2}{(1+\lambda\eta)^2}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|^2 + \frac{2\rho_{k+1}\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{(1-\eta\tau)}\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2,\end{aligned}$$

where $\rho_{k+1} = \eta\tau\rho_k$ *and* $\rho_1 = 1$.

See its proof in Appendix G.2.

Lemma 7. *Suppose the sequence* $\{\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{z}_k\}$ *are updated by Eqn. (12). That is,* $\boldsymbol{x}_{k+1} = \frac{1}{1+\lambda\eta_k}\left(\boldsymbol{x}_k - \eta_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right), \boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}, \boldsymbol{z}_{k+1} = \bar{\eta}_k\tau_k\boldsymbol{x}_{k+1} + \eta_k\tau_k\boldsymbol{y}_{k+1}$, $\boldsymbol{s}_k = \sqrt{\boldsymbol{v}_k + \nu}$. *By setting* $\eta_k = \eta$, $\bar{\eta}_k = \bar{\eta}$, $\beta_{1,k} = \beta_1$ *and* $\beta_{2,k} = \beta_2$, *then we have*

$$\mathbb{E}\left[\|\boldsymbol{m}_k - \nabla F(\boldsymbol{x}_k)\|^2\right] \leq 2(1-\beta_1)\mathbb{E}\left[\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\|^2\right] + \frac{2\Pi_k(1-\beta_1)^2L^2}{\beta_1} + \frac{2\beta_1^2\sigma^2}{b} + 2L\Pi_k',$$

where

$$\begin{aligned}\Pi_k :=& \frac{2\bar{\eta}^2}{(1+\lambda\eta)^2}\left\|\frac{\boldsymbol{w}_{k-1}}{\boldsymbol{s}_{k-1}}\right\|^2 + \frac{2\rho_k\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{(1-\eta\tau)}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2\\ \Pi_k' :=& \frac{\rho_k\eta\tau(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2,\end{aligned}$$

where $\rho_{k+1} = \eta\tau\rho_k$ *and* $\rho_1 = 1$.

see its proof in Appendix G.3.

E PROOF OF THEOREM 1

Proof. Recall our definition $\widetilde{F}_k(\boldsymbol{z}_k) = F(\boldsymbol{z}) + \frac{\lambda_k}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2 = \mathbb{E}_{\boldsymbol{\zeta}}[f(\boldsymbol{z};\boldsymbol{\zeta})] + \frac{\lambda_k}{2}\|\boldsymbol{z}\|_{\boldsymbol{s}_k}^2$, in the (13). By using the smoothness of $f(\boldsymbol{\theta};\boldsymbol{\zeta})$, we can obtain

$$
\begin{aligned}
&\widetilde{F}_{k+1}(\boldsymbol{x}_{k+1}) \\
\leq& F(\boldsymbol{x}_k) + \langle \nabla F(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2 + \lambda_{k+1}\|\boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_{k+1}}^2 \\
\overset{①}{\leq}& F(\boldsymbol{x}_k) + \langle \nabla F(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2 + \frac{\lambda_{k+1}}{1-\mu}\|\boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_k}^2 \\
\overset{②}{\leq}& F(\boldsymbol{x}_k) + \lambda_k\|\boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2 + \langle \nabla F(\boldsymbol{x}_k) + \lambda\boldsymbol{x}_k * \boldsymbol{s}_k, \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2 + \frac{\lambda}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2 \\
=& \widetilde{F}_k(\boldsymbol{x}_k) - \frac{\eta_k}{1+\lambda\eta_k}\left\langle \nabla F(\boldsymbol{x}_k) + \lambda\boldsymbol{x}_k * \boldsymbol{s}_k, \frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\rangle + \frac{L\eta_k^2}{2(1+\lambda\eta_k)^2}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|^2 + \frac{\lambda\eta_k^2}{2(1+\lambda\eta_k)^2}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|_{\boldsymbol{s}_k}^2 \\
=& \widetilde{F}_k(\boldsymbol{x}_k) + \frac{1}{2}\left\|\sqrt{\frac{\eta_k}{(1+\lambda\eta_k)\boldsymbol{s}_k}}(\nabla F(\boldsymbol{x}_k) + \lambda\boldsymbol{x}_k * \boldsymbol{s}_k - \boldsymbol{w}_k)\right\|^2 - \frac{1}{2}\left\|\sqrt{\frac{\eta_k}{(1+\lambda\eta_k)\boldsymbol{s}_k}}(\nabla F(\boldsymbol{x}_k) + \lambda\boldsymbol{x}_k * \boldsymbol{s}_k)\right\|^2 \\
&- \frac{1}{2}\left\|\sqrt{\frac{\eta_k}{(1+\lambda\eta_k)\boldsymbol{s}_k}}\boldsymbol{w}_k\right\|^2 + \frac{L\eta_k^2}{2(1+\lambda\eta_k)^2}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|^2 + \frac{\lambda\eta_k^2}{2(1+\lambda\eta_k)^2}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|_{\boldsymbol{s}_k}^2 \\
\leq& \widetilde{F}_k(\boldsymbol{x}_k) + \frac{\eta_k}{2c_1(1+\lambda\eta_k)}\|\nabla F(\boldsymbol{x}_k) - \boldsymbol{m}_k\|^2 - \frac{\eta_k}{2c_2(1+\lambda\eta_k)}\|\nabla F_k(\boldsymbol{x}_k)\|^2 \\
&- \frac{\eta_k}{2c_2(1+\lambda\eta_k)}\left[1 - \frac{c_2 L\eta_k}{c_1^2(1+\lambda\eta_k)} - \frac{c_2\lambda\eta_k}{c_1(1+\lambda\eta_k)}\right]\|\boldsymbol{w}_k\|^2 \\
\overset{③}{\leq}& \widetilde{F}_k(\boldsymbol{x}_k) + \frac{\eta_k}{2c_1(1+\lambda\eta_k)}\|\nabla F(\boldsymbol{x}_k) - \boldsymbol{m}_k\|^2 - \frac{\eta_k}{2c_2(1+\lambda\eta_k)}\|\nabla F_k(\boldsymbol{x}_k)\|^2 - \frac{\eta_k}{4c_2(1+\lambda\eta_k)}\|\boldsymbol{w}_k\|^2,
\end{aligned}
$$

where ① holds since Lemma 3 proves $\left\|\frac{\boldsymbol{s}_k}{\boldsymbol{s}_{k+1}}\right\|_\infty \in [1-\mu, 1+\mu]$ $(\forall p \in [0,1])$ in which $\mu = \frac{\beta_2 c_\infty^2}{\nu}$;
② holds because in Lemma 4, we have

$$
\frac{\lambda_{k+1}}{1-\mu}\|\boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_k}^2 \leq \lambda_k\|\boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2 + \lambda\langle \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \boldsymbol{x}_k\rangle_{\boldsymbol{s}_k} + \frac{\lambda}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2;
$$

③ holds, since we set $\eta_k \leq \frac{c_1^2(1+\lambda\eta_k)}{2c_2(L+\lambda c_1)}$ such that $\frac{c_2 L\eta_k}{c_1^2(1+\lambda\eta_k)} + \frac{c_2\lambda\eta_k}{c_1(1+\lambda\eta_k)} \leq \frac{1}{2}$.

From Lemma 7, by setting $\eta_k = \eta$, $\bar{\eta}_k = \bar{\eta}$ and $\beta_{1,k} = \beta_1$, we have

$$
\mathbb{E}\left[\|\boldsymbol{m}_k - \nabla F(\boldsymbol{x}_k)\|^2\right] \leq 2(1-\beta_1)\mathbb{E}\left[\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\|^2\right] + \frac{2\Pi_k(1-\beta_1)^2L^2}{\beta_1} + \frac{2\beta_1^2\sigma^2}{b} + 2L\Pi_k',
$$

where

$$
\begin{aligned}
\Pi_k :=& \frac{2\bar{\eta}^2}{(1+\lambda\eta)^2}\left\|\frac{\boldsymbol{w}_{k-1}}{\boldsymbol{s}_{k-1}}\right\|^2 + \frac{2\rho_k\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{(1-\eta\tau)}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2 \\
\Pi_k' :=& \frac{\rho_k\eta\tau(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2.
\end{aligned} \tag{14}
$$

Here $\rho_{k+1} = \eta\tau\rho_k$ and $\rho_1 = 1$. By considering $c_2 \geq \|\boldsymbol{s}_k\|_\infty \geq c_1$, we have

$$
\begin{aligned}
\Pi_k \leq \bar{\Pi}_k :=& \frac{2\bar{\eta}^2}{c_1(1+\lambda\eta)^2}\|\boldsymbol{w}_{k-1}\|^2 + \frac{2\rho_k\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{c_1(1-\eta\tau)}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\|\boldsymbol{w}_i\|^2 \\
\Pi_k' \leq \bar{\Pi}_k' :=& \frac{\rho_k\eta\tau(\bar{\eta}-\eta)^2}{c_1(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\|\boldsymbol{w}_i\|^2.
\end{aligned} \tag{15}
$$

Therefore, we have

$$\begin{aligned}
&\widetilde{F}_{k+1}(\boldsymbol{x}_{k+1}) \\
\leq &\widetilde{F}_k(\boldsymbol{x}_k) - \frac{\eta}{2c_2(1+\lambda\eta)}\left\|\nabla F_k(\boldsymbol{x}_k)\right\|^2 - \frac{\eta}{4c_2(1+\lambda\eta)}\left\|\boldsymbol{w}_k\right\|^2 \\
&+ \frac{\eta(1-\beta_1)}{c_1(1+\lambda\eta)}\mathbb{E}\left[\left\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\right\|^2\right] + \frac{\eta\bar{\Pi}_k(1-\beta_1)^2L^2}{c_1\beta_1(1+\lambda\eta)} + \frac{\eta\beta_1^2\sigma^2}{c_1(1+\lambda\eta)b} + \frac{\eta L\bar{\Pi}'_k}{c_1(1+\lambda\eta)}
\end{aligned} \tag{16}$$

From Lemma 5, we have

$$\begin{aligned}
&\mathbb{E}\left[\left\|\boldsymbol{m}_k - \nabla F(\boldsymbol{z}_k)\right\|^2\right] \\
\leq &(1-\beta_1)\mathbb{E}\left[\left\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\right\|^2\right] + \frac{(1-\beta_1)^2L^2}{\beta_1}\mathbb{E}\left[\left\|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\right\|^2\right] + \frac{\beta_1^2\sigma^2}{b} \\
\overset{①}{\leq} &(1-\beta_1)\mathbb{E}\left[\left\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\right\|^2\right] + \frac{(1-\beta_1)^2L^2\bar{\Pi}_k}{\beta_1} + \frac{\beta_1^2\sigma^2}{b}
\end{aligned} \tag{17}$$

where we use the results in Lemma 6 that

$$\left\|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\right\|^2 \leq \Pi_k \leq \bar{\Pi}_k.$$

Then we add Eqn. (18) and $\alpha\times$ (17) as follows:

$$\begin{aligned}
&\widetilde{F}_{k+1}(\boldsymbol{x}_{k+1}) + \alpha\mathbb{E}\left[\left\|\boldsymbol{m}_k - \nabla F(\boldsymbol{x}_k)\right\|^2\right] \\
\leq &\widetilde{F}_k(\boldsymbol{x}_k) - \frac{\eta}{2c_2(1+\lambda\eta)}\left\|\nabla F_k(\boldsymbol{x}_k)\right\|^2 - \frac{\eta}{4c_2(1+\lambda\eta)}\left\|\boldsymbol{w}_k\right\|^2 \\
&+ (1-\beta_1)\left(\frac{\eta}{c_1(1+\lambda\eta)} + \alpha\right)\mathbb{E}\left[\left\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\right\|^2\right] + \frac{\eta\bar{\Pi}_k(1-\beta_1)^2L^2}{c_1\beta_1(1+\lambda\eta)} + \frac{\eta\beta_1^2\sigma^2}{c_1(1+\lambda\eta)b} \\
&+ \frac{\eta L\bar{\Pi}'_k}{c_1(1+\lambda\eta)} + \frac{\alpha(1-\beta_1)^2L^2\bar{\Pi}_k}{\beta_1} + \frac{\alpha\beta_1^2\sigma^2}{b}
\end{aligned} \tag{18}$$

Then by setting $\alpha = \frac{\eta(1-\beta_1)}{c_1\beta_1(1+\lambda\eta)}$ and $G(\boldsymbol{x}_{k+1}) = \widetilde{F}_{k+1}(\boldsymbol{x}_{k+1}) + \frac{\eta(1-\beta_1)}{c_1\beta_1(1+\lambda\eta)}\mathbb{E}\left[\left\|\boldsymbol{m}_k - \nabla F(\boldsymbol{x}_k)\right\|^2\right]$, we can obtain

$$\begin{aligned}
G(\boldsymbol{x}_{k+1}) \leq &G(\boldsymbol{x}_k) - \frac{\eta}{2c_2(1+\lambda\eta)}\left\|\nabla F_k(\boldsymbol{x}_k)\right\|^2 - \frac{\eta}{4c_2(1+\lambda\eta)}\left\|\boldsymbol{w}_k\right\|^2 \\
&+ \frac{\eta\bar{\Pi}_k(1-\beta_1)^2L^2}{c_1\beta_1(1+\lambda\eta)} + \frac{\eta\beta_1^2\sigma^2}{c_1(1+\lambda\eta)b} + \frac{\eta L\bar{\Pi}'_k}{c_1(1+\lambda\eta)} \\
&+ \frac{\eta(1-\beta_1)^3L^2\bar{\Pi}_k}{c_1\beta_1^2(1+\lambda\eta)} + \frac{\eta(1-\beta_1)\beta_1\sigma^2}{c_1(1+\lambda\eta)b} \\
\leq &G(\boldsymbol{x}_k) - \frac{\eta}{2c_2(1+\lambda\eta)}\left\|\nabla\widetilde{F}(\boldsymbol{x}_k)\right\|^2 - \frac{\eta}{4c_2(1+\lambda\eta)}\left\|\boldsymbol{w}_k\right\|^2 \\
&+ \frac{\eta(1-\beta_1)^2L^2\bar{\Pi}_k}{c_1\beta_1^2(1+\lambda\eta)} + \frac{\eta L\bar{\Pi}'_k}{c_1(1+\lambda\eta)} + \frac{\eta\beta_1\sigma^2}{c_1(1+\lambda\eta)b}.
\end{aligned}$$

Then summing the above inequality from $k=0$ to $k=T-1$ gives

$$\begin{aligned}
&\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\left\|\nabla F_k(\boldsymbol{x}_k)\right\|^2 + \frac{1}{2}\left\|\boldsymbol{w}_k\right\|^2\right] \\
\leq &\frac{2c_2(1+\lambda\eta)}{\eta T}\left[G(\boldsymbol{x}_0) - G(\boldsymbol{x}_T)\right] + \frac{2c_2\beta_1\sigma^2}{c_1bT} + \frac{2c_2(1-\beta_1)^2L^2}{c_1\beta_1^2T}\sum_{k=0}^{T-1}\bar{\Pi}_k + \frac{2c_2L}{c_1T}\sum_{k=0}^{T-1}\bar{\Pi}'_k \\
\leq &\frac{2c_2(1+\lambda\eta)\Delta}{\eta T} + \frac{2c_2\beta_1\sigma^2}{c_1bT} + \frac{2c_2(1-\beta_1)^2L^2}{c_1\beta_1^2T}\sum_{k=0}^{T-1}\bar{\Pi}_k + \frac{2c_2L}{c_1T}\sum_{k=0}^{T-1}\bar{\Pi}'_k
\end{aligned}$$

where

$$
\begin{aligned}
&G(\boldsymbol{x}_0) - G(\boldsymbol{x}_T) \\
=&\widetilde{F}_0(\boldsymbol{x}_0) + \frac{\eta(1-\beta_1)}{c_1\beta_1(1+\lambda\eta)}\mathbb{E}\left[\|\boldsymbol{m}_{-1} - \nabla F(\boldsymbol{x}_{-1})\|^2\right] - \widetilde{F}_T(\boldsymbol{x}_T) - \frac{\eta(1-\beta_1)}{c_1\beta_1(1+\lambda\eta)}\mathbb{E}\left[\|\boldsymbol{m}_{T-1} - \nabla F(\boldsymbol{x}_{T-1})\|^2\right] \\
=&F(\boldsymbol{x}_0) - F(\boldsymbol{x}_T) - \lambda_T \|\boldsymbol{x}_T\|_{\boldsymbol{s}_T} - \frac{\eta(1-\beta_1)}{c_1\beta_1(1+\lambda\eta)}\mathbb{E}\left[\|\boldsymbol{m}_{T-1} - \nabla F(\boldsymbol{x}_{T-1})\|^2\right] \\
\leq&F(\boldsymbol{x}_0) - F(\boldsymbol{x}_T) \\
\leq&\Delta
\end{aligned}
$$

where $\Delta = F(\boldsymbol{x}_0) - F(\boldsymbol{x}_\star)$; $\boldsymbol{x}_{-1}$ and $\boldsymbol{m}_{-1}$ are two virtual points which satisfy $\boldsymbol{m}_{-1} = \nabla F(\boldsymbol{x}_{-1})$. Now we try to bound $\sum_{k=0}^{T-1}\bar{\Pi}_k$ and $\sum_{k=0}^{T-1}\bar{\Pi}_k$. Firstly, we have

$$
\begin{aligned}
\sum_{k=0}^{T-1}\bar{\Pi}_k =& \sum_{k=0}^{T-1}\left[\frac{2\bar{\eta}^2}{c_1(1+\lambda\eta)^2}\|\boldsymbol{w}_{k-1}\|^2 + \frac{2\rho_k\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{c_1(1-\eta\tau)}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\|\boldsymbol{w}_i\|^2\right] \\
=&\frac{2\bar{\eta}^2}{c_1(1+\lambda\eta)^2}\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_{k-1}\|^2\right] + \frac{2\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{c_1(1-\eta\tau)}\sum_{k=0}^{T-1}\rho_k\left[\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\|\boldsymbol{w}_i\|^2\right] \\
=&\frac{2\bar{\eta}^2}{c_1(1+\lambda\eta)^2}\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_{k-1}\|^2\right] + \frac{2\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{c_1(1-\eta\tau)}\sum_{k=0}^{T-1}\frac{1}{\rho_{k+1}}\left[\sum_{i=k}^{T-1}\rho_i\|\boldsymbol{w}_k\|^2\right] \\
\overset{①}{\leq}&\frac{2\bar{\eta}^2}{c_1(1+\lambda\eta)^2}\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_{k-1}\|^2\right] + \frac{2\bar{\eta}^2\tau(\bar{\eta}-\eta)^2}{c_1\eta(1-\eta\tau)^2}\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_k\|^2\right] \\
\overset{②}{\leq}&\frac{2\gamma^2\eta^2}{c_1}\left(\frac{1}{(1+\lambda\eta)^2} + \frac{\eta\tau(\gamma-1)^2}{(1-\eta\tau)^2}\right)\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_k\|^2\right]
\end{aligned}
$$

where ① holds since 1) $\sum_{i=k}^{T-1}\rho_i = \frac{\rho_k(1-\eta^{T-k}\tau^{T-k})}{1-\eta\tau} \leq \frac{\rho_k}{1-\eta\tau}$ and 2) $\rho_{k+1} = \eta\tau\rho_k$ and $\rho_1 = 1$; ② holds by setting $\bar{\eta} = \gamma\eta$. Similarly, we can bound

$$
\begin{aligned}
\sum_{k=0}^{T-1}\bar{\Pi}'_k =& \frac{\eta\tau(\bar{\eta}-\eta)^2}{c_1(1-\eta\tau)(1+\lambda\eta)^2}\sum_{k=0}^{T-1}\rho_k\left[\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\|\boldsymbol{w}_i\|^2\right] \\
\leq&\frac{(\bar{\eta}-\eta)^2}{c_1(1-\eta\tau)^2(1+\lambda\eta)^2}\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_k\|^2\right] \leq \frac{\eta^2(\gamma-1)^2}{c_1(1-\eta\tau)^2(1+\lambda\eta)^2}\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_k\|^2\right].
\end{aligned}
$$

Therefore, we have

$$
\begin{aligned}
\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\nabla F_k(\boldsymbol{x}_k)\|^2 + \frac{1}{2}\|\boldsymbol{w}_k\|^2\right] \leq& \frac{2c_2(1+\lambda\eta)\Delta}{\eta T} + \frac{c_2(1+\lambda\eta)\sigma^2}{c_1\beta_1 bT} + \frac{2c_2\beta_1\sigma^2}{c_1 bT} \\
&+ \frac{4c_2\gamma^2\eta^2(1-\beta_1)^2L^2}{c_1^2\beta_1^2 T}\left(\frac{1}{(1+\lambda\eta)^2} + \frac{\eta\tau(\gamma-1)^2}{(1-\eta\tau)^2}\right)\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_k\|^2\right] \\
&+ \frac{2c_2L\eta^2(\gamma-1)^2}{c_1^2(1-\eta\tau)^2(1+\lambda\eta)^2 T}\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_k\|^2\right], \\
\overset{①}{\leq}& \frac{2c_2(1+\lambda\eta)\Delta}{\eta T} + \frac{2c_2\beta_1\sigma^2}{c_1 b} + \frac{1}{4T}\sum_{k=0}^{T-1}\left[\|\boldsymbol{w}_k\|^2\right]
\end{aligned}
$$

where ① holds since we choose proper η and β_1 such that

$$
\begin{aligned}
\frac{4c_2\gamma^2\eta^2(1-\beta_1)^2L^2}{c_1^2\beta_1^2}\left(\frac{1}{(1+\lambda\eta)^2} + \frac{\eta\tau(\gamma-1)^2}{(1-\eta\tau)^2}\right) \leq& \frac{1}{8} \\
\frac{2c_2L\eta^2(\gamma-1)^2}{c_1^2(1-\eta\tau)^2(1+\lambda\eta)^2} \leq& \frac{1}{8}
\end{aligned}
\tag{19}
$$

Now we discuss how to select η and β_1 such that (19) holds. To begin with, we have $\frac{1}{(1+\lambda\eta)^2} \leq 1$ and $\frac{\eta\tau(\gamma-1)^2}{(1-\eta\tau)^2} = \frac{1+\gamma+\lambda\gamma\eta}{\gamma(1+\lambda\eta)}(\gamma-1)^2 \leq \frac{1+\gamma}{\gamma}(\gamma-1)^2 < \gamma^2$ in which $\eta\tau = \frac{\eta}{\eta+\gamma\eta+\lambda\gamma\eta^2}$. Then we only need

$$\eta \leq \min\left(\frac{c_1\beta_1}{4\sqrt{2}c_2^{0.5}(\gamma+1)^2(1-\beta_1)L}, \frac{c_1(1-\eta\tau)(1+\lambda\eta)}{4c_2^{0.5}L^{0.5}(\gamma-1)}\right)$$

So we arrive at

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\nabla F_k(\boldsymbol{x}_k)\|^2 + \frac{1}{4}\|\boldsymbol{w}_k\|^2\right] \leq \frac{2c_2(1+\lambda\eta)\Delta}{\eta T} + \frac{2c_2\beta_1\sigma^2}{c_1 b} \overset{①}{\leq} \epsilon^2,$$

where we set $T \geq \frac{6c_2(1+\lambda\eta)\Delta}{\eta\epsilon^2}$ and $\beta_1 \leq \frac{c_1 b\epsilon^2}{6c_2\sigma^2}$. This result directly bounds

$$\begin{aligned}\frac{1}{T}\sum_{k=0}^{T-1}\|\boldsymbol{s}_k * (\boldsymbol{x}_k - \boldsymbol{x}_{k+1})\|^2 =& \frac{\eta^2}{(1+\lambda\eta)^2 T}\sum_{k=0}^{T-1}\|\boldsymbol{m}_k + \lambda\boldsymbol{x}_k * \boldsymbol{s}_k\|^2 \\ \leq& \frac{\eta^2}{(1+\lambda\eta)^2 T}\sum_{k=0}^{T-1}\|\boldsymbol{w}_k\|^2 \leq \frac{4\eta^2\epsilon^2}{(1+\lambda\eta)^2}.\end{aligned}$$

Moreover, from Lemma 6, we have

$$\begin{aligned}\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\boldsymbol{y}_k - (1+\lambda\bar{\eta})\boldsymbol{x}_k\|^2 \overset{①}{\leq}& \frac{1}{T}\sum_{k=0}^{T-1}\frac{\rho_k(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2 \overset{②}{=} \frac{1}{\eta\tau T}\sum_{k=0}^{T-1}\Pi'_k \\ \frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\boldsymbol{z}_k - \boldsymbol{x}_k\|^2 \overset{①}{\leq}& \frac{1}{T}\sum_{k=0}^{T-1}\frac{\rho_k\eta\tau(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2 \overset{②}{=} \frac{1}{T}\sum_{k=0}^{T-1}\Pi'_k \\ \frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\boldsymbol{z}_{k+1} - \boldsymbol{z}_k\|^2 \overset{①}{\leq}& \frac{1}{T}\sum_{k=0}^{T-1}\left[\frac{2\bar{\eta}^2}{(1+\lambda\eta)^2}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|^2 + \frac{2\rho_{k+1}\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{(1-\eta\tau)}\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2\right] \\ \overset{②}{=}& \frac{1}{T}\sum_{k=0}^{T-1}\Pi_k\end{aligned}$$

where ① holds by using Lemma 6; ② holds by using the definition in Eqn. (14). Now remaining task is to upper bound $\frac{1}{T}\sum_{k=0}^{T-1}\Pi_k$ and $\frac{1}{T}\sum_{k=0}^{T-1}\Pi'_k$. From the bound in Eqn. (15) and the following bound on $\frac{1}{T}\sum_{k=0}^{T-1}\bar{\Pi}_k$ and $\frac{1}{T}\sum_{k=0}^{T-1}\bar{\Pi}'_k$, we have

$$\begin{aligned}\frac{1}{T}\sum_{k=0}^{T-1}\Pi_k \leq \frac{1}{T}\sum_{k=0}^{T-1}\bar{\Pi}_k \leq& \frac{2\gamma^2\eta^2}{c_1 T}\left(\frac{1}{(1+\lambda\eta)^2} + \frac{\eta\tau(\gamma-1)^2}{(1-\eta\tau)^2}\right)\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{w}_k\|^2\right] \overset{①}{\leq} \frac{c_1^3\beta_1^2\epsilon^2}{4c_2(1-\beta_1)^3L^2} \\ \frac{1}{T}\sum_{k=0}^{T-1}\Pi'_k \leq \frac{1}{T}\sum_{k=0}^{T-1}\bar{\Pi}'_k \leq& \frac{\eta^2(\gamma-1)^2}{c_1(1-\eta\tau)^2(1+\lambda\eta)^2 T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{w}_k\|^2\right] \overset{①}{\leq} \frac{c_1\epsilon^2}{4c_2 L}\end{aligned}$$

where ① holds, since 1) $\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\boldsymbol{w}_k\|^2 \leq 4\epsilon^2$; 2) we use the results in Eqn. (19) to obtain

$$\begin{aligned}\frac{2\gamma^2\eta^2}{c_1}\left(\frac{1}{(1+\lambda\eta)^2} + \frac{\eta\tau(\gamma-1)^2}{(1-\eta\tau)^2}\right) \leq& \frac{c_1^3\beta_1^2}{16c_2(c_1^2\beta_1 + 1 - \beta_1)(1-\beta_1)^2L^2} \leq \frac{c_1^3\beta_1^2}{16c_2(1-\beta_1)^3L^2} \\ \frac{\eta^2(\gamma-1)^2}{c_1(1-\eta\tau)^2(1+\lambda\eta)^2} \leq& \frac{c_1}{16c_2 L}\end{aligned}$$

Therefore, we have

$$\begin{aligned}\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\boldsymbol{y}_k - (1+\lambda\bar{\eta})\boldsymbol{x}_k\|^2 \leq& \frac{c_1\epsilon^2}{4c_2 L\eta\tau} = \frac{c_1(1+\gamma+\lambda\gamma\eta)\epsilon^2}{4c_2 L} \\ \frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\boldsymbol{z}_k - \boldsymbol{x}_k\|^2 \leq& \frac{c_1\epsilon^2}{4c_2 L} \\ \frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\boldsymbol{z}_{k+1} - \boldsymbol{z}_k\|^2 \leq& \frac{c_1^3\beta_1^2\epsilon^2}{4c_2(1-\beta_1)^3L^2}.\end{aligned}$$

Besides, we have

$$\begin{aligned}\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{m}_k-\nabla F(\boldsymbol{x}_k)\|^2\right]\leq&\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{m}_k+\lambda\boldsymbol{x}_k*\boldsymbol{s}_k-\nabla F(\boldsymbol{x}_k)-\lambda\boldsymbol{x}_k*\boldsymbol{s}_k\|^2\right]\\ \leq&\frac{2}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{m}_k+\lambda\boldsymbol{x}_k*\boldsymbol{s}_k\|^2+\|\nabla F(\boldsymbol{x}_k)+\lambda\boldsymbol{x}_k*\boldsymbol{s}_k\|^2\right]\\ =&\frac{2}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{m}_k+\lambda\boldsymbol{x}_k*\boldsymbol{s}_k\|^2+\|\nabla F_k(\boldsymbol{x}_k)\|^2\right]\\ \overset{①}{\leq}&2\left[\epsilon^2+\frac{3}{4}\times 4\epsilon^2\right]\leq 8\epsilon^2.\end{aligned}$$

where in ① we use $\boldsymbol{w}_k=\boldsymbol{m}_k+\lambda\boldsymbol{x}_k*\boldsymbol{s}_k$. In this way, we have

$$\begin{aligned}\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{m}_k-\nabla F(\boldsymbol{z}_k)\|^2\right]\leq&\frac{2}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{m}_k-\nabla F(\boldsymbol{x}_k)\|^2+\|\nabla F(\boldsymbol{x}_k)-\nabla F(\boldsymbol{z}_k)\|^2\right]\\ \leq&16\epsilon^2+\frac{2L^2}{T}\sum_{k=0}^{T-1}\mathbb{E}\left[\|\boldsymbol{x}_k-\boldsymbol{z}_k\|^2\right]\\ \leq&16\epsilon^2+\frac{c_1L\epsilon^2}{2c_2}=\frac{(c_1L+32c_2)}{2c_2}\epsilon^2.\end{aligned}$$

From Eqn. (17), we have

$$\mathbb{E}\left[\|\boldsymbol{m}_k-\nabla F(\boldsymbol{z}_k)\|^2\right]\leq(1-\beta_1)\mathbb{E}\left[\|\boldsymbol{m}_{k-1}-\nabla F(\boldsymbol{z}_{k-1})\|^2\right]+\frac{(1-\beta_1)^2L^2\bar{\Pi}_k}{\beta_1}+\frac{\beta_1^2\sigma^2}{b}.$$

By setting $\rho_{k+1}=(1-\beta_1)\rho_k$ and $\rho_0=1$, we have

$$\begin{aligned}\frac{1}{\rho_k}\mathbb{E}\left[\|\boldsymbol{m}_k-\nabla F(\boldsymbol{z}_k)\|^2\right]\leq&\frac{1}{\rho_{k-1}}\mathbb{E}\left[\|\boldsymbol{m}_{k-1}-\nabla F(\boldsymbol{z}_{k-1})\|^2\right]+\frac{(1-\beta_1)^2L^2\bar{\Pi}_k}{\beta_1\rho_k}+\frac{\beta_1^2\sigma^2}{b\rho_k}\\ \leq&\frac{1}{\rho_0}\mathbb{E}\left[\|\boldsymbol{m}_0-\nabla F(\boldsymbol{z}_0)\|^2\right]+\sum_{i=1}^{k}\left[\frac{(1-\beta_1)^2L^2\bar{\Pi}_i}{\beta_1\rho_i}+\frac{\beta_1^2\sigma^2}{b\rho_i}\right]\end{aligned}$$

For all hyper-parameters, we put their constrains together:

$$\beta_1\leq\frac{c_1b\epsilon^2}{6c_2\sigma^2},$$

where $c_1=\nu^{0.5}\leq\|\boldsymbol{s}_k\|_\infty\leq\left(c_\infty^2+\nu\right)^{0.5}=c_2$.

For η, it should satisfy

$$\eta\leq\min\left(\frac{c_1\beta_1}{4\sqrt{2}c_2^{0.5}(\gamma+1)^2(1-\beta_1)L},\frac{c_1(1-\eta\tau)(1+\lambda\eta)}{4c_2^{0.5}L^{0.5}(\gamma-1)},\frac{c_1^2(1+\lambda\eta)}{2c_2(L+\lambda c_1)}\right)$$

Considering $(1-\eta\tau)(1+\lambda\eta)=\frac{\gamma(1+\lambda\eta)^2}{1+\gamma+\lambda\gamma\eta}\geq\frac{\gamma}{\gamma+1}$, $\frac{\gamma}{(\gamma-1)(\gamma+1)}\geq\frac{1}{\gamma}$ and $c_1=\nu^{0.5}<<1$, then we have

$$\begin{aligned}\eta\leq&\min\left(\frac{c_1\beta_1}{36c_2^{0.5}(\gamma+1)^2L},\frac{c_1}{4c_2^{0.5}\gamma L^{0.5}},\frac{c_1^2}{2c_2(L+\lambda c_1)}\right)\\ =&\min\left(\frac{c_1b\epsilon^2}{36c_2^{1.5}(\gamma+1)^2L\sigma^2},\frac{c_1}{4c_2^{0.5}\gamma L^{0.5}},\frac{c_1^2(1+\lambda\eta)}{2c_2(L+\lambda c_1)}\right)=\frac{c_1b\epsilon^2}{36c_2^{1.5}(\gamma+1)^2L\sigma^2}\end{aligned}$$

where ν is often much smaller than one, and β_1 is very small. For T, we have

$$\begin{aligned}T\geq&\frac{6c_2(1+\lambda\eta)\Delta}{\eta\epsilon^2}=\mathcal{O}\left(\frac{c_2\Delta}{\epsilon^2}\frac{36c_2^{1.5}(\gamma+1)^2L\sigma^2}{c_1b\epsilon^2}\right)\\ =&\mathcal{O}\left(\frac{c_2^{2.5}(\gamma+1)^2\Delta\sigma^2L}{c_1b\epsilon^4}\right)=\mathcal{O}\left(\frac{c_\infty^{2.5}\Delta\sigma^2L}{\nu^{0.5}b\epsilon^4}\right).\end{aligned}$$

Now we compute the stochastic gradient complexity. For T iterations, the complexity is

$$\mathcal{O}(Tb) = \mathcal{O}\left(\frac{c_2^{2.5}(\gamma+1)^2\Delta\sigma^2 L}{c_1\epsilon^4}\right) = \mathcal{O}\left(\frac{c_\infty^{2.5}\Delta\sigma^2 L}{\nu^{0.5}\epsilon^4}\right).$$

The proof is completed.

□

F PROOFS OF THEOREM 2

Proof. Recall our definition $\widetilde{F}_k(\boldsymbol{\theta}_k) = F(\boldsymbol{\theta}) + \frac{\lambda_k}{2}\|\boldsymbol{\theta}\|_2^2 = \mathbb{E}_{\boldsymbol{\zeta}}[f(\boldsymbol{\theta};\boldsymbol{\zeta})] + \frac{\lambda_k}{2}\|\boldsymbol{\theta}\|_2^2$ in the (13). By setting $\beta_1' = 1-\beta_1$, then we have $\|\boldsymbol{m}_k\|_\infty \le c_\infty$ by using Lemma 3 (see Appendix D). Also we define

$$\boldsymbol{w}_k := \boldsymbol{m}_k + \lambda\boldsymbol{x}_k, \qquad \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = -\frac{\eta_k}{1+\lambda\eta_k}(\boldsymbol{m}_k + \lambda\boldsymbol{x}_k) = -\frac{\eta_k}{1+\lambda\eta_k}\boldsymbol{w}_k.$$

By using the smoothness of $f(\boldsymbol{\theta};\boldsymbol{\zeta})$, we can obtain

$$\begin{aligned}
&\widetilde{F}_{k+1}(\boldsymbol{x}_{k+1})\\
\le& F(\boldsymbol{x}_k) + \langle\nabla F(\boldsymbol{x}_k), \boldsymbol{x}_{k+1}-\boldsymbol{x}_k\rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\|^2 + \lambda_{k+1}\|\boldsymbol{x}_{k+1}\|_2^2\\
\overset{①}{\le}& F(\boldsymbol{x}_k) + \lambda_k\|\boldsymbol{x}_k\|_2^2 + \langle\nabla F(\boldsymbol{x}_k) + \lambda\boldsymbol{x}_k, \boldsymbol{x}_{k+1}-\boldsymbol{x}_k\rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\|^2 + \frac{\lambda}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\|_2^2\\
=& \widetilde{F}_k(\boldsymbol{x}_k) - \frac{\eta_k}{1+\lambda\eta_k}\langle\nabla F(\boldsymbol{x}_k) + \lambda\boldsymbol{x}_k, \boldsymbol{w}_k\rangle + \frac{L\eta_k^2}{2(1+\lambda\eta_k)^2}\|\boldsymbol{w}_k\|_2^2 + \frac{\lambda\eta_k^2}{2(1+\lambda\eta_k)^2}\|\boldsymbol{w}_k\|_2^2\\
=& \widetilde{F}_k(\boldsymbol{x}_k) + \frac{1}{2}\left\|\sqrt{\frac{\eta_k}{(1+\lambda\eta_k)}}(\nabla F(\boldsymbol{x}_k) + \lambda\boldsymbol{x}_k - \boldsymbol{w}_k)\right\|^2 - \frac{1}{2}\left\|\sqrt{\frac{\eta_k}{(1+\lambda\eta_k)}}(\nabla F(\boldsymbol{x}_k) + \lambda\boldsymbol{x}_k)\right\|^2\\
&- \frac{1}{2}\left\|\sqrt{\frac{\eta_k}{(1+\lambda\eta_k)}}\boldsymbol{w}_k\right\|^2 + \frac{L\eta_k^2}{2(1+\lambda\eta_k)^2}\|\boldsymbol{w}_k\|^2 + \frac{\lambda\eta_k^2}{2(1+\lambda\eta_k)^2}\|\boldsymbol{w}_k\|_2^2\\
\le& \widetilde{F}_k(\boldsymbol{x}_k) + \frac{\eta_k}{(1+\lambda\eta_k)}\|\nabla F(\boldsymbol{x}_k) - \boldsymbol{m}_k\|^2 - \frac{\eta_k}{(1+\lambda\eta_k)}\|\nabla F_k(\boldsymbol{x}_k)\|^2 - \frac{\eta_k}{(1+\lambda\eta_k)}\left[1 - \frac{(\lambda+L)\eta_k}{(1+\lambda\eta_k)}\right]\|\boldsymbol{w}_k\|^2\\
\overset{②}{\le}& \widetilde{F}_k(\boldsymbol{x}_k) + \frac{\eta_k}{2(1+\lambda\eta_k)}\|\nabla F(\boldsymbol{x}_k) - \boldsymbol{m}_k\|^2 - \frac{\eta_k}{2(1+\lambda\eta_k)}\|\nabla F_k(\boldsymbol{x}_k)\|^2 - \frac{\eta_k}{4(1+\lambda\eta_k)}\|\boldsymbol{w}_k\|^2,
\end{aligned}$$

where ① holds because in Lemma 4, we have

$$\frac{\lambda_{k+1}}{1-\mu}\|\boldsymbol{x}_{k+1}\|_{\boldsymbol{s}_k}^2 \le \lambda_k\|\boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2 + \lambda\langle\boldsymbol{x}_{k+1}-\boldsymbol{x}_k, \boldsymbol{x}_k\rangle_{\boldsymbol{s}_k} + \frac{\lambda}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\|_{\boldsymbol{s}_k}^2,$$

where $\boldsymbol{s}_k$ could be $\mathbf{1}$; ① holds, since we set $\eta_k \le \frac{(1+\lambda\eta_k)}{2(L+\lambda)}$ such that $\frac{L\eta_k}{(1+\lambda\eta_k)} + \frac{\lambda\eta_k}{(1+\lambda\eta_k)} \le \frac{1}{2}$.

Then in the following, we can directly follow the proof of Theorem 1. This is because the only difference between accelerated SGD and AdamW is that SGD has no the second-order moment $\boldsymbol{v}_k$, while AdamW has. By let $\boldsymbol{s}_k = \mathbf{1}$ in accelerated AdamW and setting $\beta_1' = 1-\beta_1$ in accelerated SGD, then they share the exact the same updating rules. So after setting $\beta_1' = 1-\beta_1$ in accelerated SGD, to follow the proofs of Theorem 1, we only need to verify whether the auxiliary lemmas and the proof process of Theorem 1 hold for $\boldsymbol{s}_k = \mathbf{1}$. This is the true case. Please check our auxiliary lemmas, including Lemma 3 $\sim$ 7, and the proof process of Theorem 1. Consider $\boldsymbol{s}_k = \mathbf{1}$ in accelerated SGD, we have $c_1 := 1 \le \|\boldsymbol{s}_k\|_\infty \le c_2 := 1$.

In this way, by setting $\eta \le \frac{b\epsilon^2}{36(\gamma+1)^2 L\sigma^2}$, $\beta_1 \le \frac{b\epsilon^2}{6\sigma^2}$ and $\beta_1' = 1-\beta_1$, after $T = \mathcal{O}\big(\frac{\Delta\sigma^2 L}{\nu^{0.5}b\epsilon^4}\big)$ iterations with minibatch size b and $\Delta = F(\boldsymbol{x}_0) - F(\boldsymbol{x}_\star)$, the sequence $\{(\boldsymbol{x}_k, \boldsymbol{z}_k)\}_{k=0}^T$ generated by accelerated SGD satisfies the following four properties.
a) The gradient $\nabla F_k(\boldsymbol{x}_k)$ of the sequence $\{\boldsymbol{x}_k\}_{k=0}^T$ can be upper bounded by

$$\frac{1}{T}\sum\nolimits_{k=0}^{T-1}\mathbb{E}\left[\|\nabla F_k(\boldsymbol{x}_k)\|^2 + \frac{1}{4}\|\boldsymbol{m}_k + \lambda\boldsymbol{x}_k\|^2\right] \le \epsilon^2,$$

b) The gradient moment $\boldsymbol{m}_k$ can well estimate the full gradient $\nabla F(\boldsymbol{z}_k)$ at the point $\boldsymbol{z}_k$:

$$\frac{1}{T}\sum\nolimits_{k=0}^{T-1}\mathbb{E}\Big[\|\boldsymbol{m}_k-\nabla F(\boldsymbol{z}_k)\|_2^2\Big]\leq 8\epsilon^2.$$

c) The sequence $\{\boldsymbol{x}_k,\boldsymbol{z}_k\}$ satisfies

$$\frac{1}{T}\sum_{k=0}^{T-1}\Big\{\mathbb{E}\|\boldsymbol{x}_k-\boldsymbol{x}_{k+1}\|_2^2,\mathbb{E}\|\boldsymbol{z}_{k+1}-\boldsymbol{z}_k\|_2^2,\mathbb{E}\|\boldsymbol{z}_k-\boldsymbol{x}_k\|_2^2\Big\}\leq\Big\{\frac{\eta^2\epsilon^2}{4(1+\lambda\eta)^2},\frac{\beta_1^2\epsilon^2}{4(1-\beta_1)^3L^2},\frac{\epsilon^2}{4L}\Big\}.$$

d) The total stochastic gradient complexity to achieve the above three properties is $\mathcal{O}\big(\frac{\Delta\sigma^2L}{\epsilon^4}\big)$.

The proof is completed. □

G Proofs of Auxiliary Lemmas

G.1 Proof of Lemma 3

Proof. To begin with, we assume that $\forall t\leq k$, it holds

$$\|\boldsymbol{m}_t\|_\infty\leq c_\infty,\qquad\|\boldsymbol{v}_t+\nu\|_\infty\leq c_\infty+\nu$$

Then we consider the case where $t=k+1$ as follows

$$\begin{aligned}\|\boldsymbol{m}_{k+1}\|_\infty&=\|(1-\beta_1)\boldsymbol{m}_k+\beta_1\boldsymbol{g}_k\|_\infty\leq(1-\beta_1)\|\boldsymbol{m}_k\|_\infty+\beta_1\|\boldsymbol{g}_k\|_\infty\leq c_\infty,\\ \|\boldsymbol{v}_{k+1}\|_\infty&=\big\|(1-\beta_2)\boldsymbol{v}_k+\beta_2\boldsymbol{g}_k^2\big\|_\infty\leq(1-\beta_2)\|\boldsymbol{v}_k\|_\infty+\beta_2\big\|\boldsymbol{g}_k^2\big\|_\infty\leq c_\infty^2.\end{aligned}$$

Then we derive the second results as follows:

$$\Big\|\sqrt{\frac{\boldsymbol{v}_k+\nu}{\boldsymbol{v}_{k+1}+\nu}}\Big\|_\infty=\Big\|\sqrt{1+\frac{\boldsymbol{v}_k-\boldsymbol{v}_{k+1}}{\boldsymbol{v}_{k+1}+\nu}}\Big\|_\infty=\Big\|\sqrt{1+\frac{\beta_2(\boldsymbol{v}_k-\boldsymbol{g}_k^2)}{\boldsymbol{v}_{k+1}+\nu}}\Big\|_\infty.$$

Therefore, we have

$$1-\frac{\beta_2c_\infty^2}{2(c_{s,\infty}^2+\nu)}<\sqrt{1-\frac{\beta_2c_\infty^2}{c_{s,\infty}^2+\nu}}\leq\Big\|\sqrt{\frac{\boldsymbol{v}_k+\nu}{\boldsymbol{v}_{k+1}+\nu}}\Big\|_\infty\leq\sqrt{1+\frac{\beta_2c_\infty^2}{c_{s,\infty}^2+\nu}}<1+\frac{\beta_2c_\infty^2}{2(c_{s,\infty}^2+\nu)}.$$

We complete the proof. □

G.2 Proof of Lemma 6

Proof. To begin with, we have

$$\begin{aligned}&\boldsymbol{y}_{k+1}-(1+\lambda\bar{\eta}_k)\boldsymbol{x}_{k+1}\\ =&\boldsymbol{z}_k-\bar{\eta}_k\frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}-\frac{1+\lambda\bar{\eta}_k}{1+\lambda\eta_k}\left(\boldsymbol{x}_k-\eta_k\frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right)\\ =&\bar{\eta}_{k-1}\tau_{k-1}\boldsymbol{x}_k+\eta_{k-1}\tau_{k-1}\boldsymbol{y}_k-\bar{\eta}_k\frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}-\frac{1+\lambda\bar{\eta}_k}{1+\lambda\eta_k}\left(\boldsymbol{x}_k-\eta_k\frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right)\\ =&\eta_{k-1}\tau_{k-1}\left(\boldsymbol{y}_k-(1+\lambda\bar{\eta}_{k-1})\boldsymbol{x}_k\right)-\left(\bar{\eta}_k-\frac{1+\lambda\bar{\eta}_{k-1}}{1+\lambda\eta_{k-1}}\eta_k\right)\frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}+\frac{\lambda(\eta_k-\bar{\eta}_k)}{1+\lambda\eta_k}\boldsymbol{x}_k\\ \overset{①}{=}&\eta_{k-1}\tau_{k-1}\left(\boldsymbol{y}_k-(1+\lambda\bar{\eta}_{k-1})\boldsymbol{x}_k\right)-\left(\bar{\eta}_k-\frac{1+\lambda\bar{\eta}_{k-1}}{1+\lambda\eta_{k-1}}\eta_k\right)\frac{\boldsymbol{w}_k-\lambda\sqrt{\boldsymbol{v}_k}}{\boldsymbol{s}_k}+\frac{\lambda(\eta_k-\bar{\eta}_k)}{1+\lambda\eta_k}\boldsymbol{x}_k\\ =&\eta_{k-1}\tau_{k-1}\left(\boldsymbol{y}_k-(1+\lambda\bar{\eta}_{k-1})\boldsymbol{x}_k\right)-\left(\bar{\eta}_k-\frac{1+\lambda\bar{\eta}_{k-1}}{1+\lambda\eta_{k-1}}\eta_k\right)\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\\ &+\left(\lambda\bar{\eta}_k-\frac{1+\lambda\bar{\eta}_{k-1}}{1+\lambda\eta_{k-1}}\lambda\eta_k+\frac{\lambda(\eta_k-\bar{\eta}_k)}{1+\lambda\eta_k}\right)\boldsymbol{x}_k\\ \overset{②}{=}&\eta\tau\left(\boldsymbol{y}_k-(1+\lambda\bar{\eta})\boldsymbol{x}_k\right)-\frac{\bar{\eta}-\eta}{1+\lambda\eta}\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\end{aligned}$$

where ① holds since $\boldsymbol{w}_k := \boldsymbol{m}_k + \lambda \boldsymbol{x}_k * \boldsymbol{s}_k$; ② holds since we set all $\eta_k = \eta$ and $\bar{\eta}_k = \bar{\eta}$ which gives $\tau_k = \tau = \frac{1}{\eta+\bar{\eta}+\lambda\eta\bar{\eta}}$. Therefore, by defining $\rho_{k+1} = \eta\tau\rho_k$ and $\rho_1 = 1$, then we have

$$\frac{\boldsymbol{y}_{k+1} - (1+\lambda\bar{\eta})\boldsymbol{x}_{k+1}}{\rho_{k+1}} = \frac{\boldsymbol{y}_k - (1+\lambda\bar{\eta})\boldsymbol{x}_k}{\rho_k} - \frac{1}{\rho_{k+1}}\frac{\bar{\eta}-\eta}{1+\lambda\eta}\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}(k \geq 1)$$

For $k = 0$, we have

$$\begin{aligned}
\boldsymbol{y}_1 - (1+\lambda\bar{\eta})\boldsymbol{x}_1 =& \boldsymbol{z}_0 - \bar{\eta}\frac{\boldsymbol{m}_0}{\boldsymbol{s}_0} - \frac{1+\lambda\bar{\eta}}{1+\lambda\eta}\left(\boldsymbol{x}_0 - \eta\frac{\boldsymbol{m}_0}{\boldsymbol{s}_0}\right) \\
=& \boldsymbol{z}_0 - \bar{\eta}\frac{\boldsymbol{w}_0 - \lambda\boldsymbol{s}_0 * \boldsymbol{x}_0}{\boldsymbol{s}_0} - \frac{1+\lambda\bar{\eta}}{1+\lambda\eta}\left(\boldsymbol{x}_0 - \eta\frac{\boldsymbol{w}_0 - \lambda\boldsymbol{s}_0 * \boldsymbol{x}_0}{\boldsymbol{s}_0}\right) \\
=& \boldsymbol{z}_0 - \boldsymbol{x}_0 - \frac{\bar{\eta}-\eta}{1+\lambda\eta}\frac{\boldsymbol{w}_0}{\boldsymbol{s}_0}.
\end{aligned}$$

In this way, one can obtain

$$\begin{aligned}
\frac{\boldsymbol{y}_{k+1} - (1+\lambda\bar{\eta})\boldsymbol{x}_{k+1}}{\rho_{k+1}} =& \boldsymbol{z}_0 - \boldsymbol{x}_0 - \frac{\bar{\eta}-\eta}{1+\lambda\eta}\frac{\boldsymbol{w}_0}{\boldsymbol{s}_0} - \sum_{i=1}^{k}\frac{1}{\rho_{i+1}}\frac{\bar{\eta}-\eta}{1+\lambda\eta}\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i} \\
=& -\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\frac{\bar{\eta}-\eta}{1+\lambda\eta}\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}
\end{aligned}$$

where ① hold since $\boldsymbol{z}_0 = \boldsymbol{x}_0$ and $\rho_1 = 1$. Then we can upper bound

$$\begin{aligned}
\left\|\frac{\boldsymbol{y}_{k+1} - (1+\lambda\bar{\eta})\boldsymbol{x}_{k+1}}{\rho_{k+1}}\right\|^2 =& \left\|\sum_{i=0}^{k}\frac{\rho_{k+1}(1-\eta\tau)}{\rho_{i+1}}\frac{\bar{\eta}-\eta}{\rho_{k+1}(1-\eta\tau)(1+\lambda\eta)}\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2 \\
\overset{①}{\leq}& \sum_{i=0}^{k}\frac{\rho_{k+1}(1-\eta\tau)}{\rho_{i+1}}\frac{(\bar{\eta}-\eta)^2}{\rho_{k+1}^2(1-\eta\tau)^2(1+\lambda\eta)^2}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2
\end{aligned}$$

where ① holds since 1) $\sum_{i=0}^{k}\frac{1-\eta\tau}{\rho_i} = \sum_{i=0}^{k}(\frac{1}{\rho_{i+1}} - \frac{1}{\rho_i}) = \frac{1}{\rho_{k+1}}$, and 2) Jensen' inequality. Therefore, we have

$$\|\boldsymbol{y}_{k+1} - (1+\lambda\bar{\eta})\boldsymbol{x}_{k+1}\|^2 \leq \frac{\rho_{k+1}(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2.$$

Moreover, we can also bound

$$\begin{aligned}
\|\boldsymbol{z}_{k+1} - \boldsymbol{x}_{k+1}\|^2 =& \|\bar{\eta}\tau\boldsymbol{x}_{k+1} + \eta\tau\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\|^2 \\
=& \eta\tau\|\boldsymbol{y}_{k+1} - (1+\lambda\bar{\eta})\boldsymbol{x}_{k+1}\|^2 \\
\leq& \frac{\rho_{k+1}\eta\tau(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2.
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
\|\boldsymbol{z}_{k+1} - \boldsymbol{z}_k\| =& \|\bar{\eta}\tau\boldsymbol{x}_{k+1} + \eta\tau\boldsymbol{y}_{k+1} - \boldsymbol{z}_k\| \\
\overset{①}{=}& \left\|\bar{\eta}\tau\boldsymbol{x}_{k+1} + \eta\tau\boldsymbol{y}_{k+1} - \boldsymbol{y}_{k+1} - \bar{\eta}\frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}\right\| \\
=& \left\|\bar{\eta}\tau\boldsymbol{x}_{k+1} + \eta\tau\boldsymbol{y}_{k+1} - \boldsymbol{y}_{k+1} - \bar{\eta}\frac{\boldsymbol{w}_k - \lambda\boldsymbol{x}_k * \boldsymbol{s}_k}{\boldsymbol{s}_k}\right\| \\
=& \left\|\bar{\eta}\tau(1+\lambda\eta)\left((1+\lambda\bar{\eta})\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\right) - \frac{\bar{\eta}}{1+\lambda\eta}\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\| \\
\leq& \bar{\eta}\tau(1+\lambda\eta)\|(1+\lambda\bar{\eta})\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\| + \frac{\bar{\eta}}{1+\lambda\eta}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|
\end{aligned}$$

where ① we plug in $\boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \bar{\eta}_k \frac{\boldsymbol{m}_k}{\boldsymbol{s}_k}$. Then we can upper bound

$$\begin{aligned}\|\boldsymbol{z}_{k+1} - \boldsymbol{z}_k\|^2 \leq & 2\bar{\eta}^2\tau^2(1+\lambda\eta)^2 \|(1+\lambda\bar{\eta})\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|^2 + \frac{2\bar{\eta}^2}{(1+\lambda\eta)^2}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|^2 \\ \leq & \frac{2\bar{\eta}^2}{(1+\lambda\eta)^2}\left\|\frac{\boldsymbol{w}_k}{\boldsymbol{s}_k}\right\|^2 + \frac{2\rho_{k+1}\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{(1-\eta\tau)}\sum_{i=0}^{k}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2\end{aligned}$$

The proof is completed. □

G.3 Proof of Lemma 7

Proof. From Lemma 5, we have

$$\begin{aligned}&\mathbb{E}\left[\|\boldsymbol{m}_k - \nabla F(\boldsymbol{z}_k)\|^2\right] \\ \leq & (1-\beta_{1,k})\mathbb{E}\left[\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\|^2\right] + \frac{(1-\beta_{1,k})^2L^2}{\beta_{1,k}}\mathbb{E}\left[\|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\|^2\right] + \frac{\beta_{1,k}^2\sigma^2}{b} \\ \overset{①}{\leq} & (1-\beta_{1,k})\mathbb{E}\left[\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\|^2\right] + \frac{\Pi_k(1-\beta_{1,k})^2L^2}{\beta_{1,k}} + \frac{\beta_{1,k}^2\sigma^2}{b}\end{aligned}$$

where in ①, we use the results in Lemma 6 that

$$\|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\|^2 \leq \Pi_k := \frac{2\bar{\eta}^2}{(1+\lambda\eta)^2}\left\|\frac{\boldsymbol{w}_{k-1}}{\boldsymbol{s}_{k-1}}\right\|^2 + \frac{2\rho_k\bar{\eta}^2\tau^2(\bar{\eta}-\eta)^2}{(1-\eta\tau)}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2.$$

Then we have

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{m}_k - \nabla F(\boldsymbol{x}_k)\|^2\right] \leq & 2\mathbb{E}\left[\|\boldsymbol{m}_k - \nabla F(\boldsymbol{z}_k)\|^2\right] + 2\mathbb{E}\left[\|\nabla F(\boldsymbol{z}_k) - \nabla F(\boldsymbol{x}_k)\|^2\right] \\ \leq & 2\mathbb{E}\left[\|\boldsymbol{m}_k - \nabla F(\boldsymbol{z}_k)\|^2\right] + 2L\mathbb{E}\left[\|\boldsymbol{z}_k - \boldsymbol{x}_k\|^2\right] \\ \overset{①}{\leq} & 2(1-\beta_{1,k})\mathbb{E}\left[\|\boldsymbol{m}_{k-1} - \nabla F(\boldsymbol{z}_{k-1})\|^2\right] + \frac{2\Pi_k(1-\beta_{1,k})^2L^2}{\beta_{1,k}} + \frac{2\beta_{1,k}^2\sigma^2}{b} + 2L\Pi_k',\end{aligned}$$

where in ①, we use the results in Lemma 6 that

$$\|\boldsymbol{z}_k - \boldsymbol{x}_k\|^2 \leq \Pi_k' := \frac{\rho_k\eta\tau(\bar{\eta}-\eta)^2}{(1-\eta\tau)(1+\lambda\eta)^2}\sum_{i=0}^{k-1}\frac{1}{\rho_{i+1}}\left\|\frac{\boldsymbol{w}_i}{\boldsymbol{s}_i}\right\|^2.$$

The proof is completed. □