Training Deep Neural Networks with Adaptive Momentum Inspired by the Quadratic Optimization

Tao Sun, Huaming Ling, Zuoqiang Shi, Dongsheng Li, and Bao Wang

Abstract—Heavy ball momentum is crucial in accelerating (stochastic) gradient-based optimization algorithms for machine learning. Existing heavy ball momentum is usually weighted by a uniform hyperparameter, which relies on excessive tuning. Moreover, the calibrated fixed hyperparameter may not lead to optimal performance. In this paper, to eliminate the effort for tuning the momentum-related hyperparameter, we propose a new adaptive momentum inspired by the optimal choice of the heavy ball momentum for quadratic optimization. Our proposed adaptive heavy ball momentum can improve stochastic gradient descent (SGD) and Adam. SGD and Adam with the newly designed adaptive momentum are more robust to large learning rates, converge faster, and generalize better than the baselines. We verify the efficiency of SGD and Adam with the new adaptive momentum on extensive machine learning benchmarks, including image classification, language modeling, and machine translation. Finally, we provide convergence guarantees for SGD and Adam with the proposed adaptive momentum.

Index Terms—Adaptive heavy ball, Nonconvexity, Acceleration, Momentum, Adaptive learning rate

1 INTRODUCTION

Consider the following empirical risk minimization (ERM) problem

$$\min_{x \in \mathbb{R}^d} \{ f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(s_i; x, y_i) \},$$

where $\mathcal{L}$ is the loss function, $\sigma$ is the machine learning model parameterized by $x$, and $(s_i, y_i)$ is a sample-label pair. Stochastic gradient descent (SGD) [1] is a simple yet effective algorithm to solve (1), which updates $x$ according to

$$x^{k+1} = x^k - \gamma g^k,$$

where $\gamma > 0$ is the learning rate and $g^k = \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(x^k)$ with $m \ll n$ is the mini-batch stochastic gradient of $f$. Heavy ball (HB) algorithm [2] leverages memory to accelerate GD, which updates $x$ as follows

$$x^{k+1} = x^k - \gamma (g^k + \beta (x^k - x^{k-1})), \text{ (2)}$$

where $0 \leq \beta < 1$ is the momentum hyperparameter. By introducing momentum states $(m^k)_{k \geq 0}$, we can rewrite HB as

$$m^{k+1} = \beta m^k + g^k; \quad x^{k+1} = x^k - \gamma m^{k+1}.$$

If we integrate the adaptive learning rate with the heavy ball algorithm and rescale $m^k$ and $g^k$, we obtain the following celebrated Adam algorithm [3]

$$x^{k+1} = x^k - \gamma m^{k}/[v^{k}]^{1/2}; \quad m^{k+1} = \beta m^k + (1 - \beta)g^k; \quad v^{k+1} = \alpha v^k + (1 - \alpha)[g^k]^2, \text{ (3)}$$

where $\alpha, \beta, \gamma > 0$ are three positive constants, and $[\cdot]^2$ and $[\cdot]^{1/2}$ denote the element-wise square and squared-root, respectively. These first-order algorithms are among the methods of choice for signal processing [4] and machine learning; in particular, for training deep neural networks (DNNs) [5].

The momentum in both HB and Adam is weighted by a constant $\beta$. A fine-tuned $\beta$ is crucial for accelerating SGD and Adam in training DNNs and improving their generalization [6], [7], [8], [9], [10]. Indeed, [2], [11], [12] establish the theoretical acceleration of momentum for convex optimization and training one-layer neural networks, and these results depend on specialized choices of $\beta$ and $\gamma$. Calibrating the momentum hyperparameter $\beta$ is computationally expensive. Moreover, training machine learning models may require different $\beta$ in different iterations. To the best of our knowledge, there is no principled way to choose the optimal $\beta$ for training DNNs. Therefore, it is natural and important to ask the following question:

Can we design an adaptive momentum without calibrating for the momentum-related hyperparameter to improve GD/SGD and Adam with convergence guarantees?

1.1 Contributions

We answer the above question affirmatively by replacing $\beta$ in HB [2] and Adam [3] with the following iteration-dependent adaptive

1. For the sake of simplicity, we omit the bias correction here.
scheme
\[
\beta_{k+1} = \begin{cases} 
\text{Proj}_{[0,1]} \left( 1 - \sqrt{\frac{\|g^k - g^{k-1}\|^2}{\|x^k - x^{k-1}\|^2}} \right), & k \geq 2, \\
0, & k = 0,1,
\end{cases}
\] (4)
where \(\|\cdot\|\) denotes the \(\ell_2\) norm of a given vector, and \(\text{Proj}_{[0,1]}(\cdot) := \max(0, \min(\cdot, 1 - \delta))\) with \(\delta\) be the threshold, which is simply set to \(10^{-3}\) in all the following experiments. We will study the effects of \(\delta\) in Appendix A.2.

We start from a fine-grained convergence analysis of HB for quadratic optimization, resulting in the optimal choice for momentum. Then, we design a simple sequence to approximate the optimal momentum and result in our proposed adaptive momentum [3]. The convergence of HB, proximal HB, and Adam with adaptive momentum in both convex and nonconvex settings is easy to guarantee; as a minor contribution, we show the convergence of the adaptive momentum schemes in various settings. Finally, we perform extensive experiments to show that these adaptive momentum schemes are more robust to larger learning rates, converge faster, and generalize better than the baselines.

1.2 Additional Related Works

The advances of momentum-based acceleration have been booming since the pioneering work of [2], [13]. In this section, we briefly review some of the most exciting and related works on developing and analyzing HB and Nesterov’s acceleration (a class of adaptive momentum). Also, we will discuss some of the recent development of adaptive momentum.

1.2.0.1 Heavy ball: The convergence of deterministic HB, i.e., HB with exact gradient, has been thoroughly studied by [11], [14], [15], [16], [17] in both convex and nonconvex cases. An interesting finding is that HB can escape saddle points in nonconvex optimization by using a larger learning rate than GD [18]. HB momentum has also been successfully integrated into SGD to improve training DNNs. Especially for image classification [5], [6], [7]. The authors of [19] propose and analyze HB momentum for GD [18]. HB mean GD with HB momentum.

HB achieves a convergence rate of \(\|x^k - x^*\| \leq (1 - \sqrt{\gamma \nu})^k \cdot 2\|x^0 - x^*\|\) as \(k \geq K(\nu, \epsilon)\), where \(\epsilon > 0\) is very small and \(K(\nu, \epsilon) \in \mathbb{Z}^+\) is an integer depends on \(\nu\) and \(\epsilon\).

1.2.0.2 Nesterov’s acceleration: Nesterov’s acceleration can be obtained by replacing \(\beta\) with a specially designed time-varying formula, which is a special adaptive momentum. Nesterov’s acceleration achieves the optimal convergence rate for smooth convex minimization with access to gradient information only [13], [23]. The first use of Nesterov’s acceleration in stochastic gradient is proposed by [24] but without any theoretical guarantee. The convergence of Nesterov’s acceleration can be proved for stochastic gradient using diminishing learning rates [19], [25]. In [26], [27], the authors prove that the stochastic Nesterov’s acceleration converges to a neighborhood of the minimum. Nesterov’s acceleration for over-parameterized models with a diminishing momentum has been studied in [28]. Nevertheless, in the finite-sum setting, the Nesterov’s acceleration is proved to possibly diverge with the usual choice of the learning rate and momentum [29], [30]. The possible divergence of stochastic Nesterov’s acceleration limits its use for deep learning. To this end, a restart momentum is employed for the stochastic Nesterov’s acceleration [31], which achieves further acceleration from the numerical tests, but the convergence is proved with non-trivial assumptions.

1.2.0.3 Other variants of HB and Nesterov’s acceleration: Many other momentum schemes have been developed to fix the possible divergence issue of the Nesterov’s acceleration [29], [32], [33], [34], [35], [36]. However, these algorithms are not popular in deep learning.

1.2.0.4 Other adaptive momentum: The authors in [37] develop a class of nonlinear conjugate gradient (NCG)-style adaptive momentum for HB, and they prove acceleration for quadratic function in the deterministic case and numerically show the effectiveness in training DNNs.

1.3 Notation

We denote scalars/vectors by lower case/lower case boldface letters. We denote matrices by upper case boldface letters. For \(x = (x_1, \ldots, x_d) \in \mathbb{R}^d\), we use \(\|x\|_2\) to denote its \(\ell_2\) norm. For a matrix \(A\), we use \(A^\top\) to denote its transpose and use \(\|A\|_p\) to denote its induced norm by the vector \(\ell_p\) norm. Given two sequences \(\{a_n\}\) and \(\{b_n\}\), we write \(a_n = \mathcal{O}(b_n)\) if there exists a positive constant \(0 < C < +\infty\) such that \(a_n \leq C b_n\), and we write \(a_n = \Theta(b_n)\) if there exist two positive constants \(C_1\) and \(C_2\) such that \(a_n \leq C_1 b_n\) and \(b_n \leq C_2 a_n\). We denote the set \(\{1, 2, \ldots, m\}\) as \(\left\lfloor m \right\rfloor\). We denote the gradient and Hessian of a function \(f(x) : \mathbb{R}^d \to \mathbb{R}\) by \(\nabla f(x)\) and \(\nabla^2 f(x)\), respectively.

2 Motivation and Algorithms

To get a better understanding of HB, we start from the quadratic problem. In particular, we have

\textbf{Lemma 1} Let \(f\) be a quadratic function with \(\nabla f(x) = Ax + b\), where \(A\) is positive definite. Moreover, we denote the smallest \((\lambda_{\min}(A))\) and the largest eigenvalues \((\lambda_{\max}(A))\) of \(A\) as \(\nu\) and \(L\), respectively. Then, given any fixed \(\gamma \leq 1/L\), the optimal choice for \(\beta\) is \(\beta = (1 - \sqrt{\gamma \nu})^k\). In this case, HB achieves a convergence rate of \(\|x^k - x^*\| \leq (1 - \sqrt{\gamma \nu})^k \cdot 2\|x^0 - x^*\|\) as \(k \geq K(\nu, \epsilon)\), where \(\epsilon > 0\) is very small and \(K(\nu, \epsilon) \in \mathbb{Z}^+\) is an integer depends on \(\nu\) and \(\epsilon\).

Compared with [38], Lemma 1 is an improved result for HB methods. In particular, Lemma 1 shows that the optimal hyperparameter for the HB momentum should be \((1 - \sqrt{\gamma \nu})^k\) if \(\gamma \leq 1/L\). However, the smallest eigenvalue \(\nu\) is unknown. Therefore, we construct the sequence, \(\|g^k - g^{k-1}\|/\|x^k - x^{k-1}\|\), to approximate \(\nu\). It is easy to check that \(\|g^k - g^{k-1}\|/\|x^k - x^{k-1}\| \in [\nu, L]\). Moreover, we have the following lemma, which shows that \(\|g^k - g^{k-1}\|/\|x^k - x^{k-1}\| \to \nu\).

\textbf{Lemma 2} Assume that the conditions in Lemma 1 hold and \(\{x^k\}_{k \geq 0}\) is generated by HB [2]. Also, assume \(A\) has a unique minimum singular value. If \(\gamma \leq 1/L\), for any fixed \(0 \leq \beta < 1\), we have
\[
\lim_{k \to \infty} \frac{\|g^k - g^{k-1}\|}{\|x^k - x^{k-1}\|} = \nu.
\]
2. Here, HB means GD with HB momentum.
3 EXTEND TO PROXIMAL ALGORITHMS AND ADAM

In this section, we present two extensions of ASHB. One is the proximal algorithm for composite optimization, and the other one is Adam for deep learning.

3.1 Extend to Proximal Algorithms

Proximal algorithms are used for solving the following composite optimization

$$\min \{ F(x) := f(x) + g(x) \}, \ x \in \mathbb{R}^d,$$

where we assume that the proximal map of the function $\gamma g, \gamma > 0$ is easy to obtain. In particular, when $\text{Prox}_{\gamma g}(\cdot) := \text{arg min}_x \{ g(x) + \|x - y\|^2 / 2 \}$ has a closed form solution. Composite optimization has been widely used in image processing and statistics [39], [40], [41]. To extend adaptive momentum to proximal gradient descent, we leverage the following approximation of the optimal momentum as [69] with $g^k = \nabla f(x^k)$ for $f(x)$ in (7). We call the resulting algorithm proximal adaptive heavy ball (PAHB), which is summarized in Algorithm 3.

Algorithm 3 Ada with Adaptive Momentum (Ada²m)

Require: parameters $(\gamma_k)_{k \geq 0} \subseteq (0, +\infty), \ 0 < \delta \leq 1, \ (\alpha_k)_{k \geq 0} \subseteq [0, 1)\n
Initialization: $x^0 = x^1 = 0, \ \beta_0 = 0$ for $k = 1, 2, \ldots$

step 1: $x^{k+1} = \text{Prox}_{\gamma_k g} [x^k - \gamma_k \nabla f(x^k) + \beta_k (x^k - x^{k-1})]$

step 2: update as (6b)

end for
4.1 Convergence of ASHB
We prove the convergence of ASHB in strongly convex (Theorem 1), general-convex (Theorem 2), and nonconvex (Theorem 3) settings.

**Theorem 1 (Strong convexity case)** Let $f(x)$ be strongly convex and twice-differentiable, and let $\{x^k\}_{k \geq 0}$ be generated by (6). Assume that $0 < \nu \leq \min_x \{\lambda_{\min}(\nabla^2 f(x))\} \leq \max_x \{\lambda_{\max}(\nabla^2 f(x))\} \leq L$, and Assumptions 1, 2, 3 hold. Then,

$$E\|x^K - x^*\|^2 \leq (1 - 2\gamma\nu)^K E\|x^1 - x^*\|^2 + C_1\gamma,$$

where $C_1 > 0$ is a constant independent on $\gamma$, $d$, and $K$.

The obtained convergence rate in Theorem 1 is similar to SGD for strongly convex optimization, i.e., a linear rate at the beginning. Also, the error term in the above convergence rate needs to be removed by using a diminishing learning rate. Thus, to reach $\epsilon > 0$ error for $E\|x^K - x^*\|$, we need to set $\gamma = \Theta(\epsilon)$ and $K = O(\frac{\ln \frac{1}{\epsilon}}{\epsilon})$.

For general convex cases, the convergence of Algorithm 1 can be described as follows.

**Theorem 2 (General convex case)** Let $f$ be convex and $\{x^k\}_{k \geq 1}$ be generated by (6). Suppose Assumptions 1, 2 and 3 hold. Then

$$E\left(\sum_{k=1}^{K} x^k / K\right) - \min f \leq E\|x^1 - x^*\|^2/(2\gamma K) + C_2\gamma,$$

where $C_2 > 0$ is a constant independent on $\gamma$, $d$, and $K$.

Given the desired $\epsilon$ error tolerance, we need to set $\gamma = \Theta(\epsilon)$ and let $K = O(\frac{1}{\gamma^2})$. For general nonconvex optimization problems, we measure the convergence of gradient norm following current routines. We have the following convergence guarantee for ASHB for nonconvex optimization.

**Theorem 3 (Nonconvexity case)** Let $\{x^k\}_{k \geq 1}$ be generated by (6) and suppose Assumptions 1, 2 and 3 hold. It follows that

$$\min_{1 \leq k \leq K} \{E\|\nabla f(x^k)\|^2\} \leq C_3\gamma + \frac{C_4 f(x^1) - \min f}{\gamma K},$$

where $C_3, C_4 > 0$ are constants independent on $\gamma$, $d$, and $K$.

To get the $\epsilon$ error for $\min_{1 \leq k \leq K} \{E\|\nabla f(x^k)\|^2\}$, we need to set $\gamma = \Theta(\epsilon)$ and $K = O(\frac{1}{\gamma^2})$. In summary, we cannot prove the theoretical acceleration of the proposed adaptive momentum algorithms. But the obtained results above show that the speed of ASHB can run as fast as SGD in strongly convex, general-convex, and nonconvex cases. Also, the convergence guarantees can promise the use of the proposed SGD with adaptive momentum, which is better than the stochastic Nesterov’s acceleration and its restart variants.

4.2 Convergence of PAHB
In this subsection, we analyze the convergence of PAHB; we summarize our theoretical convergence guarantee for PAHB in the following proposition.

**Proposition 1 (Convergence of PAHB)** Assume that $\{x^k\}_{k \geq 1}$ is generated by PAHB. If $g$ is nonconvex, and $\gamma_k = \frac{c(1-2\beta_k)}{2}$ with $0 < c < 1$, and $\frac{1}{2} < \delta < 1$. Then we have $\sum_k \|x^{k+1} - x^k\|^2 < +\infty$. If $g$ is convex, the result still holds when $\gamma_k = \frac{c(1-\beta_k)}{L}$ with $0 < c < 1$, and $0 < \delta < 1$.

4.3 Convergence of Ada²m
In this subsection, we discuss the convergence of Ada²m for nonconvex optimization. We summarize the convergence of Ada²m in the following proposition.

**Proposition 2 (Convergence of Ada²m)** Let $\{x^k\}_{k \geq 1}$ be generated by Ada²m. Let $\alpha_k = \frac{L}{\epsilon}$ and $\gamma_k = \frac{\sqrt{K}}{\epsilon}$ for some $\gamma > 0$, and $\|g^2\| \geq \pi > 0$. Then,

$$\min_{1 \leq k \leq K} \{E\|\nabla f(x^k)\|^2\} = O(\frac{\ln K}{\sqrt{K}}).$$

We provide technical proofs for all the above theoretical results in the appendix.

5 NUMERICAL RESULTS
In this section, we verify the efficiency of the proposed adaptive momentum for training various machine learning models, including regularized logistic regression models, ResNets, LSTMs, and transformers. In all the following experiments, we show the advantage of the adaptive momentum (4) over the baseline algorithms. We measure the convergence of gradient norm following current routines. We have the following convergence guarantee for ASHB for nonconvex optimization.

5.1 Regularized Logistic Regression
Consider training the following regularized logistic regression model

$$\min_x \{F(x) := f(x) + \lambda R(x)\},$$

where $f(x) = \frac{1}{2m} \sum_{i=1}^{m} \log(1 + \exp(-b_i a^T_i x))$ with $n$ be the sample size and $a_i \in \mathbb{R}^d$ is a training data, $b_i \in \{-1, 1\}$ be the label of $a_i$, and $\lambda = 0.001$. Here, we consider both $\ell_1 (R(x) = \|z\|_1)$ and $\ell_2 (R(x) = \|z\|_2^2)$ regularization.

For [5], we compute the exact gradient of $f(x)$ and compare PAHB with three well-studied benchmark algorithms, namely, proximal gradient descent (PGD), accelerated proximal gradient descent (APGD) [42], and proximal heavy ball (PHB) (replace $\beta_k$ with a constant $\beta$ in PAHB). We set $n = d = 1000$ and let $[a_1, a_2, \cdots, a_n]$ be the $n \times d$ multivariate-normal matrix with the covariance matrix being Toeplitz, whose first row is $[0.9^0, 0.9^1, \cdots]$. $x \in \mathbb{R}^{1000}$ is a vector whose $i$-th entry is $(-1)^i e^{-i/1000}$ for $0 < i \leq 500$ and 0 otherwise. $b_i$ is a binomial random variable with probability $a^T b$ to take the value $1$. Figure 2 shows the comparisons between the above four algorithms, where we begin with the zero initialization and use a learning rate of $0.1$ for the above four algorithms. For PHB, we set $\beta = 0.9$, which is obtained by grid search. It is seen that PHB is faster than PGD, and adaptive momentum can remarkably accelerate PHB. Moreover, PAHB is the fastest solver for solving the $\ell_2$-regularized logistic regression problem among the above four algorithms.
5.2 Training DNNs for Image Classification

5.2.1 ResNets for CIFAR10 Classification

| Model      | PreResNet20 | PreResNet56 | PreResNet110 |
|------------|-------------|-------------|--------------|
| Ada $^m$   | 91.24 ± 0.07 | 92.65 ± 0.07 | 93.31 ± 0.09 |
| AdamW      | 90.65 ± 0.09 | 92.64 ± 0.13 | 93.46 ± 0.15 |
| Ada $^m$W  | 90.99 ± 0.08 | 92.80 ± 0.04 | 93.65 ± 0.07 |

5.2.1.2 SGDM vs. ASHB: Now, we conduct the above CIFAR10 training by using SGDM and ASHB. First, we show that ASHB is more robust to different learning rates. Figure 3 (Right) shows the training curves of the four Adam-style algorithms in training PreResNet56 for CIFAR10 classification. We see that AdamW and Adam$^2$W converge remarkably faster than Adam and Ada$^m$W, and Ada$^m$W is even slightly faster than AdamW.

Second, we show that ASHB can improve the test accuracy of the trained models. We train the above three ResNet models using SGDM and ASHB for 200 epochs, respectively. For SGDM, we use the benchmark initial learning rate of 0.1 and momentum of 0.9, and we decay the learning rate by a factor of 10 at the 80-th, 120-th, and 160-th epoch, respectively. We set the weight decay to be 0.0005. Meanwhile, we train ResNets by Ada$^m$ and Ada$^m$W using the same setting as that used in Adam and AdamW. Table 3 lists the test accuracy of both models for classifying CIFAR10. We see that adaptive momentum can consistently improve test accuracy for all the three ResNet models. Figure 3 (Left) plots training curves of the four Adam-style algorithms in training PreResNet56 for CIFAR10 classification. We see that AdamW and Adam$^2$W converge remarkably faster than Adam and Ada$^m$W, and Ada$^m$W is even slightly faster than AdamW.

We train ResNet20/56/110 with pre-activation [43], denoted as PreResNet20/56/110, using batch size 128 for CIFAR10 classification, which contains 50K/10K images in the training/test set.

5.2.1.1 Adam-style algorithms: We consider two popular Adam-style algorithms that have been implemented in the latest version of PyTorch [44], namely the Adam and AdamW [45], for training ResNet[4]. We run both algorithms for 200 epochs using an initial learning rate of 0.001/0.003 for Adam/AdamW and decay the learning rate by a factor of 10 at the 80-th, 120-th, and 160-th epoch, respectively.

We perform grid search for the initial learning rate to obtain this choice for Adam and AdamW, respectively.

4. We can also incorporate the adaptive momentum with other Adam variants.
5. We perform grid search for the initial learning rate to obtain this choice for Adam and AdamW, respectively.
6. Note that the learning rate is optimized for Adam and AdamW rather than Ada$^m$ and Adam$^2$W.
ASHB plateaus slower for a given learning rate.

5.3 ResNets for ImageNet Classification

In this part, we discuss our experimental results on the 1000-way ImageNet classification task \[49\]. We train ResNet18, whose implementation is available at \[50\], using both SGD with momentum (SGDM, $\beta = 0.9$) and ASHB with five different random seeds. Following the common practice, we train each model for 90 epochs and decrease the learning rate by a factor of 10 at each certain epochs. For SGDM, we begin with an initial learning rate of 0.1 and decay it by a factor of 10 at the 30-st and 60-st epoch, respectively. For ASHB, we use different initial learning rates include 0.1, 0.3, 0.5, and 0.7, and we decay the learning rate at the 51-st and 71-st epochs by a factor of 10, respectively. Moreover, we set the weight decay to be 0.0001 for both SGDM and ASHB. Table 3 lists top-1 and top-5 accuracies of the models trained by two different optimization algorithms; we see that ASHB can outperform SGDM in classifying images. Figure 6 plots the evolution of training and test accuracies. It is clear that for a given learning rate, e.g., 0.1, ASHB converges faster than SGDM. Also, according to experiments, ASHB is more robust to large learning rates, in which SGDM will blow up, but ASHB still performs well.

5.4 Training DNNs for Natural Language Processing

5.4.0.1 LSTM.: We train two- and three-layer LSTM models for the benchmark word-level Penn Treebank (PTB) language modeling. We use the benchmark implementation and the default training settings of the LSTM model \[51\]; we switch the optimizer among Adam, AdamW, Ada$^2$m, and Ada$^2$mW. We use the fine-tuned learning rates for both Adam and AdamW, which are both 0.001. For Ada$^2$m and Ada$^2$mW, we set the learning rate to 0.005 and 0.003, respectively. Figure 7 depicts the training and test curves of different optimizers. We see that the adaptive momentum schemes can not only accelerate training, but can also improve the test perplexity by a very big margin, e.g., the best test perplexities of the three-layer LSTM trained by Adam and Ada$^2$mW are 64.4 and 60.9, respectively.

5.4.0.2 Transformers.: To further verify the efficiency of adaptive momentum for training language models, we train a transformer model for neural machine translation. Here, we train the transformer, implementation is available at \[52\], on the benchmark IWSLT14 De-En dataset. We note that AdamW with warmup is the default optimization algorithm for training transformers on this task; we compare AdamW and Ada$^2$mW on this task. We search the learning rate for both AdamW and Ada$^2$mW from the learning rate set \{0.0005, 0.0006, 0.0007, 0.0008, 0.0009\}, and we found 0.0007 is the optimal learning rate for both optimizers. We conduct experiments using five different random seeds and report their mean and standard deviation of the BLEU score (a higher BLEU score is better). For AdamW with with warmup, the mean (std) BLEU score is 35.11 (0.147). Meanwhile, the mean (std) BLEU score instead is 35.32 (0.061) for Ada$^2$mW with warmup.

6 CONCLUDING REMARKS

In this paper, we propose an adaptive momentum to eliminate the computational cost for tuning the momentum-related hyperparameter for the heavy ball method. We integrate the new adaptive momentum into several benchmark algorithms, including proximal gradient descent, stochastic gradient, and Adam. Theoretically, we establish the convergence guarantee for the above benchmark algorithms with the newly developed adaptive momentum. Empirically, we see the advantage of adaptive momentum in enhancing robustness to large learning rates, accelerating training, and improving the generalization of various machine learning models for image classification and language modeling. There are numerous avenues for future work: 1) How to interpret the adaptive momentum from the effective step size perspective \[20\], \[53\]? 2) Can we establish theoretical acceleration for the proposed adaptive momentum? 3) Can we design different optimal adaptive momentum for different optimization algorithms?

7. We decay the learning rate for ASHB later than that of SGDM since ASHB plateaus slower for a given learning rate.
8. The reported BLEU score at \[52\] is 35.02.
Appendix A

More Details of the Numerical Results

A.1 Detailed form of Laplacian matrix of a cyclic graph in numerical verification for Lemma 2

\[ L = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -1 & 0 & 0 & \cdots & -1 & 2 \end{bmatrix} \in \mathbb{R}^{d \times d} \]

A.2 Ablation Study–The Effects of \( \delta \)

In this section, we study the effects of \( \delta \) in the adaptive momentum \( \{ \theta \} \) on the performance of ASHB and Ada\( ^2 \)W. Concerning the massive computational cost, we restrict our ablative study in training PreResNet20 for CIFAR10 classification. We test the value of \( \delta \) from the set \( \{ 10^{-i} | i = 1, 2, \ldots, 9 \} \). Figure 5 shows \( \delta \) vs. test accuracy for ASHB and Ada\( ^2 \)W. It is clear that the effects of the value of \( \delta \) on the test accuracy of the trained models are negligible. In particular, the test accuracies are almost the same for different \( \delta \) provided \( \delta \) is less than 0.01.

Appendix B

Technical Lemmas

Lemma 3 Let \( f \) have Lipschitz gradient with constant \( L > 0 \) and let \( \{ x^k \}_{k \geq 0} \) be generated by \( [3] \), we have

\[ \sup_k \| x^k - x^{k-1} \| \leq \frac{\gamma R}{\delta} \]  
and

\[ \sum_{k=1}^{K} \| x^k - x^{k-1} \|^2 \leq \frac{\gamma^2 K R^2}{\delta^2}. \]  

Proof: It holds that

\[ \| \gamma g^k \| \geq \| x^{k+1} - x^k \| - \| \beta_k(x^k - x^{k-1}) \| \]

\[ = \| x^{k+1} - x^k \| - (1 - \delta) \| x^k - x^{k-1} \|. \]

Thus,

\[ \gamma g^k \leq \left( \| x^{k+1} - x^k \| - (1 - \delta) \| x^k - x^{k-1} \| \right) \]

\[ = \| x^{k+1} - x^k \|^2 - 2(1 - \delta) \| x^{k+1} - x^k \| \]

\[ \times \| x^k - x^{k-1} \| + (1 - \delta)^2 \| x^k - x^{k-1} \|^2 \]

\[ \geq \delta \| x^{k+1} - x^k \|^2 - \delta (1 - \delta) \| x^k - x^{k-1} \|^2. \]  

By using the Mathematical Induction (MI) method, we then get \( [9] \). Summing \( [11] \) from \( k = 1 \) to \( K \), we get

\[ \delta^2 \sum_{k=1}^{K} \| x^k - x^{k-1} \|^2 \leq \sum_{k=1}^{K-1} \gamma^2 \| g^k \| \leq \gamma^2 K R^2. \]

Lemma 4 Let \( f \) have Lipschitz gradient with constant \( L > 0 \) and let \( \{ x^k \}_{k \geq 0} \) be generated by \( [6] \), we have

\[ \sum_{k=1}^{K} \beta_k \mathbb{E} \langle \nabla f(x^k), x^k - x^{k-1} \rangle \leq \left( \frac{1 - \delta}{\delta} \right) L \sum_{k=1}^{K} \mathbb{E} \| x^k - x^{k-1} \|^2. \]

Proof: Direct computation yields

\[ \langle \nabla f(x^k), x^k - x^{k-1} \rangle = \langle \nabla f(x^{k-1}), x^k - x^{k-1} \rangle \]

\[ + \langle \nabla f(x^k) - \nabla f(x^{k-1}), x^k - x^{k-1} \rangle \]

\[ \leq \langle \nabla f(x^{k-1}), x^k - x^{k-1} \rangle + L \| x^k - x^{k-1} \|^2 \]

\[ = \langle \nabla f(x^{k-1}), -\gamma g^{k-1} + \beta_{k-1}(x^{k-2} - x^{k-1}) \rangle + L \| x^k - x^{k-1} \|^2. \]

Taking expectation, we then get

\[ \mathbb{E} \langle \nabla f(x^k), x^k - x^{k-1} \rangle \]

\[ \leq -\gamma \mathbb{E} \| \nabla f(x^{k-1}) \|^2 + \beta_{k-1} \mathbb{E} \langle \nabla f(x^{k-1}), (x^{k-1} - x^{k-2}) \rangle \]

\[ + L \mathbb{E} \| x^k - x^{k-1} \|^2 \]

\[ \leq \beta_{k-1} \mathbb{E} \langle \nabla f(x^{k-1}), (x^{k-1} - x^{k-2}) \rangle + L \mathbb{E} \| x^k - x^{k-1} \|^2. \]

With induction and the fact that \( x^1 = x^0 \), we then get

\[ \beta_k \mathbb{E} \langle \nabla f(x^k), x^k - x^{k-1} \rangle \leq L \sum_{i=1}^{k-1} \left( \prod_{j=1}^{k-1} \beta_j \right) \mathbb{E} \| x^i - x^{i-1} \|^2 \]

\[ \leq L \sum_{i=1}^{k} \left( 1 - \delta \right)^{k+1-i} \mathbb{E} \| x^i - x^{i-1} \|^2. \]
Thus, we have
\[
\sum_{k=1}^{K} \beta_k \mathbb{E}(\nabla f(x^k), x^k - x^{k-1}) \\
\leq L \sum_{k=1}^{K} \sum_{i=1}^{k} (1 - \delta)^{k+1-i} \mathbb{E}\|x^i - x^{i-1}\|^2 \\
\leq \frac{(1 - \delta) L}{\delta} \sum_{k=1}^{K} \mathbb{E}\|x^k - x^{k-1}\|^2.
\]

**Lemma 5** Let \( f \) have Lipschitz gradient with constant \( L > 0 \) and be convex, and let \( \{x^k\}_{k \geq 0} \) be generated by (6), we have
\[
\sum_{k=1}^{K} \beta_k \mathbb{E}(x^k - x^*, x^k - x^{k-1}) \leq \frac{(1 - \delta)}{\delta} \sum_{k=1}^{K} \mathbb{E}\|x^k - x^{k-1}\|^2.
\]

**Proof:** We can have
\[
\langle x^k - x^*, x^k - x^{k-1} \rangle = \langle x^{k-1} - x^*, x^k - x^{k-1} \rangle + \langle x^k - x^{k-1}, x^k - x^{k-1} \rangle \\
= -\gamma \langle x^{k-1} - x^*, g^{k-1} \rangle + \beta_{k-1} \langle x^{k-1} - x^*, x^{k-1} - x^{k-2} \rangle \\
+ \langle x^k - x^{k-1}, x^k - x^{k-1} \rangle + \|x^k - x^{k-1}\|^2.
\]

Taking expectation, we derive
\[
\mathbb{E}(x^k - x^*, x^k - x^{k-1}) = -\gamma \mathbb{E}(x^{k-1} - x^*, \nabla f(x^{k-1})) \\
+ \beta_{k-1} \mathbb{E}(x^{k-1} - x^*, x^{k-1} - x^{k-2}) + \mathbb{E}\|x^k - x^{k-1}\|^2 \\
+ \mathbb{E}(x^k - x^{k-1}, x^k - x^{k-1}) \\
\leq \beta_{k-1} \mathbb{E}(x^{k-1} - x^*, x^{k-1} - x^{k-2}) + \mathbb{E}\|x^k - x^{k-1}\|^2 \\
+ \mathbb{E}(x^k - x^{k-1}, x^k - x^{k-1})
\]

where we used \(-\gamma \mathbb{E}(x^{k-1} - x^*, \nabla f(x^{k-1})) \leq -\mathbb{E}(f(x^{k-1}) - f(x^*)) \leq 0\). With induction and the fact that \( x^i = x^i \), we are then led to
\[
\frac{1}{\beta_k} \mathbb{E}(x^k - x^*, x^k - x^{k-1}) \leq \sum_{i=1}^{k-1} \frac{1}{\beta_j} \mathbb{E}\|x^i - x^{i-1}\|^2 \\
\leq \sum_{i=1}^{k} (1 - \delta)^{k+1-i} \mathbb{E}\|x^i - x^{i-1}\|^2.
\]

Thus, we can get
\[
\sum_{k=1}^{K} \beta_k \mathbb{E}(x^k - x^*, x^k - x^{k-1}) \\
\leq \sum_{k=1}^{K} \sum_{i=1}^{k} (1 - \delta)^{k+1-i} \mathbb{E}\|x^i - x^{i-1}\|^2 \\
\leq \left(\frac{1 - \delta}{\delta}\right) \sum_{k=1}^{K} \mathbb{E}\|x^k - x^{k-1}\|^2.
\]

**Lemma 6** Assume \( \{x^k\}_{k \geq 0} \) is generated by Ada2m and conditions of (4) hold. Let \( \Xi_k := \gamma_k^2 \mathbb{E}\|m^k/((v^k)^{1/2})\|^2 \). We have
\[
\sum_{k=1}^{K} \Xi_k \leq \sum_{j=1}^{K-1} \gamma_{j}^2 \mathbb{E}\|\mathbf{g}^j\|^2/(\mathbf{v}^j)^{1/2},
\]

where \( \mathbf{v}^j := \sum_{i=1}^{j} |g^i|^2 \).

**Proof:** Recalling \( m^k = \sum_{j=1}^{k-1}(\Pi_{i=j+1}^k \beta_i)(1 - \beta_j)g^j \) and \( v^k = \sum_{i=1}^{k} |g^i|^2 \), we then have
\[
\sum_{i=1}^{d} \gamma_{i}^2 \mathbb{E}\|m^i/((v^i)^{1/2})\|^2 = \sum_{i=1}^{d} \gamma_{i}^2 \mathbb{E}\|m^i/((v^i)^{1/2})\|^2 \\
\leq \sum_{i=1}^{d} \gamma_{i}^2 \mathbb{E}\|\sum_{j=1}^{k-1}(\Pi_{i=j+1}^k \beta_i)g^j/((v^j)^{1/2})\|^2 \\
= \sum_{i=1}^{d} \gamma_{i}^2 \mathbb{E}\|\sum_{j=1}^{k-1}(\Pi_{i=j+1}^k \beta_i)g^j/((v^j)^{1/2})\|^2,
\]
where we used $\gamma_k = \frac{\sqrt{\gamma}}{k}$. That is further bounded by

$$
\sum_{i=1}^{d} \gamma^2 | \sum_{j=1}^{k-1} (1 - \delta)^{k-1-j} g_i^j / (\hat{v}_i^k)^{\frac{k}{2}} |^2 \\
a \leq \sum_{i=1}^{d} \gamma^2 \left( \sum_{j=1}^{k-1} (1 - \delta)^{k-1-j} \cdot \sum_{j=1}^{k-1} (1 - \delta)^{k-1-j} \frac{(g_i^j)^2}{\hat{v}_i^k} \right) \\
\leq \sum_{i=1}^{d} \gamma^2 \cdot \frac{1}{\delta} \sum_{j=1}^{k-1} (1 - \delta)^{k-1-j} ||(g_i^j)^2 / \hat{v}_i^k ||_1 \\
= \frac{\gamma^2}{\delta} \cdot \sum_{j=1}^{k-1} (1 - \delta)^{k-1-j} ||(g_i^j)^2 / \hat{v}_i^j ||_1,
$$

where $a)$ uses the fact $(\sum_{j=1}^{k-1} a_j b_j)^2 \leq \sum_{j=1}^{k-1} a_j^2 \sum_{j=1}^{k-1} b_j^2$ with $a_j = (1 - \delta) \frac{\hat{v}_i^j}{\hat{v}_i^k}$ and $b_j = \frac{1}{\delta} \frac{\hat{v}_i^k}{\hat{v}_i^j}$, and $b)$ uses $\hat{v}_i^j \leq \hat{v}_i^k$ as $j \leq k$. We can further get

$$
\sum_{j=1}^{K-1} \sum_{j=1}^{k-1} (1 - \delta)^{k-1-j} ||(g_i^j)^2 / \hat{v}_i^j ||_1 \\
= \sum_{j=1}^{K-1} \frac{1}{\delta^2} \sum_{j=1}^{k-1} (1 - \delta)^{k-1-j} ||(g_i^j)^2 / \hat{v}_i^j ||_1 \\
\leq \frac{\sum_{j=1}^{K-1} ||(g_i^j)^2 / \hat{v}_i^j ||_1}{\delta^2},
$$

where $c)$ depends on the fact that $\hat{v}_i^k = \sum_{j=1}^{K-1} (\hat{v}_i^j)^2$. The result is proved by combining the inequalities above.

**Lemma 7** Assume $\{x^k\}_{k \geq 0}$ is generated by Ada$^2$m. Let

$$
\Upsilon_k := \mathbb{E} \left( -\gamma_k (\nabla f(x^k), m^k / (\hat{v}_i^k)^{\frac{k}{2}}) \right),
$$

we then have the following result

$$
\Upsilon_k \leq -\gamma_k (1 - \beta_k) \mathbb{E} ||\nabla f(x^k)^2 / (\hat{v}_i^k)^{\frac{k}{2}} ||_1 + \beta_k \Upsilon_{k-1} + \mathcal{R}_k,
$$

with $\mathcal{R}_k := \beta_k L \Xi_{k-1} + \gamma \hat{R} (\hat{R} + D) \cdot \sum_{j=1}^{d} (1 - \frac{1}{(\hat{v}_i^j)^{\frac{k}{2}}} - \frac{1}{(\hat{v}_i^j)^{\frac{k}{2}}} \hat{v}_i^j)^2$.

**Proof:** With direct computations,

$$
\Upsilon_k = \mathbb{E} \left( -\gamma (\nabla f(x^k), m^k / (\hat{v}_i^k)^{\frac{k}{2}}) \right) + \mathbb{E} \left( -\gamma \nabla f(x^k), m^k / (\hat{v}_i^k)^{\frac{k}{2}} \right) \cdot \chi^k + \mathbb{E} \left( \gamma \nabla f(x^k), m^k / (\hat{v}_i^k)^{\frac{k}{2}} - m^k / (\hat{v}_i^k)^{\frac{k}{2}} \right) \cdot \chi^k.
$$

Now, we are going to bound the terms I and II. The Cauchy’s inequality then gives us

$$
\begin{align*}
II & \leq \mathbb{E} (\nabla f(x^k), m^k / (\hat{v}_i^k)^{\frac{k}{2}} - m^k / (\hat{v}_i^k)^{\frac{k}{2}}) \cdot \chi^k \\
& \leq \gamma \|\nabla f(x^k)\| \cdot \|m^k\| \cdot \sum_{j=1}^{d} \frac{1}{(\hat{v}_i^j)^{\frac{k}{2}}} - 1 / (\hat{v}_i^j)^{\frac{k}{2}})^2 \\
& \leq \gamma R (\hat{R} + D) \cdot \frac{d}{\sum_{j=1}^{d} (\hat{v}_i^j)^{\frac{k}{2}}} - 1 / (\hat{v}_i^j)^{\frac{k}{2}}),
\end{align*}
$$

where the last inequality comes from the fact $\hat{v}_i^{k-1} \leq \hat{v}_i^k$. The scheme of the algorithm gives us

$$
\begin{align*}
I & \leq -\gamma (1 - \beta_k) \mathbb{E} ||\nabla f(x^k)^2 / (\hat{v}_i^k)^{\frac{k}{2}} ||_1 \\
& - \gamma \beta_k \mathbb{E} \mathbb{E} \left( -\gamma (\nabla f(x^k), m^k / (\hat{v}_i^k)^{\frac{k}{2}}) \right).
\end{align*}
$$

Then, we have the following result

$$
I = -\gamma (1 - \beta_k) ||\nabla f(x^k)^2 / (\hat{v}_i^k)^{\frac{k}{2}} ||_1 - \gamma \beta_k \mathbb{E} \left( -\gamma (\nabla f(x^k), m^k / (\hat{v}_i^k)^{\frac{k}{2}}) \right).
$$

Combination of the inequalities I and II leads to the final result.

**APPENDIX C**

**PROOF OF LEMMA 1**

Let $y^k := \begin{bmatrix} x^k \\ x_{k-1}^k \end{bmatrix}$, the heavy ball can be rewritten as

$$
y^k+1 = Ty^k + \begin{bmatrix} -\gamma & b \\ 0 & 0 \end{bmatrix},
$$

where

$$
T = \begin{bmatrix} 1 + \beta & -\gamma A & -\beta I \\ I & 0 \end{bmatrix}
$$

Let $x^*$ be the minimizer of $f$, i.e., $Ax^* + b = 0$. Then, $y^* := \begin{bmatrix} x^* \\ x^* \end{bmatrix}$ satisfies $Ty^* = y^* + \begin{bmatrix} -\gamma & b \\ 0 & 0 \end{bmatrix}$. Therefore, we can get

$$
y^* - y^* = T(y^k - y^*) = T^k (y^0 - y^*).
$$

We turn to exploit the eigenvalues of $T$, i.e., the complex number $\lambda$ satisfying

$$
\det \begin{bmatrix} (\lambda - 1 - \beta) I + \gamma A & -\beta I \\ -I & \lambda I \end{bmatrix} = 0.
$$
Notice that $T$ is non-singular, all eigenvalues are nonzero. We are then led to
\[
\det \left( \begin{pmatrix} \lambda + \frac{\beta}{2} - 1 - \beta & 0 \\ -1 & \lambda \end{pmatrix} + \gamma A \right) = 0
\]
\[
\implies \det((\lambda + \frac{\beta}{2}) - (1 + \beta)I - \gamma A) = 0.
\]
Thus, for $\lambda^*$ being any eigenvalue of $(1 + \beta)I - \gamma A$, we just need to consider
\[
\lambda + \frac{\beta}{2} = \lambda^*.
\]
(17)

To guarantee the convergence, $\lambda$ is required to be $|\lambda| < 1$.

**Convergence:** a) If $0 < \gamma \leq \frac{1}{\sqrt{2}}$, then $A \geq (1 - \sqrt{2})^2I$, and $1 + \beta > \lambda^* \geq 2\sqrt{2}$, which means (17) has real solutions. The function $h(\lambda) := \lambda + \frac{\beta}{2}$ is monotonic on $[\sqrt{2}, 1]$.

Due to the fact that
\[
h(\sqrt{2}) = 2\sqrt{2} \leq h(\lambda) = \lambda^* < 1 + \beta = h(\lambda),
\]
equation (17) has one root over $(\sqrt{2}, 1)$. Similarly, it has another root in $(0, \sqrt{2})$.

b) When $\frac{1}{\sqrt{2}} \leq \gamma \leq \frac{1}{\sqrt{2}}$, $(1 - \sqrt{2})^2I \geq A \geq (1 - \sqrt{2})^2I$. And then $-2\sqrt{2} \leq \lambda^* \leq 2\sqrt{2}$. The equation has complex roots $x$ and $\bar{x}$, which obeys $x\bar{x} = |x|^2 = \beta < 1$. In this case, we need $(1 - \sqrt{2})^2 \leq \lambda^* \leq (1 + \sqrt{2})^2$, i.e., $\beta \geq \left(\frac{1 - \sqrt{2}}{1 + \sqrt{2}}\right)^2$.

**Optimal choice:** For any fixed $\gamma \leq \frac{1}{\sqrt{2}}$, $(1 + \beta) - \gamma L \leq \lambda^* \leq (1 + \beta) - \gamma \nu$.

a) If $(1 + \beta) - \gamma L \geq 2\sqrt{2}$, (that is $\beta \leq (1 - \sqrt{rL})^2$), the larger root of (17) is
\[
\lambda^* + \sqrt{(\lambda^*)^2 - 4\beta} \leq \lambda^* < 1 + \beta - \gamma \nu.
\]
We want the right side is as small as possible and then set $\beta = 0$ and
\[
\lambda_{\max} \leq 1 - \gamma \nu.
\]

b) If $(1 + \beta) - \gamma \nu \leq 2\sqrt{2}$, (that is $\beta \geq (1 - \sqrt{rL})^2$). The equation has complex roots whose norms are both $\beta$. The optimal choice is then $\beta = (1 - \sqrt{rL})^2$ and then
\[
\lambda_{\max} \leq 1 - \sqrt{r\nu}.
\]
It is easy to see that $1 - \sqrt{r\nu}$ is smaller than $1 - \gamma \nu$.

With the Gelfand’s Theorem, given a small $\epsilon > 0$, there exists $K(\epsilon, \nu)$ such that
\[
y^k - y^* \leq (1 - \sqrt{r\nu} + \epsilon)\|y^0 - y^*\|
\]
as $k \geq K(\epsilon, \nu)$.

**Appendix D**

**Proof of Lemma 2**

Note the fact that
\[
(y^{k+1} - y^k) = T(y^k - y^{k-1}).
\]
Recall the matrix [16]. For $\lambda^*$ being the largest eigenvalue of $(1 + \beta)I - \gamma A$, the larger root is $x^* = \lambda^* + \sqrt{(\lambda^*)^2 - 4\beta}$. Thus, the largest eigenvalue of $T$ is
\[
t^* := \frac{(1 + \beta - \gamma \nu) + \sqrt{(1 + \beta - \gamma \nu)^2 - 4\beta}}{2}.
\]
Let $u$ be the vector satisfying $[(1 + \beta)I - \gamma A]u = \lambda^* u$. And $u$ corresponds the minimum eigenvalue of $A$. Then, we can check that
\[
T(t^* u) = t^* (t^* u),
\]
which means $v := (t^* u) ^* $ is the eigenvector with $t^*$. Due to that $A$ has unique minimum eigenvalue, $T$ has unique maximum eigenvalue. The Jordan canonical form indicates that there exists $P = [v, ...]$ such that
\[
P(t^* A) = P^{-1} T, \lim_k \lambda_k (\Lambda/t^*)^k = 0.
\]
With the definition $z^k := \begin{pmatrix} x^k - x^{k-1} \\ x^{k-1} - x^{k-2} \end{pmatrix}$, we have
\[
z^k = T^k z^0 = P \begin{pmatrix} (t^* k)^{-3} \\ \Lambda k^{-3} \end{pmatrix} P^{-1} z^0.
\]
as $k$ is large,
\[
z^k \approx P \begin{pmatrix} (t^* k)^{-3} \\ 0 \end{pmatrix} \left\{ P^{-1} z^0 \right\} = \left\{ [P]^{-1} z^0 \right\} (t^*)^{-3} v \in \text{Range}(v).
\]
That is also $\lim_k \| \frac{g^k - g^{k-1}}{\|x^k - x^{k-1}\|} \| = 1$. Therefore, we are then led to
\[
\lim_k \| g^k - g^{k-1} \| = \lim_k \| A(x^k - x^{k-1}) \| = \nu.
\]

**Appendix E**

**Proof of Theorem 1**

Direct computations give us
\[
E\|x^{k+1} - x^*\|^2
\]
\[
= E\|x^k - x^*\|^2 - 2\gamma E\langle x^k - x^*, \nabla f(x^k) \rangle
\]
\[
+ \beta_k E\|x^k - x^{k-1}, x^k - x^*\| + E\|\beta_k(x^k - x^{k-1}) - \gamma g^k\|^2
\]
\[
\leq E\|x^k - x^*\|^2 - 2\gamma E\|f(x^k) - f(x^*)\|
\]
\[
+ \beta_k E\|x^k - x^{k-1}, x^k - x^*\|
\]
\[
+ 2\gamma^2 E\|x^k - x^{k-1}\|^2 + 2\gamma^2 E\|g^k\|^2.
\]
(18)

Note that (15) still holds, from which we can use the MI method to prove
\[
\beta_k E\langle x^k - x^*, x^k - x^{k-1} \rangle \leq \frac{(1 - \delta) R^2}{\delta^3} \gamma^2.
\]
With the strongly convex property, (18) then gives us
\[
E\|x^{k+1} - x^*\|^2 \leq (1 - 2\gamma \nu) E\|x^k - x^*\|^2
\]
\[
+ \frac{(1 - \delta) R^2}{\delta^3} \gamma^2 + 4\gamma^2 \frac{R^2}{\delta^2} \gamma^2.
\]
(19)

Therefore, we get
\[
E\|x^k - x^*\|^2 \leq (1 - 2\gamma \nu) E\|x^0 - x^*\|^2 + \frac{(1 + 3\delta) R^2}{2\delta^3 \nu} \gamma.
\]
By setting $C_1 := \frac{(1 + 3\delta) R^2}{2\delta^3 \nu}$, we then proved the result.
APPENDIX F
PROOF OF THEOREM 2
The inequality (18) still holds. Then, we get
\[ 2\gamma \left( E( f(x^k) ) - f(x^*) \right) \leq E \| x^k - x^* \|^2 - E \| x^{k+1} - x^* \|^2 + \beta_k \| x^k - x^{k-1}, x^k - x^* \| + 2\gamma E \| x^k - x^{k-1} \|^2 + 2\gamma E \| g_k \|^2. \]
Summing from \( k = 1 \) to \( K \) and using Lemma 5, we have
\[ 2\gamma \sum_{k=1}^K (E f(x^k) - f(x^*)) \leq \sum_{k=1}^K E \| x^k - x^* \|^2 + \frac{(1 - \delta)}{\delta} \sum_{k=1}^K E \| x^k - x^{k-1} \|^2 + 2\gamma E \| x^k - x^{k-1} \|^2 + 2\gamma E \| g_k \|^2 \leq \sum_{k=1}^K E \| x^k - x^* \|^2 + \frac{(1 + \delta) \gamma^2 K R^2}{\delta^2} + 2\gamma^2 R^2 K. \]
The convexity of \( f \) gives us
\[ Ef \left( \frac{\sum_{k=1}^K x^k}{K} \right) - f(x^*) \leq \frac{E \| x^k - x^* \|^2}{2\gamma K} + \frac{(1 + \delta) \gamma R^2}{2\delta^3} + \gamma R^2. \]
Letting \( C_2 := \frac{(1 + \delta) \gamma^2}{2\delta^3} + R^2 \), we then get the result.

APPENDIX G
PROOF OF THEOREM 3
The Lipschitz property yields
\[ f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{L}{2} \| x^{k+1} - x^k \|^2 \]
\[ = f(x^k) - \gamma \langle \nabla f(x^k), g_k \rangle + \beta_k \| \nabla f(x^k), x^k - x^{k-1} \| + \frac{L}{2} \| x^{k+1} - x^k \|^2. \]
Taking expectation, we then get
\[ Ef(x^{k+1}) \leq Ef(x^k) - \gamma Ef(\nabla f(x^k)) \]
\[ + Ef(\beta_k (\nabla f(x^k), x^k - x^{k-1})) + \frac{L}{2} Ef \| x^{k+1} - x^k \|^2. \]
With Lemmas 3 and 4, we are then led to
\[ \sum_{k=1}^K \gamma \frac{\| \nabla f(x^k) \|^2}{2} \leq - \frac{\delta \gamma}{1 - \delta} \sum_{k=1}^{K-1} E \| \nabla f(x^k) \|^2 \]
\[ + \frac{(1 - \delta)}{\delta} L \sum_{k=1}^{K-1} E \| x^k - x^{k-1} \|^2 + f(x^1) - \min f. \]
Therefore,
\[ \min_{1 \leq k \leq K} \{ E \| \nabla f(x^k) \|^2 \} \leq \frac{4(1 - \delta)(1 - \frac{\delta}{2}) LR^2}{\delta^3} \gamma \]
\[ + \frac{2(1 - \delta)(f(x^0) - \min f)}{\gamma K}. \]
By denoting \( C_3 := \frac{4(1 - \delta)(1 - \frac{\delta}{2}) LR^2}{\delta^3} \) and \( C_4 := 2(1 - \delta) \), we get the result.
We can directly get \( \sum_{j=1}^{k} (1 - \beta_j) \left[ - \gamma \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 \right] \). Combining (22), we are then led to

\[
\begin{align*}
\sum_{j=1}^{k} (1 - \delta)^{k-j} \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 \\
\sum_{j=1}^{k} (1 - \delta)^{k-j} \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 \\
\leq \sum_{j=1}^{k} (1 - \delta)^{k-j} \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 \\
+ \sum_{j=1}^{k} (1 - \delta)^{k-j} \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1.
\end{align*}
\]

Summation from \( k = 1 \) to \( K \),

\[
\sum_{k=1}^{K} \sum_{j=1}^{k} (1 - \delta)^{k-j} \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 \\
+ \sum_{j=1}^{k} (1 - \delta)^{k-j} \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 = \sum_{k=1}^{k} \sum_{j=1}^{k} (1 - \delta)^{k-j} \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 \\
\leq \sum_{j=1}^{k} \gamma \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1.
\]

With the Lipschitz continuity of \( \nabla f \), at the point \( x_k \), we get

\[
f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{L}{2} \| x^{k+1} - x^k \|^2.
\]

Taking total condition expectation, we get \( \mathbb{E} f(x^{k+1}) \leq \mathbb{E} f(x^k) + \mathbb{E} y_k + \frac{L}{2} \mathbb{E} \sum_{k=1}^{K-1} \Xi_k \).

Combining (22), we are then led to

\[
\frac{1}{\delta} \sum_{k=1}^{K} \gamma \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 \leq (f(x^1) - \min f) + \frac{L}{2} \sum_{k=1}^{K-1} \Xi_k.
\]

We can directly get

\[
\sum_{k=1}^{K-1} \Xi_k + \sum_{k=1}^{K} \mathbb{E} \| \nabla f(x^j)/(\psi^j - 1) \|_1 \leq \frac{1}{\delta} \sum_{k=1}^{K} \beta_k \Xi_k + \frac{\gamma \hat{R}(\hat{R} + D)^2}{K(\pi)^{1/2}}.
\]

REFERENCES

[1] H. Robbins and S. Monro, “A stochastic approximation method,” The annals of mathematical statistics, pp. 400–407, 1951.
[2] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” USSR Computational Mathematics and Mathematical Physics, vol. 4, no. 5, pp. 1–17, 1964.
[3] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” ICLR, 2015.
[4] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” SIAM Journal on Imaging Sciences, vol. 2, no. 1, pp. 183–202, 2009.
[5] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” Siam Review, vol. 60, no. 2, pp. 223–318, 2018.
[6] A. Krizhevsky, G. Hinton, et al., “Learning multiple layers of features from tiny images,” Master’s thesis, University of Toronto, 2009.
[7] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in International Conference on Machine Learning, pp. 1139–1147, 2013.
[8] N. S. Keskar and R. Socher, “Improving generalization performance by switching from Adam to SGD,” arXiv preprint arXiv:1712.07628, 2017.
[9] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, “The marginal value of adaptive gradient methods in machine learning,” in Advances in Neural Information Processing Systems, pp. 4148–4158, 2017.
[10] L. Luo, Y. Xiong, and Y. Liu, “Adaptive gradient methods with dynamic learning rate,” in International Conference on Learning Representations, 2019.
[11] E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson, “Global convergence of the heavy-ball method for convex optimization,” in 2015 European control conference (ECC), pp. 310–315, IEEE, 2015.
[12] J.-K. Wang and J. Abernethy, “Provable acceleration of neural net training via polyak’s momentum,” in ICLR, 2020.
[13] P. Ochs, J. P. Y. Chen, T. Brox, and T. Pock, “ipiasco: Inertial proximal algorithm for massive convex optimization,” SIAM Journal on Imaging Sciences, vol. 60, no. 2, pp. 223–311, 2015.
[14] Y. E. Nesterov, “A method for solving the convex programming problem with convergence rate o(1/k^2),” in Dokl. Akad. Nauk SSSR, vol. 269, pp. 543–547, 1983.
[15] P. Ochs, Y. Chen, T. Brox, and T. Pock, “ipiasco: Inertial proximal algorithm for nonconvex optimization,” SIAM Journal on Imaging Sciences, vol. 7, no. 2, pp. 1388–1419, 2014.
[16] P. Ochs, T. Brox, and T. Pock, “ipiasco: Inertial proximal algorithm for strongly convex optimization,” Journal of Mathematical Imaging and Vision, vol. 53, no. 2, pp. 171–181, 2015.
[17] T. Sun, P. Yin, D. Li, C. Huang, L. Guan, and H. Jiang, “Non-ergodic convergence analysis of heavy-ball algorithms,” in Proceedings of the AAAI Conference on Artificial Intelligence, pp. 5033–5040, 2019.
[18] T. Sun, L. Qiao, and D. Li, “Nonergodic complexity of proximal inertial gradient descent,” IEEE Transactions on Neural Networks and Learning Systems, 2020.
[19] T. Sun, D. Li, Z. Quan, H. Jiang, S. Li, and Y. Dou, “Heavy-ball algorithms always escape saddle points,” in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp. 3520–3526, International Joint Conferences on Artificial Intelligence Organization, 7 2019.
K. Yuan, B. Ying, and A. H. Sayed, “On the influence of momentum,” *Advances in Neural Information Processing Systems*, pp. 9633–9643, 2019.

M. Cohen, J. Diakonikolas, and L. Orecchia, “On acceleration with noise-corrupted gradients,” in *Advances in Neural Information Processing Systems*, pp. 9633–9643, 2019.

Y. Nesterov, “Gradient methods,” *The Journal of Machine Learning Research*, vol. 13, pp. 580–647, 2012.

S. Vaswani, F. Bach, and M. Schmidt, “Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron,” *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1195–1204, 2019.

C. Liu, M. Belkin, “Accelerating SGD with momentum for over-parameterized learning,” in *International Conference on Learning Representations*, 2020.

M. Assran and M. Rabbat, “On the convergence of nesterov’s accelerated gradient method in stochastic settings,” in *International Conference on Machine Learning*, pp. 410–420, PMLR, 2020.

B. Wang, T. M. Nguyen, T. Sun, A. L. Bertozzi, R. G. Baraniuk, and S. J. Osher, “Scheduled restart momentum for accelerated stochastic gradient descent,” *arXiv preprint arXiv:2002.10583*, 2020.

G. Lan, “An optimal method for stochastic composite optimization,” *Mathematical Programming*, vol. 133, no. 1-2, pp. 365–397, 2012.

S. Ghadimi and G. Lan, “Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework,” *SIAM Journal on Optimization*, vol. 22, no. 4, pp. 1469–1492, 2012.

S. Ghadimi and G. Lan, “Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization ii: shrinking procedures and optimal algorithms,” *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2061–2089, 2013.

Z. Allen-Zhu, “Katyusha: The first direct acceleration of stochastic gradient methods,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 8194–8244, 2017.

M. Cohen, J. Diamkoukoulas, and L. Orecchia, “On acceleration with noise-corrupted gradients,” in *International Conference on Machine Learning*, pp. 1019–1028, 2018.

B. Wang and Q. Ye, “Stochastic gradient descent with non-linear conjugate gradient-style adaptive momentum,” *arXiv preprint arXiv:2012.02188*, 2020.

R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

R. Tibshirani, “Proximal gradient descent and acceleration,” *Lecture Notes*.