AMORTIZED NESTEROV'S MOMENTUM: ROBUST AND LIGHTWEIGHT MOMENTUM FOR DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Stochastic Gradient Descent (SGD) with Nesterov's momentum is a widely used optimizer in deep learning, which is observed to have excellent generalization performance. However, due to the large stochasticity, SGD with Nesterov's momentum is not robust, i.e., its performance may deviate significantly from the expectation. In this work, we propose *Amortized Nesterov's Momentum*, a special variant of Nesterov's momentum which has more robust iterates, faster convergence in the early stage and higher efficiency. Our experimental results show that this new momentum achieves similar (sometimes better) generalization performance with little-to-no tuning. In the convex case, we provide optimal convergence rates for our new methods and discuss how the theorems explain the empirical results.

1 INTRODUCTION

In recent years, Gradient Descent (GD) (Cauchy, 1847) and its variants have been widely used to solve large scale machine learning problems. Among them, Stochastic Gradient Descent (SGD) (Robbins & Monro, 1951), which replaces gradient with an unbiased stochastic gradient estimator, is a popular choice of optimizer especially for neural network training which requires lower precision. Sutskever et al. (2013) found that using SGD with Nesterov's momentum (Nesterov, 1983; 2013), which was originally designed to accelerate deterministic convex optimization, achieves substantial speedups for training neural networks. This finding essentially turns SGD with Nesterov's momentum into the benchmarking method of neural network design, especially for classification tasks (He et al., 2016b;a; Zagoruyko & Komodakis, 2016; Huang et al., 2017). It is observed that in these tasks, the momentum technique plays a key role in achieving good generalization performance.

Adaptive methods (Duchi et al., 2011; Kingma & Ba, 2015; Tieleman & Hinton, 2012; Reddi et al., 2018), which are also becoming increasingly popular in the deep learning community, diagonally scale the gradient to speed up training. However, Wilson et al. (2017) show that these methods always generalize poorly compared with SGD with momentum (both classical momentum (Polyak, 1964) and Nesterov's momentum).

In spite of having superior generalization performance, the SGD with Nesterov's momentum used in deep learning frameworks is not robust[1], as its performance varies significantly over multiple runs. In our preliminary experiments, it has larger deviations than plain SGD with the same learning rate. This increased uncertainty slows down its convergence especially in the early stage of training and makes its single run less trustworthy.

In this work, we introduce *Amortized Nesterov's Momentum*, a special variant of Nesterov's momentum, which has more robust iterates (even more robust than plain SGD), higher efficiency and faster convergence in the early stage without losing the good generalization performance of Nesterov's momentum (and we observed improved generalization in some cases). Moreover, there is little-to-no additional tuning overhead for this new momentum.

Our high level idea is that the stochastic Nesterov's momentum can be unreliable since it is provided only by the previous stochastic iterate. The iterate potentially has large variance, which may lead

[1]In this work, *robustness* refers to the probability of an optimizer significantly deviating from its expected performance, which can be reflected by the deviations of accuracy or loss in the training process over multiple runs that start with the same initial guess.

to a false momentum that perturbs the training process. We thus propose to use the stochastic Nesterov's momentum based on several past iterates, which provides robust acceleration. In other words, instead of immediately using an iterate to provide momentum, we put the iterate into an "amortization plan" and use it later.

2 Preliminaries: SGD and Nesterov's Momentum

We start with a review of SGD and Nesterov's momentum. We discuss some subtleties in the implementation and evaluation, which contributes to the interpretation of our methods.

Notations In this paper, we use $x \in \mathbb{R}^d$ to denote the vector of model parameters. $\|\cdot\|$ and $\langle \cdot, \cdot \rangle$ denote the standard Euclidean norm and inner product, respectively. Scalar multiplication for $v \in \mathbb{R}^d$ and $\beta \in \mathbb{R}$ is denoted as $\beta \cdot v$. $f : \mathbb{R}^d \to \mathbb{R}$ denotes the loss function to be minimized and $\nabla f(x)$ represents the gradient of f evaluated at x. We denote the unbiased stochastic gradient estimator of $\nabla f(x)$ as $\nabla f_i(x)$ with the random variable i independent of x (e.g., using mini-batch). We use $x_0 \in \mathbb{R}^d$ to denote the initial guess.

SGD SGD has the following simple iterative scheme, where $\gamma \in \mathbb{R}$ denotes the learning rate:

$$x_{k+1} = x_k - \gamma \cdot \nabla f_{i_k}(x_k), \text{ for } k \geq 0.$$

Nesterov's momentum The original Nesterov's accelerated gradient (with constant step) (Nesterov, 1983; 2013) has the following scheme[2] ($y \in \mathbb{R}^d, \eta, \beta \in \mathbb{R}$ and $y_0 = x_0$):

$$\begin{aligned} y_{k+1} &= x_k - \eta \cdot \nabla f(x_k), \\ x_{k+1} &= y_{k+1} + \beta \cdot (y_{k+1} - y_k), \text{ for } k \geq 0, \end{aligned} \tag{1}$$

where we call $\beta \cdot (y_{k+1} - y_k)$ the momentum. By simply replacing $\nabla f(x_k)$ with $\nabla f_{i_k}(x_k)$, we obtain the SGD with Nesterov's momentum, which is widely used in deep learning. To make this point clear, recall that the reformulation in Sutskever et al. (2013) (scheme (2), also the Tensorflow (Abadi et al., 2016) version) and the PyTorch (Paszke et al., 2017) version (scheme (3)) have the following schemes ($v, v^{pt} \in \mathbb{R}^d$ and $v_0 = v_0^{pt} = \mathbf{0}$): for $k \geq 0$,

$$(2) \begin{cases} v_{k+1} = \beta \cdot v_k - \eta \cdot \nabla f_{i_k}(y_k + \beta \cdot v_k), \\ y_{k+1} = y_k + v_{k+1}. \end{cases} \qquad (3) \begin{cases} v_{k+1}^{pt} = \beta \cdot v_k^{pt} + \nabla f_{i_k}(x_k), \\ x_{k+1} = x_k - \eta \cdot (\beta \cdot v_{k+1}^{pt} + \nabla f_{i_k}(x_k)). \end{cases}$$

Here the notations are modified based on their equivalence to scheme (1). It can be verified that schemes (2) and (3) are equivalent to (1) through $v_k = \beta^{-1} \cdot (x_k - y_k)$ and $v_k^{pt} = \eta^{-1}\beta^{-1} \cdot (y_k - x_k)$, respectively (see Defazio (2018) for other equivalent forms of scheme (1)).

Interestingly, both PyTorch and Tensorflow[3] track the values $\{x_k\}$, which we refer to as **M-SGD**. This choice allows a consistent implementation when wrapped in a generic optimization layer (Defazio, 2018) . However, the accelerated convergence rate (in the convex case) is built upon $\{y_k\}$ (Nesterov, 2013) and $\{x_k\}$ may not possess such a theoretical improvement. We use **M-SGD2** to refer to the M-SGD that outputs $\{y_k\}$.

How to compare SGD and M-SGD? For the choice of γ for SGD and (η, β) for M-SGD, Ma & Yarats (2019) fixed the effective learning rate[4] $\gamma = \eta(1-\beta)^{-1}$ in their experiments. Here we fix $\gamma = \eta$, which is a common choice with theoretical reasons: In general smooth convex optimization, γ and η should be constrained to $O(1/L)$ in both deterministic (Nesterov, 2013) and stochastic cases[5] (Lan, 2012), where L is the smoothness constant. From the interpretation in Allen-Zhu & Orecchia (2017), η represents the learning rate for the gradient descent "inside" Nesterov's method.

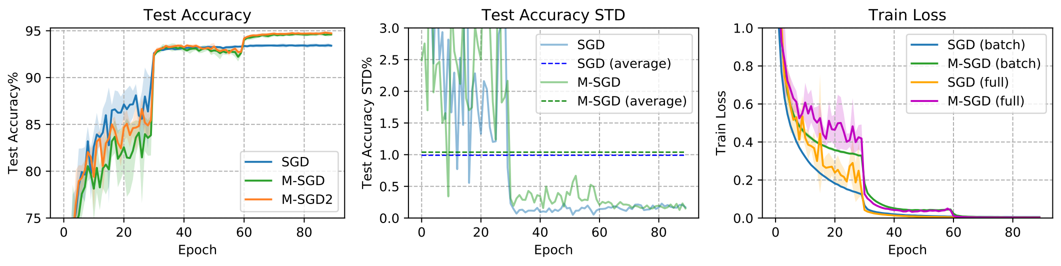
We report the results of training ResNet34 (He et al., 2016b) on CIFAR-10 (Krizhevsky et al., 2009) as our basic case study in Figure 1. In this paper, all the multiple runs start with the same initial

[2]We exchange the notations of x and y in Nesterov (2013).

[3]Tensorflow tracks the values $\{y_k + \beta \cdot v_k\} = \{x_k\}$.

[4]A discussion on this choice is given in Appendix C.1.

[5]η corresponds to $\beta_t^{-1}\gamma_t = O(1/L)$ in AC-SA (Lan, 2012), see Appendix B.1.



(a) (b) (c)

Figure 1: ResNet34 on CIFAR-10. Initial learning rate $\eta_0 = 0.1$, momentum $\beta = 0.9$, run 5 seeds (same x_0). In (a) (c), we plot mean curves with shaded bands indicating ± 1 standard deviation. (b) shows the standard deviation of test accuracy and its average over 90 epochs. Best viewed in color.

guess x_0. Figure 1a shows that SGD is faster in the first 60 epochs but is surpassed by M-SGD in the final stage, which verifies the importance of momentum for achieving high accuracy. Figure 1b justifies that M-SGD potentially has larger performance deviations than SGD.

Train-batch loss vs. Full-batch loss In Figure 1c, train-batch loss stands for the average of batch losses forwarded in an epoch, which is commonly used to indicate the training process in deep learning. Full-batch loss is the average loss over the entire training dataset evaluated at the end of each epoch. In terms of optimizer evaluation, full-batch loss is much more informative than train-batch loss as it reveals the robustness of an optimizer. However, full-batch loss is way too expensive to evaluate. On the other hand, test accuracy is "too informative" since it is related to generalization. Considering the basic usage of momentum in deep learning, we mainly use test accuracy to evaluate optimizers. We give a discussion on this issue in Appendix C.2.

M-SGD vs. M-SGD2 We also plot the curve of M-SGD2 in Figure 1a. Compared with M-SGD, M-SGD2 is more robust and achieves slightly better performance in this experiment (see Table 4 in Appendix A for the detailed data). This can be explained following the interpretation in Hinton (2012) that $\{x_k\}$ are the points after "jump" (or "gamble") and $\{y_k\}$ are the points after "correction".

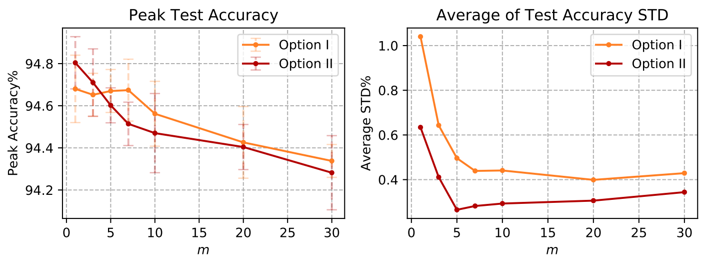
3 Amortized Nesterov's Momentum

Algorithm 1 AM1-SGD

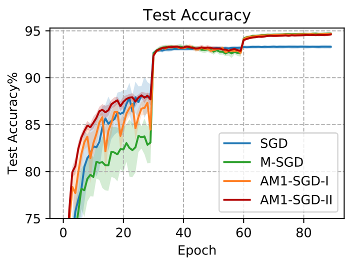
```
Input: Initial guess $x_0$, learning rate $\eta$, momentum $\beta$, amortization length $m$, iteration number $K$.
Initialize: $\tilde{x} \leftarrow x_0$, $\tilde{x}^+ \leftarrow \mathbf{0}$ {a running average}.
1: for $k = 0, \dots, K-1$ do
2:     $x_{k+1} = x_k - \eta \cdot \nabla f_{i_k}(x_k)$.
3:     $\tilde{x}^+ \leftarrow \tilde{x}^+ + \frac{1}{m} \cdot x_{k+1}$.
4:     if $(k+1) \mod m = 0$ then
5:         $x_{k+1} \leftarrow x_{k+1} + \beta \cdot (\tilde{x}^+ - \tilde{x})$. {adding amortized momentum}
6:         $\tilde{x} \leftarrow \tilde{x}^+, \tilde{x}^+ \leftarrow \mathbf{0}$.
7:     end if
8: end for
Output: Option I: $x_K$, Option II: $\tilde{x}$.        * The symbol '$\leftarrow$' denotes assignment.
```

In this section, we formally introduce SGD with Amortized Nesterov's Momentum (AM1-SGD) in Algorithm 1 with the following remarks:

Options It can be verified that if $m = 1$, AM1-SGD with Option I degenerates to M-SGD and Option II corresponds to M-SGD2. Just like the case for M-SGD and M-SGD2, the accelerated convergence rate is built upon Option II while Option I is easier to be implemented in a generic

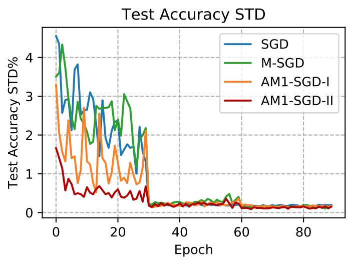


(a) Sweeping m in $\{3, 5, 7, 10, 20, 30\}$. Run 5 seeds.



(b) Fixing $m = 5$. Run 20 seeds.

Figure 2: ResNet34 on CIFAR-10. For all methods, $\eta_0 = 0.1, \beta = 0.9$. Labels of AM1-SGD are 'AM1-SGD-$\{$*Option*$\}$'. Shaded bands (or bars) indicate ± 1 standard deviation. Best viewed in color.



METHOD	PEAK ACCURACY	Avg. STD
SGD	$93.310\% \pm 0.167\%$	0.928%
M-SGD	$94.710\% \pm 0.169\%$	0.995%
AM1-SGD-I	$94.675\% \pm 0.177\%$	**0.587%**
AM1-SGD-II	$94.619\% \pm 0.152\%$	**0.314%**

Figure 3 & Table 1: Detailed data of the curves in Figure 2b. Best viewed in color.

optimization layer[6]. Intuitively, Option I is SGD with amortized momentum and Option II applies an m-iterations tail averaging on Option I.

Efficiency We can improve the efficiency of Algorithm 1 by maintaining a running scaled momentum $\tilde{v}^+ \triangleq m \cdot (\tilde{x}^+ - \tilde{x})$ instead of the running average $\tilde{x}^+$, by replacing the following steps in Algorithm 1:

Initialize: $\tilde{x} \leftarrow x_0, \tilde{v}^+ \leftarrow -m \cdot x_0,$ **Step 3:** $\tilde{v}^+ \leftarrow \tilde{v}^+ + x_{k+1}.$

Step 5: $x_{k+1} \leftarrow x_{k+1} + (\beta/m) \cdot \tilde{v}^+.$ **Step 6:** $\tilde{x} \leftarrow \tilde{x} + (1/m) \cdot \tilde{v}^+, \tilde{v}^+ \leftarrow -m \cdot \tilde{x}.$

Then, in one m-iterations loop, for each of the first $m - 1$ iterations, AM1-SGD requires 1 vector addition and 1 scaled vector addition. At the m-th iteration, it requires 1 vector addition, 1 scalar-vector multiplication and 3 scaled vector additions. In comparison, M-SGD (standard PyTorch) requires 1 vector addition, 1 (in-place) scalar-vector multiplication and 2 scaled vector additions per iteration. Thus, as long as $m > 2$, AM1-SGD has lower amortized cost than M-SGD. For memory complexity, AM1-SGD requires one more auxiliary buffer than M-SGD.

Hyperparameter tuning As the notations suggest, η and β in AM1-SGD have the same definitions as in M-SGD. Thus, the only tuning overhead is m, and hence we did a parameter sweep for it on our basic case study. We plot the peak (mean) and the average deviation of test accuracies over 5 runs against m in Figure 2a. Note that $m = 1$ corresponds to the results of M-SGD and M-SGD2 given in Figure 1. It is now clear that m introduces a trade-off between peak accuracy and robustness (see also Appendix A.1). Figure 2a suggests that $m = 5$ is a good choice for this task. For simplicity, and also as a recommended setting, we fix $m = 5$ for the rest of experiments in this paper.

A momentum that increases robustness To provide a stronger justification, we ran 20 seeds with $m = 5$ in Figure 2b and the detailed data are given in Figure 3 & Table 1. The results show that the amortized momentum significantly increases the robustness. Intuitively, the gap between Option I and Option II can be understood as the effect of tail averaging. However, the large gap between

[6]To implement Option II, we can either maintain another identical network for the shifted point $\tilde{x}$ or temporarily change the network parameters in the evaluation phase.

Option I and SGD is somewhat mysterious, which basically differentiates AM1-SGD from a simple interpolation in-between M-SGD and SGD.

Learning rate scheduler issue We observed that when we use schedulers with a large decay factor and the momentum β is too large for the task (e.g., 0.995 for the task of this section), there would be a performance drop after the learning rate reduction. We believe that it is caused by the different cardinalities of iterates being averaged in $\tilde{x}^+$, which leads to a false momentum. This issue is resolved by restarting the algorithm after each learning rate reduction as discussed in Appendix A.4.

3.1 AM2-SGD: A variant with identical iterations

Algorithm 2 AM2-SGD

```
Input: Initial guess $x_0$, amortization length $m$, a point table $\phi = [\phi_1 \quad \cdots \quad \phi_m] \in \mathbb{R}^{d\times m}$, learning
    rate $\eta$, momentum $\beta$, iteration number $K$.
Initialize: $\phi_j^0 = x_0, \forall j \in [m]$*. $\{j_k \mid j_k \in [m]\}_{k=0}^{K-1}$ is a sequence of uniformly random indexes.
    If Option II is used, $\bar{\phi}^0 = x_0$. {a running average for the point table $\phi$}
1: for $k = 0, \ldots, K-1$ do
2:     $\phi_{j_k}^{k+1} = x_k - \eta \cdot \nabla f_{i_k}(x_k)$ and keep other entries unchanged (i.e., $\phi_j^{k+1} = \phi_j^k$ for $j \neq j_k$).
3:     $x_{k+1} = \phi_{j_k}^{k+1} + \beta \cdot (\phi_{j_{k+1}}^{k+1} - \phi_{j_k}^k)$. {adding amortized momentum}
4:     if Option II then $\bar{\phi}^{k+1} = \bar{\phi}^k + \frac{1}{m} \cdot \left(\phi_{j_k}^{k+1} - \phi_{j_k}^k\right)$.
5: end for
Output: Option I: $x_K$, Option II: $\bar{\phi}^K$.                * $[m]$ denotes the set $\{1, \ldots, m\}$.
```

While enjoying an improved efficiency, AM1-SGD does not have identical iterations, which to some extent limits its extensibility to other settings (e.g., asynchronous setting). In this section, we propose a variant of Amortized Nesterov's Momentum (AM2-SGD, Algorithm 2) to address this problem. To show the characteristics of AM2-SGD, we make the following remarks:

Trading memory for extensibility In expectation, the point table ϕ stores the most recent m iterations and thus the output $\bar{\phi}^K$ is an m-iterations tail average, which connects to AM1-SGD. The relation between AM1-SGD and AM2-SGD resembles that of SVRG (Johnson & Zhang, 2013) and SAGA (Defazio et al., 2014), the most popular methods in finite-sum convex optimization: to reuse the information from several past iterates, we can either maintain a "snapshot" that aggregates the information or keep the iterates in a table.

Options and convergence As in the case of AM1-SGD, if $m = 1$, AM2-SGD with Option I corresponds to M-SGD and Option II is M-SGD2. In our preliminary experiments, the convergence of AM2-SGD is similar to AM1-SGD and it also has the learning rate scheduler issue. We did a small parameter sweep for AM2-SGD in Appendix A and observed that Option I is consistently worse than Option II. For Option II, when using the same m, we found that the performance of AM2-SGD is slightly better than AM1-SGD, which suggests that randomly incorporating past iterates beyond m iterations helps. We also set $m = 5$ for AM2-SGD for its evaluation due to the similarity.

Additional randomness $\{j_k\}$ AM2-SGD is a bit tricky in randomness. In our implementation, at each iteration, we sample a random index in $[m]$ as j_{k+1} and obtain the stored index j_k. We observed that with Option I, AM2-SGD has much larger deviations than AM1-SGD, which we believe is caused by the additional random indexes $\{j_k\}$.

4 Convergence Results

In this section, we establish optimal convergence rates for AM1-SGD and AM2-SGD in the general convex setting. Our analysis sheds light on the understanding of the empirical results in Figure 2. For generality, we focus on the following convex composite problem:

$$\min_{x\in X} \Big\{ F(x) \triangleq f(x) + h(x) \Big\}, \tag{4}$$

where $X \subseteq \mathbb{R}^d$ is a non-empty closed convex set and h is a proper convex function with its proximal operator $\text{prox}_{\alpha h}(\cdot)$[7] available. We denote $x^\star \in X$ as a solution to problem (4) and $x_0 \in X$ as the initial guess. All the proofs in this paper are given in Appendix B.

Unlike its usage in deep learning, the momentum parameter β is always a variable in general convex analysis. For the simplicity of analysis and interpretation purpose, we reformulate AM1-SGD (Algorithm 1) and AM2-SGD (Algorithm 2) into the following schemes[8]($z \in X, \alpha \in \mathbb{R}$):

AM1-SGD (reformulated, proximal)

```
Initialize: $\tilde{x}_0 = z_0 = x_0, S = K/m$.
1: for $s = 0, \dots, S-1$ do
2:     for $j = 0, \dots, m-1$ do
3:         $k = sm + j$.
4:         $x_k = (1-\beta_s) \cdot z_k + \beta_s \cdot \tilde{x}_s$.
5:         $z_{k+1} = \text{prox}_{\alpha_s h}\{z_k - \alpha_s \cdot \nabla f_{i_k}(x_k)\}$.
6:         $(x_{k+1} = (1-\beta_s) \cdot z_{k+1} + \beta_s \cdot \tilde{x}_s.)$
7:     end for
8:     $\tilde{x}_{s+1} = \frac{1}{m}\sum_{j=1}^{m} x_{sm+j}$.
9: end for
Output: $\tilde{x}_S$.
```

AM2-SGD (reformulated, proximal)

```
Initialize: $z_0 = \phi_j^0 = x_0, \forall j \in [m]$.
1: for $k = 0, \dots, K-1$ do
2:     Sample $j_k$ uniformly in $[m]$.
3:     $x_k^{j_k} = (1-\beta_k) \cdot z_k + \beta_k \cdot \phi_{j_k}^k$.
4:     $z_{k+1} = \text{prox}_{\alpha_k h}\{z_k - \alpha_k \cdot \nabla f_{i_k}(x_k^{j_k})\}$.
5:     $\phi_{j_k}^{k+1} = (1-\beta_k) \cdot z_{k+1} + \beta_k \cdot \phi_{j_k}^k$.
6: end for
Output: $\bar{\phi}^K = \frac{1}{m}\sum_{j=1}^{m} \phi_j^K$.
```

We show in Appendix B.1 that when $h \equiv 0$ and β is a constant, the reformulated schemes AM1-SGD and AM2-SGD are equivalent to Algorithm 1 and Algorithm 2 through $\alpha_s = \eta(1-\beta_s)^{-1}$ and $\alpha_k = \eta(1-\beta_k)^{-1}$. Note that in the reformulated AM1-SGD, Step 6 is only for a unified proof and not needed since the sequence $\{x_k\}$ is totally defined by $\{z_k\}$.

Comparing the reformulated schemes, we see that their iterations can be generalized as follows:

$$\begin{aligned} x &= (1-\beta) \cdot z + \beta \cdot y, \\ z^+ &= \text{prox}_{\alpha h}\{z - \alpha \cdot \nabla f_i(x)\}, \\ y^+ &= (1-\beta) \cdot z^+ + \beta \cdot y. \end{aligned} \tag{5}$$

This type of scheme is first proposed in Auslender & Teboulle (2006), which represents one of the simplest variants of the Nesterov's methods (see Tseng (2008) for other variants). The scheme is then modified into various settings (Lan, 2012; Ghadimi & Lan, 2012; 2016; Zhou et al., 2019; Lan et al., 2019) to achieve acceleration. For a random process $i_0, i_1, \dots$, we assume the sequence $\{i_k\}$ is i.i.d. and use the notation $\mathbb{E}_{i_k}[\,\cdot\,] = \mathbb{E}[\,\cdot \mid (i_0, \dots, i_{k-1})]$. We also need the following assumptions:

Assumptions. *For some $L \geq 0, M \geq 0, \sigma \geq 0$,*

(a) $0 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \frac{L}{2}\|y-x\|^2 + M\|y-x\|, \forall x, y \in X.$[9]

(b) $\mathbb{E}_i[\nabla f_i(x)] = \nabla f(x), \forall x \in X.$

(c) $\mathbb{E}_i\left[\|\nabla f_i(x) - \nabla f(x)\|^2\right] \leq \sigma^2, \forall x \in X.$

These assumptions are identical to those used in Ghadimi & Lan (2012) (with $\mu = 0$), which cover several important classes of convex problems (e.g., f can be smooth or non-smooth convex functions and the (sub)gradient oracle can be deterministic, see Ghadimi & Lan (2012) for more discussion). The following lemma serves as a cornerstone for the convergence analysis of our algorithms.

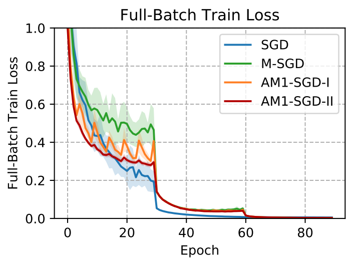
Lemma 1. *If $\alpha(1-\beta) < 1/L$, the update scheme (5) satisfies the following recursion:*

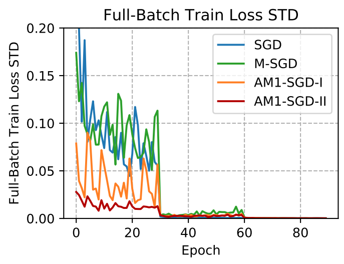
$$\begin{aligned} \frac{1}{1-\beta}\big(F(y^+) - F(x^\star)\big) \leq \frac{\beta}{1-\beta}\big(F(y) - F(x^\star)\big) + \frac{1}{2\alpha}\left(\|z - x^\star\|^2 - \|z^+ - x^\star\|^2\right) \\ + \frac{(\|\nabla f(x) - \nabla f_i(x)\| + M)^2}{2(\alpha^{-1} - L(1-\beta))} + \langle \nabla f(x) - \nabla f_i(x), z - x^\star \rangle . \end{aligned}$$

[7]$\forall x \in \mathbb{R}^d, \text{prox}_{\alpha h}(x) \triangleq \arg\min_{u \in X}\left\{\frac{1}{2}\|u-x\|^2 + \alpha h(u)\right\}$, see Parikh et al. (2014).

[8]For simplicity, we assume K is divisible by m.

[9]When $M > 0$, f is not necessarily differentiable and we keep using the notation $\nabla f(x)$ to denote an arbitrary subgradient of f at x for consistency.





METHOD	Avg. STD
SGD	0.0329
M-SGD	0.0339
AM1-SGD-I	**0.0142**
AM1-SGD-II	**0.0056**

Figure 4 & Table 2: ResNet18 with pre-activation on CIFAR-10. For all methods, $\eta_0 = 0.1, \beta = 0.9$, run 20 seeds. For AM1-SGD, $m = 5$ and its labels are formatted as 'AM1-SGD-{*Option*}'. Shaded bands indicate ± 1 standard deviation. Best viewed in color.

Based on this key recursion, we establish the convergence rates for AM1-SGD and AM2-SGD by casting their iterations as instances of scheme (5).

Theorem 1. *For the reformulated AM1-SGD, suppose we choose*

$$\beta_s = \frac{s}{s+2} \text{ and } \alpha_s = \frac{\lambda_1}{L(1-\beta_s)} \text{ with } \lambda_1 = \min\left\{\frac{2}{3}, \frac{L\|x_0 - x^\star\|}{2\sqrt{m}\sqrt{\sigma^2 + M^2}(S+1)^{\frac{3}{2}}}\right\}. \tag{6}$$

Then,

(a) *The output $\tilde{x}_S$ satisfies*

$$\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) \leq \frac{3Lm\|x_0 - x^\star\|^2}{(K+m)^2} + \frac{8\|x_0 - x^\star\|\sqrt{\sigma^2 + M^2}}{\sqrt{K+m}} \triangleq \mathcal{K}_0(m).$$

(b) *If the variance has a "light tail", i.e., $\mathbb{E}_i\left[\exp\left\{\|\nabla f_i(x) - \nabla f(x)\|^2/\sigma^2\right\}\right] \leq \exp\{1\}, \forall x \in X$, and X is compact, denoting $D_X \triangleq \max_{x\in X}\|x - x^\star\|$, for any $\Lambda \geq 0$, we have*

$$\text{Prob}\left\{F(\tilde{x}_S) - F(x^\star) \leq \mathcal{K}_0(m) + \frac{4\Lambda\sigma\left(3\|x_0 - x^\star\| + \sqrt{6}D_X\right)}{3\sqrt{K+m}}\right\}$$
$$\geq 1 - \left(\exp\{-\Lambda^2/3\} + \exp\{-\Lambda\}\right).$$

Remarks for Theorem 1a: Theorem 1a gives the expected performance, which is related to the mean accuracy curve in Figure 2b. We see that increasing m improves the dependence on variance σ but deteriorates the $O(L/K^2)$ term (i.e., the acceleration). By connecting to Figure 2b, we believe that the improved performance in the first 30 epochs is the result of a better control on σ and the slightly lowered peak accuracy is caused by the reduced acceleration effect. Regarding $\mathcal{K}_0(m)$, its minimum is obtained at either $m = 1$ or $m = K$. Note that for AM1-SGD, m is strictly constrained in $\{1, \ldots, K\}$. It can be verified that when $m = K$, AM1-SGD becomes the modified mirror descent SA (Lan, 2012), or under the Euclidean setting, the SGD that outputs the average of the whole history, which is rarely used in practice. In this case, the convergence rate in Theorem 1a becomes the corresponding $O(L/K + (\sigma + M)/\sqrt{K})$ (cf. Theorem 1 in Lan (2012)). Thus, we can regard AM1-SGD as a smooth transition between AC-SA and the modified mirror descent SA.

Remarks for Theorem 1b: Theorem 1b provides a sub-Gaussian confidence bound, which depicts the large deviations of full-batch loss. The additional compactness and "light tail" assumptions are similarly required in Nemirovski et al. (2009); Lan (2012); Ghadimi & Lan (2012). Recently, Juditsky et al. (2019) established sub-Gaussian confidence bounds under weaker assumptions by truncating the gradient. However, as indicated by the authors, their technique cannot be used for accelerated algorithms due to the accumulation of bias. From Theorem 1b, we see that increasing m leads to smaller deviations with the same probability, which to some extent explains the reduced deviation of test accuracy in Figure 3 & Table 1. To provide a stronger justification for Theorem 1, we evaluate AM1-SGD based on full-batch loss in Figure 4 & Table 2. Here we choose training ResNet18 with pre-activation (He et al., 2016a) on CIFAR-10 as the case study (the test accuracy is reported in Appendix A.5).

For AM2-SGD, we only give the expected convergence results as follows.

Theorem 2. *For the reformulated AM2-SGD, if we choose*

$$\beta_k = \frac{k/m}{k/m+2} \text{ and } \alpha_k = \frac{\lambda_2}{L(1-\beta_k)} \text{ with } \lambda_2 = \min\left\{\frac{2}{3}, \frac{L\left\|x_0 - x^\star\right\|}{\sqrt{2m}(\sigma+M)\left(\frac{K-1}{m}+2\right)^{\frac{3}{2}}}\right\},$$

the output $\bar{\phi}^K$ satisfies

$$\mathbb{E}\left[F(\bar{\phi}^K)\right] - F(x^\star) \le \frac{4(m^2-m)\big(F(x_0)-F(x^\star)\big)+3Lm\left\|x_0-x^\star\right\|^2}{(K+2m-1)^2} + \frac{4\sqrt{2}\left\|x_0-x^\star\right\|(\sigma+M)}{\sqrt{K+2m-1}}.$$

Remarks for Theorem 2: In comparison with Theorem 1a, Theorem 2 has an additional term $F(x_0)-F(x^\star)$ in the upper bound, which is inevitable. This difference comes from different restrictions on the choice of m. For AM2-SGD, $m \ge 1$ is the only requirement. Since it is impossible to let $m \gg K$ to obtain an improved rate, this additional term is inevitable. As a sanity check, we can let $m \to \infty$ to obtain a point table with almost all x_0, and then the upper bound becomes exactly $F(x_0) - F(x^\star)$. Comparing the rates, we see that when using the same m, AM2-SGD has slightly better dependence on σ, which explains the observation in Figure 5 that AM2-SGD is slightly faster than AM1-SGD. In some cases, there exists an optimal choice of $m > 1$ in Theorem 2. However, the optimal choice could be messy and thus we omit the discussion here.

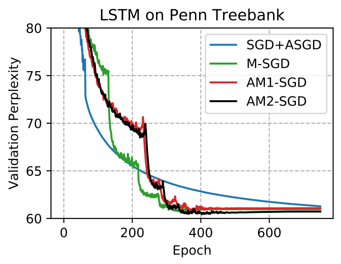
If $m = O(1)$, Theorems 1 and 2 establish the optimal $O(L/K^2 + (\sigma+M)/\sqrt{K})$ rate in the convex setting (see Lan (2012) for optimality), which verifies AM1-SGD and AM2-SGD as variants of the Nesterov's method (Nesterov, 1983; 2013). In the following we discuss some connections between AM1-SGD and Katyusha (Allen-Zhu, 2018), the recent breakthrough in finite-sum convex optimization. We believe that such a discussion sheds light on why the experimental results are more significant than what the theorem predicts (which is at most a constant level effect).

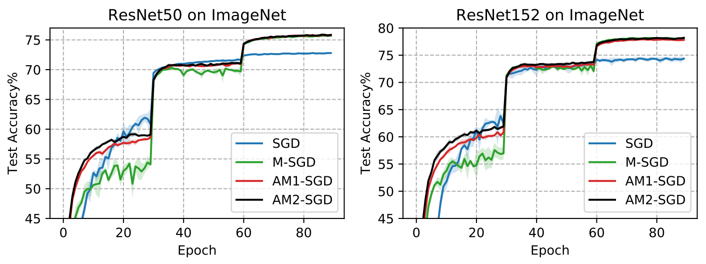
Connections with Katyusha Our original inspiration of AM1-SGD comes from the construction of Katyusha, which uses a previously calculated "snapshot" point to provide momentum, i.e., Katyusha momentum. The high level idea of Katyusha momentum is that it works as a "magnet" inside an epoch of SVRG updates, which "stabilizes" the iterates so as to make Nesterov's momentum effective (Allen-Zhu, 2018). In theory, the key effect of Katyusha momentum is that it allows the tightest possible variance bound for the stochastic gradient estimator of SVRG (cf. Lemma 2.4 and its comments in Allen-Zhu (2018)). In this sense, we can interpret Katyusha momentum as a variance reducer that further reduces the variance of SVRG. AM1-SGD also uses an aggregated point to provide momentum and it shares many structural similarities with Katyusha, which we show in Appendix B.5. We thus conjecture that the amortized momentum can also reduce the variance of SGD (i.e., σ). However, in theory, following a similar analysis of Katyusha, we cannot guarantee a reduction of σ in the worst case.

5 Performance Evaluation

In this section, we provide more empirical results for AM1-SGD and AM2-SGD. We choose to evaluate Option II for both methods since it corresponds to our analysis in the previous section. The performance of Option I is observed to be more unpredictable on convolutional networks (see Appendix A). Note that for all the experiments in this paper, AM1-SGD and AM2-SGD use exactly the same pair of (η, β) as M-SGD. Basically, the hyperparameter choice is to optimize the performance of M-SGD and then we directly evaluate AM1-SGD and AM2-SGD using that choice with $m = 5$.

We trained ResNet50 and ResNet152 (He et al., 2016b) on the ILSVRC2012 dataset ("ImageNet") (Russakovsky et al., 2015) shown in Figure 5b. For all methods, we used 0.1 initial learning rate and 0.9 momentum, which is the typical choice for this task. We performed a restart after each learning rate reduction as discussed in Appendix A.4. We believe that this helps the training process and also does not incur any additional overhead. We report the peak accuracy in Table 3. AM1-SGD and AM2-SGD perform well on the ImageNet experiments. Especially for the ResNet50 experiment, both AM1-SGD and AM2-SGD converge faster than M-SGD during the entire training process.





(a) LSTM on Penn Treebank. (b) ResNet on ImageNet. Run 3 seeds.

Figure 5: Convergence of LSTM and ResNet. We plot the curve of validation perplexity and test accuracy, respectively. Shaded bands indicate ± 1 standard deviation. Best viewed in color.

Table 3: Detailed perplexity and accuracy results for Figure 5.

METHOD	**Penn Treebank (Perplexity)**		**ImageNet (Peak Accuracy)**	
	Validation	Test	ResNet50	ResNet152
SGD (+ASGD)	61.283	59.068	$72.801\% \pm 0.103\%$	$74.439\% \pm 0.229\%$
M-SGD	60.747	58.355	$75.711\% \pm 0.062\%$	$78.192\% \pm 0.099\%$
AM1-SGD	60.734	**57.977**	$\mathbf{75.838}\% \pm 0.023\%$	$77.926\% \pm 0.269\%$
AM2-SGD	**60.434**	58.233	$\mathbf{75.863}\% \pm 0.297\%$	$\mathbf{78.194}\% \pm 0.147\%$

We also did a language model experiment on Penn Treebank dataset (Marcus et al., 1993). We used the LSTM (Hochreiter & Schmidhuber, 1997) model defined in Merity et al. (2017) and followed the experimental setup in its released code. We only changed the learning rate and momentum in the setup. The baseline is SGD+ASGD[10] (Polyak & Juditsky, 1992) with constant learning rate 30 as used in Merity et al. (2017). For the tuning of M-SGD, following Lucas et al. (2019), we chose $\beta = 0.99$ and used the scheduler that reduces the learning rate by half when the validation loss has not decreased for 15 epochs. We swept η from $\{5, 2.5, 1, 0.1, 0.01\}$ and found that $\eta = 2.5$ resulted in the lowest validation perplexity for M-SGD. We thus ran AM1-SGD and AM2-SGD with this η, β and $m = 5$. Due to the small decay factor, we did not restart AM1-SGD and AM2-SGD after learning rate reductions. The validation perplexity curve is plotted in Figure 5a. We report validation perplexity and test perplexity in Table 3. We see that AM2-SGD achieves the lowest validation perplexity while AM1-SGD has the best test performance, both of which are better than M-SGD and SGD+ASGD. This experiment is directly comparable with the one in Lucas et al. (2019).

In the appendices, we provide extra results: the robustness when using large β (Appendix A.2), a CIFAR-100 experiment (Appendix A.6) and comparison with classical momentum (Polyak, 1964), AggMo (Lucas et al., 2019) and QHM (Ma & Yarats, 2019) (Appendix A.3).

6 Conclusions

We presented Amortized Nesterov's Momentum, which is a special variant of Nesterov's momentum that utilizes several past iterates to provide the momentum. Based on this idea, we designed two different realizations, namely, AM1-SGD and AM2-SGD. Both of them are simple to implement with little-to-no additional tuning overhead and have more robust iterates than their vanilla counterparts. Our theory provides optimal convergence rates for them in the convex case and explains their observed convergence behaviors. Our empirical results demonstrate that both AM1-SGD and AM2-SGD achieve similar (sometimes better) generalization performance compared with M-SGD. AM1-SGD has a lightweight design, which can serve as an excellent replacement for M-SGD in large-scale deep learning tasks. AM2-SGD, due to its extensibility and good performance, could be favorable for more restrictive tasks (e.g., asynchronous training).

[10]SGD+ASGD is to run SGD and switch to averaged SGD (ASGD) when a threshold is met.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221):1–51, 2018. URL `http://jmlr.org/papers/v18/16-410.html`.

Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. In *ITCS*, 2017.

Alfred Auslender and Marc Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM Journal on Optimization*, 16(3):697–725, 2006.

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

Aaron Defazio. On the curved geometry of accelerated optimization. *arXiv preprint arXiv:1812.04634*, 2018.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pp. 1646–1654, 2014.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012.

Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016a.

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, 2016b.

Geoffrey Hinton. Lecture 6C : The momentum method, 2012. URL `http://www.cs.toronto.edu/~hinton/coursera_lectures.html`.

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

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Anatoli Juditsky, Alexander Nazin, Arkadi Nemirovsky, and Alexandre Tsybakov. Algorithms of robust stochastic optimization based on mirror descent method. *arXiv preprint arXiv:1907.02707*, 2019.

Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2):365–397, 2012.

Guanghui Lan, Arkadi Nemirovski, and Alexander Shapiro. Validation analysis of mirror descent stochastic approximation method. *Mathematical programming*, 134(2):425–458, 2012.

Guanghui Lan, Zhize Li, and Yi Zhou. A unified variance-reduced accelerated gradient method for convex optimization. *arXiv preprint arXiv:1905.12412*, 2019.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

James Lucas, Shengyang Sun, Richard Zemel, and Roger Grosse. Aggregated momentum: Stability through passive damping. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Syxt5oC5YQ`.

Jerry Ma and Denis Yarats. Quasi-hyperbolic momentum and adam for deep learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=S1fUpoR5FQ`.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Dokl. Akad. Nauk SSSR*, volume 269, pp. 543–547, 1983.

Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ryQu7f-RZ`.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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, 2013.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. 2008.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4148–4158, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Kaiwen Zhou, Fanhua Shang, and James Cheng. A simple stochastic variance reduced algorithm with fast convergence rates. In *International conference on machine learning*, pp. 5980–5989, 2018.

Kaiwen Zhou, Qinghua Ding, Fanhua Shang, James Cheng, Danli Li, and Zhi-Quan Luo. Direct Acceleration of SAGA using Sampled Negative Momentum. In *AISTATS*, pp. 1602–1610, 2019.

Appendices

A EXTRA EXPERIMENTAL RESULTS

In this appendix, we provide more experimental results to further evaluate the Amortized Nesterov's Momentum. Table 4 shows the detailed data of the parameter sweep experiments, where the convergence curves of these results are given in Appendix A.1. In Appendix A.2, we compare the robustness of AM1-SGD and M-SGD on large momentum parameters. In Appendix A.3, we empirically compare the Amortized Nesterov's Momentum with classical momentum (Polyak, 1964), aggregated momentum (Lucas et al., 2019) and quasi-hyperbolic momentum (Ma & Yarats, 2019). We discuss the issues with learning rate schedulers in Appendix A.4. We report the test accuracy results of the ResNet18 experiment (in Section 4) in Appendix A.5. A CIFAR-100 experiment is provided in Appendix A.6. We also provide a sanity check for our implementation in Appendix A.7.

Table 4: Peak test accuracy and average accuracy STD of training ResNet34 on CIFAR-10 over 5 runs (including the detailed data of the curves in Figure 1 and Figure 2a). For all the methods, $\eta_0 = 0.1, \beta = 0.9$. Multiple runs start with the same x_0.

METHOD	**DESCRIPTION**	**PEAK ACCURACY**	**Avg. STD**
SGD	Standard Pytorch	$93.460\% \pm 0.175\%$	0.991%
M-SGD	Standard Pytorch	$94.680\% \pm 0.160\%$	1.040%
AM1-SGD	Option I, $m = 1$, sanity check	$94.690\% \pm 0.212\%$	0.912%
AM1-SGD	Option I, $m = 3$	$94.652\% \pm 0.102\%$	0.643%
AM1-SGD	Option I, $m = 5$	$94.670\% \pm 0.102\%$	0.496%
AM1-SGD	Option I, $m = 7$	$94.674\% \pm 0.147\%$	0.439%
AM1-SGD	Option I, $m = 10$	$94.562\% \pm 0.154\%$	0.441%
AM1-SGD	Option I, $m = 20$	$94.426\% \pm 0.170\%$	0.399%
AM1-SGD	Option I, $m = 30$	$94.338\% \pm 0.078\%$	0.429%
M-SGD2	AM1-SGD (Opt. II, $m = 1$)	$94.804\% \pm 0.124\%$	0.634%
AM1-SGD	Option II, $m = 3$	$94.710\% \pm 0.160\%$	0.411%
AM1-SGD	Option II, $m = 5$	$94.602\% \pm 0.083\%$	0.265%
AM1-SGD	Option II, $m = 7$	$94.514\% \pm 0.103\%$	0.282%
AM1-SGD	Option II, $m = 10$	$94.470\% \pm 0.188\%$	0.293%
AM1-SGD	Option II, $m = 20$	$94.404\% \pm 0.107\%$	0.306%
AM1-SGD	Option II, $m = 30$	$94.282\% \pm 0.176\%$	0.344%
AM2-SGD	Option I, $m = 1$, sanity check	$94.712\% \pm 0.178\%$	0.822%
AM2-SGD	Option I, $m = 5$	$94.592\% \pm 0.103\%$	0.591%
AM2-SGD	Option I, $m = 10$	$94.506\% \pm 0.117\%$	0.737%
AM2-SGD	Option I, $m = 20$	$94.318\% \pm 0.088\%$	0.741%
AM2-SGD	Option II, $m = 5$	$94.706\% \pm 0.059\%$	0.263%
AM2-SGD	Option II, $m = 10$	$94.554\% \pm 0.203\%$	0.280%
AM2-SGD	Option II, $m = 20$	$94.420\% \pm 0.144\%$	0.246%

A.1 THE EFFECT OF m ON CONVERGENCE

We show in Figure 6 how m affects the convergence of test accuracy. The results show that increasing m speeds up the convergence in the early stage. While for AM1-SGD the convergences of Option I and Option II are similar, AM2-SGD with Option II is consistently better than with Option I in this experiment. It seems that AM2-SGD with Option I does not benefit from increasing m and the algorithm is not robust. Thus, we do not recommend using Option I for AM2-SGD.

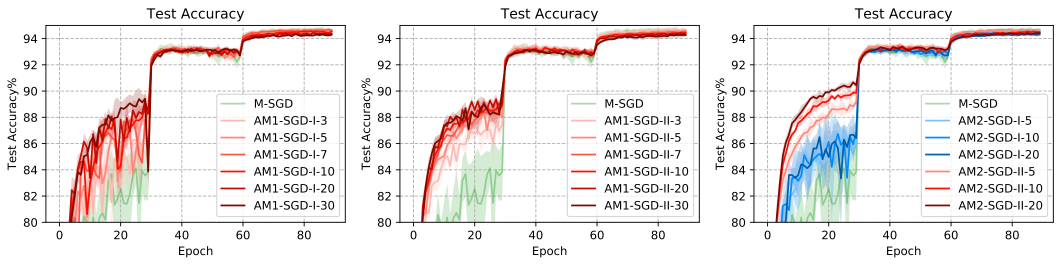
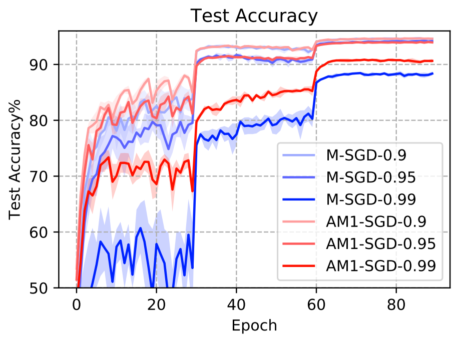


Figure 6: Convergence of test accuracy from the parameter sweep experiments in Table 4. Labels are formatted as 'AM1/2-SGD-$\{Option\}$-$\{m\}$' . Best viewed in color.

A.2 Robustness on large momentum parameters

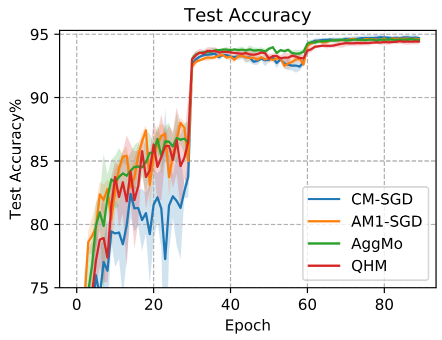
We compare the robustness of M-SGD and AM1-SGD when β is large in Figure 7 & Table 5. For fair comparison, AM1-SGD uses Option I. As we can see, the STD error of M-SGD scales up significantly when β is larger and the performance is more affected by a large β compared with AM1-SGD.



METHOD	PEAK ACCURACY	Avg. STD
M-SGD-0.9	$94.680\% \pm 0.160\%$	1.040%
M-SGD-0.95	$94.238\% \pm 0.081\%$	1.200%
M-SGD-0.99	$88.458\% \pm 0.223\%$	2.561%
AM1-SGD-0.9	$94.670\% \pm 0.102\%$	0.496%
AM1-SGD-0.95	$94.016\% \pm 0.110\%$	0.581%
AM1-SGD-0.99	$90.856\% \pm 0.170\%$	0.900%

Figure 7 & Table 5: ResNet34 on CIFAR-10. $\eta_0 = 0.1, \beta \in \{0.9, 0.95, 0.99\}$, run 5 seeds (the $\beta = 0.9$ results are copied from Table 4). Labels are formatted as "$\{Algorithm\}$-$\{\beta\}$". Best viewed in color.

A.3 Comparison with other momentum



METHOD	PEAK ACCURACY	Avg. STD
CM-SGD	$94.764\% \pm 0.186\%$	1.107%
AM1-SGD	$94.670\% \pm 0.102\%$	0.496%
AggMo	$94.610\% \pm 0.152\%$	0.777%
QHM	$94.454\% \pm 0.212\%$	1.073%
M-SGD	$94.680\% \pm 0.160\%$	1.040%

Figure 8 & Table 6: ResNet34 on CIFAR-10. Run 5 seeds. The results of AM1-SGD and M-SGD are copied from Table 4. Best viewed in color.

In this section, we compare AM1-SGD (Option I) with classical momentum (Polyak, 1964), AggMo (Lucas et al., 2019) and QHM (Ma & Yarats, 2019) in our basic case study (training ResNet34 on

CIFAR-10). Since we are not aware of what makes a fair comparison with these methods (e.g., it is not clear what is the effective learning rate for AM1-SGD), we compare them based on the default hyperparameter settings suggested by their papers.

Classical Momentum The SGD with classical momentum (CM-SGD) that is widely used in deep learning has the following scheme (standard PyTorch) ($v^{cm} \in \mathbb{R}^d, v_0^{cm} = \mathbf{0}$):

$$\begin{aligned} v_{k+1}^{cm} &= \beta \cdot v_k^{cm} + \nabla f_{i_k}(x_k), \\ x_{k+1} &= x_k - \eta \cdot v_{k+1}^{cm}, \text{ for } k \geq 0. \end{aligned}$$

CM-SGD with its typical hyperparameter settings ($\eta_0 = 0.1, \beta = 0.9$) is observed to achieve similar generalization performance as M-SGD. However, CM-SGD is more unstable and prone to oscillations (Lucas et al., 2019), which makes it less robust than M-SGD as shown in Table 6.

Aggregated Momentum (AggMo) AggMo combines multiple momentum buffers, which is inspired by the passive damping from physics literature (Lucas et al., 2019). AggMo uses the following update rules (for $t = 1, \ldots, T$, $v^{(t)} \in \mathbb{R}^d, v_0^{(t)} = \mathbf{0}$):

$$\begin{aligned} v_{k+1}^{(t)} &= \beta^{(t)} \cdot v_k^{(t)} - \nabla f_{i_k}(x_k), \text{ for } t = 1, \ldots, T, \\ x_{k+1} &= x_k + \frac{\eta}{T} \cdot \sum_{t=1}^{T} v_{k+1}^{(t)}, \text{ for } k \geq 0. \end{aligned}$$

We used the exponential hyperparameter setting recommended in the original work with the scale-factor $a = 0.1$ fixed, $\beta^{(t)} = 1 - a^{t-1}$, for $t = 1, \ldots, T$ and choosing T in $\{2, 3, 4\}$. We found that $T = 2$ gave the best performance in this experiment. As shown in Figure 8 & Table 6, with the help of passive damping, AggMo is more stable and robust compared with CM-SGD.

Quasi-hyperbolic Momentum (QHM) Ma & Yarats (2019) introduce the immediate discount factor $\nu \in \mathbb{R}$ for the momentum scheme, which results in the QHM update rules ($\alpha \in \mathbb{R}, v^{qh} \in \mathbb{R}^d, v_0^{qh} = \mathbf{0}$):

$$\begin{aligned} v_{k+1}^{qh} &= \beta \cdot v_k^{qh} + (1 - \beta) \cdot \nabla f_{i_k}(x_k), \\ x_{k+1} &= x_k - \alpha \cdot (\nu \cdot v_{k+1}^{qh} + (1 - \nu) \cdot \nabla f_{i_k}(x_k)), \text{ for } k \geq 0. \end{aligned}$$

Here we used the recommended hyperparameter setting for QHM ($\alpha_0 = 1.0, \beta = 0.999, \nu = 0.7$).

Figure 8 shows that AM1-SGD, AggMo and QHM achieve faster convergence in the early stage while CM-SGD has the highest peak accuracy. In terms of robustness, huge gaps are observed when comparing AM1-SGD with the remaining methods in Table 6. Note that AM1-SGD is more efficient than both QHM and AggMo, and is as efficient as CM-SGD.

We also plot the convergence of train-batch loss for all the methods in Figure 9. Despite of showing worse generalization performance, both QHM and AggMo perform better on reducing the train-batch loss in this experiment, which is consistent with the results reported in Ma & Yarats (2019); Lucas et al. (2019).

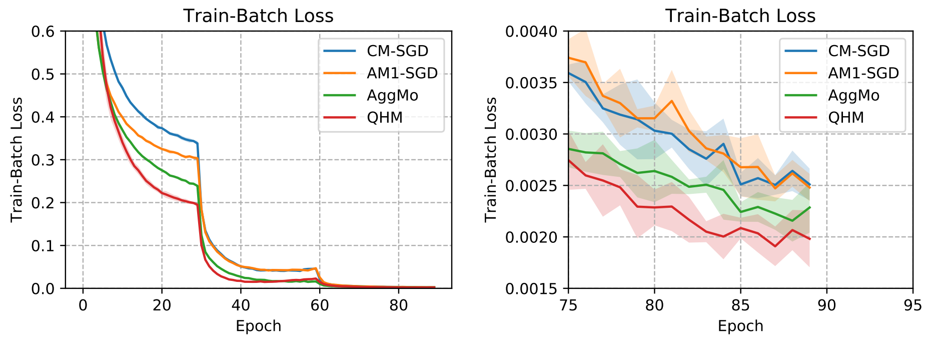
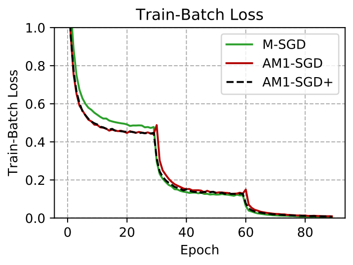
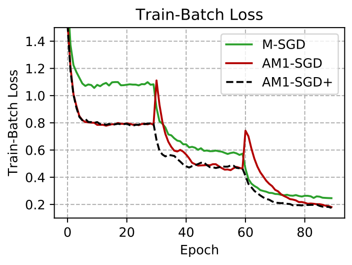


Figure 9: Train-batch loss results. Best viewed in color.

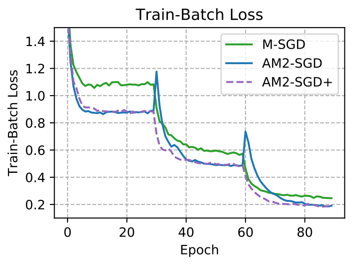
A.4 Issues with learning rate schedulers



(a) $\beta = 0.95$



(b) $\beta = 0.995$



(c) $\beta = 0.995$

Figure 10: ResNet18 on CIFAR-10. $\eta_0 = 0.1, \beta \in \{0.95, 0.995\}$. '+' represents performing a restart after each learning rate reduction.

We show in Figure 10 that when β is large for the task, using step learning rate scheduler with decay factor 10, a performance drop is observed after each reduction. Both Option I and Option II have this issue and the curves are basically identical. Here we only use Option II. We fix this issue by performing a restart after each learning rate reduction (labeled with '+'). We plot the train-batch loss here because we find the phenomenon is clearer in this way. If $\beta = 0.9$, there is no observable performance drop in this experiment.

For smooth-changing schedulers such as the cosine annealing scheduler (Loshchilov & Hutter, 2016), the amortized momentum works well as shown in Figure 11.

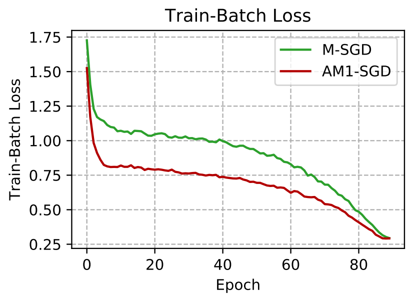
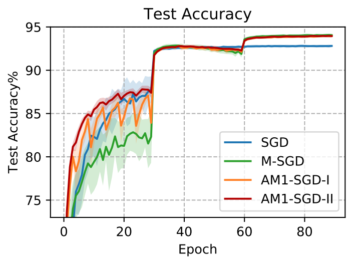


Figure 11: ResNet18 on CIFAR-10. Cosine annealing scheduler (without restarts), $\eta_0 = 0.1, \beta = 0.995$.

A.5 Test accuracy results of Figure 4 & Table 2

We report the test accuracy results of the experiments in Section 4 in Figure 12 & Table 7. These results are reminiscent of the ResNet34 experiments (Figure 3 & Table 1).



METHOD	PEAK ACCURACY	Avg. STD
SGD	92.812% ± 0.181%	1.005%
M-SGD	94.072% ± 0.166%	1.102%
AM1-SGD-I	93.968% ± 0.154%	**0.543%**
AM1-SGD-II	93.976% ± 0.181%	**0.315%**

Figure 12 & Table 7: ResNet18 with pre-activation on CIFAR-10. For all methods, $\eta_0 = 0.1, \beta = 0.9$, run 20 seeds. For AM1-SGD, $m = 5$ and its labels are formatted as 'AM1-SGD-{*Option*}'. Shaded bands indicate ± 1 standard deviation. Best viewed in color.

A.6 CIFAR-100 experiment

We report the results of training DenseNet121 (Huang et al., 2017) on CIFAR-100 in Figure 13, which shows that both AM1-SGD and AM2-SGD perform well before the final learning rate reduction. However, the peak accuracies are lowered around 0.6% compared with M-SGD. We also notice that SGD reduces the train-batch loss at an incredibly fast rate and the losses it reaches are consistently lower than other methods in the entire 300 epochs. However, this performance is not reflected in the convergence of test accuracy. We believe that this phenomenon suggests that the DenseNet model is actually "overfitting" M-SGD (since in the ResNet experiments, M-SGD always achieves a lower train loss than SGD after the final learning rate reduction).

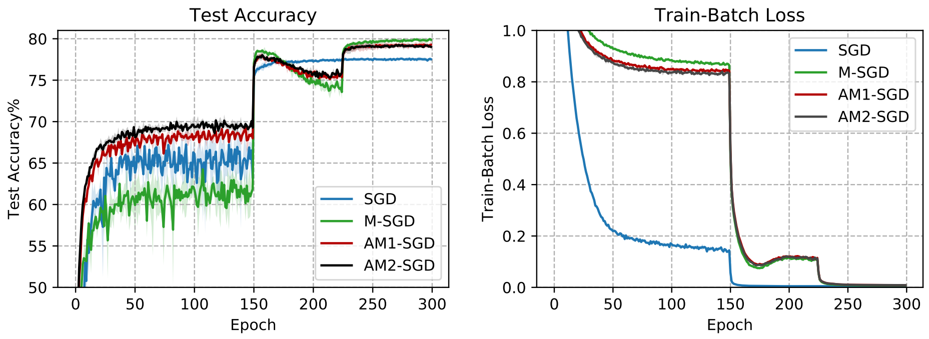


Figure 13: DenseNet121 on CIFAR-100. For all methods, $\eta_0 = 0.1, \beta = 0.9$, run 3 seeds. AM1-SGD and AM2-SGD use Option II and $m = 5$. Shaded bands indicating ± 1 standard deviation. Best viewed in color.

A.7 A SANITY CHECK

When $m = 1$, both AM1-SGD and AM2-SGD are equivalent to M-SGD, we plot their convergence in Figure 14 as a sanity check (the detailed data is given in Table 4).

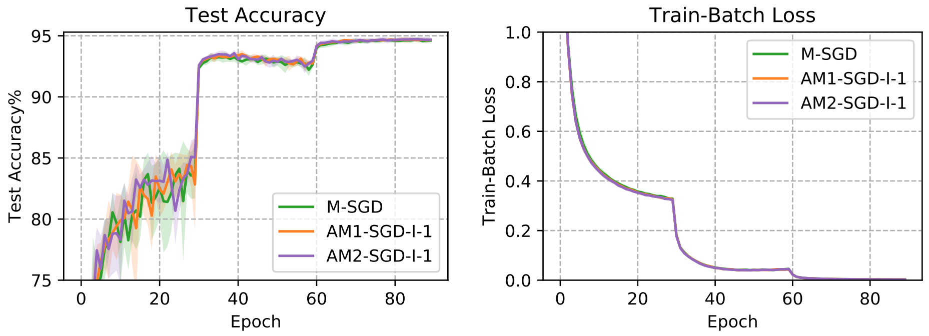


Figure 14: A sanity check. Labels are formatted as 'AM{1/2}-SGD-{*Option*}-{m}'.

We observed that when $m = 1$, both AM1-SGD and AM2-SGD have a lower STD error than M-SGD. We believe that it is because they both maintain the iterates without scaling, which is numerically more stable than M-SGD (M-SGD in standard PyTorch maintains a scaled buffer, i.e., $v_k^{pt} = \eta^{-1}\beta^{-1} \cdot (y_k - x_k)$).

B MISSING PARTS IN SECTION 4

B.1 THE REFORMULATIONS

When $h \equiv 0$ and β is a constant, we do the reformulations by eliminating the sequence $\{z_k\}$.

For the reformulated AM2-SGD,

$$\begin{aligned} x_k^{j_k} &= (1-\beta)\cdot z_k + \beta\cdot\phi_{j_k}^k, \\ z_{k+1} &= z_k - \alpha\cdot\nabla f_{i_k}(x_k^{j_k}), \\ \phi_{j_k}^{k+1} &= (1-\beta)\cdot z_{k+1} + \beta\cdot\phi_{j_k}^k, \\ \big(x_{k+1}^{j_{k+1}} &= (1-\beta)\cdot z_{k+1} + \beta\cdot\phi_{j_{k+1}}^{k+1}\big). \end{aligned} \quad \xrightarrow[\text{Eliminate } \{z_k\}]{\alpha(1-\beta)=\eta} \quad \begin{aligned} \phi_{j_k}^{k+1} &= x_k^{j_k} - \eta\cdot\nabla f_{i_k}(x_k^{j_k}), \\ x_{k+1}^{j_{k+1}} &= \phi_{j_k}^{k+1} + \beta\cdot\left(\phi_{j_{k+1}}^{k+1} - \phi_{j_k}^k\right). \end{aligned}$$

The reformulated AM2-SGD **Algorithm 2**

For the reformulated AM1-SGD, when $h \equiv 0$, the inner loops are basically SGD,

$$\begin{aligned} x_k &= (1-\beta)\cdot z_k + \beta\cdot\tilde{x}_s, \\ z_{k+1} &= z_k - \alpha\cdot\nabla f_{i_k}(x_k), \\ (x_{k+1} &= (1-\beta)\cdot z_{k+1} + \beta\cdot\tilde{x}_s.) \end{aligned} \quad \xrightarrow[\text{Eliminate } \{z_k\}]{\alpha(1-\beta)=\eta} \quad x_{k+1} = x_k - \eta\cdot\nabla f_{i_k}(x_k).$$

At the end of each inner loop (i.e., when $(k+1) \mod m = 0$), we have

$$x_{(s+1)m} = (1-\beta)\cdot z_{(s+1)m} + \beta\cdot\tilde{x}_s,$$

while at the beginning of the next inner loop,

$$x_{(s+1)m} = (1-\beta)\cdot z_{(s+1)m} + \beta\cdot\tilde{x}_{s+1},$$

which means that we need to set $x_{k+1} \leftarrow x_{k+1} + \beta\cdot(\tilde{x}_{s+1} - \tilde{x}_s)$.

We also give the reformulation of M-SGD (scheme (1)) to the Auslender & Teboulle (2006) scheme for reference:

$$\begin{aligned} x_k &= (1-\beta)\cdot z_k + \beta\cdot y_k, \\ z_{k+1} &= z_k - \alpha\cdot\nabla f_{i_k}(x_k), \\ y_{k+1} &= (1-\beta)\cdot z_{k+1} + \beta\cdot y_k, \\ \big(x_{k+1} &= (1-\beta)\cdot z_{k+1} + \beta\cdot y_{k+1}\big). \end{aligned} \quad \xrightarrow[\text{Eliminate } \{z_k\}]{\alpha(1-\beta)=\eta} \quad \begin{aligned} y_{k+1} &= x_k - \eta\cdot\nabla f_{i_k}(x_k), \\ x_{k+1} &= y_{k+1} + \beta\cdot(y_{k+1} - y_k). \end{aligned}$$

Auslender & Teboulle (2006)
(AC-SA (Lan, 2012)) **Nesterov (1983; 2013)**

AC-SA (in the Euclidean case) maps to the Auslender & Teboulle (2006) scheme through (in the original notations)

$$\begin{cases} x = x^{md} \\ z = x \\ y = x^{ag} \\ 1-\beta = \beta_t^{-1} \\ \alpha = \gamma_t \end{cases}.$$

Thus, $\eta = \alpha(1-\beta) = \beta_t^{-1}\gamma_t = O(1/L)$ (cf. Corollary 1 in Lan (2012)). Intuition for the Auslender & Teboulle (2006) scheme can be found in Remark 2 in Lan (2012).

B.2 Proof of Lemma 1

This Lemma is provided in a similar way as in Lan (2012); Ghadimi & Lan (2012) under a more general setting that allows non-Euclidean norms in the assumptions, we give a proof here for completeness.

The scheme (5) is copied here for reference:

$$\begin{aligned}x &= (1-\beta)\cdot z + \beta\cdot y.\\ z^+ &= \operatorname{prox}_{\alpha h}\{z - \alpha\cdot\nabla f_i(x)\}.\\ y^+ &= (1-\beta)\cdot z^+ + \beta\cdot y.\end{aligned}$$

Based on the convexity (Assumption (a)), we have

$$\begin{aligned}f(x) - f(x^\star) \le\ & \underbrace{\left\langle \nabla f(x), x - z\right\rangle}_{R_0} + \underbrace{\left\langle \nabla f(x) - \nabla f_i(x), z - x^\star\right\rangle}_{R_1} + \underbrace{\left\langle \nabla f_i(x), z - z^+\right\rangle}_{R_2}\\ &+ \underbrace{\left\langle \nabla f_i(x), z^+ - x^\star\right\rangle}_{R_3}.\end{aligned} \tag{7}$$

We upper bound the terms on the right side one-by-one.

For R_0,

$$R_0 \overset{(\star)}{=} \frac{\beta}{1-\beta}\left\langle \nabla f(x), y - x\right\rangle \le \frac{\beta}{1-\beta}\big(f(y) - f(x)\big), \tag{8}$$

where $(\star)$ uses the relation between x and z, i.e., $(1-\beta)\cdot(x-z) = \beta\cdot(y-x)$.

For R_2, based on Assumption (a), we have

$$f(y^+) - f(x) + \left\langle \nabla f(x), x - y^+\right\rangle \le \frac{L}{2}\left\|x - y^+\right\|^2 + M\left\|x - y^+\right\|.$$

Then, noting that $x - y^+ = (1-\beta)\cdot(z - z^+)$, we can arrange the above inequality as

$$\begin{aligned}R_2 \le\ & \frac{L(1-\beta)}{2}\left\|z - z^+\right\|^2 + \frac{1}{1-\beta}\big(f(x) - f(y^+)\big) + \left\langle \nabla f(x) - \nabla f_i(x), z^+ - z\right\rangle\\ &+ M\left\|z - z^+\right\|\\ \le\ & \frac{L(1-\beta)}{2}\left\|z - z^+\right\|^2 + \frac{1}{1-\beta}\big(f(x) - f(y^+)\big) + \big(\left\|\nabla f(x) - \nabla f_i(x)\right\| + M\big)\left\|z - z^+\right\|.\end{aligned}$$

Using Young's inequality with $\zeta > 0$, we obtain

$$R_2 \le \frac{L(1-\beta)+\zeta}{2}\left\|z - z^+\right\|^2 + \frac{1}{1-\beta}\big(f(x) - f(y^+)\big) + \frac{(\|\nabla f(x) - \nabla f_i(x)\| + M)^2}{2\zeta}. \tag{9}$$

For R_3, based on the optimality condition of $\operatorname{prox}_{\alpha h}\{z - \alpha\cdot\nabla f_i(x)\}$ and denoting $\partial h(z^+)$ as a subgradient of h at z^+, we have for any $u \in X$,

$$\begin{aligned}\left\langle \alpha\cdot\partial h(z^+) + z^+ - z + \alpha\cdot\nabla f_i(x), u - z^+\right\rangle &\ge 0,\\ \left\langle \nabla f_i(x), z^+ - u\right\rangle &\le \left\langle \partial h(z^+), u - z^+\right\rangle + \frac{1}{\alpha}\left\langle z^+ - z, u - z^+\right\rangle\\ &\le h(u) - h(z^+) + \frac{1}{\alpha}\left\langle z^+ - z, u - z^+\right\rangle.\end{aligned}$$

Choosing $u = x^\star$,

$$\begin{aligned}R_3 &\le h(x^\star) - h(z^+) + \frac{1}{\alpha}\left\langle z^+ - z, x^\star - z^+\right\rangle\\ &\overset{(\star)}{=} h(x^\star) - h(z^+) + \frac{1}{2\alpha}\left(\|z - x^\star\|^2 - \left\|z^+ - x^\star\right\|^2 - \left\|z^+ - z\right\|^2\right),\end{aligned} \tag{10}$$

where $(\star)$ follows from $\|a+b\|^2 = \|a\|^2 + \|b\|^2 + 2\langle a, b\rangle$.

Finally, by upper bounding (7) using (8), (9), (10), we conclude that

$$\begin{aligned} f(x) - f(x^\star) \le R_1 &+ \frac{\beta}{1-\beta}\big(f(y) - f(x)\big) + \frac{L(1-\beta) + \zeta - \alpha^{-1}}{2}\left\|z - z^+\right\|^2 \\ &+ \frac{1}{1-\beta}\big(f(x) - f(y^+)\big) + h(x^\star) - h(z^+) + \frac{(\|\nabla f(x) - \nabla f_i(x)\| + M)^2}{2\zeta} \\ &+ \frac{1}{2\alpha}\left(\|z - x^\star\|^2 - \left\|z^+ - x^\star\right\|^2\right), \end{aligned}$$

After simplification,

$$\begin{aligned} \frac{1}{1-\beta}\big(f(y^+) - f(x^\star)\big) \le \frac{\beta}{1-\beta}\big(f(y) - f(x^\star)\big) &+ \frac{L(1-\beta) + \zeta - \alpha^{-1}}{2}\left\|z - z^+\right\|^2 \\ &+ h(x^\star) - h(z^+) + \frac{(\|\nabla f(x) - \nabla f_i(x)\| + M)^2}{2\zeta} + R_1 \\ &+ \frac{1}{2\alpha}\left(\|z - x^\star\|^2 - \left\|z^+ - x^\star\right\|^2\right). \end{aligned} \tag{11}$$

Note that with the convexity of h and $y^+ = (1-\beta)\cdot z^+ + \beta \cdot y$, we have

$$\begin{aligned} h(y^+) &\le (1-\beta)h(z^+) + \beta h(y), \\ h(z^+) &\ge \frac{1}{1-\beta}h(y^+) - \frac{\beta}{1-\beta}h(y). \end{aligned}$$

Using the above inequality and choosing $\zeta = \alpha^{-1} - L(1-\beta) > 0 \Rightarrow \alpha(1-\beta) < 1/L$, we can arrange (11) as

$$\begin{aligned} \frac{1}{1-\beta}\big(F(y^+) - F(x^\star)\big) \le \frac{\beta}{1-\beta}\big(F(y) - F(x^\star)\big) &+ \frac{1}{2\alpha}\left(\|z - x^\star\|^2 - \left\|z^+ - x^\star\right\|^2\right) \\ &+ \frac{(\|\nabla f(x) - \nabla f_i(x)\| + M)^2}{2(\alpha^{-1} - L(1-\beta))} + R_1. \end{aligned}$$

B.3 Proof of Theorem 1a

Using Assumption (c), Lemma 1 with

$$\begin{cases} x = x_k \\ z = z_k \\ z^+ = z_{k+1} \\ y = \tilde{x}_s \\ y^+ = x_{k+1} \\ \alpha = \alpha_s \\ \beta = \beta_s \end{cases}, \tag{12}$$

and taking expectation, if $\alpha_s(1-\beta_s) < 1/L$, we have

$$\begin{aligned} &\frac{1}{1-\beta_s}\big(\mathbb{E}_{i_k}\left[F(x_{k+1})\right] - F(x^\star)\big) + \frac{1}{2\alpha_s}\mathbb{E}_{i_k}\left[\left\|z_{k+1} - x^\star\right\|^2\right] \\ &\le \frac{\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{2\alpha_s}\|z_k - x^\star\|^2 + \frac{(\sigma + M)^2}{2(\alpha_s^{-1} - L(1-\beta_s))}. \end{aligned}$$

Summing the above inequality from $k = sm, \ldots, sm+m-1$, we obtain

$$\begin{aligned} &\frac{1}{(1-\beta_s)m}\sum_{j=1}^{m}\big(\mathbb{E}\left[F(x_{sm+j})\right] - F(x^\star)\big) + \frac{1}{2\alpha_s m}\mathbb{E}\left[\left\|z_{(s+1)m} - x^\star\right\|^2\right] \\ &\le \frac{\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{2\alpha_s m}\|z_{sm} - x^\star\|^2 + \frac{(\sigma + M)^2}{2(\alpha_s^{-1} - L(1-\beta_s))}, \end{aligned}$$

Using the definition of $\tilde{x}_{s+1}$ and convexity,

$$\begin{aligned}
&\frac{\alpha_s}{1-\beta_s}\big(\mathbb{E}\left[F(\tilde{x}_{s+1})\right] - F(x^\star)\big) + \frac{1}{2m}\mathbb{E}\left[\left\|z_{(s+1)m} - x^\star\right\|^2\right] \\
&\le \frac{\alpha_s\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{2m}\left\|z_{sm} - x^\star\right\|^2 + \frac{\alpha_s(\sigma^2+M^2)}{\alpha_s^{-1} - L(1-\beta_s)}.
\end{aligned} \tag{13}$$

It can be verified that with the choices $\beta_s = \frac{s}{s+2}$ and $\alpha_s = \frac{\lambda_1}{L(1-\beta_s)}$, the following holds for $s \ge 0$,

$$\frac{\alpha_{s+1}\beta_{s+1}}{1-\beta_{s+1}} \le \frac{\alpha_s}{1-\beta_s} \text{ and } \beta_0 = 0. \tag{14}$$

Thus, by telescoping (13) from $s = S-1, \ldots, 0$, we obtain

$$\begin{aligned}
&\frac{\alpha_{S-1}}{1-\beta_{S-1}}\big(\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star)\big) + \frac{1}{2m}\mathbb{E}\left[\left\|z_{Sm} - x^\star\right\|^2\right] \\
&\le \frac{1}{2m}\left\|x_0 - x^\star\right\|^2 + \sum_{s=0}^{S-1}\frac{\alpha_s(\sigma^2+M^2)}{\alpha_s^{-1} - L(1-\beta_s)},
\end{aligned}$$

and thus,

$$\begin{aligned}
\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) &\le \frac{2L}{\lambda_1 m(S+1)^2}\left\|x_0 - x^\star\right\|^2 + \frac{4L(\sigma^2+M^2)}{\lambda_1(S+1)^2}\sum_{s=0}^{S-1}\frac{\alpha_s^2}{1-\alpha_s(1-\beta_s)L} \\
&\overset{(a)}{\le} \frac{2L}{\lambda_1 m(S+1)^2}\left\|x_0 - x^\star\right\|^2 + \frac{3\lambda_1(\sigma^2+M^2)}{L(S+1)^2}\sum_{s=0}^{S-1}(s+2)^2 \\
&\overset{(b)}{\le} \frac{2L}{\lambda_1 m(S+1)^2}\left\|x_0 - x^\star\right\|^2 + \frac{8\lambda_1(\sigma^2+M^2)(S+1)}{L},
\end{aligned}$$

where (a) follows from $\lambda_1 \le \frac{2}{3}$ and (b) holds because $0 \le x \mapsto (x+2)^2$ is non-decreasing and thus

$$\sum_{s=0}^{S-1}(s+2)^2 \le \int_0^S (x+2)^2 dx \le \frac{(S+2)^3}{3} \le \frac{8(S+1)^3}{3}.$$

Denoting

$$\lambda_1^\star \triangleq \frac{L\left\|x_0 - x^\star\right\|}{2\sqrt{m}\sqrt{\sigma^2+M^2}(S+1)^{\frac{3}{2}}},$$

and based on the choice of $\lambda_1 = \min\left\{\frac{2}{3}, \lambda_1^*\right\}$, if $\lambda_1^* \le \frac{2}{3}$, we have

$$\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) \le \frac{8\left\|x_0 - x^\star\right\|\sqrt{\sigma^2+M^2}}{m^{\frac{1}{2}}(S+1)^{\frac{1}{2}}}.$$

If $\lambda_1^* > \frac{2}{3}$,

$$\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) \le \frac{3L\left\|x_0 - x^\star\right\|^2}{m(S+1)^2} + \frac{4\left\|x_0 - x^\star\right\|\sqrt{\sigma^2+M^2}}{m^{\frac{1}{2}}(S+1)^{\frac{1}{2}}}.$$

Thus, we conclude that

$$\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) \le \frac{3L\left\|x_0 - x^\star\right\|^2}{m(S+1)^2} + \frac{8\left\|x_0 - x^\star\right\|\sqrt{\sigma^2+M^2}}{m^{\frac{1}{2}}(S+1)^{\frac{1}{2}}}.$$

Substituting $S = K/m$ completes the proof.

B.4 Proof of Theorem 1b

In order to prove Theorem 1b, we need the following known result for the martingale difference (cf. Lemma 2 in Lan et al. (2012)):

Lemma 2. *With $N > 0$, let $\xi_0, \xi_1, \ldots, \xi_{N-1}$ be a sequence of i.i.d. random variables, for $t = 0, \ldots, N-1$, $\sigma_t > 0$ be a deterministic number and $\psi_t = \psi_t(\xi_0, \ldots, \xi_t)$ be a deterministic measurable function such that $\mathbb{E}_{\xi_t}[\psi_t] = 0$ a.s. and $\mathbb{E}_{\xi_t}\left[\exp\{\psi_t^2/\sigma_t^2\}\right] \le \exp\{1\}$ a.s.. Then for any $\Lambda \ge 0$,*

$$\text{Prob}\left\{\sum_{t=0}^{N-1} \psi_t \ge \Lambda \sqrt{\sum_{t=0}^{N-1} \sigma_t^2}\right\} \le \exp\{-\Lambda^2/3\}.$$

To start with, using Lemma 1 with the parameter mapping (12), we have

$$\begin{aligned}
&\frac{1}{1-\beta_s}\big(F(x_{k+1}) - F(x^\star)\big) + \frac{1}{2\alpha_s}\|z_{k+1} - x^\star\|^2 \\
&\le \frac{\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{2\alpha_s}\|z_k - x^\star\|^2 \\
&\quad + \frac{(\|\nabla f(x_k) - \nabla f_{i_k}(x_k)\| + M)^2}{2(\alpha_s^{-1} - L(1-\beta_s))} + \langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star \rangle \\
&\le \frac{\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{2\alpha_s}\|z_k - x^\star\|^2 + \frac{M^2}{\alpha_s^{-1} - L(1-\beta_s)} \\
&\quad + \frac{\|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|^2}{\alpha_s^{-1} - L(1-\beta_s)} + \langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star \rangle.
\end{aligned}$$

Summing the above inequality from $k = sm, \ldots, sm+m-1$ and using the choice $\alpha_s = \frac{\lambda_1}{L(1-\beta_s)}$ with $\lambda_1 \le \frac{2}{3}$, we obtain

$$\begin{aligned}
&\frac{\alpha_s}{1-\beta_s}\big(F(\tilde{x}_{s+1}) - F(x^\star)\big) + \frac{1}{2m}\left\|z_{(s+1)m} - x^\star\right\|^2 \\
&\le \frac{\alpha_s\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{2m}\|z_{sm} - x^\star\|^2 + 3\alpha_s^2 M^2 \\
&\quad + \frac{3\alpha_s^2}{m}\sum_{k=sm}^{sm+m-1}\|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|^2 + \frac{\alpha_s}{m}\sum_{k=sm}^{sm+m-1}\langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star\rangle.
\end{aligned}$$

With our parameter choices, the relations in (14) hold and thus we can telescope the above inequality from $s = S-1, \ldots, 0$,

$$\begin{aligned}
\frac{\alpha_{S-1}}{1-\beta_{S-1}}\big(F(\tilde{x}_S) - F(x^\star)\big) \le \frac{1}{2m}\|x_0 - x^\star\|^2 &+ 3M^2\sum_{s=0}^{S-1}\alpha_s^2 \\
&+ \underbrace{\frac{3}{m}\sum_{k=0}^{K-1}\alpha_{\lfloor k/m\rfloor}^2\|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|^2}_{R_4} \\
&+ \underbrace{\frac{1}{m}\sum_{k=0}^{K-1}\alpha_{\lfloor k/m\rfloor}\langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star\rangle}_{R_5}.
\end{aligned} \tag{15}$$

Denoting $\mathcal{V}_k^2 \triangleq \|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|^2$, $\bar{\alpha} = \sum_{k=0}^{K-1}\alpha_{\lfloor k/m\rfloor}^2 = m\sum_{s=0}^{S-1}\alpha_s^2$, for R_4, by Jensen's inequality, we have

$$\mathbb{E}\left[\exp\left\{\frac{1}{\bar{\alpha}}\sum_{k=0}^{K-1}\alpha_{\lfloor k/m\rfloor}^2\mathcal{V}_k^2/\sigma^2\right\}\right] \le \frac{1}{\bar{\alpha}}\sum_{k=0}^{K-1}\alpha_{\lfloor k/m\rfloor}^2\mathbb{E}\left[\exp\left\{\mathcal{V}_k^2/\sigma^2\right\}\right] \overset{(\star)}{\le} \exp\{1\},$$

where $(\star)$ uses the additional assumption $\mathbb{E}_{i_k}\left[\exp\left\{\mathcal{V}_k^2/\sigma^2\right\}\right] \le \exp\{1\}$.

Then, based on Markov's inequality, we have for any $\Lambda \ge 0$,

$$\operatorname{Prob}\left\{\exp\left\{\frac{1}{\bar{\alpha}}\sum_{k=0}^{K-1}\alpha_{\lfloor k/m\rfloor}^2\mathcal{V}_k^2/\sigma^2\right\} \ge \exp\{\Lambda+1\}\right\} \le \exp\{-\Lambda\},$$
$$\operatorname{Prob}\left\{R_4 \ge (\Lambda+1)\sigma^2 m\sum_{s=0}^{S-1}\alpha_s^2\right\} \le \exp\{-\Lambda\}. \tag{16}$$

For R_5, since we have $\mathbb{E}_{i_k}\left[\alpha_{\lfloor k/m\rfloor}\left\langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star\right\rangle\right] = 0$ and

$$\mathbb{E}_{i_k}\left[\exp\left\{\frac{\alpha_{\lfloor k/m\rfloor}^2\left\langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star\right\rangle^2}{\alpha_{\lfloor k/m\rfloor}^2\sigma^2 D_X^2}\right\}\right] \le \mathbb{E}_{i_k}\left[\exp\left\{\mathcal{V}_k^2/\sigma^2\right\}\right] \le \exp\{1\},$$

which is based on the "light tail" assumption, using Lemma 2, we obtain

$$\operatorname{Prob}\left\{R_5 \ge \Lambda\sigma D_X\sqrt{m\sum_{s=0}^{S-1}\alpha_s^2}\right\} \le \exp\{-\Lambda^2/3\}. \tag{17}$$

Combining (15), (16) and (17), based on the parameter setting (cf. (6)) and using the notation

$$\mathcal{K}_0(m) \triangleq \frac{3Lm\left\|x_0 - x^\star\right\|^2}{(K+m)^2} + \frac{8\left\|x_0 - x^\star\right\|\sqrt{\sigma^2+M^2}}{\sqrt{K+m}},$$
$$R_6 \triangleq \frac{12L\sigma^2}{\lambda_1(S+1)^2}\sum_{s=0}^{S-1}\alpha_s^2 + \frac{4L\sigma D_X}{\lambda_1(S+1)^2\sqrt{m}}\sqrt{\sum_{s=0}^{S-1}\alpha_s^2},$$

we conclude that

$$\operatorname{Prob}\left\{F(\tilde{x}_S) - F(x^\star) \le \mathcal{K}_0(m) + \Lambda R_6\right\} \ge 1 - \left(\exp\{-\Lambda^2/3\} + \exp\{-\Lambda\}\right).$$

For R_6, using the choice of α_s and λ_1, we obtain

$$\begin{aligned} R_6 \le \frac{4\sqrt{6}\sigma D_X}{3\sqrt{K+m}} + \frac{8\lambda_1\sigma^2(S+1)}{L} &\le \frac{4\sqrt{6}\sigma D_X}{3\sqrt{K+m}} + \frac{4\sigma^2\left\|x_0 - x^\star\right\|}{\sqrt{K+m}\sqrt{\sigma^2+M^2}} \\ &\le \frac{4\sigma\left(3\left\|x_0 - x^\star\right\| + \sqrt{6}D_X\right)}{3\sqrt{K+m}}, \end{aligned}$$

which completes the proof.

B.5 Deriving AM1-SGD from Katyusha

Katyusha has the following scheme (non-proximal, in the original notations, σ is the strong convexity parameter, cf. Algorithm 1 with Option I in Allen-Zhu (2018))[11]:

```
**Initialize:** $\widetilde{x}^0 = y_0 = z_0 = x_0, \eta = \frac{1}{3L}, \omega = 1 + \alpha\sigma$.
1: **for** $s = 0, \ldots, S-1$ **do**
2:     Compute and store $\nabla f(\widetilde{x}^s)$.
3:     **for** $j = 0, \ldots, m-1$ **do**
4:         $k = sm + j$.
5:         $x_k = \tau_1 \cdot z_k + \tau_2 \cdot \widetilde{x}^s + (1 - \tau_1 - \tau_2) \cdot y_k$.
6:         $\widetilde{\nabla}_k = \nabla f_{i_k}(x_k) - \nabla f_{i_k}(\widetilde{x}^s) + \nabla f(\widetilde{x}^s)$.
7:         $z_{k+1} = z_k - \alpha \cdot \widetilde{\nabla}_k$.
8:         $y_{k+1} = x_k - \eta \cdot \widetilde{\nabla}_k$.
9:     **end for**
10:    $\widetilde{x}^{s+1} = \left(\sum_{j=0}^{m-1} \omega^j\right)^{-1} \cdot \sum_{j=0}^{m-1} \omega^j \cdot y_{sm+j+1}$.
11: **end for**
**Output:** $\widetilde{x}^S$.
```

We can eliminate the sequence $\{z_k\}$ in this scheme. Note that in the parameter setting of Katyusha, we have $\eta = \alpha\tau_1$, and thus

$$\begin{aligned}
x_{k+1} &= \tau_1 \cdot z_{k+1} + \tau_2 \cdot \widetilde{x}^s + (1 - \tau_1 - \tau_2) \cdot y_{k+1} \\
&= \tau_1 \cdot z_k - \eta \cdot \widetilde{\nabla}_k + \tau_2 \cdot \widetilde{x}^s + (1 - \tau_1 - \tau_2) \cdot y_k + (1 - \tau_1 - \tau_2) \cdot (y_{k+1} - y_k) \\
&= x_k - \eta \cdot \widetilde{\nabla}_k + (1 - \tau_1 - \tau_2) \cdot (y_{k+1} - y_k) \\
&= y_{k+1} + (1 - \tau_1 - \tau_2) \cdot (y_{k+1} - y_k).
\end{aligned}$$

Hence, the inner loops can be written as

$$\begin{aligned}
y_{k+1} &= x_k - \eta \cdot \widetilde{\nabla}_k, \\
x_{k+1} &= y_{k+1} + (1 - \tau_1 - \tau_2) \cdot (y_{k+1} - y_k),
\end{aligned}$$

which is the Nesterov's scheme (scheme (1)). At the end of each inner loop (when $k = sm + m - 1$),

$$x_{(s+1)m} = \tau_1 \cdot z_{(s+1)m} + \tau_2 \cdot \widetilde{x}^s + (1 - \tau_1 - \tau_2) \cdot y_{(s+1)m},$$

while at the beginning of the next inner loop,

$$x_{(s+1)m} = \tau_1 \cdot z_{(s+1)m} + \tau_2 \cdot \widetilde{x}^{s+1} + (1 - \tau_1 - \tau_2) \cdot y_{(s+1)m},$$

which means that we need to set $x_{(s+1)m} \leftarrow x_{(s+1)m} + \tau_2 \cdot (\widetilde{x}^{s+1} - \widetilde{x}^s)$.

Then, the following is an equivalent scheme of Katyusha:

```
**Initialize:** $\widetilde{x}^0 = y_0 = x_0, \eta = \frac{1}{3L}, \omega = 1 + \alpha\sigma$.
1: **for** $s = 0, \ldots, S-1$ **do**
2:     **for** $j = 0, \ldots, m-1$ **do**
3:         $k = sm + j$.
4:         $y_{k+1} = x_k - \eta \cdot \widetilde{\nabla}_k$.
5:         $x_{k+1} = y_{k+1} + (1 - \tau_1 - \tau_2) \cdot (y_{k+1} - y_k)$.
6:     **end for**
7:     $\widetilde{x}^{s+1} = \left(\sum_{j=0}^{m-1} \omega^j\right)^{-1} \cdot \sum_{j=0}^{m-1} \omega^j \cdot y_{sm+j+1}$.
8:     $x_{(s+1)m} \leftarrow x_{(s+1)m} + \tau_2 \cdot (\widetilde{x}^{s+1} - \widetilde{x}^s)$.
9: **end for**
**Output:** $\widetilde{x}_S$.
```

Now it is clear that the inner loops use Nesterov's momentum and the Katyusha momentum is injected for every m iterations. If we replace the SVRG estimator $\widetilde{\nabla}_k$ with $\nabla f_{i_k}(x_k)$, set $1 - \tau_1 -$

[11]We change the notation x_{k+1} to x_k.

$\tau_2 = 0$, which is to eliminate Nesterov's momentum, and use a uniform average for $\widetilde{x}^{s+1}$, the above scheme becomes exactly AM1-SGD (Algorithm 1).

If we only replace the SVRG estimator $\widetilde{\nabla}_k$, the scheme can be regarded as adding amortized momentum to M-SGD. This scheme requires tuning the ratio of Nesterov's momentum and amortized momentum. In our preliminary experiments, after suitable tuning, we observed some performance improvement. However, this scheme increases the complexity, which we do not consider it worthwhile.

A recent work (Zhou et al., 2018) shows that when $1 - \tau_1 - \tau_2 = 0$, which is to solely use Katyusha momentum, one can still derive optimal rates and the algorithm is greatly simplified. Their proposed algorithm (i.e., MiG) is structurally more similar to AM1-SGD.

B.6 Proof of Theorem 2

Using Assumption (c), Lemma 1 with

$$\begin{cases} x = x_k^{j_k} \\ z = z_k \\ z^+ = z_{k+1} \\ y = \phi_{j_k}^k \\ y^+ = \phi_{j_k}^{k+1} \\ \alpha = \alpha_k \\ \beta = \beta_k \end{cases},$$

and taking expectation, if $\alpha_k(1-\beta_k) < 1/L$, we have

$$\begin{aligned} &\frac{1}{1-\beta_k}\mathbb{E}_{i_k,j_k}\left[F(\phi_{j_k}^{k+1}) - F(x^\star)\right] + \frac{1}{2\alpha_k}\mathbb{E}_{i_k,j_k}\left[\|z_{k+1} - x^\star\|^2\right] \\ &\leq \frac{\beta_k}{1-\beta_k}\mathbb{E}_{j_k}\left[F(\phi_{j_k}^k) - F(x^\star)\right] + \frac{1}{2\alpha_k}\|z_k - x^\star\|^2 + \frac{(\sigma+M)^2}{2(\alpha_k^{-1} - L(1-\beta_k))}. \end{aligned} \tag{18}$$

Note that

$$\begin{aligned} &\mathbb{E}_{i_k,j_k}\left[F(\phi_{j_k}^{k+1}) - F(x^\star)\right] \\ &= \mathbb{E}_{i_k,j_k}\left[\sum_{j=1}^m \left(F(\phi_j^{k+1}) - F(x^\star)\right) - \sum_{j\neq j_k}^m \left(F(\phi_j^k) - F(x^\star)\right)\right] \\ &= \mathbb{E}_{i_k,j_k}\left[\sum_{j=1}^m \left(F(\phi_j^{k+1}) - F(x^\star)\right)\right] - \mathbb{E}_{j_k}\left[\sum_{j\neq j_k}^m \left(F(\phi_j^k) - F(x^\star)\right)\right]. \end{aligned}$$

Dividing both sides of (18) by m and then adding $\frac{1}{(1-\beta_k)m}\mathbb{E}_{j_k}\left[\sum_{j\neq j_k}^m \left(F(\phi_j^k) - F(x^\star)\right)\right]$ to both sides, we obtain

$$\begin{aligned} &\frac{1}{1-\beta_k}\mathbb{E}_{i_k,j_k}\left[\frac{1}{m}\sum_{j=1}^m F(\phi_j^{k+1}) - F(x^\star)\right] + \frac{1}{2\alpha_k m}\mathbb{E}_{i_k,j_k}\left[\|z_{k+1} - x^\star\|^2\right] \\ \leq\ & -\frac{1}{m}\mathbb{E}_{j_k}\left[F(\phi_{j_k}^k) - F(x^\star)\right] + \frac{1}{1-\beta_k}\left(\frac{1}{m}\sum_{j=1}^m F(\phi_j^k) - F(x^\star)\right) + \frac{1}{2\alpha_k m}\|z_k - x^\star\|^2 \\ &+ \frac{(\sigma+M)^2}{2m(\alpha_k^{-1} - L(1-\beta_k))} \\ =\ & \frac{1 - \frac{1-\beta_k}{m}}{1-\beta_k}\left(\frac{1}{m}\sum_{j=1}^m F(\phi_j^k) - F(x^\star)\right) + \frac{1}{2\alpha_k m}\|z_k - x^\star\|^2 + \frac{(\sigma+M)^2}{2m(\alpha_k^{-1} - L(1-\beta_k))}. \end{aligned} \tag{19}$$

It can be verified that with our parameters choice: $\beta_k = \frac{k/m}{k/m+2}$ and $\alpha_k = \frac{\lambda_2}{L(1-\beta_k)}$, the following holds for $k \geq 0$,

$$\alpha_{k+1}\frac{1-\frac{1-\beta_{k+1}}{m}}{1-\beta_{k+1}} \leq \frac{\alpha_k}{1-\beta_k} \text{ and } \beta_0 = 0.$$

Then, we can telescope (19) from $k = K-1, \dots, 0$, which results in

$$\begin{aligned}
&\frac{\alpha_{K-1}}{1-\beta_{K-1}}\mathbb{E}\left[\frac{1}{m}\sum_{j=1}^{m} F(\phi_j^K) - F(x^\star)\right] + \frac{1}{2m}\mathbb{E}\left[\|z_K - x^\star\|^2\right] \\
&\leq \frac{\lambda_2(m-1)}{Lm}\big(F(x_0) - F(x^\star)\big) + \frac{1}{2m}\|x_0 - x^\star\|^2 + \sum_{k=0}^{K-1}\frac{\alpha_k(\sigma+M)^2}{2m(\alpha_k^{-1} - L(1-\beta_k))}.
\end{aligned}$$

Using the definition of $\bar{\phi}^K$ and convexity, we obtain

$$\begin{aligned}
\mathbb{E}\left[F(\bar{\phi}^K) - F(x^\star)\right] &\leq \frac{1-\beta_{K-1}}{\alpha_{K-1}}\left(\frac{\lambda_2(m-1)}{Lm}\big(F(x_0) - F(x^\star)\big) + \frac{1}{2m}\|x_0 - x^\star\|^2\right) \\
&\quad + \frac{1-\beta_{K-1}}{\alpha_{K-1}}\sum_{k=0}^{K-1}\frac{\alpha_k(\sigma+M)^2}{2m(\alpha_k^{-1} - L(1-\beta_k))} \\
&\overset{(a)}{=} \frac{4(m-1)\big(F(x_0) - F(x^\star)\big)}{m\left(\frac{K-1}{m}+2\right)^2} + \frac{2L\|x_0 - x^\star\|^2}{\lambda_2 m\left(\frac{K-1}{m}+2\right)^2} \\
&\quad + \frac{3\lambda_2(\sigma+M)^2}{2Lm\left(\frac{K-1}{m}+2\right)^2}\sum_{k=0}^{K-1}\left(\frac{k}{m}+2\right)^2 \\
&\overset{(b)}{\leq} \frac{4(m-1)\big(F(x_0) - F(x^\star)\big)}{m\left(\frac{K-1}{m}+2\right)^2} + \frac{2L\|x_0 - x^\star\|^2}{\lambda_2 m\left(\frac{K-1}{m}+2\right)^2} \\
&\quad + \frac{4\lambda_2(\sigma+M)^2\left(\frac{K-1}{m}+2\right)}{L},
\end{aligned} \tag{20}$$

where (a) uses $\lambda_2 \leq \frac{2}{3}$, (b) follows from simple integration arguments and that $\frac{K}{m} + 2 \leq 2\left(\frac{K-1}{m}+2\right)$ since $K \geq 1, m \geq 1$.

Based on the choice of

$$\lambda_2 = \min\left\{\frac{2}{3}, \frac{L\|x_0 - x^\star\|}{\sqrt{2m}(\sigma+M)\left(\frac{K-1}{m}+2\right)^{\frac{3}{2}}}\right\},$$

(20) can be further upper bounded as

$$\mathbb{E}\left[F(\bar{\phi}^K) - F(x^\star)\right] \leq \frac{4(m-1)\big(F(x_0) - F(x^\star)\big)}{m\left(\frac{K-1}{m}+2\right)^2} + \frac{3L\|x_0 - x^\star\|^2}{m\left(\frac{K-1}{m}+2\right)^2} + \frac{4\sqrt{2}\|x_0 - x^\star\|(\sigma+M)}{m^{\frac{1}{2}}\left(\frac{K-1}{m}+2\right)^{\frac{1}{2}}}.$$

C MISCELLANIES

C.1 COMPARISON OF SGD AND M-SGD

Ma & Yarats (2019) normalize the momentum buffer of M-SGD, which results in the following formulation ($\alpha \in \mathbb{R}, v^{qh} \in \mathbb{R}^d, v_0^{qh} = \mathbf{0}$):

$$\begin{aligned} v_{k+1}^{qh} &= \beta \cdot v_k^{qh} + (1-\beta) \cdot \nabla f_{i_k}(x_k), \\ x_{k+1} &= x_k - \alpha \cdot (\beta \cdot v_{k+1}^{qh} + (1-\beta) \cdot \nabla f_{i_k}(x_k)), \text{ for } k \geq 0. \end{aligned}$$

This scheme is equivalent to the PyTorch formulation (scheme (3)) through $v_k^{pt} = (1-\beta)^{-1} \cdot v_k^{qh}$ and $\eta = \alpha(1-\beta)$.

Based on this formulation, α is understood as the effective learning rate (i.e., the vector it scales has the same cardinality as a gradient) and the experiments in Ma & Yarats (2019) were conducted with fixed $\alpha = 1$. Their results indicate that when using the same effective learning rate, M-SGD and SGD achieve similar performance and thus they suspect that the benefit of momentum basically comes from using sensible learning rates.

In the convex case, their results are explainable. For simplicity, we consider deterministic smooth convex optimization. In theory, to obtain the optimal convergence rate, the effective learning rate α is set to a very large $O(k/L)$, which can be derived from Theorem 1 or Theorem 2 by setting $\sigma = 0, M = 0, m = 1$ (then λ_1 or λ_2 is always $\frac{2}{3}$ since the other term is ∞). If we fix $\alpha = \frac{2}{3L}$ for both methods, GD has an $O(1/K)$ convergence rate (cf. Theorem 2.1.13 in Nesterov (2013)). For the Nesterov's method, if we use $\beta_k = \frac{k}{k+2}$, it has the convergence rate (applying Lemma 1):

$$\frac{1}{1-\beta_k}\big(F(y_{k+1}) - F(x^\star)\big) + \frac{3L}{4}\left\|z_{k+1} - x^\star\right\|^2 \leq \frac{\beta_k}{1-\beta_k}\big(F(y_k) - F(x^\star)\big) + \frac{3L}{4}\left\|z_k - x^\star\right\|^2,$$

$$F(y_K) - F(x^\star) \leq \frac{3L\left\|x_0 - x^\star\right\|^2}{2(K+1)}.$$

Thus, in this case, both GD and the Nesterov's method yield an $O(1/K)$ rate, and thus we expect them to have similar performance. This analysis suggests that the acceleration effect basically comes from choosing a large effective learning rate, which corresponds to the observations in Ma & Yarats (2019).

However, what is special about the Nesterov's method is that it finds a legal way to adopt a large α that breaks the $O(1/L)$ limitation. If GD uses the same large α, we would expect it to be unstable and potentially diverge. In this sense, Nesterov's momentum can be understood as a "stabilizer". In our basic case study (ResNet34 on CIFAR-10), if we align the effective learning rate and set $\gamma = 1.0$ for SGD, the peak accuracy is improved but the performance is highly unstable and not robust, which is 2.205% average STD of test accuracy over 5 runs. The significance of QHM (Ma & Yarats, 2019) is that with suitable tuning, it achieves much faster convergence without changing the effective learning rate. Our work focuses on the robustness and our intuition is rooted in convex analysis, which are basically why we set $\gamma = \eta$.

C.2 TRAINING EVALUATION

Due to the mechanism of back-propagation, evaluating train-batch loss basically incurs no overhead. It can be efficiently used to indicate the training progress. However, it can only be treated as a coarse approximation to the full-batch loss as shown in Figure 1c. If batch normalization (Ioffe & Szegedy, 2015) or dropout (Srivastava et al., 2014) is used in training, the model changes during the training phase, which makes train-batch loss less accurate. More importantly, train-batch loss is always observed to be statistically stable, which omits many important characteristics of an optimizer such as robustness, oscillations, etc. We include a comparison of train-batch loss and full-batch loss on training ResNet18 with pre-activation on CIFAR-10 (the experiment in Section 4) in Figure 15.

We also notice that for different optimizers, even if their convergences on train-batch loss are indistinguishable, their convergences on test accuracy can vary greatly. We show two examples in Figure 16, where the ImageNet experiment is from Section 5 and the CIFAR-10 experiment is from Table 4 with $m = 10$.

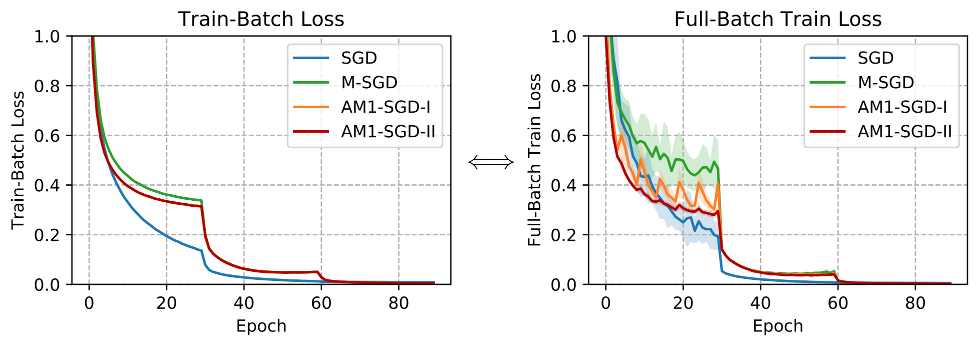


Figure 15: Train-batch loss vs. full-batch loss. Best viewed in color.

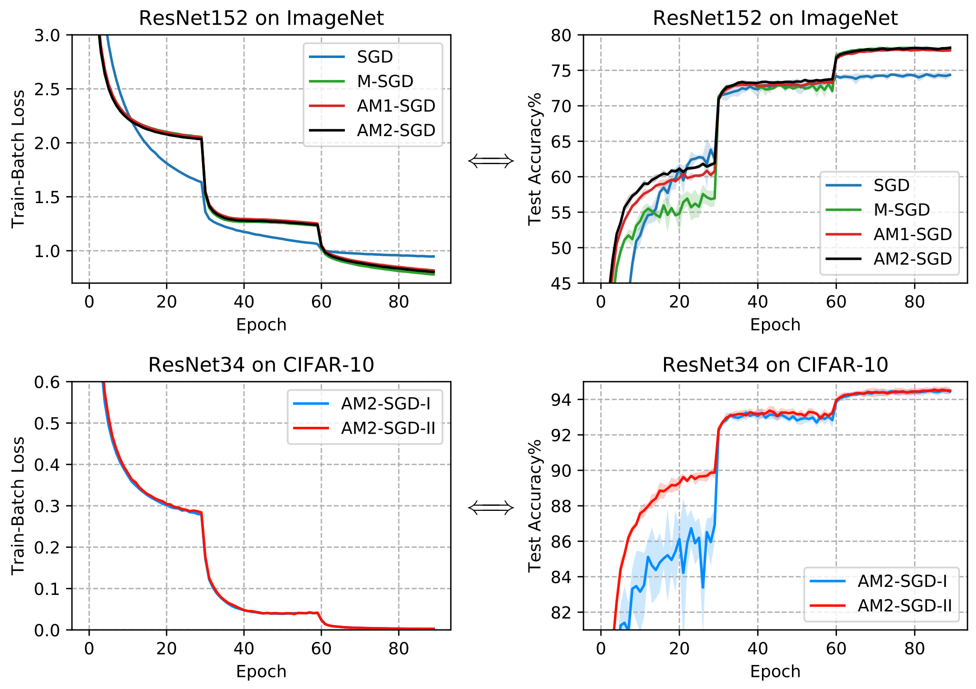


Figure 16: Train-batch loss vs. test accuracy. Best viewed in color.

D Experimental Setup

All of our experiments were conducted using PyTorch (Paszke et al., 2017) library.

D.1 Classification Setup

CIFAR-10 & CIFAR-100 Our implementation (e.g., ResNet and DenseNet implementations, data pre-processing) generally follows the repository `https://github.com/kuangliu/pytorch-cifar`. All the CIFAR experiments in this paper used a single GPU in a mix of RTX2080Ti, TITAN Xp and TITAN V. The batch size is fixed to 128. We used cross-entropy loss with 0.0005 weight decay and used batch normalization (Ioffe & Szegedy, 2015). Data augmentation includes random 32-pixel crops with a padding of 4-pixel and random horizontal flips with 0.5 probability. We used step (or multi-step) learning rate scheduler with a decay factor 10. For the CIFAR-10 experiments, we trained 90 epochs and decayed the learning rate every 30 epochs. For the CIFAR-100 experiments, we trained 300 epochs and decayed the learning rate at 150 epoch and 225 epoch following the settings in DenseNet (Huang et al., 2017).

ImageNet In the ImageNet experiments, we tried both ResNet50 and ResNet152 (He et al., 2016b). The training strategy is the same as the PyTorch's official repository `https://github.com/pytorch/examples/tree/master/imagenet`, which uses a batch size of 256. The learning rate starts at 0.1 and decays by a factor of 10 every 30 epochs. Also, we applied weight decay with 0.0001 decay rate to the model during the training. For the data augmentation, we applied random 224-pixel crops and random horizontal flips with 0.5 probability. Here, we run all experiments across 8 NVIDIA P100 GPUs for 90 epochs.

D.2 Language Model Setup

We followed the implementation in the repository `https://github.com/salesforce/awd-lstm-lm` and trained word level Penn Treebank with LSTM without fine-tuning or continuous cache pointer augmentation for 750 epochs. The experiments were conducted on a single RTX2080Ti. We used the default hyperparameter tuning except for learning rate and momentum: The LSTM has 3 layers containing 1150 hidden units each, embedding size is 400, gradient clipping has a maximum norm 0.25, batch size is 80, using variable sequence length, dropout for the layers has probability 0.4, dropout for the RNN layers has probability 0.3, dropout for the input embedding layer has probability 0.65, dropout to remove words from embedding layer has probability 0.1, weight drop (Merity et al., 2017) has probability 0.5, the amount of $\ell 2$-regularization on the RNN activation is 2.0, the amount of slowness regularization applied on the RNN activation is 1.0 and all weights receive a weight decay of 0.0000012.