# ACMo: Angle-Calibrated Moment Methods for Stochastic Optimization

**Xunpeng Huang,** [1] **Runxin Xu,** [2,*]**, Hao Zhou,** [1] **Zhe Wang,** [3,*] **Zhengyang Liu,** [4,*] **Lei Li** [1]

[1] Bytedance AI Lab, Shanghai, China
[2] Peking University, Beijing, China
[3] Ohio State University, Columbus , Ohio , United States
[4] Beijing Institute of Technology, Beijing, China
{huangxunpeng, zhouhao.nlp, lileilab}@bytedance.com,
runxinxu@gmail.com, wang.10982@osu.edu, zhengyang@bit.edu.cn

## Abstract

Stochastic gradient descent (SGD) is a widely used method for its outstanding generalization ability and simplicity. Adaptive gradient methods have been proposed to further accelerate the optimization process. In this paper, we revisit existing adaptive gradient optimization methods with a new interpretation. Such new perspective leads to a refreshed understanding of the roles of second moments in stochastic optimization. Based on this, we propose *Angle-Calibration Moment method* (ACMo), a novel stochastic optimization method. It enjoys the benefits of second moments with only first moment updates. Theoretical analysis shows that ACMo is able to achieve the same convergence rate as mainstream adaptive methods. Experiments on a variety of CV and NLP tasks demonstrate that ACMo has a comparable convergence to state-of-the-art Adam-type optimizers, and even a better generalization performance in most cases. The code is available at https://github.com/Xunpeng746/ACMo.

## Introduction

Deep neural network has been widely adopted in different applications because of its excellent performance, which always requires a huge amount of data for training. Calculating the full gradient of data and performing the full gradient descent (GD) become computationally expensive. Therefore, stochastic gradient descent (SGD) has become very popular for training deep neural networks. Empirically, in each step of the training, SGD samples a mini-batch of data and applies gradient descent with the corresponding stochastic gradients computed on the mini-batch.

In practice however, the vanilla SGD does not always produce good results, in which case many SGD variants are proposed. Especially, relevant work has shown that incorporating momentum information into SGD can help with its optimization process. Specifically, by introducing the first moment, the SGD momentum can help the model escape from some saddle points, and therefore improve its generalization. Intuitively, vanilla SGD walks along the steepest path, whereas the added momentum makes the optimization process smoother and quicker, thus helping the model to barrel through narrow valleys.

Additionally, *second moments* are used to adapt the learning rate of model parameters (Duchi, Hazan, and Singer 2011), performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features and larger updates (i.e. high learning rates) for those associated with infrequent features. This always accelerates the optimization process towards the objective. Moreover, Adam is proposed to utilize both of first and second moments for enjoying both of their benefits (Kingma and Ba 2015). Currently, it is one of the most widely used methods for neural network training.

Although Adam has achieved a great success, we argue that introducing extra second moments is not necessarily the best way to boost the optimization efficiency. First, by designing special optimization problems, the solution of adaptive gradient methods with second moments may fall into a local minima of pool generalization (Wilson et al. 2017). Second, keeping a copy of second moments (with the same size as the parameters) brings high memory overhead, leading to smaller mini-batch size for training. This may adversely affect the performance of many applications that are sensitive to training batch size. Given the above concerns, could we design an optimization approach that *only uses first moments but enjoys the benefits of both first (better generalization by escaping saddle points) and second (fast convergence with adaptive learning) moments*?

In this paper, we offer an affirmative to the question by proposing an Angle-Calibrated Moment method (ACMo) for stochastic optimization. ACMo takes a further step by explicitly requiring that the opposite direction of current descent be in acute angles with both the current gradient and the directions of historical mini-batch updating. Given the angle constraint, ACMo is likely to ensure descents for all mini-batch losses, while guaranteeing sufficient descent of current mini batches. Although ACMo has abandoned the second moment, on which many other methods rely, it still takes the advantage of fast convergence as those methods. We summarize our contributions below

- We propose ACMo, which is a new SGD variant. To the best of our knowledge, ACMo is the first approach without relying on second moments, but still has comparable convergence speed as compared with Adam-type methods.
- We provide a novel view and an intuitive analysis to under-

[*] Work is done while at Bytedance AI Lab.

stand the effect of second moments, which may shed light on the following work in this field .
- We provide theoretical results for the gradient norm convergence of ACMo on the nonconvex settings, which illustrates that ACMo can offer the same convergence rate comparing with the Adam-type optimizers. Experimental results on different CV and NLP tasks show that, even without second moments, ACMo can display convergence speed that is on par with SOTA Adam-type optimizers, while obtains even better generalization in most cases.

**Notations.** In the rest of this paper, we have parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ where $d$ denotes the dimension of parameters. The $l_2$ norm of a given vector $\boldsymbol{\theta}$ is expressed by $\|\boldsymbol{\theta}\|= \sqrt{\sum_{i=1}^{d} \theta_i^2}$. With slightly abuse of notation, we represent arithmetic symbols as element-wise operations for vectors, e.g., $\mathbf{a}^2 = [\mathrm{a}_1^2, \mathrm{a}_2^2, \ldots]^T, \mathbf{a}/\mathbf{b} = [\mathrm{a}_1/\mathrm{b}_1, \mathrm{a}_2/\mathrm{b}_2, \ldots]^T$. We denote $\lfloor x \rfloor$ as the greatest integer less than or equal to the real number $x$. Given any integers $x, y$, where $y > 0$, we denote $x \pmod y$ as the remainder of the Euclidean division of $x$ by $y$. In the finite-sum loss function, $f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta})$, the number of instances and the loss of the $i$-th training data are represented as $n$ and $f_i(\boldsymbol{\theta})$, respectively. Besides, we denote $f_{\mathbb{A}}(\boldsymbol{\theta})$ when we feed a collection of samples, i.e., $f_{\mathbb{A}}(\boldsymbol{\theta}) := \frac{1}{|\mathbb{A}|}\sum_{i\in\mathbb{A}} f_i(\boldsymbol{\theta})$. For an optimization algorithm, if its update paradigm can be formulated as

$$\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} \ \hat{f}_{\mathbb{A}_t}(\boldsymbol{\theta}, \boldsymbol{\theta}_t), \tag{1}$$

we denote eq. 1 as the iteration auxiliary problem (similar to Nesterov and Polyak (2006)).

## Related Work

In this section, we will introduce the development of the neural network optimizers.

**First Moment Optimizers** SGD-momentum and Nesterov accelerated gradient (Nesterov 2013) are widely used in training large-scale neural networks, but because of the learning rate issue, their excellent generalization ability is brittle.

**Second Moment Optimizers** To accelerate the convergence, researchers began to focus on the design of adaptive gradient methods. Adagrad (Duchi, Hazan, and Singer 2011) introduced the second moment to obtain a self-adaptive learning rate, thus freeing researcher of the troubles of parameter tuning. The update rules of Adagrad can be formulated as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \cdot \mathbf{g}_t/\sqrt{\mathbf{v}_t}$, where $\mathbf{g}_t$ denotes the stochastic gradient, $\mathbf{v}_t$ is the accumulation of gradient's second moments, i.e., $\mathbf{v}_t = \sum_{\tau=1}^{t} \mathbf{g}_\tau^2$, and $\alpha_t$ is the decreasing learning rate with $\alpha_t = \Theta(1/\sqrt{t})$. Theoretically, Adagrad improved the convergence of regret from $O(\sqrt{d/T})$ to $O(1/\sqrt{T})$ for the convex objectives with sparse gradients. However, in practice, people realize that adaptive gradient of Adagrad, i.e., $\mathbf{g}_t/\sqrt{\mathbf{v}_t}$ goes to zero very quickly due to the fact that $\mathbf{v}_t$ accumulates to large number quickly as the algorithm proceeds, and they often require optimizers to have a lower memory cost for training a larger mini-batch. To make it through, RMSProp (Hinton, Srivastava, and Swersky 2012) uses the exponential decay in second moments to control the accumulation speed of second moment in Adagrad, and min-max squared graidient is introduced to implement Adagrad with a memory-efficient way (Anil et al. 2019).

**Adam-type Optimizers** To take both the benefits from first and second moments, Adam was proposed to incorporate the momentum into RMSProp. The detailed procedure of Adam can be formulated as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \cdot \mathbf{m}_t/\sqrt{\mathbf{v}_t}$, where $\mathbf{m}_t$ is the exponential decay of momentum, i.e., $\mathbf{m}_t = \sum_{\tau=1}^{t}(1-\beta_1)\beta_1^{t-\tau}\mathbf{g}_\tau$, and $\mathbf{v}_t$ is the exponential decay of second moments, i.e., $\mathbf{v}_t = \sum_{\tau=1}^{t}(1-\beta_2)\beta_2^{t-\tau}\mathbf{g}_\tau^2$.

The faster convergence, robust hyper-parameters and good performance on bunch of tasks make Adam become one of the most successful optimizers. Adam was proved to be divergent in certain convex cases, and Amsgrad was proposed to correct the direction of Adam (Reddi, Kale, and Kumar 2018).

Although convergence, the generalization ability of adaptive algorithms is worse than SGD-momentum in many tasks. Thus, a lot of studies are proposed to improve the generalization performance of Adam-type methods by making some connections between Adam and SGD-momentum, e.g., ND-Adam (Zhang et al. 2017), AdamW (Loshchilov and Hutter 2017), SWATS (Keskar and Socher 2017), Adabound (Luo et al. 2019), PAdam (Chen et al. 2020), etc.

All the aforementioned methods try to find the connection between Adam and SGD-momentum, thus all these algorithms take the second moment adaptation as a grant. Different from previous work, we revisit the original idea of second moment adaptation, and propose an angle based algorithm without second moments. Such intuition makes ACMo have a simpler update, a lower memory overhead, a good generalization performance, and a comparable convergence to SOTA Adam-type optimizers.

## Proposed ACMo

In this section, we propose a novel optimizer, the Angle-Calibrated Moment method (ACMo), for both fast convergence and ideal generalization *with only first moments*.

We first give some theoretical analysis, showing that incorporating second moments into optimization approximately equals to penalize the projection of the current descent direction on previous gradients. Besides, such penalty helps facilitate the descent of the current batch loss while not harming the descent effects of previous iterations as much as possible. After that, we replace the projection penalty with the inner product penalty in iteration auxiliary problems, which partially preserve geometric properties of second moments (protecting effects of previous updates) to expect a fast convergence. Besides, the new iteration auxiliary problem can be considered as a general form of SGD-momentum updates to expect a good generalization ability. Finally, with sufficient descent constraints of the current batch loss, we propose

ACMo whose update paradigm is

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \left( \mathbf{g}_t + \beta_t \cdot \frac{\|\mathbf{g}_t\|}{\|\hat{\mathbf{m}}_{t-1}\| + \delta_t} \cdot \hat{\mathbf{m}}_{t-1} \right), \quad (2)$$

where $\mathbf{g}_t$ and $\mathbf{m}_t$ are mini-batch gradients and angle-calibrated moments at iteration $t$, respectively.

## Second Moments Work as Projection Penalty to Preserve Previous Descent

In this section, we revisit the iterations of existing adaptive algorithms from a novel point of view and denote the essential effect of second moments which is to penalize the projection of the current descent direction on previous gradients. Such penalty is desired to decrease the cumulative loss by guaranteeing a certain descent on the current batch loss with little increasing previous batch loss.

One may notice that almost all the adaptive methods utilize second moments to adjust the individual magnitude of their updates. However, the efficient calculation of second moments is only proposed in Adagrad (Duchi, Hazan, and Singer 2011). Here is the original update of Adagrad:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \mathbf{G}_t^{-1/2} \mathbf{g}_t, \quad (3)$$

where $\mathbf{G}_t = \sum_{\tau=1}^{t} \mathbf{g}_\tau \mathbf{g}_\tau^T$, and $\mathbf{g}_\tau$ denotes the stochastic gradient calculated at iteration $\tau$. The iteration auxiliary problem of Adagrad corresponding to eq. 3 can be formulated as

$$\begin{aligned} &\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} \overline{f}(\boldsymbol{\theta}), \\ &\overline{f}(\boldsymbol{\theta}) \coloneqq \underbrace{(\boldsymbol{\theta} - \boldsymbol{\theta}_t)^T \mathbf{g}_t}_{T_1} + \underbrace{\frac{1}{2\alpha_t} (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^T \mathbf{G}_t^{1/2} (\boldsymbol{\theta} - \boldsymbol{\theta}_t)}_{T_2}. \end{aligned} \quad (4)$$

To investigate the properties of second moments, we provide $\overline{f}(\boldsymbol{\theta})$ an upper bound, $\hat{f}_{\mathbb{A}_t}(\boldsymbol{\theta})$, with the Young's inequality (Young 1912), and minimize it with another iteration auxiliary problem formulated as

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{t+1} &= \arg\min \hat{f}_{\mathbb{A}_t}(\boldsymbol{\theta}), \\ \hat{f}_{\mathbb{A}_t}(\boldsymbol{\theta}) &= \underbrace{(\boldsymbol{\theta} - \boldsymbol{\theta}_t)^T \mathbf{g}_t}_{T_1} + \underbrace{\frac{1}{2\alpha_t} \|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2}_{T_3} \\ &\quad + \underbrace{\frac{1}{8\alpha_t} \sum_{\tau=1}^{t} \left\| (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^T \mathbf{g}_\tau \right\|^2}_{T_4} \end{aligned} \quad (5)$$

where $\mathbb{A}_t$ denotes the chosen mini-batch at iteration $t$. Thus, the global minima $\hat{\boldsymbol{\theta}}_{t+1}$ of eq. 5 can also be considered as an approximate solution of eq. 4. Investigating $\hat{f}_{\mathbb{A}_t}(\boldsymbol{\theta})$, we find $T_1 + T_3$ is the second-order Taylor polynomial of $f_{\mathbb{A}_t}(\boldsymbol{\theta})$ near $\boldsymbol{\theta}_t$, which is completely the same with the objective function of SGD iteration auxiliary problem. Besides, the term $T_4$ in $\hat{f}_{\mathbb{A}_t}(\boldsymbol{\theta})$ can be considered as a penalty for the projections of current descent $\boldsymbol{\theta} - \boldsymbol{\theta}_t$ on the previous gradients $\mathbf{g}_\tau$. If we approximate $\mathbf{g}_\tau$ in $T_4$ with $\nabla f_{\mathbb{A}_\tau}(\boldsymbol{\theta}_t)$ due to smoothness assumption, and replace the projection regularization ($T_4$) to some constraints, we can obtain an approximate optimization problem with a hard margin formulated as

$$\begin{aligned} \arg\min_{\boldsymbol{\theta}} \ & (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^T \mathbf{g}_t + \frac{1}{2\alpha_t} \|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2 \\ \text{s.t.} \ & (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^T \nabla f_{\mathbb{A}_\tau}(\boldsymbol{\theta}_t) = 0, \tau \le t. \end{aligned} \quad (6)$$

The constraints in eq. 6 denote the descent direction, i.e., $\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t$ is desire to be orthogonal to $\nabla f_{\mathbb{A}_\tau}(\boldsymbol{\theta}_t)$ for any $\tau \le t$, which plays a similar role to $T_4$ in eq. 5. In the following, we provide a explanation about such constraint optimization problem from a random shuffling perspective.

Before each epoch begins, the whole training dataset $\mathbb{A}$ is usually randomly shuffled, and is partitioned into mini-batch of equal size $\{\mathbb{A}_0, \mathbb{A}_1, \ldots, \mathbb{A}_{p-1}\}$. Then the algorithm is fed with the samples in the fixed order, say, first $\mathbb{A}_0$, then $\mathbb{A}_1$, and so on. The whole procedure repeats after each iteration over the whole dataset. Let $\nabla f_{\mathbb{A}_t}(\boldsymbol{\theta}_t)$ be the gradient calculated at iteration $t$ by using the sample in subset $\mathbb{A}_t$, where $0 \le t < p$. Note that the loss function is the average of all the samples, e.g., $\frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta})$. If we utilize $\nabla f_{\mathbb{A}_t}(\boldsymbol{\theta}_t)$ to directly update parameters like SGD, e.g., $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla f_{\mathbb{A}_t}(\boldsymbol{\theta}_t)$, the batch loss $\sum_{i \in \mathbb{A}_t} f_i(\boldsymbol{\theta})$ will decrease since it aligns with the opposite direction of its gradient, e.g., $(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t)^T \nabla f_{\mathbb{A}_t}(\boldsymbol{\theta}_t) = -\|\nabla f_{\mathbb{A}_t}(\boldsymbol{\theta}_t)\|^2 < 0$. However, for the loss corresponding to the sample not in $\mathbb{A}_t$, e.g., $f_{\mathbb{A}_\tau}(\boldsymbol{\theta}),\ \tau \ne t$, it is highly possible that $(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t)^T \nabla f_{\mathbb{A}_\tau}(\boldsymbol{\theta}_t) > 0$. In other words, only using $-\nabla f_{\mathbb{A}_t}(\boldsymbol{\theta}_t)$ as update direction will decrease the loss corresponding to $\mathbb{A}_t$ but increase the loss except $\mathbb{A}_t$. Hence, with the orthogonal requirements, i.e., the constraints in eq. 6, one can consider that *Adagrad decreases the cumulative loss by guaranteeing a descent on the current batch loss while does not increase previous batch loss sharply.*

**Algorithm 1** **A**ngle-**C**alibrated **Mo**ment method.

**Input:** initial point $\boldsymbol{\theta}_0 \in \mathcal{X}$; step size $\{\alpha_t\}$, momentum parameters $\{\beta_t\}$
set $\hat{\mathbf{m}}_0 = 0$
**for** $t = 1$ to $T$ **do**
  $\mathbf{g}_t = \nabla f_{\mathbb{A}_t}(\boldsymbol{\theta}_t)$
  $\hat{\beta}_t = \beta_t \|\mathbf{g}_t\| / (\|\hat{\mathbf{m}}_{t-1}\| + \delta_t)$
  $\hat{\mathbf{m}}_t = \mathbf{g}_t + \Psi(\hat{\beta}_t, \hat{\beta}_{t-1}) \cdot \hat{\mathbf{m}}_{t-1}$
  $\boldsymbol{\theta}_{t+1} = \Pi_{\mathcal{X}} (\boldsymbol{\theta}_t - \alpha_t \cdot \hat{\mathbf{m}}_t)$
**end for**
**Return:** $\boldsymbol{\theta}_{\mathrm{o}}$ with a discrete distribution as $P(\mathrm{o} = i) = \alpha_{i-1} \Big/ \left(\sum_{\tau=1}^{T-1}\right) \alpha_\tau,\ 2 \le \mathrm{o} \le T.$

## Angle-Calibrated Moments Warrant Descents

In this section, we first enhance the iteration auxiliary problem eq. 5. Then, we make the opposite direction of current descent forms acute angles with both current mini-batch gradient and some moments to introduce our ACMo in Algorithm 1.

Enlightened by the analysis in section , we realize that adding the projection penalty ($T_4$ in eq. 5) is to ensure that

the descent direction does not increase previous mini-batch losses. If we replace it with a weighted inner products penalty, we can even expect the descent direction make decrease of both the current mini-batch loss and the previous mini-batch losses. Hence, we can formulate the objective of the new iteration auxiliary problem as:

$$\tilde{f}_{\mathbb{A}_t}(\boldsymbol{\theta}) = (\boldsymbol{\theta}-\boldsymbol{\theta}_t)^T \mathbf{g}_t + \frac{L_t}{2}\|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|^2 + (\boldsymbol{\theta}-\boldsymbol{\theta}_t)^T \hat{\mathbf{g}}_t, \quad \text{where} \quad \hat{\mathbf{g}}_t \coloneqq \hat{\beta}_t \sum_{\tau=1}^{t} w_\tau \mathbf{g}_\tau, \tag{7}$$

where $L_t$ denotes the constant of smoothness, and $w_\tau$ denotes the weight to measure the approximation confidence about $\mathbf{g}_\tau$. For updating $\boldsymbol{\theta}$ through minimizing eq. 7, the optimum of the quadratic function satisfies

$$\begin{aligned} &\frac{\partial \tilde{f}_{\mathbb{A}_t}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{t+1}} = \mathbf{g}_t + L_t(\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_t) + \hat{\mathbf{g}}_t = 0 \\ \Leftrightarrow &\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\hat{\mathbf{g}}_t + \mathbf{g}_t}{L_t}. \end{aligned} \tag{8}$$

Notice that if we set $\hat{\beta}_t = 1$ and $w_\tau = \beta_0^\tau$ in eq. 7, eq. 8 corresponds to the update paradigm of SGD-momentum coincidentally. Thus, we may get the benefit of generalization performance from the iteration auxiliary problems. Different from SGD-momentum, we want to guarantee a sufficient descent for the loss of current mini-batch. Hence, $\hat{\beta}_t$ is requested to have

$$\begin{aligned} &\hat{\beta}_t := \frac{\beta_t \|\mathbf{g}_t\|}{\left\|\sum_{\tau=1}^{t-1} w_\tau \mathbf{g}_\tau\right\| + \delta_t}, \ \beta_t \le 1 \\ \Rightarrow &\hat{\beta}_t \le \frac{\|\mathbf{g}_t\|}{\left\|\sum_{\tau=1}^{t-1} w_\tau \mathbf{g}_\tau\right\|} \\ \Rightarrow &\hat{\beta}_t \left\|\sum_{\tau=1}^{t-1} w_\tau \mathbf{g}_\tau\right\| \le \|\mathbf{g}_t\|, \end{aligned} \tag{9}$$

for Eq. 7. Notice that the last inequality of eq. 9 implies current mini-batch loss descent as

$$\begin{aligned} &f_{\mathbb{A}_t}(\boldsymbol{\theta}_{t+1}) - f_{\mathbb{A}_t}(\boldsymbol{\theta}_t) \\ \overset{①}{\le} &(\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_t)^T \mathbf{g}_t + \frac{L_t}{2}\|\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_t\|^2 \\ \overset{②}{=} &-\left[\frac{\|\mathbf{g}_t\|^2 + \hat{\mathbf{g}}_t^T \mathbf{g}_t}{L_t}\right] + \frac{L_t}{2}\cdot\frac{\|\mathbf{g}_t+\hat{\mathbf{g}}_t\|^2}{L_t^2} \\ = &-\frac{\|\mathbf{g}_t\|^2}{2L_t} + \frac{\|\hat{\mathbf{g}}_t\|^2}{2L_t} \overset{③}{\le} 0, \end{aligned} \tag{10}$$

where ① follows from the smoothness assumption, ② is established due to eq. 8 and ③ is from eq. 9. Besides, we denote $\sum_{\tau=1}^{t-1} w_\tau \mathbf{g}_\tau$ as $\hat{\mathbf{m}}_{t-1}$ with the following iteration to generate $w_\tau$s automatically,

$$\hat{\mathbf{m}}_t = \mathbf{g}_t + \beta_t \cdot \frac{\|\mathbf{g}_t\|}{\|\hat{\mathbf{m}}_{t-1}\| + \delta_t} \cdot \hat{\mathbf{m}}_{t-1}, \tag{11}$$

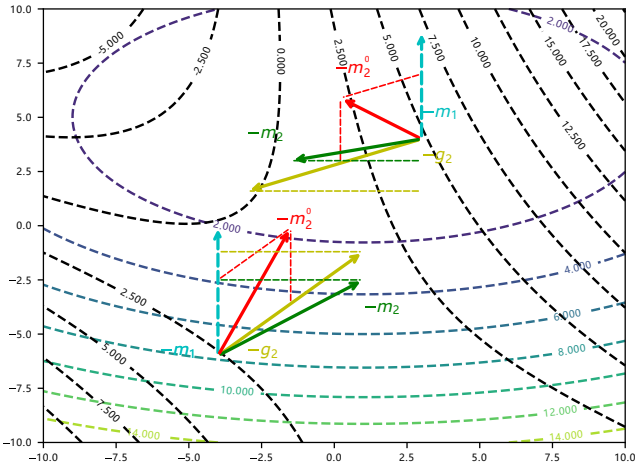


Figure 1: Cyan: previous moments. Yellow: current mini-batch negative gradients. Green: descent directions for auxiliary problems with the projection regularization, e.g., Adagrad. **Red**: descent directions for auxiliary problems with ACMo regularization.

where $\beta_t$ and $\delta_t$ are hyper-parameters. From a geometric perspective, $\hat{\mathbf{m}}_t$ is the angle bisector of $\mathbf{g}_t$ and $\hat{\mathbf{m}}_{t-1}$ coincidentally when $\beta_t = 1$ and $\delta_t = 0$ (see Figure 1). Then, we can reformulate the iteration auxiliary problem of ACMo as:

$$\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} \frac{L_t}{2}\|\boldsymbol{\theta}-\boldsymbol{\theta}_t\|^2 + (\boldsymbol{\theta}-\boldsymbol{\theta}_t)^T \mathbf{g}_t + \beta_t \cdot \frac{\|\mathbf{g}_t\|}{\|\hat{\mathbf{m}}_{t-1}\| + \delta_t} \cdot (\boldsymbol{\theta}-\boldsymbol{\theta}_t)^T \hat{\mathbf{m}}_{t-1}, \tag{12}$$

where we can obtain the update paradigm of ACMo as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \left(\mathbf{g}_t + \beta_t \cdot \frac{\|\mathbf{g}_t\|}{\|\hat{\mathbf{m}}_{t-1}\| + \delta_t} \cdot \hat{\mathbf{m}}_{t-1}\right). \tag{13}$$

In summary, we observe that (i) if $\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}$ forms acute angles with both $\mathbf{g}_t$ and $\nabla f_{\mathbb{A}_\tau}(\boldsymbol{\theta}_t)$, rather than penalizing the projection as Adagrad, we can obtain descent on both current and previous batches; (ii) to handle the case when the estimation of accumulative weighted gradients using $\hat{\mathbf{m}}_{t-1}$ is not accurate, we expect the current gradient $\mathbf{g}_t$ to dominate the descent direction for guaranteeing a sufficient descent of current mini-batch loss. Hence, we propose a new first moment optimizer, which attach the properties of the second moment to first moment iterations, inspired by the iteration auxiliary problem of Adagrad. The proposed algorithm ACMo is shown in Algorithm 1. Note that, $\Psi(\cdot)$ is a function to guarantee a sufficient descent of the iteration auxiliary problem. In practice, we usually set $\Psi(\hat{\beta}_t, \hat{\beta}_{t-1}) = \hat{\beta}_t$.

## Theoretical Results

In this section, we provide the convergence about gradient norm in expectation for our ACMo in nonconvex settings. Our theoretical results show that ACMo obtains the same

convergence rate with Adam-type optimizers. All details of our proof can be found in our supplementary material. We list assumptions required in convergence analysis, and then provide the main theoretical results.

**Assumption 1.** *We assume the loss function $f(\boldsymbol{\theta})$ is differentiable, and has L-Lipschitz gradient, i.e., for any feasible solution $\boldsymbol{\theta}_i, \boldsymbol{\theta}_j \in \mathbb{R}^d$, $\|\nabla f_i(\boldsymbol{\theta}_i) - \nabla f_j(\boldsymbol{\theta}_j)\| \leq L\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|$.*

**Assumption 2.** *We assume the objective $f(\boldsymbol{\theta})$ is lower bounded, which means $\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) > -\infty$.*

**Assumption 3.** *For the mini-batch loss $f_{\mathbb{A}_t}(\boldsymbol{\theta})$ at iteration t, we assume stochstic gradients, e.g., $\nabla f_{\mathbb{A}_t}(\boldsymbol{\theta})$ and $\nabla f_i(\boldsymbol{\theta})$ satisfy*

$$\mathbb{E}\left[\nabla f_{\mathbb{A}_t}(\boldsymbol{\theta})\right] = \nabla f(\boldsymbol{\theta}), \quad \|\nabla f_{\mathbb{A}_t}(\boldsymbol{\theta})\| \leq G,$$
$$\max_i \left\{\|\nabla f_i(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta})\|^2\right\} \leq \sigma^2.$$

**Theorem 1.** *Suppose Assumption 1, 2, 3 hold. If we set $\epsilon \geq 0$, $\delta \geq \sigma$, $\beta_t \leq 1/50$, $0 < a_0 \leq 3/(4L + 1240)$ and*

$$\alpha_t = \frac{a_0}{\sqrt{t}}, \Psi(\hat{\beta}_{t+1}, \hat{\beta}_t) = \min\left\{\hat{\beta}_{t+1}, \sqrt{\frac{t+1}{t}}\hat{\beta}_t\right\},$$

*without loss of generality. Then, the output of ACMo satisfies*

$$\mathbb{E}\left[\|\nabla f(\boldsymbol{\theta}_o)\|^2\right] \leq \frac{C_0}{\sqrt{T}} + \frac{C_1 \log(T)}{\sqrt{T}}$$

*where $C_0$ and $C_1$ are constants independent with $T$, $n$, $d$ and presented in our proof.*

Theorem 1 shows that ACMo has the same convergence rate as Adam-type optimizers (Chen et al. 2019). Comparing with theoretical results provided in Chen et al. (2020), ours has additional $O(\log T/\sqrt{T})$ term. Since we do not require the condition about the sparsity of gradients as Chen et al. (2019). Besides, we show that rather than requesting the coefficient sequence of first moments to be non-increasing, a gently increasing sequence ($\Psi(\cdot)$ in ACMo) can keep an $\tilde{O}(1/\sqrt{T})$ convergence rate for ACMo, which expands the range of hyper-parameters selection.

## Experiments

In this section, we conduct extensive experiments on image classification and neural machine translation tasks. We want to demonstrate the generalization performance and the efficiency of ACMo, as compared with other adaptive gradient methods, e.g, Adam (Kingma and Ba 2015), Amsgrad (Reddi, Kale, and Kumar 2018), Adamw (Loshchilov and Hutter 2017), PAdam (Chen et al. 2020), and Adabound (Luo et al. 2019) and SGD-momentum. Besides, we conduct Adagrad and SGD on an additional CV task to validate our intuition of second moments acceleration effect.

**Hyperparameter Tuning** Hyperparameters in optimizers can exert great impact on ultimate solutions found by optimization algorithms. In our experiments, we tune over hyperparameters in the following way. For all optimizers in our experiments, we choose the best initial set of step size from $\{1e-1, 5e-2, 1e-2, 5e-3, \ldots, 5e-5\}$.

*SGD-momentum:* We tune the coefficient of momentum from $\{0.9, 0.8, \ldots, 0.1\}$.

*Adam, AMSGrad, AdamW, PAdam, and Adabound:* For image classification tasks, we turn over $\beta_1$ values of $\{0.9, 0.99\}$ and $\beta_2$ values of $\{0.99, 0.999\}$ and the perturbation value $\epsilon = 1e-8$. For nerual machine translation tasks, we set $\beta_1$ and $\beta_2$ as the suggested values of Transformer (Vaswani et al. 2017), where $\beta_1 = 0.9$ and $\beta_2 = 0.98$. Besides, for PAdam, we choose the best hyperparameter $p$ from $\{1/4, 1/8, 1/16\}$.

*ACMo:* We directly apply the default hyperparameters, i.e., $\beta_t = 0.9$, $\Psi(\hat{\beta}_t, \hat{\beta}_{t-1}) = \hat{\beta}_t$, for all our experiments.

### Experiments on Image Classification Tasks

In the image classification tasks, adaptive gradient methods display fast convergence, but poorer generalization results when compared to the well tuned SGD momentum, which includes proper hyper parameters for the learning and weight decay. Note that introducing weight decay is equivalent to adding $l_2$ regularization to the objectives, and has a significant impact on the generalization ability of optimizers. Hence, our experiments were conducted from two perspectives as guidelines: (i) We record the convergence of training loss and the test accuracy of all optimizers with fixed weight decays (they optimize the same objective function). (ii) We adopt the optimal weight decays for all optimizers, then investigate their generalization ability based on the test accuracy.

The paper then proceeds to introduce the experimental settings for the image classification tasks. We used two datasets CIFAR-10, CIFAR-100 (Krizhevsky, Hinton et al. 2009), and tested three different CNN architectures including VGGNet (Simonyan and Zisserman 2015), ResNet (He et al. 2016) and DenseNet (Huang et al. 2017). To achieve stable convergence, we ran 200 epochs, and set the learning rate to decay by $0.1$ every $50$ epochs. We performed cross-validation to choose the best hyper-parameters for for all the optimizers.

**Experiment with Fixed Weight Decay:** We first evaluated CIFAR-100 dataset. In this experiment, the values of weight decay in all optimizers were fixed, and were chosen to be the weight decay in SGD-momentum when it achieves the maximum test accuracy.

*Though ACMo was second only to SGD momentum in terms of generalization performance, it was significantly faster than not only the latter, but also Adam and AMSGrad in terms of convergence speed. Its convergence speed is comparable with SOTA Adam variants.* From the first row of Figure 2, i.e, the training curves of three tests, we observed that ACMo significantly outperforms Adam, and has comparable rate with PAdam and Adabound when we fixed the weight decay. Also, in the second row of Figure 2, i,e, the test accuracy, our ACMo outperformed all benchmarks except only SGD-momentum. However, the training loss converges much slower when implemented with SGD-momentum which is also recorded in other literature.

**Experiment with Optimal Weight Decay** We conduct experiments on CIFAR-10 dataset, and find the optimal weight

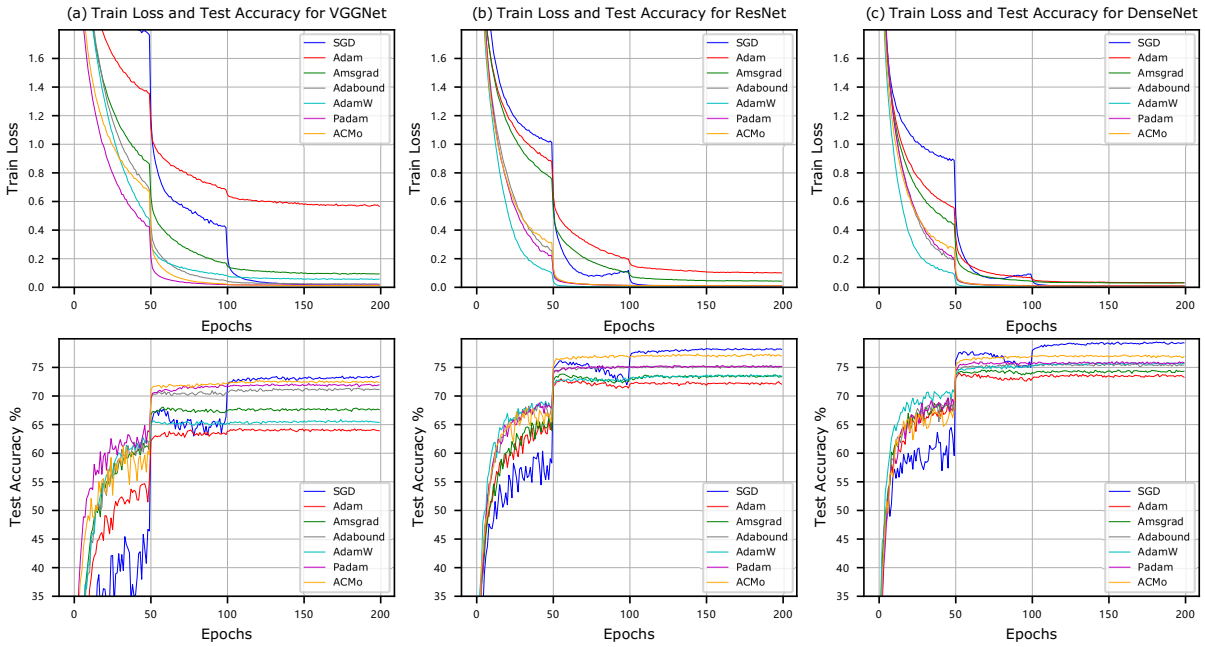


Figure 2: Learning curves of optimizers for CNNs on CIFAR-100 image classification task.Top: training loss. Bottom:test accuracy.

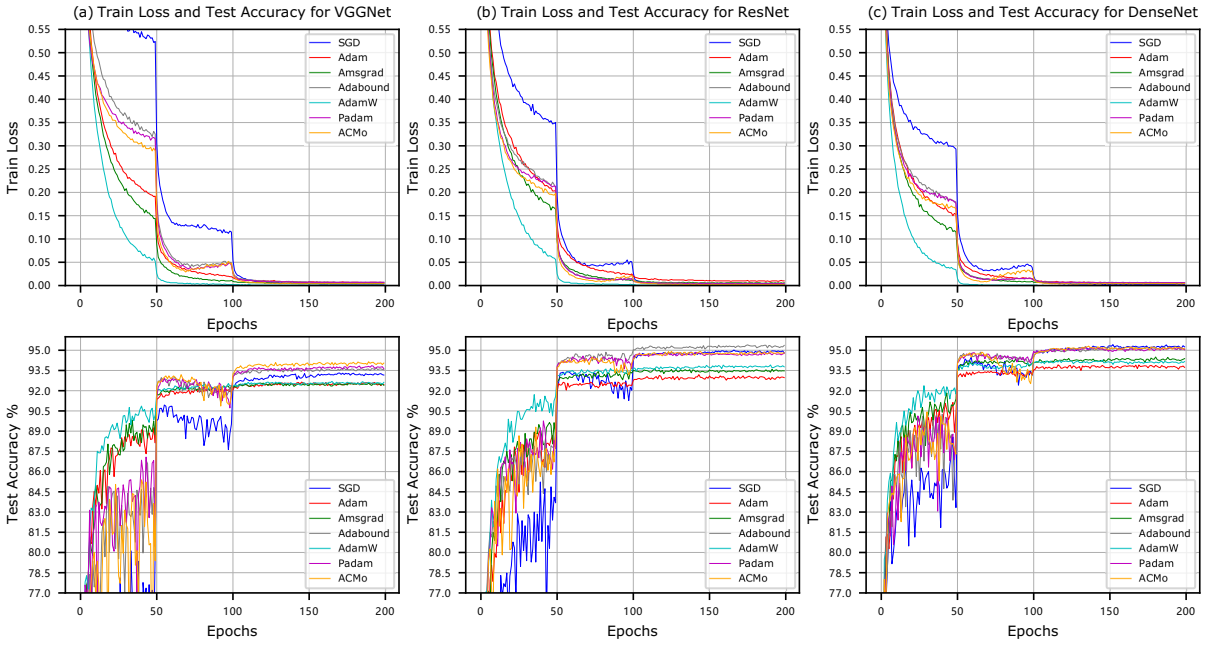


Figure 3: Learning curves of optimizers for CNNs on CIFAR-10 image classification task. Top: training loss. Bottom: test accuracy.

decay for each optimizer which can achieve the best test accuracy.

*ACMo obtains the top three results of generalization performance in different architectures with comparable efficiency to SOTA Adam-type optimizers.* As shown in Figure 3, Adam, AdamW and Amsgrad did not perform as well as other optimizers in the plots of test accuracy. In this weight decay setting, PAdam, Adabound and ACMo can even outperform SGD-momentum on VGGNet at the expense of sacrificing some efficiency. Nonetheless, they were able to outperform Adam by more than 2 percentage points for the test accuracy. The experimental results show that optimizers which converge faster than ACMo generalize worse.

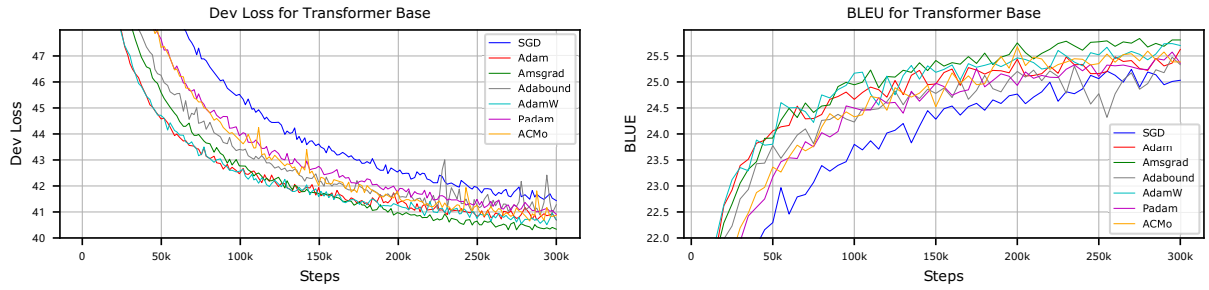


Figure 4: Learning curves of optimizers for Transformer-base on WMT'14 EN-DE machine translation task. (Dev loss & BLEU score)

## Experiments on Neural Machine Translation Tasks

Different from image classification tasks, in NLP tasks with attention models, i.e., neural machine translation (NMT), adaptive gradient methods are still mainstream optimizers (Zhang et al. 2019). Especially, Adam-type optimizers have huge advantages over first moment methods in both convergence and generalization performance in *the fixed learning rate setting*. To validate the efficiency and generalization of ACMo with complicated models, we perform experiments on WMT'14 EN-DE dataset with Transformer (Vaswani et al. 2017).

Now we introduce the experimental settings for NMT tasks. For preprocessing, sentences were encoded using byte-pair encoding , which has a shared sourcetarget vocabulary of about 37000 tokens. Besides, we utilize 4 Tesla-V100-PCIE-(16GB) GPUs to train the Tansformer base, where we set batch token size as 4096 per GPU in the training process. In order to exclude the influence of learning tricks, we conduct experiments with fixed step size and gradient clipping.

*ACMo obtains a comparable convergence and generalization performance to mainstream adaptive gradient methods, and a better performance to SGD-momentum from both convergence and generalization in the fix learning rate setting.* From Figure 4, we observe that ACMo descends a bit slowly in the early stage of training due to the lack of adaptive learning rate. Even though, the rapid descent in the middle stage of optimization help ACMo to surpass some adaptive gradient methods, i.e., Adabound and PAdam. Finally, ACMo obtains a comparable Dev Loss and BLEU results to that in Adam and AdamW, and better results to SGD-momentum on the fixed learning rate setting. These results match the records in other literature (Zhang et al. 2019) (Variants of SGD usually have a worse performance on attention models).

## Experiments for Validating Second Moments Intuition

In this section, we aim to validate our intuition about, *Adagrad decreases the cumulative loss by guaranteeing a descent on the current batch loss while does not increase previous batch loss sharply*.

We utilize commonly used Adagrad, rather than Adagrad with full matrices, to optimize VGG-16 on CIFAR-10. During the training process, we sampled some iterations, and check the descent on current mini-batch loss and the ascent on other mini-batch losses. Then, we replace Adagrad to SGD update in different learning rate at the sampled iterations, and check mini-batch losses. In Figure 5, we noticed the ascent on other mini-batch losses for Adagrad is smaller than SGDs, when they have similar descents on current mini-batch loss. Hence, we validate that the acceleration mechanism of second moments.

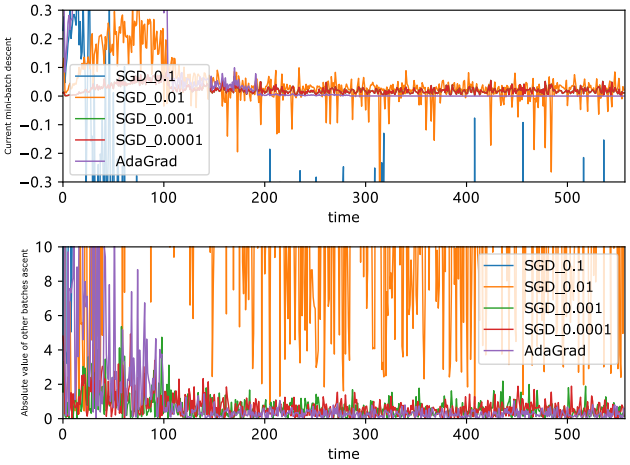


Figure 5: The intuition validation experiments. Top: the current mini-batch descent. Bottom: other mini-batch ascents

## Conclusion

In this paper, we revisited the existing adaptive optimization methods from a novel point of view. We found that the widely used second moments essentially penalize the projection of the current descent direction on previous gradients. Following such a new idea, we proposed a new method ACMo. It removes the second moments and constructs a decent direction by forming acute angles with both current and (approximated) previous gradients. We analyzed its convergence property in the nonconvex setting, and denote that ACMo shares the same convergence about gradient norm with SOTA Adam-type optimizers. Last, extensive experiments on CV and NLP tasks validated its efficiency and generalization ability.

## Acknowledgments

Thanks to the anonymous reviewers for their helpful comments and suggestions. We also thank Qin Wang for her continuous support, Yitan Li and Ning Miao for their valuable discussions. ZL is supported by Beijing Institute of Technology Research Fund Program for Young Scholars and National Natural Science Foundation of China (No. 62002017).

## References

Anil, R.; Gupta, V.; Koren, T.; and Singer, Y. 2019. Memory-Efficient Adaptive Optimization for Large-Scale Learning. *arXiv preprint arXiv:1901.11150* .

Chen, J.; Zhou, D.; Tang, Y.; Yang, Z.; Cao, Y.; and Gu, Q. 2020. Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 3267–3275. International Joint Conferences on Artificial Intelligence Organization. doi:10.24963/ijcai.2020/452. URL https://doi.org/10.24963/ijcai.2020/452. Main track.

Chen, X.; Liu, S.; Sun, R.; and Hong, M. 2019. On the Convergence of A Class of Adam-Type Algorithms for Non-Convex Optimization. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=H1x-x309tm.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hinton, G.; Srivastava, N.; and Swersky, K. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on* 14: 8.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Keskar, N. S.; and Socher, R. 2017. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628* .

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*. URL http://arxiv.org/abs/1412.6980.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Loshchilov, I.; and Hutter, F. 2017. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101* .

Luo, L.; Xiong, Y.; Liu, Y.; and Sun, X. 2019. Adaptive Gradient Methods with Dynamic Bound of Learning Rate. In *Proceedings of the 7th International Conference on Learning Representations*. New Orleans, Louisiana.

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

Nesterov, Y.; and Polyak, B. T. 2006. Cubic regularization of Newton method and its global performance. *Mathematical Programming* 108(1): 177–205.

Reddi, S. J.; Kale, S.; and Kumar, S. 2018. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=ryQu7f-RZ.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. URL http://arxiv.org/abs/1409.1556.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wilson, A. C.; Roelofs, R.; Stern, M.; Srebro, N.; and Recht, B. 2017. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, 4148–4158.

Young, W. H. 1912. On classes of summable functions and their Fourier series. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 87(594): 225–229.

Zhang, J.; Karimireddy, S. P.; Veit, A.; Kim, S.; Reddi, S. J.; Kumar, S.; and Sra, S. 2019. Why ADAM Beats SGD for Attention Models. *arXiv preprint arXiv:1912.03194* .

Zhang, Z.; Ma, L.; Li, Z.; and Wu, C. 2017. Normalized direction-preserving adam. *arXiv preprint arXiv:1709.04546* .