# SUPER-ADAM: Faster and Universal Framework of Adaptive Gradients

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Adaptive gradient methods have shown excellent performance for solving many machine learning problems. Although multiple adaptive methods were recently studied, they mainly focus on either empirical or theoretical aspects and also only work for specific problems by using specific adaptive learning rates. It is desired to design a universal framework for practical algorithms of adaptive gradients with theoretical guarantee to solve general problems. To fill this gap, we propose a faster and universal framework of adaptive gradients (*i.e.*, SUPER-ADAM) by introducing a universal adaptive matrix that includes most existing adaptive gradient forms. Moreover, our framework can flexibly integrates the momentum and variance reduced techniques. In particular, our novel framework provides the convergence analysis support for adaptive gradient methods under the nonconvex setting. In theoretical analysis, we prove that our new algorithm can achieve the best known complexity of $\tilde{O}(\epsilon^{-3})$ for finding an $\epsilon$-stationary point of nonconvex optimization, which matches the lower bound for stochastic smooth nonconvex optimization. In numerical experiments, we employ various deep learning tasks to validate that our algorithm consistently outperforms the existing adaptive algorithms.

## 1 Introduction

Stochastic Gradient Descent (SGD) is widely used in many large-scale machine learning problems such as training deep neural networks (DNNs), due to only requiring a mini-batch samples or even one sample at each iteration. Adaptive gradient methods are one of the most important variants of SGD, which use adaptive learning rates and possibly incorporate momentum techniques, so they generally require little parameter tuning and enjoy faster convergence rate than the SGD. At the same time, compared to SGD, adaptive gradient methods escape saddle points faster [21]. Thus, recently adaptive gradient methods have been widely developed and studied. For example, the first adaptive gradient method *i.e.*, Adagrad has been proposed in [8], which significantly outperforms the vanilla SGD in the sparse gradient setting. Subsequently, Adagrad has been found that it does not be well competent to the nonconvex problems and dense gradient setting. To address these drawbacks, some variants of Adagrad, *e.g.*, Adadelta [27], Adam [14], have been presented by using exponential moving average instead of the arithmetic average.

Adam [14] recently has been shown great successes in current machine learning problems, *e.g.*, it is a default method of choice for training DNNs. Unfortunately, Reddi et al. [19] still show that Adam is frequently divergent in some settings where the gradient information quickly disappear. To deal with this issue, some versions of Adam algorithm, *e.g.*, AMSGrad [19], YOGI [26], Generalized Adam [4], have been developed and studied. Specifically, AMSGrad [19] applies an extra 'long term memory' variable to preserve the past gradient information in order to handle the convergence issue in Adam.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Table 1: Convergence properties of the representative adaptive gradient algorithms for finding an $\epsilon$-stationary point of the **non-convex** stochastic problem (1), i.e., $\mathbb{E}\|\nabla f(x)\| \leq \epsilon$ or its equivalent variants. For fair comparison, we only give the gradient complexity and convergence rate in **the worst case** without considering the sparsity of stochastic gradient. Here $T$ denotes the whole number of iteration, $b$ denotes mini-batch size, and $d$ denotes dimension of data. **ALR** denotes adaptive learning rate. **1** denotes the smoothness of each component function $f(x;\xi)$; **2** denotes the smoothness of objective function $f(x) = \mathbb{E}_{\xi}[f(x;\xi)]$; **3** denotes the bounded noisy gradient $\nabla f(x;\xi)$; **4** denotes the bounded true gradient $\nabla f(x)$; **5** denotes the smoothness of true gradient $\nabla f(x)$.

| **Algorithm** | **Reference** | **Complexity** | **Convergence Rate** | **ALR** | **Conditions** |
|---|---|---|---|---|---|
| Adam/ YOGI | [26] | $O(\epsilon^{-4})$ | $O(\frac{1}{\sqrt{T}} + \frac{1}{\sqrt{b}})$ | specific | **1**, **2**, **3**, **4** |
| Generalized Adam | [4] | $\tilde{O}(\epsilon^{-4})$ | $O(\frac{\sqrt{\log(T)}}{T^{1/4}})$ | specific | **2**, **3**, **4** |
| Padam | [3] | $O(\epsilon^{-4})$ | $O(\frac{1}{\sqrt{T}} + \frac{1}{T^{1/4}})$ | specific | **2**, **3**, **4** |
| Adaptive SGD | [15] | $\tilde{O}(\epsilon^{-4})$ | $O(\frac{\sqrt{\ln(T)}}{\sqrt{T}} + \frac{\sqrt{\ln(T)}}{T^{1/4}})$ | specific | **2** |
| AdaGrad-Norm | [24] | $\tilde{O}(\epsilon^{-4})$ | $O(\frac{\sqrt{\log(T)}}{T^{1/4}})$ | specific | **2**, **4** |
| Ada-Norm-SGD | [6] | $\tilde{O}(\epsilon^{-3.5})$ | $\tilde{O}(\frac{1}{T^{2/7}})$ | specific | **2**, **5** |
| AdaBelief | [29] | $\tilde{O}(\epsilon^{-4})$ | $O(\frac{\sqrt{\log(T)}}{T^{1/4}})$ | specific | **2**, **3**, **4** |
| ADAM$^+$ | [17] | $O(\epsilon^{-3.5})$ | $O(\frac{1}{T^{2/7}})$ | specific | **2**, **5** |
| STORM | [7] | $\tilde{O}(\epsilon^{-3})$ | $O(\frac{(\ln(T))^{3/4}}{\sqrt{T}} + \frac{\sqrt{\ln(T)}}{T^{1/3}})$ | specific | **1**, **3**, **4** |
| SUPER-ADAM ($\tau = 0$) | Ours | $\tilde{O}(\epsilon^{-4})$ | $O(\frac{\sqrt{\ln(T)}}{\sqrt{T}} + \frac{\sqrt{\ln(T)}}{T^{1/4}})$ | universal | **2** |
| SUPER-ADAM ($\tau = 1$) | Ours | $\tilde{O}(\epsilon^{-3})$ | $O(\frac{\sqrt{\ln(T)}}{\sqrt{T}} + \frac{\sqrt{\ln(T)}}{T^{1/3}})$ | universal | **1** |

Please **note that** we only consider the adaptive STORM [7] algorithm (The convergence rate of STORM comes from the last line of the proof of Theorem 1 in [7]). Padam [3] uses the $\ell_\infty$-norm bounded gradients, while the other methods such as Adam [26] use the $\ell_2$-norm bounded gradients. For fair comparison, we omit the data dimension $d$ in the convergence rate of Padam [3].

YOGI [26] introduces an adaptive denominator constant, and studies the effect of the mini-batch size in its convergence. Chen et al.[4] have studied the convergence of a class of Adam-type algorithms for nonconvex optimization. Liu et al. [16] have studied the variances of the adaptive algorithm with the coordinate-wise learning rates. Although these adaptive gradient methods show good empirical performance, their generalization performances are worse than the SGD (with momentum) on many deep learning tasks due to using the coordinate-wise learning rates [25]. Thus, recently some adaptive gradient methods have been proposed to improve the generalization performance of Adam algorithm. For example, AdamW [18] and Padam [3] improve the generalization performance of Adam by decoupling weight decay regularization and introducing a partial adaptive parameter, respectively. More recently, the norm version of AdaGrad (*i.e.*, AdaGrad-Norm) [24] has been proposed to reach state-of-the-art results in some deep learning tasks without sacrificing generalization.

So far, the above adaptive gradient methods still suffer from high sample complexity of $O(\epsilon^{-4})$ for finding $\epsilon$-stationary point in the worst case without considering sparsity of gradient. More recently, some faster variance-reduced adaptive gradient methods such as STORM [7], Adaptive Normalized SGD [6], ADAM$^+$ [17] have been proposed. For example, STORM applies the momentum-based variance reduced technique to obtain a lower sample complexity of $\tilde{O}(\epsilon^{-3})$. Although several adaptive methods were recently developed, they can only solve specific problems with focusing on either pure theoretical or empirical aspects. It is desired to design a universal framework for the adaptive methods on both theoretical analysis and practical algorithms to solve general problems. Note that the Adagrad [8] algorithm uses an adaptive matrix obtained by concatenating the subgradient sequence, but it still is a class of specific adaptive learning rate. For example, the diagonal adaptive matrix used in Adagrad [8] and Sadagrad [5] is the same to the adaptive learning rate in Adaptive-SGD [15]. In addition, Duchi et al. [8] and Chen et al. [5] only study convergence properties of the proposed algorithms in the convex setting.

To fill this gap, in the paper, we propose a faster and universal framework of adaptive gradients (i.e., SUPER-ADAM) by introducing a universal adaptive matrix. Moreover, we provide a novel convergence analysis framework for the adaptive gradient methods under the nonconvex setting. In summary, our main **contributions** are in three-fold:

1) We propose a faster and universal framework of adaptive gradients (i.e., SUPER-ADAM) by introducing a universal adaptive matrix that includes most existing adaptive gradient forms. Moreover, our framework can flexibly integrate the momentum and variance-reduced techniques.
2) We provide a novel convergence analysis framework for the adaptive gradient methods in the nonconvex setting under the milder conditions (Please see Table 1).
3) We apply a momentum-based variance reduced gradient estimator [7, 22] to our algorithm (SUPER-ADAM ($\tau = 1$)), which make our algorithm reach a faster convergence rate than the classic adaptive methods. Under smoothness of each component function $f(x;\xi)$, we prove that the SUPER-ADAM ($\tau = 1$) achieves the best known complexity of $\tilde{O}(\epsilon^{-3})$ for finding an $\epsilon$-stationary point of nonconvex optimization, which matches the lower bound for stochastic smooth nonconvex optimization [1]. Under smoothness of the true function $f(x)$, we prove that the SUPER-ADAM ($\tau = 0$) achieves the complexity of $\tilde{O}(\epsilon^{-4})$.

# 2 Preliminaries

In the paper, we consider solving the following stochastic optimization problem:

$$\min_{x\in\mathcal{X}} f(x) := \mathbb{E}_{\xi\sim\mathcal{D}}[f(x;\xi)] \tag{1}$$

where $f(x)$ is smooth and possibly nonconvex function, and $\xi$ is a random variable following an unknown distribution $\mathcal{D}$. Here $\mathcal{X} \subseteq \mathbb{R}^d$ is a compact and convex set, or $\mathcal{X} = \mathbb{R}^d$. The problem (1) is frequently appears in many machine learning applications such as the expectation loss minimization.

## 2.1 Notation

$I_d$ denotes a $d$-dimensional identity matrix. $\mathrm{diag}(a) \in \mathbb{R}^d$ denotes a diagonal matrix with diagonal entries $a = (a_1, \cdots, a_d)$. For two vectors $u$ and $v$, $u^p$ $(p > 0)$ denotes the element-wise power operation, $u/v$ denotes the element-wise division and $\max(u, v)$ denotes the element-wise maximum. $\langle u, v\rangle$ denotes the inner product of two vectors $u$ and $v$. For a vector $x \in \mathbb{R}^d$, $\|x\|$ denotes the $\ell_2$ norm of $x$. For two sequences $\{a_n\}$ and $\{b_n\}$, we write $a_n = O(b_n)$ if there exists a positive constant $C$ such that $a_n \leq Cb_n$, and $\tilde{O}(\cdot)$ hides logarithmic factors. Given a convex closed set $\mathcal{X}$, we define a projection operation $\mathcal{P}_{\mathcal{X}}(y) = \min_{x\in\mathcal{X}} \|x - y\|^2$.

## 2.2 Adaptive Algorithms

In the subsection, we review the existing typical adaptive gradient methods. Recently, many adaptive algorithms have been proposed to solve the problem (1), and achieve good performances. For example, Adagrad [8] is the first adaptive gradient method with adaptive learning rate for each individual dimension. Specifically, it adopts the following update form:

$$x_{t+1} = x_t - \eta_t \frac{g_t}{\sqrt{v_t}}, \quad v_t = \frac{1}{t}\sum_{j=1}^{t} g_j^2, \tag{2}$$

where $g_t = \nabla f(x_t;\xi_t)$ is the stochastic gradient, and $\eta_t = \frac{\eta}{\sqrt{t}}$ with $\eta > 0$ is the step size. In fact, $\eta_t$ only is the basic learning rate that is the same for all coordinates of variable $x_t$, while $\frac{\eta_t}{\sqrt{v_{t,i}}}$ is the effective learning rate for the $i$-th coordinate of $x_t$, which varies across the coordinates.

Adam [14] is one of the most popular exponential moving average variant of Adagrad, which combines the exponential moving average technique with momentum acceleration. It takes the following update form:

$$\begin{aligned} &m_t = \alpha_1 m_{t-1} + (1-\alpha_1)\nabla f(x_t;\xi_t), \quad v_t = \alpha_2 v_{t-1} + (1-\alpha_2)(\nabla f(x_t;\xi_t))^2 \\ &\hat{m}_t = m_t/(\alpha_1)^t, \quad \hat{v}_t = v_t/(\alpha_2)^t, \quad x_{t+1} = x_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t}}, \end{aligned} \tag{3}$$

where $\alpha_1 > 0, \alpha_2 > 0$ and $\eta_t = \frac{\eta}{\sqrt{t}}$ with $\eta > 0$. However, Reddi et al. [19] found a divergence issue of the Adam algorithm, and proposed a modified version of Adam (i.e., Amsgrad), which adopts a new step instead of the debiasing step in (3) to ensure the decay of the effective learning rate:

$$\hat{v}_t = \max(\hat{v}_{t-1}, v_t), \quad x_{t+1} = x_t - \eta_t \frac{m_t}{\sqrt{\hat{v}_t}}. \tag{4}$$

**Algorithm 1** SUPER-ADAM Algorithm

1: **Input:** Total iteration $T$, and tuning parameters $\{\mu_t, \alpha_t\}_{t=1}^T$, $\gamma > 0$ ;
2: **Initialize:** $v_1 = 0$, and $x_1 \in \mathcal{X}$, sample one point $\xi_1$, compute $g_1 = \nabla f(x_1; \xi_1)$;
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Generate the adaptive matrix $H_t \in \mathbb{R}^{d \times d}$;
// Given two examples to update $H_t$:
5: Case 1: given $\beta \in (0, 1)$ and $\lambda > 0$ to update $H_t$,
6: $v_t = \beta v_{t-1} + (1 - \beta)\nabla f(x_t; \xi_t)^2$;
7: $H_t = \text{diag}(\sqrt{v_t} + \lambda)$;
8: Case 2: given $\lambda > 0$ to update $H_t$,
9: $b_t = b_{t-1} + \|\nabla f(x_t; \xi_t)\|^2$;
10: $H_t = \big(b_t + \lambda\big)^{1/2} I_d$;
11: Update $\tilde{x}_{t+1} = \arg\min_{x \in \mathcal{X}} \big\{\langle g_t, x\rangle + \frac{1}{2\gamma}(x - x_t)^T H_t (x - x_t)\big\}$;
12: Update $x_{t+1} = (1 - \mu_t)x_t + \mu_t \tilde{x}_{t+1}$;
13: Sample one point $\xi_{t+1}$, and compute $g_{t+1} = \alpha_{t+1}\nabla f(x_{t+1}; \xi_{t+1}) + (1 - \alpha_{t+1})\big[g_t - \tau\big(\nabla f(x_{t+1}; \xi_{t+1}) - \nabla f(x_t; \xi_{t+1})\big)\big]$, where $\tau \in \{0, 1\}$;
14: **end for**
15: **Output:** $x_\zeta$ chosen uniformly random from $\{x_t\}_{t=1}^T$.

Due to the coordinate-wise learning rates, these adaptive gradient methods frequently have worse generalization performance than its non-adaptive counterpart (e.g., SGD) on many deep learning tasks [25]. Thus, recently some effective adaptive gradient methods [24, 15, 7] were proposed with adopting the global adaptive learning rates instead of coordinate-wise counterparts. For example, AdaGrad-Norm [24] applies a global adaptive learning rate to the following update form for all $t \geq 1$

$$x_t = x_{t-1} - \eta \frac{\nabla f(x_{t-1}; \xi_{t-1})}{b_t}, \quad b_t^2 = b_{t-1}^2 + \|\nabla f(x_{t-1}; \xi_{t-1})\|^2, \; b_0 > 0, \tag{5}$$

where $\eta > 0$. At the same time, adaptive-SGD [15] adopts another global adaptive learning rate to the following update form:

$$\eta_t = \frac{k}{\big(w + \sum_{i=1}^{t-1} \|\nabla f(x_i; \xi_i)\|^2\big)^{1/2+\varepsilon}}, \quad x_{t+1} = x_t - \eta_t \nabla f(x_t; \xi_t), \tag{6}$$

where $k > 0$, $w > 0$, and $\varepsilon \geq 0$. Subsequently, STORM [7] uses not only a global adaptive learning rate but also the variance-reduced technique with the following update form:

$$\begin{aligned} \eta_t &= \frac{k}{\big(w + \sum_{i=1}^{t} \|\nabla f(x_i; \xi_i)\|^2\big)^{1/3}}, \quad x_{t+1} = x_t - \eta_t g_t, \\ g_{t+1} &= \nabla f(x_{t+1}; \xi_{t+1}) + (1 - c\eta_t^2)(g_t - \nabla f(x_t; \xi_{t+1})), \end{aligned} \tag{7}$$

where $k > 0$, $w > 0$ and $c > 0$, and $g_t$, $g_{t+1}$ are variance-reduced stochastic gradients.

To make adaptive algorithms reach a good generalization as SGD, the AdaBelief [29] optimizer was proposed to adapt the stepsize according to the 'belief' in the current gradient direction:

$$\begin{aligned} & m_t = \alpha_1 m_{t-1} + (1 - \alpha_1)\nabla f(x_t; \xi_t), \quad v_t = \alpha_2 v_{t-1} + (1 - \alpha_2)(\nabla f(x_t; \xi_t) - m_t)^2 + \varepsilon \\ & \hat{m}_t = m_t/(\alpha_1)^t, \quad \hat{v}_t = v_t/(\alpha_2)^t, \quad x_{t+1} = x_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}, \quad \forall\, t \geq 1 \end{aligned} \tag{8}$$

where $\alpha_1 > 0, \alpha_2 > 0$, and $\eta_t = \frac{\eta}{\sqrt{t}}$ with $\eta > 0$, and $\varepsilon > 0$.

## 3 SUPER-ADAM Algorithm

In the section, we propose a faster and universal framework of adaptive gradients (i.e., SUPER-ADAM) by introducing a universal adaptive matrix that includes most existing adaptive gradient forms. Specifically, our SUPER-ADAM algorithm is summarized in Algorithm 1.

At the step 4 in Algorithm 1, we generate an adaptive matrix $H_t$ based on stochastic gradient information, which can include both global and coordinate-wise learning rates. For example, $H_t$

generated from **case 1** in Algorithm 1 is similar to the adaptive learning rate used in Adam [14]. $H_t$ generated from **case 2** in Algorithm 1 is equivalent to a global learning rate, which is similar to the adaptive learning rate used in the AdaGrad-Norm [24] and Adaptive-SGD [15]. Moreover, we can also obtain the new adaptive learning rates by generating some specific adaptive matrices. For example, we can design an adaptive matrix $H_t$ defined as:

$$b_t = \frac{|\langle \nabla f(x_t;\xi_t) - \nabla f(x_{t-1};\xi_t), x_t - x_{t-1}\rangle|}{\|x_t - x_{t-1}\|^2}, \quad H_t = (b_t + \lambda) I_d, \tag{9}$$

where $\lambda > 0$. In fact, this adaptive matrix $H_t$ generates a Barzilai-Borwein step size for nonconvex stochastic optimization problems based on Barzilai-Borwein technique [2]. We can also generate a new adaptive matrix $H_t$ defined as:

$$v_t = \beta v_{t-1} + (1-\beta)(\nabla f(x_t;\xi_t) - g_t)^2, \quad H_t = \text{diag}(\sqrt{v_t} + \lambda), \tag{10}$$

where $\beta > 0$ and $\lambda > 0$. This new adaptive matrix $H_t$ is similar to the adaptive learning rate in [29].

At the step 11 in Algorithm 1, we apply a generalized gradient descent iteration to update the parameter $x$ based on the adaptive matrix $H_t$, defined as

$$\tilde{x}_{t+1} = \arg\min_{x\in\mathcal{X}} \big\{ \langle g_t, x\rangle + \frac{1}{2\gamma}(x - x_t)^T H_t (x - x_t) \big\}, \tag{11}$$

where $\gamma > 0$ is a constant stepsize. At the step 12 in Algorithm 1, we introduce a weighted solution $x_{t+1} = (1-\mu_t)x_t + \mu_t \tilde{x}_{t+1}$ based on a dynamic weighted parameter $\mu_t$ to effectively incorporate both momentum and variance reduced techniques. Under this case, our algorithm can flexibly utilize various adaptive learning rates at the step 11. By the step 11 of Algorithm 1, we have $\tilde{x}_{t+1} \in \mathcal{X}$. Since $\mathcal{X}$ is convex set, we should choose $\mu_t \in (0, 1]$ to ensure $x_{t+1} \in \mathcal{X}$ at the step 12 of Algorithm 1. We consider a special case, i.e., $\mathcal{X} = \mathbb{R}^d$. By the step 11, we can obtain

$$\tilde{x}_{t+1} = x_t - \gamma H_t^{-1} g_t, \tag{12}$$

where $\gamma > 0$. Then by the step 12, we have

$$x_{t+1} = (1-\mu_t)x_t + \mu_t \tilde{x}_{t+1} = x_t - \gamma\mu_t H_t^{-1} g_t, \tag{13}$$

where $\mu_t \in (0, 1]$. Under this case, $\gamma\mu_t$ denotes the basic stepsize as $\eta_t$ in the above (3) of Adam algorithm, and $H_t^{-1}$ denotes the adaptive stepsize as $\frac{1}{\sqrt{\hat{v}_t}}$ in the above (3) of Adam algorithm.

At the step 13 of Algorithm 1, we use the stochastic gradient estimator $g_{t+1}$ for all $t \geq 1$:

$$g_{t+1} = \alpha_{t+1}\nabla f(x_{t+1};\xi_{t+1}) + (1-\alpha_{t+1})\big[g_t - \tau\big(\nabla f(x_{t+1};\xi_{t+1}) - \nabla f(x_t;\xi_{t+1})\big)\big], \tag{14}$$

where $\tau \in \{0, 1\}$ and $\alpha_{t+1} \in (0, 1]$ for all $t \geq 1$. When $\tau = 1$, we have $g_{t+1} = \nabla f(x_{t+1};\xi_{t+1}) + (1-\alpha_{t+1})\big(g_t - \nabla f(x_{t+1};\xi_{t+1})\big)$ for all $t \geq 1$, which is a stochastic gradient estimator used in STORM [7]. When $\tau = 0$, we have $g_{t+1} = \alpha_{t+1}\nabla f(x_{t+1};\xi_{t+1}) + (1-\alpha_{t+1})g_t$ for all $t \geq 1$, which is a stochastic gradient estimator used in the Adam [14].

# 4 Theoretical Analysis

In this section, we study the convergence properties of our algorithm (SUPER-ADAM) under some mild conditions. All detailed proofs are in the supplementary materials.

## 4.1 Some Mild Assumptions

**Assumption 1.** *Variance of stochastic gradient is bounded,* i.e., *there exists a constant $\sigma > 0$ such that for all $x \in \mathcal{X}$, it follows $\mathbb{E}\|\nabla f(x;\xi) - \nabla f(x)\|^2 \leq \sigma^2$.*

**Assumption 2.** *The function $f(x)$ is bounded from below in $\mathcal{X}$,* i.e., $f^* = \inf_{x\in\mathcal{X}} f(x)$.

**Assumption 3.** *Assume the adaptive matrix $H_t$ for all $t \geq 1$ satisfies $(H_t)^T = H_t$ and $\lambda_{\min}(H_t) \geq \rho > 0$, where $\lambda_{\min}(H_t)$ denotes the smallest eigenvalue of the symmetrical matrix $H_t$.*

Assumption 1 is commonly used in stochastic optimization [10, 7]. Assumption 2 ensures the feasibility of the problem (1). Assumption 3 guarantees that the adaptive matrices $\{H_t\}$ used in our algorithm are positive definite, which is easy to achieve in our algorithm by choosing the tuning parameter $\lambda > 0$. In fact, many adaptive algorithms also implicitly use Assumption 3. For example,

Zaheer et al.[26] and Zhuang et al.[29] used the following iteration form to update the variable $x$: $x_{t+1} = x_t - \eta_t \frac{m_t}{\sqrt{v_t}+\varepsilon}$ for all $t \geq 0$ and $\varepsilon > 0$, which is equivalent to $x_{t+1} = x_t - \eta_t H_t^{-1} m_t$ with $H_t = \mathrm{diag}(\sqrt{v_t} + \varepsilon)$. Clearly, we have $H_t \succeq \varepsilon I_d \succ 0$. Ward et al.[24] applied a global adaptive learning rate to the update form in (5), which is equivalent to the following form: $x_t = x_{t-1} - \eta H_t^{-1} \nabla f(x_{t-1}; \xi_{t-1})$ with $H_t = b_t I_d$. By the above (5), we have $H_t \succeq \cdots \succeq H_0 = b_0 I_d \succ 0$. Li et al. [15] and Cutkosky et al. [7] applied a global adaptive learning rate to the update forms in (6) and (7), which is equivalent to $x_{t+1} = x_t - H_t^{-1} g_t$, where $H_t = (1/\eta_t) I_d$ and $\eta_t = k/\big(w + \sum_{i=1}^{t} \|\nabla f(x_i; \xi_i)\|^2\big)^{\alpha}$ with $k > 0, w > 0, \alpha \in (0, 1)$. By the above (6) and (7), we have $H_t \succeq \cdots \succeq H_0 = (w^{\alpha}/k) I_d \succ 0$. Reddi et al. [19] and Chen et al. [3] used the condition $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$, and let $H_t = \mathrm{diag}(\sqrt{\hat{v}_t})$, thus we have $H_t \succeq \cdots \succeq H_1 = \mathrm{diag}(\sqrt{\hat{v}_1}) = \sqrt{1-\alpha_2}\mathrm{diag}(|\nabla f(x_1; \xi_1)|) \succeq 0$. Without loss of generality, choosing an initial point $x_1$ and let $(\nabla f(x_1; \xi_1))_j \neq 0$ for all $j \in [d]$, we have $H_t \succeq \cdots \succeq H_1 \succ 0$. When $\mathcal{X} = \mathbb{R}^d$ and $H_t \neq I_d$, interestingly, our SUPER-ADAM algorithm also includes a class of novel momentum-based quasi-Newton algorithms by choosing an approximated Hessian matrix $H_t$. In fact, the quasi-Newton algorithms [23, 11, 28] generally require the bounded approximated Hessian matrices, i.e., $\bar{\kappa} \leq \lambda_{\min}(H_t) \leq \lambda_{\max}(H_t) \leq \hat{\kappa}$ for all $t \geq 1$, where $\hat{\kappa} \geq \bar{\kappa} > 0$. Overall, the above Assumption 3 is reasonable and mild. Note that we mainly focus on the adaptive algorithms in this paper, while the quasi-Newton algorithms only are the by-products of our algorithm.

### 4.2 A Universal Convergence Measure

We provide a universal and reasonable measure to evaluate convergence of our algorithm for both constrained and unconstrained optimizations, defined as

$$\mathcal{M}_t = \|\nabla f(x_t) - g_t\| + \frac{\rho}{\gamma}\|\tilde{x}_{t+1} - x_t\|. \tag{15}$$

In the following, we consider the existing four cases:

**1)** When $\mathcal{X} = \mathbb{R}^d$ and $H_t = I_d$, we have $\rho = 1$, and $\tilde{x}_{t+1} = x_t - \gamma g_t$. Then we have

$$\mathcal{M}_t = \|\nabla f(x_t) - g_t\| + \|g_t\| \geq \|\nabla f(x_t)\|. \tag{16}$$

When $\mathcal{M}_t \to 0$, we have $\|\nabla f(x_t)\| \to 0$.

**2)** When $\mathcal{X} = \mathbb{R}^d$ and $H_t \neq I_d$, we have $\tilde{x}_{t+1} = x_t - \gamma H_t^{-1} g_t$. Then we have

$$\mathcal{M}_t = \|\nabla f(x_t) - g_t\| + \rho\|H_t^{-1} g_t\|, \tag{17}$$

where $\rho$ is the smallest eigenvalue of the matrix $H_t$. In our algorithm, when iteration step $t$ increases, $\|H_t\|$ generally will become large, and then $\rho H_t^{-1} \to I_d$, i.e., $\|\rho H_t^{-1}\| \to 1$. With growing iteration step $t$ , thus we have

$$\mathcal{M}_t = \|\nabla f(x_t) - g_t\| + \|\rho H_t^{-1} g_t\| \to \|\nabla f(x_t) - g_t\| + \|g_t\| \geq \|\nabla f(x_t)\|. \tag{18}$$

When $\mathcal{M}_t \to 0$, we have $\|\nabla f(x_t)\| \to 0$.

**3)** When $\mathcal{X} \neq \mathbb{R}^d$ and $H_t = I_d$, we have $\rho = 1$. According to [10], if $x^* \in \mathcal{X}$ is a stationary point or local minimum of the problem (1), we have $x^* = \mathcal{P}_{\mathcal{X}}(x^* - \gamma \nabla f(x^*))$. If $\mathcal{M}_t = \frac{1}{\gamma}\|\tilde{x}_{t+1} - x_t\| + \|\nabla f(x_t) - g_t\| \to 0$, we have $g_t \to \nabla f(x_t)$ and $x_t \to \tilde{x}_{t+1} = \mathcal{P}_{\mathcal{X}}(x_t - \gamma g_t) \to \mathcal{P}_{\mathcal{X}}(x_t - \gamma \nabla f(x_t))$.

**4)** When $\mathcal{X} \neq \mathbb{R}^d$ and $H_t \neq I_d$, let $w_t(x) = \frac{1}{2} x^T H_t x$, then we define a prox-function associated with $w_t(x)$ as in [10], defined as:

$$V_t(x, x_t) = w_t(x) - \big[w_t(x_t) + \langle \nabla w_t(x_t), x - x_t \rangle\big] = \frac{1}{2}(x - x_t)^T H_t (x - x_t). \tag{19}$$

Thus, the step 11 of Algorithm 1 is equivalent to the following generalized projection problem:

$$\tilde{x}_{t+1} = \min_{x \in \mathcal{X}} \big\{ \langle g_t, x \rangle + \frac{1}{\gamma} V_t(x, x_t) \big\}, \tag{20}$$

where $\gamma > 0$. Following the Lemma 1 in [10], we define a generalized projected gradient $\mathcal{G}_{\mathcal{X}}(x_t, g_t, \gamma) = \frac{1}{\gamma}(x_t - \tilde{x}_{t+1})$. When $\mathcal{M}_t = \|\nabla f(x_t) - g_t\| + \frac{\rho}{\gamma}\|\tilde{x}_{t+1} - x_t\| \to 0$, we have $g_t \to \nabla f(x_t)$ and $\rho\|\mathcal{G}_{\mathcal{X}}(x_t, g_t, \gamma)\| \to 0$. Thus we can obtain $\|\mathcal{G}_{\mathcal{X}}(x_t, \nabla f(x_t), \gamma)\| \to 0$, where $x_t$ is a stationary point or local minimum of the problem (1).

Overall, our convergence measure $\mathbb{E}[\mathcal{M}_t]$ is reasonable and is tighter than the existing measures such as $\mathbb{E}\|\nabla f(x_t)\|$ for unconstrained optimization.

### 4.3 Convergence Analysis of SUPER-ADAM ($\tau = 1$)

In this subsection, we provide the convergence analysis of our SUPER-ADAM ($\tau = 1$) algorithm when using the momentum-based variance reduced gradient estimator [7, 22].

**Assumption 4.** *Each component function $f(x;\xi)$ is L-smooth for all $\xi \in \mathcal{D}$, such that*

$$\|\nabla f(x;\xi) - \nabla f(y;\xi)\| \leq L\|x - y\|, \ \forall x, y \in \mathcal{X}.$$

Assumption 4 is widely used in the variance-reduced algorithms [9, 7]. According to Assumption 4, we have $\|\nabla f(x) - \nabla f(y)\| = \|\mathbb{E}[\nabla f(x;\xi) - \nabla f(y;\xi)]\| \leq \mathbb{E}\|\nabla f(x;\xi) - \nabla f(y;\xi)\| \leq L\|x - y\|$ for all $x, y \in \mathcal{X}$. So the function $f(x)$ is $L$-smooth.

**Theorem 1.** *In Algorithm 1, under the Assumptions (**1,2,3,4**), given $\tau = 1$, $\mu_t = \frac{k}{(m+t)^{1/3}}$ and $\alpha_{t+1} = c\mu_t^2$ for all $t \geq 0$, $0 < \gamma \leq \frac{\rho m^{1/3}}{4kL}$, $\frac{1}{k^3} + \frac{10L^2\gamma^2}{\rho^2} \leq c \leq \frac{m^{2/3}}{k^2}$, $m \geq \max\big(\frac{3}{2}, k^3, (ck)^3, \frac{8^{3/2}}{(3k)^{3/2}}\big)$ and $k > 0$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\mathcal{M}_t\big] \leq \frac{2\sqrt{2G}m^{1/6}}{T^{1/2}} + \frac{2\sqrt{2G}}{T^{1/3}}, \tag{21}$$

*where $G = \frac{\rho(f(x_1)-f^*)}{k\gamma} + \frac{m^{1/3}\rho^2\sigma^2}{8k^2L^2\gamma^2} + \frac{\rho^2k^2c^2\sigma^2}{4L^2\gamma^2}\ln(m+T)$.*

**Remark 1.** *Given $k = O(1)$ and $m = O(1)$, we have $G = O\big(\ln(T)\big)$. Thus, our SUPER-ADAM ($\tau = 1$) algorithm has convergence rate of $\tilde{O}\big(\frac{1}{T^{1/3}}\big)$. Consider $\frac{1}{T^{1/3}} \leq \epsilon$, we have $T \geq \epsilon^{-3}$. Since our SUPER-ADAM algorithm requires one sample to estimate the stochastic gradient $g_t$ at each iteration, and needs $T$ iterations. Thus, our SUPER-ADAM ($\tau = 1$) algorithm has a sample complexity of $1 \cdot T = \tilde{O}(\epsilon^{-3})$ for finding an $\epsilon$-stationary point of the problem* (1).

### 4.4 Convergence Analysis of SUPER-ADAM ($\tau = 0$)

In this subsection, we provide the convergence analysis of our SUPER-ADAM ($\tau = 0$) algorithm when using the basic momentum stochastic gradient estimator [14].

**Assumption 5.** *The function $f(x) = \mathbb{E}_\xi[f(x;\xi)]$ is L-smooth such that*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \ \forall x, y \in \mathcal{X}.$$

Assumption 5 is widely used in the adaptive algorithms [26, 4, 29]. Clearly, Assumption 5 is milder than Assumption 4.

**Theorem 2.** *In Algorithm 1, under the Assumptions (**1,2,3,5**), given $\tau = 0$, $\mu_t = \frac{k}{(m+t)^{1/2}}$, $\alpha_{t+1} = c\mu_t$ for all $t \geq 0$, $0 < \gamma \leq \frac{\rho m^{1/2}}{8Lk}$, $\frac{8L\gamma}{\rho} \leq c \leq \frac{m^{1/2}}{k}$, $m \geq \max\{k^2, (ck)^2\}$ and $k > 0$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\mathcal{M}_t\big] \leq \frac{2\sqrt{2M}m^{1/4}}{T^{1/2}} + \frac{2\sqrt{2M}}{T^{1/4}},$$

*where $M = \frac{\rho(f(x_1)-f^*)}{\gamma k} + \frac{\rho\sigma^2}{\gamma kL} + \frac{2\rho m\sigma^2}{\gamma kL}\ln(m+T)$.*

**Remark 2.** *Given $k = O(1)$ and $m = O(1)$, we have $M = O\big(\ln(T)\big)$. Thus, our SUPER-ADAM ($\tau = 0$) algorithm has convergence rate of $\tilde{O}\big(\frac{1}{T^{1/4}}\big)$. Consider $\frac{1}{T^{1/4}} \leq \epsilon$, we have $T \geq \epsilon^{-4}$. Since our SUPER-ADAM algorithm requires one sample to estimate the stochastic gradient $g_t$ at each iteration, and needs $T$ iterations. Thus, our SUPER-ADAM ($\tau = 0$) algorithm has a sample complexity of $1 \cdot T = \tilde{O}(\epsilon^{-4})$ for finding an $\epsilon$-stationary point of the problem* (1).

## 5 Differences between Our Algorithm and Related Algorithms

In this section, we show some significance differences between our algorithm and the most related algorithms, i.e., STORM algorithm [7] and ADAM-type algorithms [14, 19, 29]. Although our SUPER-ADAM ($\tau = 1$) algorithm uses the same stochastic gradient estimator used in the STORM, there exist some significance differences besides using different learning rates:

1) Our algorithm focuses on both constrained and unconstrained optimizations, but STORM only focuses on unconstrained optimization. Since we only assume constrained set is convex, our results are also suitable to unconstrained optimization.

2) In our algorithm, we introduce a weighted solution $x_{t+1}$ at the step 12 to effectively incorporate variance reduced and momentum techniques. Under this case, our SUPER-ADAM algorithm can flexibly collaborate with various adaptive learning rates at the step 11. In fact, this is one of important novelties of our paper. However, the STORM only uses a simple gradient descent iteration.

3) In our theoretical analysis, our algorithm uses a new convergence metric defined in (15) for both constrained and unconstrained optimizations. While the STORM uses a common metric $\mathbb{E}\|\nabla f(x_t)\|$ only for unconstrained optimization.

Similarly, although our SUPER-ADAM ($\tau = 0$) algorithm use the same stochastic gradient estimator used in these ADAM-type algorithms, there exist some significance differences besides using different adaptive learning rates: These ADAM-type algorithms use a decreasing learning rate $\eta_t = \frac{\eta}{\sqrt{t}}$ (Please see the above (3), (4) and (8)) besides an adaptive learning rate, while our algorithm only use a constant learning rate $\gamma$ besides an adaptive learning rate. Moreover, we introduces a weighted solution $x_{t+1}$ (at the step 12) with a decreasing weighted parameter $\mu_t = \frac{k}{\sqrt{m+t}}$ (Please see Theorem 2) and use a decreasing parameter $\alpha_{t+1} = c\mu_t$ in our gradient estimator, while these ADAM-type algorithms only use a constant parameter $\alpha_1$ in their gradient estimator. Under this case, our algorithm use these decreasing parameters $\mu_t$ and $\alpha_{t+1}$ to control the noises in our gradient estimator, so our algorithm do not require some additional assumptions such as the bounded (stochastic) gradient assumption in our convergence analysis (Please see Table 1). For example, when $\tau = 0$, our gradient estimator is $g_{t+1} = \alpha_{t+1}\nabla f(x_{t+1};\xi_{t+1}) + (1-\alpha_{t+1})g_t$. Intuitively, with growing $t$, $\alpha_{t+1} = \frac{ck}{\sqrt{m+t}}$ will become small, so the noises added in our gradient estimator $g_{t+1}$ will also become less.

# 6 Numerical Experiments

In this section, we conduct some experiments to empirically evaluate our SUPER-ADAM algorithm on two deep learning tasks as in [17]: **image classification** on CIFAR-10, CIFAR-100 and Image-Net datasets and **language modeling** on Wiki-Text2 dataset (Please see Table 2). In the experiments, we compare our SUPER-ADAM algorithm against several state-of-the-art adaptive gradient algorithms, including: (1) Adam [14], (2) Amsgrad [19], (3) AdaGrad-Norm [15], (4) ADAM$^+$ [17], (5) STORM [7] and (6) AdaBelief [29]. For SUPER-ADAM, we consider two cases $\tau = 1$ and $\tau = 0$. All experiments are run over a machine with Intel Xeon E5-2683 CPU and 4 Nvidia Tesla P40 GPUs.

Table 2: Summary of setups in the experiments.

| *Task* | *Architecture* | *Dataset* |
|---|---|---|
| Image Classification | ResNet18 | CIFAR-10 |
| Image Classification | VGG19 | CIFAR-100 |
| Image Classification | ResNet34 | Image-Net |
| Language Modeling | Two-layer LSTM | Wiki-Text2 |
| Language Modeling | Transformer | Wiki-Text2 |

## 6.1 Image Classification Task

In the experiment, we conduct image classification task on CIFAR-10, CIFAR-100 and Image-Net datasets. We show results for CIFAR-10 and CIFAR-100 in the main-text and for Image-net in the Appendix due to space limitation. Specifically, we perform training over ResNet-18 [12] and VGG-19 [20] on CIFAR-10 and CIFAR-100 datasets. For all the optimizers, we set the batch size as 128 and trains for 200 epochs. For the learning rates and other hyper-parameters, we do grid search and report the best one for each optimizer. In Adam,Amsgrad and AdaBelief, we set the learning rate as 0.001. In AdaGrad-Norm, the best learning rate is 17 for CIFAR-10 and 10 for CIFAR-100, respectively. In Adam$^+$, we use the recommended tuning parameters in [17]. In STORM, the best result is obtained when $w = 6$, $k = 10$ and $c = 100$ for CIFAR-10, while $w = 3$, $k = 10$ and $c = 100$ for CIFAR-100. For our SUPER-ADAM algorithm. For both CIFAR-10 and CIFAR-100 data-set, we set $k = 1$, $m = 100$, $c = 40$, $\gamma = 0.001$ when $\tau = 1$, and $k = 1$, $m = 100$, $c = 20$, $\gamma = 0.001$ when $\tau = 0$;

In Figures 1 and 2, we show both train and test errors and accuracy results of the CIFAR-10 and CIFAR-100 datasets, respectively. Our algorithm consistently outperforms the other optimizers with a great margin, especially when we set $\tau = 1$. When we set $\tau = 0$, our SUPER-ADAM algorithm gets the comparable performances with Adam/AmsGrad.

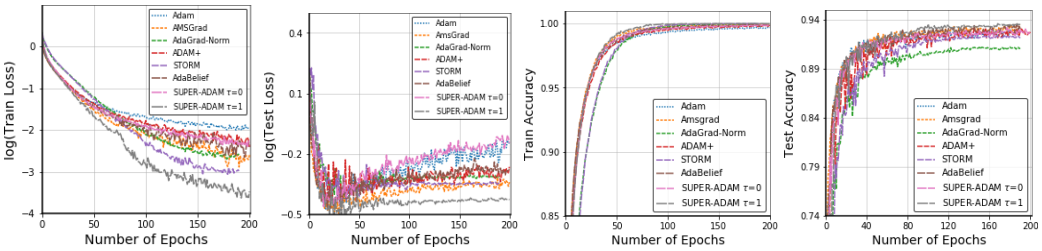


Figure 1: Experimental Results of CIFAR-10 by Different Optimizers over ResNet-18.

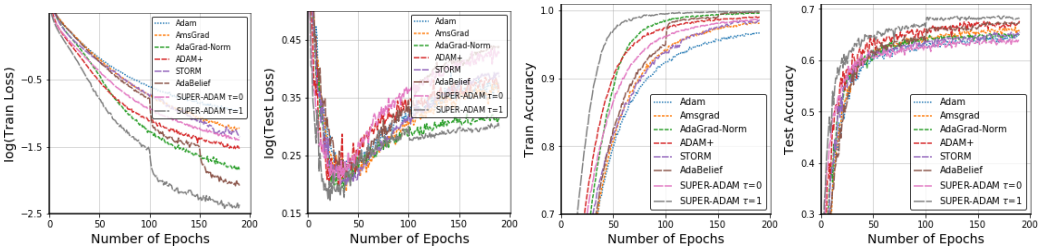


Figure 2: Experimental Results of CIFAR-100 by Different Optimizers over VGG-19.

### 6.2 Language Modeling Task

In the experiment, we conduct language modeling task on the Wiki-Text2 dataset. Specifically, we train a 2-layer LSTM [13] and a 2-layer Transformer over the WiKi-Text2 dataset. We present the results trained with LSTM here and defer the results for Transformer to the Appendix. For the LSTM, we use 650 dimensional word embeddings and 650 hidden units per-layer. What's more, we set the batch size as 20 and trains for 40 epochs with dropout rate 0.5. We also clip the gradients by norm 0.25 in case of the exploding gradient in LSTM. We also decrease the learning by 4 whenever the validation error increases. For the learning rate, we also do grid search and report the best one for each optimizer. In Adam and Amsgrad algorithms, we set the learning rate as 0.001 in LSTM In AdaGrad-Norm algorithm, the best learning rate is 40. In Adam$^+$ algorithm, we use the learning rate 20. In AdaBelief algorithm, we set the learing rate 0.1. In STORM algorithm, we set $w = 50$, $k = 10$ and $c = 100$. In our SUPER-ADAM algorithm, we set $k = 1$, $m = 100$, $c = 40$, $\gamma = 0.001$ when $\tau = 1$, while $k = 1$, $m = 100$, $c = 20$, $\gamma = 0.01$ when $\tau = 0$;

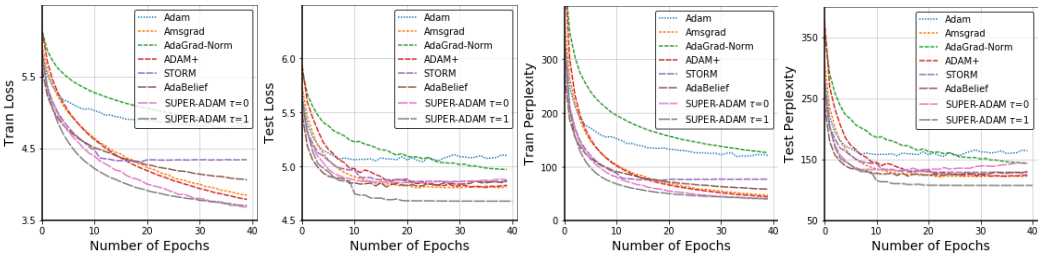


Figure 3: Experimental Results of WikiText-2 by Different Optimizers over LSTM.

In Figure 3, we show both train and test perplexities (losses) for different optimizers. When $\tau = 1$, our algorithm outperforms all the other optimizers. When $\tau = 0$, our optimizer gets a comparable performance with Adam.

## 7 Conclusions

In the paper, we proposed a new faster and universal adaptive gradient framework, which introduces a universal adaptive matrix to include most existing adaptive gradient forms and can also flexibly work with the momentum and variance reduced techniques. Moreover, we provided a novel convergence analysis framework for the adaptive gradient methods under the nonconvex setting. Experimental studies were conducted on both image classification and language modeling tasks, and all empirical results verify the superior performance of our proposed algorithm.

# References


[1] Y. Arjevani, Y. Carmon, J. C. Duchi, D. J. Foster, N. Srebro, and B. Woodworth. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

[2] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

[3] J. Chen, D. Zhou, Y. Tang, Z. Yang, and Q. Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018.

[4] X. Chen, S. Liu, R. Sun, and M. Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[5] Z. Chen, Y. Xu, E. Chen, and T. Yang. Sadagrad: Strongly adaptive stochastic gradient methods. In *International Conference on Machine Learning*, pages 913–921. PMLR, 2018.

[6] A. Cutkosky and H. Mehta. Momentum improves normalized sgd. *arXiv preprint arXiv:2002.03305*, 2020.

[7] A. Cutkosky and F. Orabona. Momentum-based variance reduction in non-convex sgd. In *Advances in Neural Information Processing Systems*, pages 15210–15219, 2019.

[8] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[9] C. Fang, C. Li, Z. Lin, and T. Zhang. Near-optimal non-convex optimization via stochastic path integrated differential estimator. *Advances in Neural Information Processing Systems*, 31:689, 2018.

[10] S. Ghadimi, G. Lan, and H. Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305, 2016.

[11] D. Goldfarb, Y. Ren, and A. Bahamou. Practical quasi-newton methods for training deep neural networks. *arXiv preprint arXiv:2006.08877*, 2020.

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

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] X. Li and F. Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 983–992. PMLR, 2019.

[16] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[17] M. Liu, W. Zhang, F. Orabona, and T. Yang. Adam+: A stochastic method with adaptive variance reduction. *arXiv preprint arXiv:2011.11985*, 2020.

[18] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[19] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[21] M. Staib, S. Reddi, S. Kale, S. Kumar, and S. Sra. Escaping saddle points with adaptive gradient methods. In *International Conference on Machine Learning*, pages 5956–5965. PMLR, 2019.

[22] Q. Tran-Dinh, N. H. Pham, D. T. Phan, and L. M. Nguyen. Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization. *arXiv preprint arXiv:1905.05920*, 2019.

[23] X. Wang, S. Ma, D. Goldfarb, and W. Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[24] R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pages 6677–6686. PMLR, 2019.

[25] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in neural information processing systems*, pages 4148–4158, 2017.

[26] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, pages 9793–9803, 2018.

[27] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[28] Q. Zhang, F. Huang, C. Deng, and H. Huang. Faster stochastic quasi-newton methods. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[29] J. Zhuang, T. Tang, Y. Ding, S. C. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in Neural Information Processing Systems*, 33, 2020.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Please see Section 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] Please see the supplemental material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]