MaxVA: Fast Adaptation of Step Sizes by Maximizing Observed Variance of Gradients

Chen Zhu[1], Yu Cheng[2], Zhe Gan[2], Furong Huang[1],
Jingjing Liu[3], and Tom Goldstein[1]

[1] University of Maryland {chenzhu,furongh,tomg}@umd.edu
[2] Microsoft {yu.cheng,zhe.gan,jingjl}@microsoft.com
[3] Tsinghua University jjliu@air.tsinghua.edu.cn

Abstract. Adaptive gradient methods such as RMSProp and Adam use exponential moving estimate of the squared gradient to compute adaptive step sizes, achieving better convergence than SGD in face of noisy objectives. However, Adam can have undesirable convergence behaviors due to unstable or extreme adaptive learning rates. Methods such as AMSGrad and AdaBound have been proposed to stabilize the adaptive learning rates of Adam in the later stage of training, but they do not outperform Adam in some practical tasks such as training Transformers [38]. In this paper, we propose an adaptive learning rate principle, in which the running mean of squared gradient in Adam is replaced by a weighted mean, with weights chosen to maximize the estimated variance of each coordinate. This results in a faster adaptation to the local gradient variance, which leads to more desirable empirical convergence behaviors than Adam. We prove the proposed algorithm converges under mild assumptions for nonconvex stochastic optimization problems, and demonstrate the improved efficacy of our adaptive averaging approach on machine translation, natural language understanding and large-batch pretraining of BERT.

1 Introduction

Stochastic Gradient Descent (SGD) and its variants are commonly used for training deep neural networks because of their effectiveness and efficiency. In their simplest form, gradient methods train a network by iteratively moving each parameter in the direction of the negative gradient (or the running average of gradients) of the loss function on a randomly sampled mini-batch of training data. A scalar learning rate is also applied to control the size of the update. In contrast, *adaptive* stochastic gradient methods use coordinate-specific learning rates, which are inversely proportional to the square root of the running mean of squared gradients [37,12,19]. Such methods are proposed to improve the stability of SGD on non-stationary problems, and have achieved success in different fields across Speech, Computer Vision, and Natural Language Processing.

Large pretrained Transformer-based language models have achieved remarkable successes in various language tasks [10,25,20,31,5]. The original Transformer architecture (Post-LN Transformers) often demonstrates better performance than its Pre-LN variant [24], but its gradient has high variance during training. A warmup learning rate schedule or small initial adaptive learning rates [23] are required for its convergence. [49] shows that SGD fails to train Transformers without gradient clipping, and adaptivity is important for stabilizing optimization under the heavy-tailed noise in Transformer's gradients. This indicates that the strategy of AdaBound [27], which is to transition from Adam into SGD, may fail on Post-LN Transformers (see Appendix D for instance). However, the adaptive learning rate of Adam can be unstable in the later stage of training, and such instability sometimes leads to sub-optimal solutions or even non-convergent behavior on some simple problems [33,27]. AMSGrad [33] was proposed to deal with this issue by computing the adaptive learning rate with an update rule that guarantees monotonically decaying adaptive learning rates for each coordinate, but to our knowledge, it has not been widely deployed to enhance Adam for training Transformer-based language models.

In this work, we explore a different approach to improving the stability of adaptive learning rates. We propose *Maximum Variation Averaging* (MaxVA), which computes the running average of squared gradients using dynamic, rather than constant, coordinate-wise weights. These weights are chosen so that the estimated variance of gradients is maximized, to enable a faster adaptation to the changing variance of gradients. The MaxVA weights for maximizing this variance have a simple closed-form solution that requires little storage or computational cost. With MaxVA, the adaptive optimizer 1) takes a smaller step size when abnormally large gradient is present, to improve stability; 2) takes a larger step size when abnormally small gradient is prevent, to avoid spurious minima and achieve better generalization [22]; 3) takes a steady step size when gradients are stable and within estimated deviation, to ensure convergence [33]. In the large-batch setting of BERT pretraining, where the total number of iterations is sharply reduced and a faster adaptation in each step is more important, MaxVA achieves faster convergence and obtain models with better test performance on downstream tasks than both Adam and Lamb [45]. Extensive experiments on both synthetic and practical datasets demonstrate that MaxVA leads to an improved adaptability and stability for Adam, yielding better test set performance than Adam on a variety of tasks. We also prove MaxVA converges under mild assumptions in the nonconvex stochastic optimization setting.

2 Preliminary and Definitions

By default, all vector-vector operators are element-wise in the following sections. Let $\theta \in \mathbb{R}^d$ be the parameters of the network to be trained, $\ell(x;\theta)$ is the loss of the model with parameters θ evaluated at x. Our goal is to minimize the expected risk on the data distribution defined as:

$$f(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[\ell(x;\theta)\right]. \tag{1}$$

In most deep learning problems, only a finite number of potentially noisy samples can be used to approximate Eq. 1, and the gradients are computed on randomly sampled minibatches during training. Stochastic regularizations such as Dropout [36] are commonly used for training Transformer-based language models [38,51], which further adds to the randomness of the gradients. Thus, it is important to design optimizers that tolerate noisy gradients. ADAM [19] is an effective optimizer that adapts to such noisy gradients. It keeps exponential moving averages m_t and v_t of past gradients $g_1, ..., g_{t-1}$, defined as:

$$\tilde{m}_t = \alpha \tilde{m}_{t-1} + (1-\alpha) g_t, \quad m_t = \frac{\tilde{m}_t}{1-\alpha^{t+1}},$$
$$\tilde{v}_t = \beta \tilde{v}_{t-1} + (1-\beta) g_t^2, \quad v_t = \frac{\tilde{v}_t}{1-\beta^{t+1}},$$

where $\alpha, \beta \in [0,1]$, $g_t = \nabla_\theta \ell(x_t; \theta_t)$ is the gradient of the t-th minibatch x_t, $\tilde{m}_0 = \tilde{v}_0 = 0$, and m_t, v_t corrects this zero-initialization bias of $\tilde{m}_t, \tilde{v}_t$ [19]. ADAM updates the parameters with the estimated moments as $\theta_{t+1} = \theta_t - \eta_t \frac{m_t}{\sqrt{v_t}+\epsilon}$, where $\epsilon > 0$ is a small constant for numerical stability.

If we assume that the distribution of the stochastic gradient is constant within the effective horizon of the running average, then m_t and v_t will be estimates of the first and second moments of the gradient g_t [2]. Same as other adaptive methods such as ADAM and the recently proposed AdaBelief [52], we adopt this assumption throughout training. With this assumption, at time t, we assume $\mathbb{E}[m_t] \approx \nabla f_t$, $\mathbb{E}[v_t] \approx \nabla f_t^2 + \sigma_t^2$, where σ_t^2 is the variance of g_t. ADAM, RMSPROP and other variants that divide the update steps by $\sqrt{v_t}$ can be seen as adapting to the gradient variance under this assumption when m_t is small. These adaptive methods take smaller step sizes when the estimated variance $\sigma_t^2 = v_t - m_t^2$ is high. Higher local gradient variance indicates higher local curvature, and vice versa. In certain quadratic approximations to the loss function, this variance is proportional to the curvature [34] (Eq. 12 of our paper). Therefore, like a diagonal approximation to Newton's method, such adaptative learning rates adapt to the curvature and can accelerate the convergence of first-order methods.

However, the adaptive learning rate $\eta_t/(\sqrt{v_t}+\epsilon)$ of ADAM and RMSPROP can take extreme values, causing convergence to undesirable solutions [42,8]. [33] gave one such counter example where gradients in the correct direction are large but occur at a low frequency, and ADAM converges to the solution of maximum regret. They solve this issue by keeping track of the maximum v_t for each coordinate throughout training with a new variable $\hat{v}_t$, and replace the adaptive learning rate with $\eta_t/\sqrt{\hat{v}_t}$ to enforce monotonically descreasing learning rates. Extremely small adaptive learning rates can also cause undesirable convergence behavior, as demonstrated by a counter example from [27].

3 Maximizing the Variance of Running Estimations

Motivation. We propose to mitigate the undesirable convergence issue of ADAM by changing the constant running average coefficient β for the second moment into an adaptive one. The idea is to allow β_t to adopt the value that maximizes

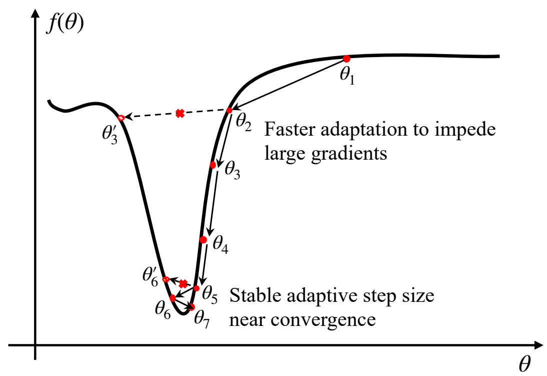

Fig. 1. An illustrative example of MaxVA. It adopts a smaller adaptive learning rate when an abnormally large gradient appears at θ_2 by choosing a smaller β, to prevent overshooting to θ'_3. When the gradient is more stable near convergence at θ_5, it uses a larger β for the slowest change in adaptive learning rate, to prevent overshooting to θ'_6.

the estimated variance of the gradient at each iteration t. As a result, our method will assign a higher coefficient $(1-\beta_t)$ to g_t when it deviates too much from the estimated mean, resulting in a smaller step size when the gradient is too large to avoid overshooting, and a larger step size when the gradient is abnormally small to avoid spurious local minima. By contrast, if the gradient is stable and close to the estimated mean, which often happens near a flat minimum, our method will assign minimum coefficient to g_t to maintain the value of v_t and take steady steps towards the minimum. Therefore, our method can use β_t as the adaptive running average coefficient to take steps that are cautious enough to avoid instability and spurious minima but aggressive enough to make progress. An illustrative example is given in Figure 1.

Maximum Variation Averaging. Formally, we estimate the variance of the gradient at each coordinate by keeping track of the zeroth, first, and second moments of the gradient as functions of the adaptive running average coefficient β_t, denoted as $w_t(\beta_t)$, $\tilde{u}_t(\beta_t)$ and $\tilde{v}_t(\beta_t)$, respectively:

$$w_t(\beta_t) = \beta_t w_{t-1}(\beta_{t-1}) + (1-\beta_t), \tag{2}$$

$$\tilde{u}_t(\beta_t) = \beta_t \tilde{u}_{t-1}(\beta_{t-1}) + (1-\beta_t) g_t, \tag{3}$$

$$\tilde{v}_t(\beta_t) = \beta_t \tilde{v}_{t-1}(\beta_{t-1}) + (1-\beta_t) g_t^2. \tag{4}$$

The zeroth moment $w_t(\beta_t)$ is used to normalize $\tilde{u}_t(\beta_t)$ and $\tilde{v}_t(\beta_t)$ to achieve bias-corrected estimates $u_t(\beta_t) = \tilde{u}_t(\beta_t)/w_t(\beta_t)$ and $v_t(\beta_t) = \tilde{v}_t(\beta_t)/w_t(\beta_t)$ for the first and second moments, so that the estimates are not biased towards zero ($\tilde{m}_0 = \tilde{v}_0 = 0$) [19].

Under our assumptions, the bias-corrected local estimate of the gradient variance is $\sigma_t^2 = \tilde{v}_t(\beta_t)/w_t(\beta_t) - [\tilde{u}_t(\beta_t)/w_t(\beta_t)]^2$. Taking the arg max for σ_t^2, we find the β_t that achieves the maximal variance for each coordinate i:

$$\beta_{t,i} = \arg\max_{\beta} \sigma_{t,i}^2 = \arg\max_{\beta} v_{t,i}(\beta) - [u_{t,i}(\beta)]^2. \tag{5}$$

We call our approach to finding adaptive running average coefficient β_t *Maximum Variation Averaging* (MaxVA). We plug MaxVA into Adam and its variant LaProp [53], which results in two novel algorithms, MAdam and LaMAdam, listed in Algorithm 1 and Algorithm 2 (in the Appendix). Different from Adam, LaProp uses v_t to normalize the gradients before taking the running average, which results in higher empirical stability under various hyperparameters. Note, we only apply the adaptive β_t to the *second* moment $u_t(\beta_t)$ used for scaling the learning rate; m_t is still an exponential moving average *with a constant coefficient* α of the gradient for MAdam or the normalized gradient for LaMAdam.

Algorithm 1 MAdam

Input: Learning rate $\{\eta_t\}_{t=1}^T$, parameter $0 < \alpha < 1$, $0 < \underline{\beta} < \bar{\beta} < 1$, $\epsilon > 0$
Set $\tilde{m}_0 = \tilde{u}_0 = \tilde{v}_0 = w_0 = 0$
for $t = 1$ **to** T **do**
  Draw samples S_t from training set
  Compute $g_t = \frac{1}{|S_t|}\sum_{x_k \in \mathcal{S}_t} \nabla \ell(x_k; \theta_t)$
  $\tilde{m}_t = \alpha \tilde{m}_{t-1} + (1-\alpha) g_t$
  $\tilde{\beta}_t = \arg\max_{\beta} v_t(\beta) - u_t^2(\beta)$ ▷ see Eq 6
  $\beta_t = \max(\underline{\beta}, \min(\bar{\beta}, \tilde{\beta}_t))$
  $\tilde{u}_t = \beta_t \tilde{u}_{t-1} + (1-\beta_t) g_t$
  $\tilde{v}_t = \beta_t \tilde{v}_{t-1} + (1-\beta_t) g_t^2$
  $w_t = \beta_t w_{t-1} + (1-\beta_t)$
  $\theta_t = \theta_{t-1} - \eta_t \frac{\sqrt{w_t}}{1-\alpha^t} \frac{\tilde{m}_t}{\sqrt{\tilde{v}_t}+\epsilon}$

Finding β_t via a Closed-form Solution. The maximization for β_t in Eq. 5 is quadratic and has a relatively simple closed-form solution that produces maximal σ_t^2 for each coordinate:

$$\beta_t = \frac{\Delta g_t^2 + \sigma_{t-1}^2}{w_{t-1}(\Delta g_t^2 - \sigma_{t-1}^2) + \Delta g_t^2 + \sigma_{t-1}^2}, \tag{6}$$

where all variables are vectors and all the operations are elementwise, $\Delta g_t = (g_t - u_{t-1})$ is the deviation of the gradient g_t from the estimated mean u_{t-1}, $\sigma_{t-1}^2 = v_{t-1} - u_{t-1}^2$ is the estimated variance, and we have abbreviated $u_{t-1}(\beta_{t-1})$, $v_{t-1}(\beta_{t-1})$ and $w_{t-1}(\beta_{t-1})$ into u_{t-1}, v_{t-1} and w_{t-1}. We use this abbreviation in the following sections, and defer the derivation of Eq. 6 to Appendix A.

Implementation Notes. We apply MaxVA in every step except for the first step, where the gradient variance one can observe is zero. So for Algorithm 1 and Algorithm 2 we define:

$$\tilde{u}_1 = (1-\beta_1) g_1, \tilde{v}_1 = (1-\beta_1) g_1^2, w_1 = 1 - \beta_1. \tag{7}$$

The coefficient β_1 for $t = 1$ is set to a constant that is the same as typical values for ADAM. To obtain a valid running average, we clip β_t so that $\underline{\beta} \leq \beta_t \leq \bar{\beta}$, where the typical values are $\underline{\beta} = 0.5, 0.98 \leq \bar{\beta} \leq 1$. For convenience, we set $\beta_1 = \bar{\beta}$ by default. For $t > 1$, since $0 < \beta_t \leq 1$, w_t will monotonically increase from $(1-\beta_1)$ to 1. Before clipping, for any g_t, u_{t-1}, v_{t-1} satisfying $v_{t-1} - u_{t-1}^2 > 0$ in Eq. 6, we have $\beta_t \in [1/(1+w_{t-1}), 1/(1-w_{t-1})]$. As a result, the lower bound that we use ($\underline{\beta} = 0.5$) is tight and does not really change the value of β_t, and as $t \to \infty$, $w_t \to 1$ and $\beta_t \in [0.5, \infty]$. We have a special case at $t = 2$, where β_t is a constant $1/(2 - \beta_1)$.

In practice, we also add a small coefficient $\delta > 0$ to the denominator of Eq. 6 to prevent division by zero, which will have negligible effect on the value of β_t and does not violate the maximum variation objective (Eq. 5). All the derivations for these conclusions are deferred to Appendix C.

Effect of Maximum Variation Averaging. By definition, we have $\sigma_{t-1}^2 \geq 0$, but in most cases $\sigma_{t-1}^2 > 0$. When $\sigma_{t-1}^2 > 0$, we define a new variable $R_t = \Delta g_t^2 / \sigma_{t-1}^2$, which represents the degree of deviation of gradient g_t from the current estimated average. Then, we can rewrite:

$$\beta_t = \frac{R_t + 1}{(1 + w_t)R_t + 1 - w_t}. \tag{8}$$

From Eq. 8, we can see β_t monotonically decreases from $1/(1-w_t)$ to $1/(1+w_t)$ as R_t increases from 0 to ∞, and equals to 1 when $R_t = 1$. As a result, for each coordinate, if $R_t \gg 1$, g_t deviates much more than σ_{t-1} from u_{t-1}, and MaxVA will find a smaller β_t and therefore a higher weight $(1 - \beta_t)$ on g_t^2 to adapt to the change faster. This helps to avoid overshooting when abnormally large gradient is present (see Figure 1), and avoids spurious sharp local minima where gradients are abnormally small. With a faster response to abnormal gradients, MaxVA is better at handling the heavy-tailed distribution of gradients in the process of training Transformers [49]. In practice, v_t tends to be larger than ADAM/LAPROP using a constant $\bar{\beta}$, but as we will show in the experiments, using a larger learning rate counters such an effect and achieves better results.

On the other hand, if $R_t < 1$, or the deviation of the gradient g_t from the current running mean u_{t-1} is within the estimated standard deviation σ_{t-1}, we will use $\bar{\beta}$ to update $\tilde{v}_t$, which is the smallest change we allow for $\tilde{v}_t$. This tends to happen in the later phase of training, where the gradient variance decreases. MaxVA will adopt a steady step towards convergence by finding the slowest rate to update $\tilde{v}_t$. This allows large values of $\tilde{v}_t$ to last for a longer horizon even compared with setting β_t to a constant $\bar{\beta}$ on the same sequence, since we have assigned more mass to large gradients, which can be seen as an adaptive version of AMSGRAD. Note that MaxVA and AMSGRAD can be complementary approaches if applied together, which we have found helpful for Image Classification on CIFAR10/100.

Convergence Analysis. We prove the convergence of MaxVA in the nonconvex stochastic optimization setting. For the sake of simplicity, we analyze the case where $\alpha = 0$, which is effectively applying MaxVA to RMSPROP. We leave the analysis for $\alpha \neq 0$ for future research. We assume the function ℓ is L-smooth in θ, i.e., there exists a constant L such that for all $\theta_1, \theta_2 \in \mathbb{R}^d, x \in \mathcal{X}$,

$$\|\nabla_\theta \ell(x; \theta_1) - \nabla_\theta \ell(x; \theta_2)\| \leq L\|\theta_1 - \theta_2\|. \tag{9}$$

This automatically implies that $f(\theta) = \mathbb{E}[\ell(x; \theta)]$ is L-smooth. Such a smoothness assumption holds for networks with smooth activation functions, e.g., Transformers that use the GELU activation [17]. We also need to assume function ℓ has bounded gradient, i.e., $\|\nabla_\theta \ell(x; \theta)\|_\infty \leq G$ for all $\theta \in \mathbb{R}^d, x \in \mathcal{X}$. As typically used in the analysis of stochastic first-order methods [46,13], we assume the stochastic gradient has bounded variance: $\mathbb{E}[[\nabla_\theta \ell(x; \theta)]_i - [\nabla_\theta f(\theta)]_i]^2 \leq \sigma^2$ for all $\theta \in \mathbb{R}^d$. Further, we assume the batch size increases with time as $b_t = t$, which is also adopted in the analysis of signSGD [4], and holds in our large batch experiments. Theorem 1 gives a "worst-case" convergence rate of MaxVA to a stationary point under these assumptions, where the dependence of β_t on g_t is ignored and we only consider the worst-case of β_t in each step. The proof is given in Appendix B.

Theorem 1. *Define $w_0 = 1$. Let $\eta_t = \eta$ and $b_t = t$ for all $t \in [T]$. Furthermore, we assume $\epsilon, \underline{\beta}, \bar{\beta}, \eta$ are chosen such that $\eta \leq \frac{\epsilon}{2L}$, $1 - \underline{\beta} \leq \frac{\epsilon^2}{16G^2}$, and $\bar{\beta} \leq 2\underline{\beta}$. Then for θ_t generated using MAdam, we have the following bound:*

$$\mathbb{E}\|\nabla f(\theta_a)\|^2 \leq O\left(\frac{f(\theta_1) - f(\theta^*)}{\eta T} + \frac{2\sigma d G}{\epsilon\sqrt{T}}\right), \tag{10}$$

where θ^* is an optimal solution to minimize the objective in Eq. 1, and θ_a is an iterate uniformly randomly chosen from $\{\theta_1, ..., \theta_T\}$.

4 Experiments on Synthetic Data

For a quantitative control of the stochasticity and data distribution, which affects the difficulty of the problem and the efficacy of the optimizers, we compare MAdam and the baselines in two sets of synthetic data, and demonstrate the efficacy of MaxVA with statistical significance on a large number of instances. The first dataset simulates prevalent machine learning settings, where mini-batch stochastic gradient methods are applied on a finite set of samples, on which we show MAdam fixes the nonconvergence issue of Adam and achieves faster convergence rate than AMSGrad. The second dataset evaluates the algorithms under different curvatures and gradient noise levels, where we show MAdam achieves both lower loss and variance than fine-tuned Adam at convergence.

4.1 Convergence with Stochastic Gradients

Since MaxVA maximizes the variance and the gradient converges to zero in most cases, MAdam biases towards larger v_t than Adam but does not require v_t to

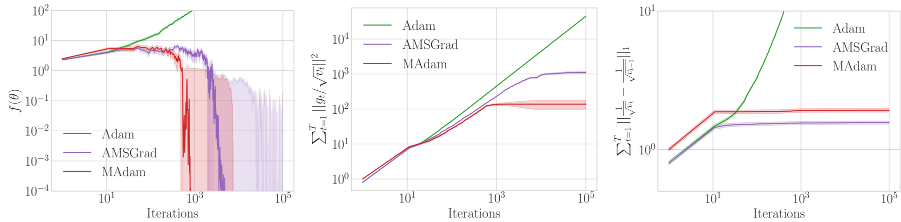

Fig. 2. Median and standard error (100 runs) of objective value ($f(\theta)$), accumulated update size ($\sum_{t=1}^{T}||g_t/\sqrt{v_t}||^2$) and total change in adaptive learning rate ($\sum_{t=1}^{T}||\frac{1}{\sqrt{v_t}}-\frac{1}{\sqrt{v_{t-1}}}||_1$) for ADAM, AMSGRAD, MADAM on the problem in Eq. 11.

be monotonically increasing, which is like an adaptive version of AMSGRAD. To highlight the difference, we compare ADAM, MADAM and AMSGRAD on the synthetic dataset from [8] simulating training with stochastic mini batches on a finite set of samples. Formally, let $\mathbb{1}_{[\cdot]}$ be the indicator function. We consider the problem $\min_\theta f(\theta) = \sum_{i=1}^{11} \ell_i(\theta)$ where

$$\ell_i(\theta) = \begin{cases} \mathbb{1}_{i=1} 5.5\theta^2 + \mathbb{1}_{i\neq 1}(-0.5\theta^2), & \text{if } |\theta| \leq 1; \\ \mathbb{1}_{i=1}(11|\theta| - 5.5) + \mathbb{1}_{i\neq 1}(-|\theta| + 0.5), & \text{otherwise.} \end{cases} \tag{11}$$

At every step, a random index i is sampled uniformly from $i \in [11]$, and the gradient $\nabla \ell_i(\theta)$ is used by the optimizer. The only stationary point where $\nabla f(\theta) = 0$ is $\theta = 0$. We set $\alpha = 0, \beta = 0.9$ for ADAM and AMSGRAD. For MADAM, we set $\alpha = 0, (\underline{\beta}, \bar{\beta}) = (0.5, 1)$. We select the best *constant* learning rates for the three algorithms, see Appendix E for details.

We plot the median and standard error of the objective ($f(\theta)$), accumulated update size ($S_1 = \sum_{t=1}^{T}||g_t/\sqrt{v_t}||^2$), and total change in adaptive step size ($S_2 = \sum_{t=1}^{T}||\frac{1}{\sqrt{v_t}} - \frac{1}{\sqrt{v_{t-1}}}||_1$) over 100 runs in Figure 11.The optimal learning rates for these optimziers are different, so for fair comparisons, we have ignored the constant learning rate in S_1 and S_2. From the curves of $f(\theta)$, we can see ADAM diverges, and MADAM converges faster than AMSGRAD in the later stage. As shown by the S_2 curves, the adaptive step sizes of MADAM and AMSGRAD all converged to some constant values after about 10 steps, but MADAM converges faster on both $f(\theta)$ and S_1, indicating the adaptive step size found by MADAM fits the geometry of the problem better than AMSGRAD. This also shows $S_1 + S_2$ of MADAM has a smaller slope than AMSGRAD in the log-scale plots after 10 iterations, leading to a faster theoretical convergence rate in the bound given by [8]. The slightly larger variation in adaptive step sizes of MADAM at the beginning of training, shown by the larger S_2 values, demonstrates MADAM adapts faster to the changing gradients than AMSGRAD, achieved by dynamically selecting $\beta < 0.9$.

4.2 Convergence in the Noisy Quadratic Model

We analyze the ability of MADAM to adapt to curvature and gradient noise on the simple but illustrative Noisy Quadratic Model (NQM), which has been

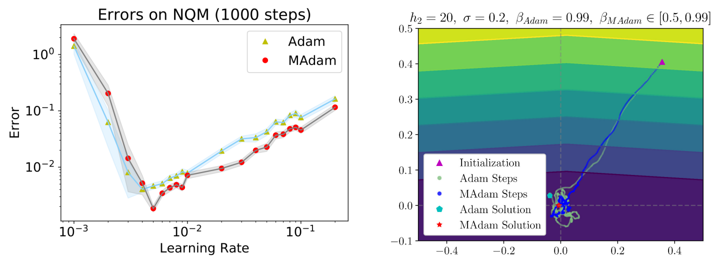

Fig. 3. Results on NQM. The left figure shows the mean and standard error of the loss under different learning rates η, computed over 100 runs at each point. We select the best β for Adam at each η. The best results (mean and variance) of Adam and MAdam are 1.84e-3 (2.51e-4) and 4.05e-3 (4.84e-4) respectively. Figure on the right gives a qualitative example of the trajectories of two approaches.

widely adopted for analyzing optimization dynamics [34,43,48,50]. The loss function is defined as $f(\theta) = \mathbb{E}_{x \sim \mathcal{N}(0,\sigma^2 I)} \left[\frac{1}{2} \sum_{i=1}^{d} h_i(\theta_i - x_i)^2 \right]$, where x is a noisy observation of the ground-truth parameter $\theta^* = 0$, simulating the gradient noise in stochastic optimization, and h_i represents the curvature of the system in d dimensions. In each step, the optimizers use the following noisy gradient for coordinate i, from which we can see the gradient's variance is proportional to the curvature h_i^2:

$$\nabla_{\theta_i} \ell(\sigma\epsilon_i; \theta_i) = h_i(\theta_i - \sigma\epsilon_i), \epsilon_i \sim \mathcal{N}(0, 1). \tag{12}$$

To validate the effectiveness of MaxVA, we compare MAdam with Adam under a variety of different curvatures h and noise level σ on an NQM with $d = 2$. For each setting of h and σ, we test both algorithms on a variety of learning rates. For Adam, we additionally choose the best β and report the best results. See Appendix F for details. We run each setting 100 times to report the mean and standard error. MAdam consistently achieves 30-40% lower average loss with smaller standard error in all settings. Figure 3 shows the results for one of the settings, from which we find the best result of MAdam is better than Adam under any choice of β and learning rate, confirming the advantage of MaxVA. From the qualitative example, MaxVA also demonstrates smaller variance near convergence, enabled by a quicker response to impede the noise with a smaller β_t. More experimental results under other settings are provided in Appendix F.

5 Experiments on Practical Datasets

In this section, we evaluate MAdam and LaMAdam on a variety of tasks against well-calibrated baselines: IWSLT'14 DE-EN/WMT'16 EN-DE for neural machine translation, the GLUE benchmark for natural language understanding, and pretraining the BERT-Base model. We also provide results on image classification. We use the decoupled weight decay [26] in all our experiments. Across all the plots in this section, we define the average step size at time t as the average

Model	CIFAR-10	CIFAR-100	ImageNet
SGD	95.44 (.04)	79.62 (.07)	70.18
Adam	95.37 (.03)	78.77 (.07)	66.54
LaProp	95.34 (.03)	78.36 (.07)	70.02
AdaBelief	95.30*	77.30*	70.08
MAdam (ours)	**95.51** (.09)	**79.32** (.08)	69.96
LaMAdam (ours)	95.38 (.11)	79.21 (.11)	**70.16**

Table 1. Comparing adaptive methods with exhaustively fine-tuned SGD on CIFAR10/100 and ImageNet. CIFAR10/100 experiments are the median (standard error) over 4 runs. *: The results of AdaBelief are from their paper [52] with a ResNet34, while our results are with ResNet18.

of $|\eta_t m_t/(\sqrt{v_t}+\epsilon)|$ for Adam/MAdam and $|\eta_t m_t|$ for LaProp/LaMAdam over all the entries.

5.1 Image Classification

To evaluate the effectiveness of MaxVA for image classification, we compare with SGD, Adam, LaProp [53] and AdaBelief [52] in training ResNet18 [16] on CIFAR10, CIFAR100 and ImageNet. On all the datasets, we perform a grid search for the learning rate and weight decay, and report the best results for each method in Table 1. For CIFAR10/100, we train ResNet18 with a batch size of 128 for 200 epochs. We also find AMSGrad [33] improves the classification accuracy of all adaptive methods evaluated on CIFAR10/100, so we apply AMSGrad in all experiments with adaptive methods. On ImageNet, we use the implementation from torchvision and the default multi-step learning rate schedule. We do not use AMSGrad in this case. Further details are in Appendix G.

Despite achieving a marginal improvement on CIFAR10, adaptive methods often underperforms carefully tuned SGD on CIFAR100 and ImageNet when training popular architectures such as ResNet, as confirmed by [42,50,23]. Nevertheless, with the proposed MaxVA, we shrink the gap between adaptive methods and carefully tuned SGD on these image classification datasets, and achieve top-1 accuracy very close to SGD on ImageNet. Note our results with ResNet18 is better than the recent AdaBelief's results with ResNet34 on CIFAR10/CIFAR100 (95.51/79.32 vs. 95.30/77.30 approximately), as well as AdaBelief with ResNet18 on ImageNet (70.16 vs. 70.08) [52].

5.2 Neural Machine Translation

We train Transformers from scratch with LaProp and LaMAdam on IWSLT'14 German-to-English (DE-EN) translation [6] and WMT'16 English-to-German (EN-DE) translation, based on the implementation of fairseq.[4] We do not compare with SGD, since it is unstable for Transformers [49]. We also show in Appendix D that AdaBound cannot achieve any good result without degenerating into Adam. More details are in Appendix H.

[4] https://github.com/pytorch/fairseq

Method	IWSLT'14 DE-EN	WMT'16 EN-DE
RAdam	35.51	-
AdaBelief	35.90	-
LaProp(ours)	35.98 (0.06)	27.02
LaMAdam(ours)	**36.09** (0.04)	**27.11**

Table 2. BLEU score for training transformers on machine translation datasets. We report the median and standard error for IWSLT'14 over 5 runs. Results of other meethods are from the AdaBelief paper [52].

IWSLT'14 DE-EN has 160k training examples, on which we use a Transformer with 512-dimensional word embeddings and 1024 FFN dimensions. We train it for 60k iterations, with up to 4096 tokens in each minibatch. Results are listed in Table 2. Note the baseline's BLEU score is already 1.22 higher than the best results reported in [23] using the same model. As shown in Appendix H, LaMAdam uses much smaller update size than LaProp, and it is not able for LaProp to achieve better results even when we scale its learning rate to get similar update sizes as LaMAdam, indicating MaxVA helps to find a better minimum not achievable by using constant β.

WMT'16 EN-DE has 4.5M training examples, where same as [29], we use a larger Transformer with 1024-dimensional word embeddings and 4096 FFN dimensions. Each batch has up to 480k tokens. We train for 32k iterations using the same inverse square root learning rate schedule as [38]. We evaluate the *single model* BLEU on newstest2013, unlike [23] where models in the last 20 epochs are averaged to get the results. As shown in Table 2, LaMAdam also achieves better results.

5.3 General Language Understanding Evaluation (GLUE)

Method	MNLI (Acc)	QNLI (Acc)	QQP (Acc)	RTE (Acc)	SST-2 (Acc)	MRPC (Acc)	CoLA (Mcc)	STS-B (Pearson)
Reported	87.6	92.8	91.9	78.7	94.8	90.2	63.6	91.2
Adam	87.70 (.03)	92.85 (.06)	91.80 (.03)	79.25 (.71)	94.75 (.08)	88.50 (.24)	61.92 (1.1)	91.17 (.13)
LaProp	87.80 (.04)	92.85 (.13)	91.80 (.03)	78.00 (.46)	94.65 (.11)	89.20 (.20)	63.01 (.61)	91.17 (.06)
MAdam	**87.90** (.08)	92.95 (.07)	91.85 (.03)	79.60 (.66)	94.85 (.12)	89.70 (.17)	63.33 (.60)	91.28 (.03)
LaMAdam	87.80 (.03)	**93.05** (.05)	**91.85** (.05)	**80.15** (.64)	**95.15** (.15)	**90.20** (.20)	**63.84** (.85)	**91.36** (.04)

Table 3. Results (median and variance) on the dev sets of GLUE based on finetuning the RoBERTa-base model ([25]), from 4 runs with the same hyperparameter but different random seeds.

To evaluate MaxVA for transfer learning, we fine-tune pre-trained RoBERTa-base model [25] on 8 of the 9 tasks of the GLEU benchmark [39]. Following prevalent validation settings [10,20,31], we report the median and standard error for fine-tuning the RoBERTa-base model [25] over 4 runs where only the random seeds are changed. The results are in Table 3. MAdam and LaMAdam give

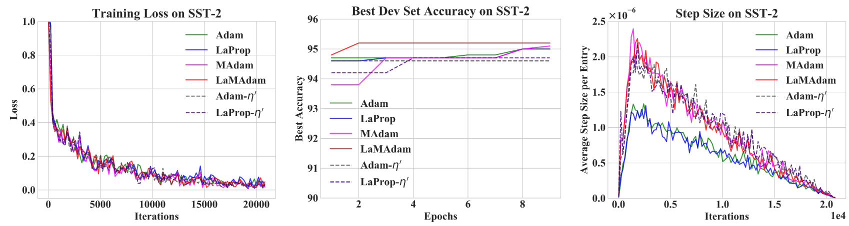

Fig. 4. Training loss, validation accuracy and step size of various optimization methods on SST-2. All optimizers here use $\lambda = 0.1$. Adam and LaProp use (η, β)=(1e-5, 0.98), MAdam and LaMAdam use $(\eta, \underline{\beta}, \bar{\beta})$=(4e-5, 0.5, 0.98), Adam-η' and LaProp-η' use (η, β)=(1.6e-5, 0.98).

better scores than the corresponding baselines in the 8 tasks. More experimental details are in Appendix I.

To highlight the difference of the optimizers, we compare the training loss, dev set accuracy and the average step size on SST-2, as shown in Figure 4. Different from Machine Translation experiments where we train the Transformers from scratch, the adaptive step size of MAdam/LaMAdam is higher in this transfer learning setting. The ratio of the learning rate and step size of MaxVA to non-MaxVA optimizers are 4 and 1.8 respectively on GLUE, while on IWSLT'14 the two ratios are 2 and (approximately) 0.875. Because we start from a pre-trained model, the heavy tail of the gradient is alleviated, just as the BERT model in the later stage of training as shown by [49], and the curvature of the loss landscape should be smaller. Therefore, MaxVA selects larger adaptive step sizes for better convergence. Same as in the Machine Translation experiments, the highest test accuracy of Adam/LaProp cannot reach the same value as MAdam/LaMAdam by simply scaling the base learning rate η to reach similar step sizes as MAdam/LaMAdam.

5.4 Large-batch Pretraining for BERT

We use the NVIDIA BERT pretraining repository to perform large-batch pre-training for BERT-Base model on the Wikipedia Corpus only.[5] Each run takes about 52 hours on 8 V100 GPUs. Training is divided into two phases: the first phase uses a batch size of 64K with input sequence length 128 for 7,038 steps; the second phase uses a batch size 32K with input sequence length 512 for 1563 steps. The total of steps is significantly smaller than the 1,000,000 steps used in the small-batch training of [10]. Therefore, a faster adaptation to curvature in each step is more important.

This point is validated by the faster convergence of MAdam in both phases, as shown in the training loss curves in Figure 5. Contrary to the observation by [45], Adam even converges faster than Lamb in the earlier iterations. [45] only

[5] Note the results from the repository are for BERT-Large trained with additional data from BookCorpus.

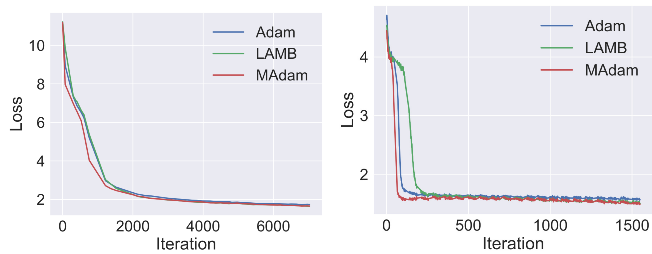

Fig. 5. Training losses of Adam, Lamb and MAdam on Wikipedia Corpus in the two training phases.

explored weight decay of up to 0.01 for Adam, but we find using larger weight decay of 0.1 together with gradient clipping ($\|g_t\|_2 \leq 1$, same as Lamb) stabilizes Adam. We inherit this setting for MAdam. For MAdam and Adam, we do a grid search on the learning rate of phase 1 while keeping the ratios of learning rate in phase 1 and phase 2 to the same as Lamb. We use $\bar{\beta} = 0.999, \underline{\beta} = 0.5$ for MAdam. For Lamb, we use the default setting from the aforementioned repository.

The faster adaptation of MaxVA improves the stability, which enables MAdam to use a much larger learning rate to achieve faster convergence than Adam. The best learning rate for MAdam is 3.4e-3. We tried learning rates in {7e-4, 8e-4, 9e-4, 1e-3} for Adam, and find it always diverges when the learning rate is higher or equal to 9e-4. The best result of Adam is achieved with learning rate 8e-4. MAdam achieves a training loss of 1.492, while Lamb achieves a training loss of 1.507, and Adam has the worst training loss 1.568. The test scores of the models pretrained with MAdam/Lamb/Adam are 88.53/87.60/88.07 (F1) and 82.10/81.40/80.78 (Accuracy) on SQuAD v1.1 and MNLI, respectively.

6 Related Work

Various adaptive methods have been proposed and broadly applied in deep learning [19,12,37,47]. [33] proposed to compute the adaptive learning rate with the coordinate-wise maximum value of v_t so that the adaptive learning rate does not increase. AdaBound [27] clips the adaptive learning rate of Adam with a decreasing upper bound and an increasing lower bound. Lookahead [50] computes weight updates by looking ahead at the sequence of "fast weights" generated by another optimizer. Padam [7] improves the generalization of adaptive methods by choosing a proper exponent for the v_t of AMSGrad. LAPROP [53] uses local running estimation of the variance to normalize the gradients, resulting in higher empirical stability. RAdam [23] was recently invented to free Adam from the warmup schedule for training Transformers. [28] found that using a linear warmup over $2 \cdot (1 - \beta_2)^{-1}$ iterations for Adam achieves almost the same convergence as RAdam. [44] proposes Layer-wise Adaptive Rate Scaling (LARS), and scales the batch size to 16,384 for training ResNet50. LAMB [45] applies a similar layer-wise learning rate on Adam to improve LARS on training BERT. Starting from a similar motivation of adapting to the curvature, the recent work AdaBelief [52] directly estimates the exponential running average of the gradient deviation to compute the adaptive step sizes. Our approach finds the averaging coefficients β_t automatically by maximizing the estimated variance for a faster adaptation to the curvature, which could be complementary to all the aforementioned methods, and is the first to explore in this direction to our knowledge.

7 Conclusion

In this paper, we present Maximum Variation Averaging (MaxVA), a novel adaptive learning rate scheme that replaces the exponential running average of squared gradient with an adaptive weighted mean. In each step, MaxVA chooses the weight β_t for each coordinate, such that the esimated gradient variance is maximized. This enables MaxVA to: (1) take smaller steps when large curvatures or abnormally large gradients are present, which leads to more desirable convergence behaviors in face of noisy gradients; (2) adapt faster to the geometry of the objective, achieving faster convergence in the large-batch setting. We illustrate how our method improves convergence by a better adaptation to variance, and demonstrate strong empirical results on a wide range of tasks. We prove MaxVA converges in the nonconvex stochastic optimization setting under mild assumptions.

8 Appendix

The appendix is available at `https://arxiv.org/pdf/2006.11918.pdf`.

References

1. Agirre, E., M‘arquez, L., Wicentowski, R. (eds.): Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007). ACL (2007)
2. Balles, L., Hennig, P.: Dissecting adam: The sign, magnitude and variance of stochastic gradients. In: ICML. pp. 404–413 (2018)
3. Bentivogli, L., Dagan, I., Dang, H.T., Giampiccolo, D., Magnini, B.: The fifth PASCAL recognizing textual entailment challenge. In: TAC (2009)
4. Bernstein, J., Wang, Y.X., Azizzadenesheli, K., Anandkumar, A.: signsgd: Compressed optimisation for non-convex problems. In: ICML. pp. 560–569 (2018)
5. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
6. Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., Federico, M.: Report on the 11th iwslt evaluation campaign, iwslt 2014. In: IWSLT. vol. 57 (2014)
7. Chen, J., Zhou, D., Tang, Y., Yang, Z., Gu, Q.: Closing the generalization gap of adaptive gradient methods in training deep neural networks. arXiv:1806.06763 (2018)
8. Chen, X., Liu, S., Sun, R., Hong, M.: On the convergence of a class of adam-type algorithms for non-convex optimization. ICLR (2019)
9. Dagan, I., Glickman, O., Magnini, B.: The PASCAL recognising textual entailment challenge. In: Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment. Springer (2006)

10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL. pp. 4171–4186 (2019)
11. Dolan, W.B., Brockett, C.: Automatically constructing a corpus of sentential paraphrases. In: Proceedings of the International Workshop on Paraphrasing (2005)
12. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. JMLR (2011)
13. Ghadimi, S., Lan, G.: Stochastic first-and zeroth-order methods for nonconvex stochastic programming. SIAM Journal on Optimization **23**(4), 2341–2368 (2013)
14. Giampiccolo, D., Magnini, B., Dagan, I., Dolan, B.: The third PASCAL recognizing textual entailment challenge. In: Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing (2007)
15. Haim, R.B., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., Szpektor, I.: The second pascal recognising textual entailment challenge. In: Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment (2006)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
17. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
18. Iyer, S., Dandekar, N., Csernai, K.: First quora dataset release: Question pairs (2017), `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) ICLR (2015)
20. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. ICLR (2020)
21. Levesque, H.J., Davis, E., Morgenstern, L.: The Winograd schema challenge. In: AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning (2011)
22. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Advances in neural information processing systems. pp. 6389–6399 (2018)
23. Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond. ICLR (2020)
24. Liu, L., Liu, X., Gao, J., Chen, W., Han, J.: Understanding the difficulty of training transformers. arXiv:2004.08249 (2020)
25. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692 (2019)
26. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2018)
27. Luo, L., Xiong, Y., Liu, Y., Sun, X.: Adaptive gradient methods with dynamic bound of learning rate. ICLR (2019)
28. Ma, J., Yarats, D.: On the adequacy of untuned warmup for adaptive optimization. arXiv:1910.04209 (2019)
29. Ott, M., Edunov, S., Grangier, D., Auli, M.: Scaling neural machine translation. In: WMT. pp. 1–9 (2018)
30. Park, D.S., Chan, W., Zhang, Y., Chiu, C.C., Zoph, B., Cubuk, E.D., Le, Q.V.: Specaugment: A simple data augmentation method for automatic speech recognition. Interspeech pp. 2613–2617 (2019)

31. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv:1910.10683 (2019)
32. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000+ questions for machine comprehension of text. In: EMNLP (2016)
33. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond. ICLR (2018)
34. Schaul, T., Zhang, S., LeCun, Y.: No more pesky learning rates. In: ICML. pp. 343–351 (2013)
35. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP (2013)
36. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR (2014)
37. Tieleman, T., Hinton, G.: Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA (2012)
38. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
39. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: Glue: A multi-task benchmark and analysis platform for natural language understanding. EMNLP (2018)
40. Warstadt, A., Singh, A., Bowman, S.R.: Neural network acceptability judgments. arXiv preprint 1805.12471 (2018)
41. Williams, A., Nangia, N., Bowman, S.R.: A broad-coverage challenge corpus for sentence understanding through inference. In: NAACL (2018)
42. Wilson, A.C., Roelofs, R., Stern, M., Srebro, N., Recht, B.: The marginal value of adaptive gradient methods in machine learning. In: Neurips. pp. 4148–4158 (2017)
43. Wu, Y., Ren, M., Liao, R., Grosse, R.: Understanding short-horizon bias in stochastic meta-optimization. arXiv:1803.02021 (2018)
44. You, Y., Gitman, I., Ginsburg, B.: Scaling SGD batch size to 32k for imagenet training. CoRR **abs/1708.03888** (2017)
45. You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., Hsieh, C.J.: Large batch optimization for deep learning: Training bert in 76 minutes. In: ICLR (2020)
46. Zaheer, M., Reddi, S., Sachan, D., Kale, S., Kumar, S.: Adaptive methods for nonconvex optimization. In: NeurIPS. pp. 9793–9803 (2018)
47. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. CoRR (2012)
48. Zhang, G., Li, L., Nado, Z., Martens, J., Sachdeva, S., Dahl, G., Shallue, C., Grosse, R.B.: Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In: NeurIPS. pp. 8194–8205 (2019)
49. Zhang, J., Karimireddy, S.P., Veit, A., Kim, S., Reddi, S., Kumar, S., Sra, S.: Why are adaptive methods good for attention models? NeurIPS **33** (2020)
50. Zhang, M.R., Lucas, J., Ba, J., Hinton, G.E.: Lookahead optimizer: k steps forward, 1 step back. In: NeurIPS (2019)
51. Zhu, C., Cheng, Y., Gan, Z., Sun, S., Goldstein, T., Liu, J.: Freelb: Enhanced adversarial training for natural language understanding. In: ICLR (2020)
52. Zhuang, J., Tang, T., Tatikonda, S., Dvornek, N., Ding, Y., Papademetris, X., Duncan, J.S.: Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In: NeurIPS (2020)
53. Ziyin, L., Wang, Z.T., Ueda, M.: Laprop: a better way to combine momentum with adaptive gradient. arXiv:2002.04839 (2020)