AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients

Juntang Zhuang\textsuperscript{1}; Tommy Tang\textsuperscript{2}; Yifan Ding\textsuperscript{3}; Sekhar Tatikonda\textsuperscript{1}; Nicha Dvornek\textsuperscript{1}; Xenophon Papademetris\textsuperscript{1}; James S. Duncan\textsuperscript{1}

\textsuperscript{1}Yale University; \textsuperscript{2}University of Illinois at Urbana-Champaign; \textsuperscript{3}University of Central Florida

\{j.zhuang;sekhar.tatikonda;nicha.dvornek;xenophon.papademetris; james.duncan\}@yale.edu; tommymt2@illinois.edu; yf.ding@knights.ucf.edu

Abstract

Most popular optimizers for deep learning can be broadly categorized as adaptive methods (e.g. Adam) and accelerated schemes (e.g. stochastic gradient descent (SGD) with momentum). For many models such as convolutional neural networks (CNNs), adaptive methods typically converge faster but generalize worse compared to SGD; for complex settings such as generative adversarial networks (GANs), adaptive methods are typically the default because of their stability. We propose AdaBelief to simultaneously achieve three goals: fast convergence as in adaptive methods, good generalization as in SGD, and training stability. The intuition for AdaBelief is to adapt the stepsize according to the "belief" in the current gradient direction. Viewing the exponential moving average (EMA) of the noisy gradient as the prediction of the gradient at the next time step, if the observed gradient greatly deviates from the prediction, we distrust the current observation and take a small step; if the observed gradient is close to the prediction, we trust it and take a large step. We validate AdaBelief in extensive experiments, showing that it outperforms other methods with fast convergence and high accuracy on image classification and language modeling. Specifically, on ImageNet, AdaBelief achieves comparable accuracy to SGD. Furthermore, in the training of a GAN on Cifar10, AdaBelief demonstrates high stability and improves the quality of generated samples compared to a well-tuned Adam optimizer. Code is available at https://github.com/juntang-zhuang/Adabelief-Optimizer

1 Introduction

Modern neural networks are typically trained with first-order gradient methods, which can be broadly categorized into two branches: the accelerated stochastic gradient descent (SGD) family \cite{1}, such as Nesterov accelerated gradient (NAG) \cite{2}, SGD with momentum \cite{3} and heavy-ball method (HB) \cite{4}; and the adaptive learning rate methods, such as Adagrad \cite{5}, AdaDelta \cite{6}, RMSProp \cite{7} and Adam \cite{8}. SGD methods use a global learning rate for all parameters, while adaptive methods compute an individual learning rate for each parameter.

Compared to the SGD family, adaptive methods typically converge fast in the early training phases, but have poor generalization performance \cite{9,10}. Recent progress tries to combine the benefits of both, such as switching from Adam to SGD either with a hard schedule as in SWATS \cite{11}, or with a smooth transition as in AdaBound \cite{12}. Other modifications of Adam are also proposed: AMSGrad \cite{13} fixes the error in convergence analysis of Adam, Yogi \cite{14} considers the effect of minibatch size, MSVAG \cite{15} dissects Adam as sign update and magnitude scaling, RAdam \cite{16} rectifies the variance of learning rate, Fromage \cite{17} controls the distance in the function space, and AdamW \cite{18} decouples weight decay from gradient descent. Although these modifications achieve better accuracy compared to Adam, their generalization performance is typically worse than SGD on large-scale...
datasets such as ImageNet \cite{19}; furthermore, compared with Adam, many optimizers are empirically unstable when training generative adversarial networks (GAN) \cite{20}.

To solve the problems above, we propose “AdaBelief”, which can be easily modified from Adam. Denote the observed gradient at step $t$ as $g_t$, and its exponential moving average (EMA) as $m_t$. Denote the EMA of $g_t^2$ and $(g_t - m_t)^2$ as $v_t$ and $s_t$, respectively. $m_t$ is divided by $\sqrt{m_t}$ in Adam, while it is divided by $\sqrt{t}$ in AdaBelief. Intuitively, $\frac{1}{\sqrt{t}}$ is the “belief” in the observation: viewing $m_t$ as the prediction of the gradient, if $g_t$ deviates much from $m_t$, we have weak belief in $g_t$, and take a small step; if $g_t$ is close to the prediction $m_t$, we have a strong belief in $g_t$, and take a large step. We validate the performance of AdaBelief with extensive experiments. Our contributions can be summarized as:

- We theoretically analyze the convergence property of AdaBelief in both convex optimization and non-convex stochastic optimization.
- We validate the performance of AdaBelief with extensive experiments: AdaBelief achieves fast convergence as in adaptive gradient methods, good generalization as in the SGD family, and training stability in complex settings such as GAN.
- We propose AdaBelief, which can be easily modified from Adam without extra parameters.

2 Methods

2.1 Details of AdaBelief Optimizer

Notations  By the convention in \cite{8}, we use the following notations:

- $f(\theta) \in \mathbb{R}$, $\theta \in \mathbb{R}^d$: $f$ is the loss function to minimize, $\theta$ is the parameter in $\mathbb{R}^d$
- $\prod f_{\mathcal{M}}(y) = \arg\min_{x \in \mathcal{X}} |M^{1/2}(x - y)|$: projection of $y$ onto a convex feasible set $\mathcal{F}$. $g_t$: the gradient at step $t$$m_t$: exponential moving average (EMA) of $g_t$$v_t$, $s_t$: $v_t$ is the EMA of $g_t^2$, $s_t$ is the EMA of $(g_t - m_t)^2$$\alpha$, $\epsilon$: $\alpha$ is the learning rate, default is $10^{-3}$; $\epsilon$ is a small number, typically set as $10^{-8}$$\beta_1$, $\beta_2$: smoothing parameters, typical values are $\beta_1 = 0.9$, $\beta_2 = 0.999$$\beta_1$, $\beta_2$ are the momentum for $m_t$ and $v_t$ respectively at step $t$, and typically set as constant (e.g. $\beta_1 = \beta_1$, $\beta_2 = \beta_2, \forall t \in \{1, 2, ..., T\}$

Algorithm 1: Adam Optimizer

| Initialize $\theta_0$, $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$ |
| While $\theta_t$ not converged |
| $t \leftarrow t + 1$ |
| $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ |
| $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)g_t$ |
| $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$ |
| Bias Correction $\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$, $\hat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$ |
| Update $\theta_t \leftarrow \prod f_{\mathcal{F}, \sqrt{v_t}}(\theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}})$ |

Algorithm 2: AdaBelief Optimizer

| Initialize $\theta_0$, $m_0 \leftarrow 0$, $s_0 \leftarrow 0$, $t \leftarrow 0$ |
| While $\theta_t$ not converged |
| $t \leftarrow t + 1$ |
| $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ |
| $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)g_t$ |
| $s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2)(g_t - m_t)^2$ |
| Bias Correction $\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$, $\hat{s}_t \leftarrow \frac{s_t + t}{1 - \beta_2^t}$ |
| Update $\theta_t \leftarrow \prod f_{\mathcal{F}, \sqrt{s_t}}(\theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{s}_t + \epsilon}})$ |

Comparison with Adam  Adam and AdaBelief are summarized in Algo. 1 and Algo. 2, where all operations are element-wise, with differences marked in blue. Note that no extra parameters are introduced in AdaBelief. Specifically, in Adam, the update direction is $m_t / \sqrt{v_t}$, where $v_t$ is the EMA of $g_t^2$; in AdaBelief, the update direction is $m_t / \sqrt{\hat{v}_t}$, where $s_t$ is the EMA of $(g_t - m_t)^2$. Intuitively, viewing $m_t$ as the prediction of $g_t$, AdaBelief takes a large step when observation $g_t$ is close to prediction $m_t$, and a small step when the observation greatly deviates from the prediction. $\epsilon$ represents bias-corrected value. Note that an extra $\epsilon$ is added to $s_t$ during bias-correction, in order to
we demonstrate AdaBelief’s advantage over Adam in the “large gradient, small stepsize” case. In this case, the algorithm oscillates in a “steep and narrow” valley, hence both Adam and AdaBelief take a large stepsize, because the denominator \( \sqrt{s} \) is a small value.

(2) In region (2), the algorithm oscillates in a “steep and narrow” valley, hence both \(|g_t|\) and \(|g_t - g_{t-1}|\) are large. An ideal optimizer should decrease its stepsize, while SGD takes a large step (proportional to \( m_t \)). Adam and AdaBelief take a small step because the denominators \( \sqrt{s} \) and \( \sqrt{v} \) are large.

(3) In region (3), we demonstrate AdaBelief’s advantage over Adam in the “large gradient, small curvature” case. In this case, \(|g_t|\) and \(v_t\) are large, but \(|g_t - g_{t-1}|\) and \(s_t\) are small; this could happen because of a small learning rate \( \alpha \). In this case, an ideal optimizer should increase its stepsize. SGD uses a large stepsize \((\sim \alpha |g_t|)\); in Adam, the denominator \( \sqrt{v} \) is large, hence the stepsize is small; in AdaBelief, denominator \( \sqrt{s} \) is small, hence the stepsize is large as in an ideal optimizer.

To sum up, AdaBelief scales the update direction by the change in gradient, which is related to the Hessian. Therefore, AdaBelief considers curvature information and performs better than Adam.

### AdaBelief uses curvature information

Update formulas for SGD, Adam and AdaBelief are:

\[
\Delta \theta_t^{SGD} = -\alpha m_t, \quad \Delta \theta_t^{Adam} = -\alpha m_t / \sqrt{v_t}, \quad \Delta \theta_t^{AdaBelief} = -\alpha m_t / \sqrt{s_t}
\]

Note that we name \( \alpha \) as the “learning rate” and \(|\Delta \theta_t|\) as the “stepsize” for the \( t \)th parameter. With a 1D example in Fig. 1, we demonstrate that AdaBelief uses the curvature of loss functions to improve training as summarized in Table 1 with a detailed description below:

(1) In region (1) in Fig. 1, the loss function is flat, hence the gradient is close to 0. In this case, an ideal optimizer should take a large stepsize. The stepsize of SGD is proportional to the EMA of the gradient, hence is small in this case; while both Adam and AdaBelief take a large stepsize, because the denominator (\( \sqrt{v} \) and \( \sqrt{m} \)) is a small value.

### AdaBelief considers the sign of gradient in denominator

We show the advantages of AdaBelief with a 2D example in this section, which gives us more intuition for high dimensional cases. In Fig. 2, we consider the loss function: \( f(x, y) = |x| + |y| \). Note that in this simple problem, the gradient in each axis can only take \( \{1, -1\} \). Suppose the start point is near the \( x \)-axis, e.g. \( y_0 \approx 0, x_0 < 0 \). Optimizers will oscillate in the \( y \) direction, and keep increasing in the \( x \) direction.

Suppose the algorithm runs for a long time (\( t \) is large), so the bias of EMA (\( \beta = \frac{1}{2} \)) is small:

\[
m_t = EMA(g_0, g_1, \ldots g_t) \approx E(g_t), \quad m_{t,x} \approx E g_{t,x} = 1, \quad m_{t,y} \approx E g_{t,y} = 0
\]

\[
v_t = EMA(g_0^2, g_1^2, \ldots g_t^2) \approx E(g_t^2), \quad v_{t,x} \approx E g_{t,x}^2 = 1, \quad v_{t,y} \approx E g_{t,y}^2 = 1.
\]
Figure 2: Left: Consider $f(x, y) = |x| + |y|$. Blue vectors represent the gradient, and the cross represents the optimal point. The optimizer oscillates in the $y$ direction, and keeps moving forward in the $x$ direction. Right: Optimization process for the example on the left. Note that denominator $\sqrt{\tau_{t,x}} = \sqrt{\tau_{t,y}}$ for Adam, hence the same stepsize in $x$ and $y$ direction; while $\sqrt{\tau_{t,x}} < \sqrt{\tau_{t,y}}$, hence AdaBelief takes a large step in the $x$ direction, and a small step in the $y$ direction.

In practice, the bias correction step will further reduce the error between the EMA and its expectation if $g_t$ is a stationary process [8]. Note that:

$$s_t = EMA((g_0 - m_0)^2, \ldots, (g_t - m_t)^2) \approx E[(g_t - E g_t)^2] = \text{Var}g_t, \quad s_{t,x} \approx 0, \quad s_{t,y} \approx 1 \quad (4)$$

An example of the analysis above is summarized in Fig. 2. From Eq. 3 and Eq. 4, note that in Adam, $v_x = v_y$; this is because the update of $v_t$ only uses the amplitude of $g_t$ and ignores its sign, hence the stepsize for the $x$ and $y$ direction is the same $1/\sqrt{v_{t,x}} = 1/\sqrt{v_{t,y}}$. AdaBelief considers both the magnitude and sign of $g_t$, and $1/\sqrt{\tau_{t,x}} \gg 1/\sqrt{\tau_{t,y}}$, hence takes a large step in the $x$ direction and a small step in the $y$ direction, which matches the behaviour of an ideal optimizer.

**Update direction in Adam is close to “sign descent” in low-variance case** In this section, we demonstrate that when the gradient has low variance, the update direction in Adam is close to “sign descent”, hence deviates from the gradient. This is also mentioned in [15].

Under the following assumptions: (1) assume $g_t$ is drawn from a stationary distribution, hence after bias correction, $E v_t = (E g_t) + \text{Var}g_t$. (2) low-noise assumption, assume $(E g_t)^2 \gg \text{Var}g_t$, hence we have $E g_t/\sqrt{E v_t} \approx E g_t/\sqrt{(E g_t)^2} = \text{sign}(E g_t)$. (3) low-bias assumption, assume $\beta_1^t (\beta_1$ to the power of $t$) is small, hence $m_t$, as an estimator of $E g_t$, has a small bias $\beta_1^t E g_t$. Then

$$\Delta g_t^{\text{Adam}} = -\alpha m_t/\sqrt{v_{t,x}} \approx -\alpha \frac{E g_t}{\sqrt{(E g_t)^2 + \text{Var}g_t + \epsilon}} \approx -\alpha \frac{E g_t}{||E g_t||} = -\alpha \text{sign}(E g_t) \quad (5)$$

In this case, Adam behaves like a “sign descent”; in 2D cases the update is $\pm 45^\circ$ to the axis, hence deviates from the true gradient direction. The “sign update” effect might cause the generalization gap between adaptive methods and SGD (e.g. on ImageNet) [23, 9]. For AdaBelief, when the variance of $g_t$ is the same for all coordinates, the update direction matches the gradient direction; when the variance is not uniform, AdaBelief takes a small (large) step when the variance is large (small).

**Numerical experiments** In this section, we validate intuitions in Sec. 2.2. Examples are shown in Fig. 3 and we refer readers to more video examples [9] for better visualization. In all examples, compared with SGD with momentum and Adam, AdaBelief reaches the optimal point at the fastest speed. Learning rate is $\alpha = 10^{-3}$ for all optimizers. For all examples except Fig. 3(d), we set the parameters of AdaBelief to be the same as the default in Adam [8], $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, and set momentum as 0.9 for SGD. For Fig. 3(d), to match the assumption in Sec. 2.2 we set $\beta_1 = \beta_2 = 0.3$ for both Adam and AdaBelief, and set momentum as 0.3 for SGD.

(a) Consider the loss function $f(x, y) = |x| + |y|$ and a starting point near the $x$ axis. This setting corresponds to Fig. 2. Under the same setting, AdaBelief takes a large step in the $x$ direction, and a small step in the $y$ direction, validating our analysis. More examples such as $f(x, y) = |x|/10 + |y|$ are in the supplementary videos.

(b) For an inseparable $L_1$ loss, AdaBelief outperforms other methods under the same setting.

(c) For an inseparable $L_2$ loss, AdaBelief outperforms other methods under the same setting.

(d) We set $\beta_1 = \beta_2 = 0.3$ for Adam and AdaBelief, and set momentum as 0.3 in SGD. This corresponds to settings of Eq. 5. For the loss $f(x, y) = |x|/10 + |y|$, $g_t$ is a constant for a large region, hence $||E g_t|| \gg \text{Var}g_t$. As mentioned in [8], $E m_t = (1 - \beta_t^t) E g_t$, hence a

---

https://www.youtube.com/playlist?list=PL7KkG3n9bER6YmMLrKJ5wocjlvP7aWoOu
(a) loss function is
\[ f(x, y) = |x| + |y| \]
(b) \[ f(x, y) = |x + y| + |x - y| / 10 \]
(c) \[ f(x, y) = (x + y)^2 + (x - y)^2 / 10 \]
\[ \beta_1 = \beta_2 = 0.3 \]
(d) \[ f(x, y) = |x| / 10 + |y| \]

(e) Trajectory for Beale function in 2D.
(f) Trajectory for Beale function in 3D.
(g) Trajectory for Rosenbrock function in 2D.
(h) Trajectory for Rosenbrock function in 3D.

Figure 3: Trajectories of SGD, Adam and AdaBelief. AdaBelief reaches optimal point (marked as orange cross in 2D plots) the fastest in all cases. We refer readers to video examples.

smaller \( \beta \) decreases \( ||m_t - \mathbb{E}g_t|| \) faster to 0. Adam behaves like a sign descent (45° to the axis), while AdaBelief and SGD update in the direction of the gradient.

(e)-(f) Optimization trajectory under default setting for the Beale [24] function in 2D and 3D.
(g)-(h) Optimization trajectory under default setting for the Rosenbrock [25] function.

Above cases occur frequently in deep learning Although the above cases are simple, they give hints to local behavior of optimizers in deep learning, and we expect them to occur frequently in deep learning. Hence, we expect AdaBelief to outperform Adam in general cases. Other works in the literature [13, 12] claim advantages over Adam, but are typically substantiated with carefully-constructed examples. Note that most deep networks use ReLU activation [26], which behaves like an absolute value function as in Fig. 3(a). Considering the interaction between neurons, most networks behave like case Fig. 3(b), and typically are ill-conditioned (the weight of some parameters are far larger than others) as in the figure. Considering a smooth loss function such as cross entropy or a smooth activation, this case is similar to Fig. 3(c). The case with Fig. 3(d) requires \( ||m_t| \| \|Eg_t| \| \| \| \|G_t \) and this typically occurs at the late stages of training, where the learning rate \( \alpha \) is decayed to a small value, and the network reaches a stable region.

2.3 Convergence analysis in convex and non-convex optimization

Similar to [13, 12, 27], for simplicity, we omit the de-biasing step (analysis applicable to de-biased version). Proof for convergence in convex and non-convex cases is in the appendix.

Optimization problem For deterministic problems, the problem to be optimized is \( \min_{\theta \in \mathcal{F}} f(\theta) \); for online optimization, the problem is \( \min_{\theta \in \mathcal{F}} \sum_{t=1}^{T} f_t(\theta) \), where \( f_t \) can be interpreted as loss of the model with the chosen parameters in the \( t \)-th step.

Theorem 2.1. (Convergence in convex optimization) Let \( \{\theta_t\} \) and \( \{s_t\} \) be the sequence obtained by AdaBelief, let \( 0 \leq \beta_2 < 1, \alpha_t = \frac{\alpha}{\sqrt{t}}, \beta_1 = \beta_2, 0 \leq \beta_1 \leq 1, s_t = s_{t+1}, \forall t \in [T] \). Let \( \theta \in \mathcal{F} \), where \( \mathcal{F} \subset \mathbb{R}^d \) is a convex feasible set with bounded diameter \( D_\infty \). Assume \( f(\theta) \) is a convex function and \( ||g_t||_\infty \leq G_\infty / 2 \) (hence \( ||g_t - m_t||_\infty \leq G_\infty \) and \( s_{t+1} \geq c > 0, \forall t \in [T], \theta \in \mathcal{F} \). Denote the
optimal point as $\theta^*$. For $\theta_t$ generated with AdaBelief, we have the following bound on the regret:

$$
\sum_{t=1}^T [f_t(\theta_t) - f_t(\theta^*)] \leq \frac{D_2^2 \sqrt{2T}}{2(1 - \beta_1)^3} \sum_{i=1}^d \frac{s_{T,i}^{1/2}}{N} \left(1 + \beta_1 \right) \alpha \sqrt{\log T} \sum_{i=1}^d \frac{||g_i^T||_2}{2 \sqrt{C(1 - \beta_1)^3}} \sum_{i=1}^d \beta_1 s_{i,T}^{1/2} \alpha
$$

Corollary 2.1.1. Suppose $\beta_{1,t} = \beta_1 \lambda^t$, $0 < \lambda < 1$ in Theorem 2.1, then we have:

$$
\sum_{i=1}^T [f_t(\theta_t) - f_t(\theta^*)] \leq \frac{D_2^2 \sqrt{2T}}{2(1 - \beta_1)^3} \sum_{i=1}^d \frac{s_{T,i}^{1/2}}{N} \left(1 + \beta_1 \right) \alpha \sqrt{\log T} \sum_{i=1}^d \frac{||g_i^T||_2}{2 \sqrt{C(1 - \beta_1)^3}} \sum_{i=1}^d \beta_1 s_{i,T}^{1/2} \alpha
$$

For the convex case, Theorem 2.1 implies the regret of AdaBelief is upper bounded by $O(\sqrt{T})$. Conditions for Corollary 2.1.1 can be relaxed to $\beta_{1,t} = \beta_1 / t$ as in [13], which still generates $O(\sqrt{T})$ regret. Similar to Theorem 4.1 in [8] and corollary 1 in [13], where the term $\sum_{i=1}^d v_{T,i}^{1/2}$ exists, we have $\sum_{i=1}^d s_{T,i}^{1/2}$, Without further assumption, $\sum_{i=1}^d s_{T,i}^{1/2} < dG_\infty$ since $||g_t - m_t||_\infty < G_\infty$ as assumed in Theorem 2.1, and $dG_\infty$ is constant. The literature [8] [13] exerts a stronger assumption that $\sum_{i=1}^d \sqrt{T} v_{T,i}^{1/2} \leq dG_\infty \sqrt{T}$. Our assumption could be similar or weaker, because $E_{st} = Var g_t \leq E_{t}^2 = E_{t}v_t$, then we get better regret than $O(\sqrt{T})$.

Theorem 2.2. (Convergence for non-convex stochastic optimization) Under the assumptions:

- $f$ is differentiable; $||\nabla f(x) - \nabla f(y)|| \leq L ||x - y||$, $\forall x, y$; $f$ is also lower bounded.
- The noisy gradient is unbiased, and has independent noise, i.e. $g_t = \nabla f(\theta_t) + \zeta_t, E_{\zeta_t} = 0, \zeta \sim \zeta$, $\forall t, j \in N, t \neq j$.
- At step $t$, the algorithm can access a bounded noisy gradient, and the true gradient is also bounded.
  i.e. $||\nabla f(\theta_t)|| \leq H, ||g_t|| \leq H, \forall t > 1$.

Assume $\min_{j \in [d]} (s_{t,j}) \geq c > 0$, noise in gradient has bounded variance, $Var(g_t) = \sigma_t^2 \leq 2, \forall t \in N$, then the proposed algorithm satisfies:

$$
\min_{t \in [T]} E\left[||\nabla f(\theta_t)||^2\right] \leq \frac{H^2}{\sqrt{\sigma_t}} \left[ C_1 \alpha^2 (H^2 + \alpha^2)(1 + \log T) + C_2 \frac{d \alpha}{\sqrt{\lambda}} + C_3 \frac{d \alpha}{\sqrt{c}} + C_4 \frac{\sigma}{\sqrt{c}} - \right]
$$

as in [27], $C_1$, $C_2$, $C_3$ are constants independent of $d$ and $T$, and $C_4$ is a constant independent of $T$.

Corollary 2.2.1. If $c > C_1 H$ and assumptions for Theorem 2.2 are satisfied, we have:

$$
\frac{1}{T} \sum_{t=1}^T E\left[\alpha_t^2 ||\nabla f(\theta_t)||^2\right] \leq \frac{1}{T} \frac{1}{\sqrt{T}} \left[ C_1 \alpha^2 (H^2 + \alpha^2)(1 + \log T) + C_2 \frac{d \alpha}{\sqrt{\lambda}} + C_3 \frac{d \alpha}{\sqrt{c}} + C_4 \frac{\sigma}{\sqrt{c}} - \right]
$$

Theorem 2.2 implies the convergence rate for AdaBelief in the non-convex case is $O(\log T/\sqrt{T})$, which is similar to Adam-type optimizers [13][27]. Note that regret bounds are derived in the worst possible case, while empirically AdaBelief outperforms Adam mainly because the cases in Sec. 2.2 occur more frequently. It is possible that the above bounds are loose; we will try to derive a tighter bound in the future.

3 Experiments

We performed extensive comparisons with other optimizers, including SGD [8], AdaBound [12], Yogi [14], Adam [8], MSVAG [15], RAdam [16], Fromage [17] and AdamW [18]. The experiments include: (a) image classification on Cifar dataset [28] with VGG [29], ResNet [30] and DenseNet [31], and image recognition with ResNet on ImageNet [32]; (b) language modeling with LSTM [33] on Penn TreeBank dataset [34]; (c) Wasserstein-GAN (WGAN) [21] on Cifar10 dataset. We emphasize (c) because prior work focuses on convergence and accuracy, yet neglects training stability.

Hyperparameter tuning We performed a careful hyperparameter tuning in experiments. On image classification and language modeling we use the following:

- AdaBelief: We use the default parameters of Adam: $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \alpha = 10^{-3}$.
- SGD, Fromage: We set the momentum as 0.9, which is the default for many networks such as ResNet [30] and DenseNet [31]. We search learning rate among $\{ 10.0, 1.0, 0.1, 0.01, 0.001 \}$.
- Adam, Yogi, RAdam, MSVAG, AdaBound: We search for optimal $\beta_1$ among $\{ 0.5, 0.6, 0.7, 0.8, 0.9 \}$,
CNNs on image classification. We experiment with VGG11, ResNet34 and DenseNet121 on Cifar10 and Cifar100 dataset. We use the official implementation of AdaBound, hence achieved an exact replication of [12]. For each optimizer, we search for the optimal hyperparameters, and report the mean and standard deviation of test-set accuracy (under optimal hyperparameters) for 3 runs with random initialization. As Fig. 4 shows, AdaBelief achieves fast convergence as in adaptive methods such as Adam while achieving better accuracy than SGD and other methods.

We then train a ResNet18 on ImageNet, and report the accuracy on the validation set in Table 2. Due to the heavy computational burden, we could not perform an extensive hyperparameter search; instead, we report the result of AdaBelief with the default parameters of Adam (\(\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}\)) and decoupled weight decay as in [16, 18]; for other optimizers, we report the best result in the literature. AdaBelief outperforms other adaptive methods and achieves comparable accuracy to SGD (70.08 vs. 70.23), which closes the generalization gap between adaptive methods and SGD. Experiments validate the fast convergence and good generalization performance of AdaBelief.

LSTM on language modeling. We experiment with LSTM on the Penn TreeBank dataset [34], and report the perplexity (lower is better) on the test set in Fig. 5. We report the mean and standard deviation across 3 runs. For both 2-layer and 3-layer LSTM models, AdaBelief achieves the lowest
perplexity, validating its fast convergence as in adaptive methods and good accuracy. For the 1-layer model, the performance of AdaBelief is close to other optimizers.

Generative adversarial networks Stability of optimizers is important in practice such as training of GANs, yet recently proposed optimizers often lack experimental validations. The training of a GAN alternates between generator and discriminator in a mini-max game, and is typically unstable [20]; SGD often generates mode collapse, and adaptive methods such as Adam and RMSProp are recommended in practice [38, 37, 39]. Therefore, training of GANs is a good test for the stability. We experiment with one of the most widely used models, the Wasserstein-GAN (WGAN) [21] and the improved version with gradient penalty (WGAN-GP) [37] using a small model with vanilla CNN generator. Using each optimizer, we train the model for 100 epochs, generate 64,000 fake images from noise, and compute the Frechet Inception Distance (FID) [40] between the fake images and real dataset (60,000 real images). FID score captures both the quality and diversity of generated images and is widely used to assess generative models (lower FID is better). For each optimizer, under its optimal hyperparameter settings, we perform 5 runs of experiments, and report the results in Fig. 6 and Fig. 7. AdaBelief significantly outperforms other optimizers, and achieves the lowest FID score.

Besides the small model above, we also experiment with a large model using a ResNet generator and spectral normalization in the discriminator (SN-GAN). Results are summarized in Table. 3. Compared with a vanilla GAN, all FID scores are lower because the SN-GAN is more advanced. Compared with other optimizers, AdaBelief achieves the lowest FID with both large and small GANs.

Remarks Recent research on optimizers tries to combine the fast convergence of adaptive methods with high accuracy of SGD. AdaBound [12] achieves this goal on Cifar, yet its performance on ImageNet is still inferior to SGD [35]. Padam [35] closes this generalization gap on ImageNet; writing the update as $\theta_{t+1} = \theta_t - \alpha m_t / v_t^p$, SGD sets $p = 0$, Adam sets $p = 0.5$, and Padam searches $p$ between 0 and 0.5 (outside this region Padam diverges [35, 41]). Intuitively, compared to Adam, by using a smaller $p$, Padam sacrifices the adaptivity for better generalization as in SGD; however, without good adaptivity, Padam loses training stability. As in Table 4 compared with Padam, AdaBelief achieves a much lower FID score in the training of GAN, meanwhile achieving slightly higher accuracy on ImageNet classification. Furthermore, AdaBelief has the same number of parameters as Adam, while Padam has one more parameter hence is harder to tune.
Figure 7: Left to right: real images, samples from WGAN, WGAN-GP (both trained by AdaBelief).

Table 3: FID (lower is better) of a SN-GAN with ResNet generator on Cifar10.

| Optimizer  | FID     |
|------------|---------|
| AdaBelief  | 12.52 ± 0.16 |
| RAdam      | 12.70 ± 0.12 |
| RMSProp    | 12.15 ± 0.12 |
| Adam       | 13.08 ± 0.19 |
| Fromage    | 12.75 ± 0.15 |
| Yogi       | 14.32 ± 0.15 |
| SGD        | 40.70 ± 0.34 |
| MSVAG      | 48.35 ± 3.34 |
| AdaBound   | 55.65 ± 2.35 |

Table 4: Comparison of AdaBelief and Padam. Higher Acc (lower FID) is better. ‡ is from [35].

| Optimizer | p=1/2 (Adam) | p=2/5 | p=1/4 | p=1/5 | p=1/8 | p=1/16 | p = 0 (SGD) |
|-----------|--------------|-------|-------|-------|-------|--------|-------------|
| AdaBelief | 70.38        | 63.79 | -     | 70.07 | -     | 70.07  | 70.07       |
| Padam     | 63.0 ± 4.1   | 96.6 ± 4.5 | 97.5 ± 2.8 | 126.4 ± 49.6 | 401.5 ± 33.2 | 328.1 ± 37.2 | 362.6 ± 43.9 | 469.3 ± 7.9 |

4 Related works

This work considers the update step in first-order methods. Other directions include Lookahead [42] which updates “fast” and “slow” weights separately, and is a wrapper that can combine with other optimizers; variance reduction methods [43, 44, 45] which reduce the variance in gradient; and LARS [46] which uses a layer-wise learning rate scaling. AdaBelief can be combined with these methods. Other variants of Adam have also been proposed (e.g. NosAdam [47], Sadam [48] and Adax [49]).

Besides first-order methods, second-order methods (e.g. Newton’s method [50], Quasi-Newton method and Gauss-Newton method [51, 52, 51], L-BFGS [53], Natural-Gradient [54, 55], Conjugate-Gradient [56]) are widely used in conventional optimization. Hessian-free optimization (HFO) [57] uses second-order methods to train neural networks. Second-order methods typically use curvature information and are invariant to scaling [58] but have heavy computational burden, and hence are not widely used in deep learning.

5 Conclusion

We propose the AdaBelief optimizer, which adaptively scales the stepsize by the difference between predicted gradient and observed gradient. To our knowledge, AdaBelief is the first optimizer to achieve three goals simultaneously: fast convergence as in adaptive methods, good generalization as in SGD, and training stability in complex settings such as GANs. Furthermore, AdaBelief has the same parameters as Adam, hence is easy to tune. We validate the benefits of AdaBelief with intuitive examples, theoretical convergence analysis in both convex and non-convex cases, and extensive experiments on real-world datasets.

Broader Impact

Optimization is at the core of modern machine learning, and numerous efforts have been put into it. To our knowledge, AdaBelief is the first optimizer to achieve fast speed, good generalization and training stability. AdaBelief can be used for the training of all models that can numerically estimate parameter gradients, hence can boost the development and application of deep learning models.
This work mainly focuses on the theory part, and the social impact is mainly determined by each application rather than by optimizer.

**Acknowledgments and Disclosure of Funding**

This research is supported by NIH grant R01NS035193.

**References**

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

[2] Yu Nesterov, “A method of solving a convex programming problem with convergence rate of(1/k^2),” in *Sov. Math. Dokl*, 1983, vol. 27.

[3] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, “On the importance of initialization and momentum in deep learning,” in *International conference on machine learning*, 2013, pp. 1139–1147.

[4] Boris T Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[5] John Duchi, Elad Hazan, and Yoram Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of machine learning research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[6] Matthew D Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.

[7] Alex Graves, “Generating sequences with recurrent neural networks,” *arXiv preprint arXiv:1308.0850*, 2013.

[8] Diederik P Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

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

[10] Kaifeng Lyu and Jian Li, “Gradient descent maximizes the margin of homogeneous neural networks,” *arXiv preprint arXiv:1906.05890*, 2019.

[11] Nitish Shirish Keskar and Richard Socher, “Improving generalization performance by switching from adam to sgd,” *arXiv preprint arXiv:1712.07628*, 2017.

[12] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun, “Adaptive gradient methods with dynamic bound of learning rate,” *arXiv preprint arXiv:1902.09843*, 2019.

[13] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar, “On the convergence of adam and beyond,” *arXiv preprint arXiv:1904.09237*, 2019.

[14] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar, “Adaptive methods for nonconvex optimization,” in *Advances in neural information processing systems*, 2018, pp. 9793–9803.

[15] Lukas Balles and Philipp Hennig, “Dissecting adam: The sign, magnitude and variance of stochastic gradients,” *arXiv preprint arXiv:1705.07774*, 2017.

[16] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han, “On the variance of the adaptive learning rate and beyond,” *arXiv preprint arXiv:1908.03265*, 2019.

[17] Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming-Yu Liu, “On the distance between two neural networks and the stability of learning,” *arXiv preprint arXiv:2002.03432*, 2020.

[18] Ilya Loshchilov and Frank Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.
[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, 2014, pp. 2672–2680.

[21] Martin Arjovsky, Soumith Chintala, and Léon Bottou, “Wasserstein gan,” arXiv preprint arXiv:1701.07875, 2017.

[22] Marc Toussaint, “Lecture notes: Some notes on gradient descent,” 2012.

[23] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar, “signsgd: Compressed optimisation for non-convex problems,” arXiv preprint arXiv:1802.04434, 2018.

[24] Evelyn ML Beale, “On minimizing a convex function subject to linear inequalities,” Journal of the Royal Statistical Society: Series B (Methodological), vol. 17, no. 2, pp. 173–184, 1955.

[25] HoHo Rosenbrock, “An automatic method for finding the greatest or least value of a function,” The Computer Journal, vol. 3, no. 3, pp. 175–184, 1960.

[26] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, “Deep sparse rectifier neural networks,” in Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 315–323.

[27] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong, “On the convergence of a class of adam-type algorithms for non-convex optimization,” arXiv preprint arXiv:1808.02941, 2018.

[28] Alex Krizhevsky, Geoffrey Hinton, et al., “Learning multiple layers of features from tiny images,” 2009.

[29] Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

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

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.

[33] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Ynpeng Wang, “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data,” Transportation Research Part C: Emerging Technologies, vol. 54, pp. 187–197, 2015.

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

[35] Jinghui Chen and Quanquan Gu, “Closing the generalization gap of adaptive gradient methods in training deep neural networks,” arXiv preprint arXiv:1806.06763, 2018.

[36] Alec Radford, Luke Metz, and Soumith Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv preprint arXiv:1511.06434, 2015.

[37] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, “Improved techniques for training gans,” in Advances in neural information processing systems, 2016, pp. 2234–2242.

[38] Ian Goodfellow, “Nips 2016 tutorial: Generative adversarial networks,” arXiv preprint arXiv:1701.00160, 2016.

[39] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville, “Improved training of wasserstein gans,” in Advances in neural information processing systems, 2017, pp. 5767–5777.
[40] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in Advances in neural information processing systems, 2017, pp. 6626–6637.

[41] Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu, “On the convergence of adaptive gradient methods for nonconvex optimization,” arXiv preprint arXiv:1808.05671, 2018.

[42] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton, “Lookahead optimizer: k steps forward, 1 step back,” in Advances in Neural Information Processing Systems, 2019, pp. 9593–9604.

[43] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex Smola, “Stochastic variance reduction for nonconvex optimization,” in International conference on machine learning, 2016, pp. 314–323.

[44] Rie Johnson and Tong Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” in Advances in neural information processing systems, 2013, pp. 315–323.

[45] Jerry Ma and Denis Yarats, “Quasi-hyperbolic momentum and adam for deep learning,” arXiv preprint arXiv:1810.06801, 2018.

[46] Yang You, Igor Gitman, and Boris Ginsburg, “Scaling sgd batch size to 32k for imagenet training,” arXiv preprint arXiv:1708.03888, vol. 6, 2017.

[47] Haiwen Huang, Chang Wang, and Bin Dong, “Nostalgic adam: Weighting more of the past gradients when designing the adaptive learning rate,” arXiv preprint arXiv:1805.07557, 2018.

[48] Guanghui Wang, Shiyin Lu, Weiwei Tu, and Lijun Zhang, “Sadam: A variant of adam for strongly convex functions,” arXiv preprint arXiv:1905.02957, 2019.

[49] Wenjie Li, Zhaoyang Zhang, Xinjiang Wang, and Ping Luo, “Adax: Adaptive gradient descent with exponential long term memory,” arXiv preprint arXiv:2004.09740, 2020.

[50] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe, Convex optimization, Cambridge university press, 2004.

[51] Robert WM Wedderburn, “Quasi-likelihood functions, generalized linear models, and the gauss—newton method,” Biometrika, vol. 61, no. 3, pp. 439–447, 1974.

[52] Nicol N Schraudolph, “Fast curvature matrix-vector products for second-order gradient descent,” Neural computation, vol. 14, no. 7, pp. 1723–1738, 2002.

[53] Jorge Nocedal, “Updating quasi-newton matrices with limited storage,” Mathematics of computation, vol. 35, no. 151, pp. 773–782, 1980.

[54] Shun-Ichi Amari, “Natural gradient works efficiently in learning,” Neural computation, vol. 10, no. 2, pp. 251–276, 1998.

[55] Razvan Pascanu and Yoshua Bengio, “Revisiting natural gradient for deep networks,” arXiv preprint arXiv:1301.3584, 2013.

[56] Magnus R Hestenes, Eduard Stiefel, et al., “Methods of conjugate gradients for solving linear systems,” Journal of research of the National Bureau of Standards, vol. 49, no. 6, pp. 409–436, 1952.

[57] James Martens, “Deep learning via hessian-free optimization.,” in ICML, 2010, vol. 27, pp. 735–742.

[58] Roberto Battiti, “First-and second-order methods for learning: between steepest descent and newton’s method,” Neural computation, vol. 4, no. 2, pp. 141–166, 1992.
Appendix

A. Detailed Algorithm of AdaBelief

Notations  By the convention in [8], we use the following notations:

- \( f(\theta) \in \mathbb{R}, \theta \in \mathbb{R}^d \): \( f \) is the loss function to minimize, \( \theta \) is the parameter in \( \mathbb{R}^d \)
- \( g_t \): the gradient and step \( t \)
- \( \alpha, c \): \( \alpha \) is the learning rate, default is 10\(^{-3} \); \( c \) is a small number, typically set as 10\(^{-8} \)
- \( \beta_1, \beta_2 \): smoothing parameters, typical values are \( \beta_1 = 0.9, \beta_2 = 0.999 \)
- \( m_t \): exponential moving average (EMA) of \( g_t \)
- \( v_t, s_t \): \( v_t \) is the EMA of \( g_t^2 \), \( s_t \) is the EMA of \(( g_t - m_t \)^2\)
- \( \prod_{\mathcal{F}, \mathcal{M}}(y) = \text{argmin}_{x \in \mathcal{X}} \| M^{1/2}(x - y) \|

Algorithm 1: AdaBelief

Initialize \( \theta_0 \n\)
\[ m_0 \leftarrow 0, s_0 \leftarrow 0, t \leftarrow 0 \]

While \( \theta_t \) not converged
\[ t \leftarrow t + 1 \]
\[ g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1}) \]
\[ m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t \]
\[ s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2) (g_t - m_t)^2 \]

If AMGrad
\[ s_t \leftarrow \max(s_t, s_{t-1}) \]

Bias Correction
\[ \hat{m}_t \leftarrow m_t/(1 - \beta_1^t), \quad \hat{s}_t \leftarrow (s_t+\epsilon)/(1 - \beta_2^t) \]

Update
\[ \theta_t \leftarrow \prod_{\mathcal{F}, \sqrt{T}}(\theta_{t-1} - \frac{\hat{m}_t}{\sqrt{\hat{s}_t+\epsilon}}) \]

B. Convergence analysis in convex online learning case (Theorem 2.1 in main paper)

For the ease of notation, we absorb \( \epsilon \) into \( s_t \). Equivalently, \( s_t \geq c > 0, \forall t \in [T] \). For simplicity, we omit the debiasing step in theoretical analysis as in [13]. Our analysis can be applied to the de-biased version as well.

**Lemma 1.** [7] For any \( Q \in S_+^d \) and convex feasible set \( \mathcal{F} \subset \mathbb{R}^d \), suppose \( u_1 = \min_{x \in \mathcal{F}} \| Q^{1/2}(x - z_1) \| \) and \( u_2 = \min_{x \in \mathcal{F}} \| Q^{1/2}(x - z_2) \| \), then we have \( \| Q^{1/2}(u_1 - u_2) \| \leq \| Q^{1/2}(z_1 - z_2) \| \).

**Theorem 2.** Let \( \{ \theta_t \} \) and \( \{ s_t \} \) be the sequence obtained by the proposed algorithm, let \( 0 \leq \beta_2 < 1, \alpha_t = \frac{\sqrt{\alpha}}{\sqrt{t}}, \beta_1 = 1, 0 \leq \beta_1 \leq \beta_1 < 1, s_{t-1} \leq s_t, \forall t \in [T] \). Let \( \theta \in \mathcal{F} \), where \( \mathcal{F} \subset \mathbb{R}^d \) is a convex feasible set with bounded diameter \( D_{\infty} \). Assume \( f(\theta) \) is a convex function and \( \| g_t \|_{\infty} \leq G_{\infty}/2 \) (hence \( \| g_t - m_t \|_{\infty} \leq G_{\infty} \)) and \( s_{t+1} \geq c > 0, \forall t \in [T], \theta \in \mathcal{F} \). Denote the optimal point as \( \theta^* \). For \( \theta_t \) generated with Algorithm[7], we have the following bound on the regret:

\[
\sum_{t=1}^{T} f_t(\theta_t) - f_t(\theta^*) \leq \frac{D_{\infty}^2 \sqrt{T}}{2(1 - \beta_1)} \sum_{i=1}^{d} s_{t, i}^{1/2} + \frac{(1 + \beta_1) \alpha \sqrt{1 + \log T}}{2\sqrt{c}(1 - \beta_1)^3} \sum_{i=1}^{d} \| g_{t, i} \|_{\infty}^2 \]

**Proof:**
\[
\theta_{t+1} = \prod_{\mathcal{F}, \sqrt{s_t}}(\theta_t - \alpha_t s_t^{-1/2} m_t) = \min_{\theta \in \mathcal{F}} \| \theta_{t}^{1/4} \theta - (\theta_t - \alpha_t s_t^{-1/2} m_t) \| \]
Note that $\prod_{j=1}^{T}(\theta^{*}) = \theta^{*}$ since $\theta^{*} \in \mathcal{F}$. Use $\theta^{*}$ and $\theta_{i,t}$ to denote the $i$th dimension of $\theta^{*}$ and $\theta_{t}$ respectively. From lemma (1), using $u_1 = \theta_{t+1}$ and $u_2 = \theta^{*}$, we have:

\[
\left\| s_{t}^{1/4}(\theta_{t+1} - \theta^{*}) \right\|^2 \leq \left\| s_{t}^{1/4}(\theta_{t} - \alpha_{t}s_{t}^{-1/2}m_{t} - \theta^{*}) \right\|^2 \\
= \left\| s_{t}^{1/4}(\theta_{t} - \theta^{*}) \right\|^2 + \alpha_{t}^{2}\left\| s_{t}^{-1/4}m_{t} \right\|^2 - 2\alpha_{t}(m_{t}, \theta_{t} - \theta^{*}) \\
= \left\| s_{t}^{1/4}(\theta_{t} - \theta^{*}) \right\|^2 + \alpha_{t}^{2}\left\| s_{t}^{-1/4}m_{t} \right\|^2 - 2\alpha_{t}(\beta_{1t}m_{t-1} + (1 - \beta_{1t})\theta_{t}, \theta_{t} - \theta^{*})
\]

(1)

Note that $\beta_{1} \in [0, 1)$ and $\beta_{2} \in [0, 1)$, rearranging inequality (1), we have:

\[
g_{t}, \theta_{t} - \theta^{*} \leq \frac{1}{2\alpha_{t}(1 - \beta_{1t})}\left[\left| s_{t}^{1/4}(\theta_{t} - \theta^{*}) \right|^2 - \left| s_{t}^{1/4}(\theta_{t+1} - \theta^{*}) \right|^2 \right] \\
+ \frac{\beta_{1t}}{2(1 - \beta_{1t})}\alpha_{t}\left| s_{t}^{-1/4}m_{t} \right|^2 + \frac{\beta_{1t}}{2\alpha_{t}(1 - \beta_{1t})}\left| s_{t}^{1/4}(\theta_{t} - \theta^{*}) \right|^2
\]

(Cauchy-Schwarz and Young’s inequality: $ab \leq \frac{a^2}{2} + \frac{b^2}{2} \forall \epsilon > 0$) (2)

By convexity of $f$, we have:

\[
\sum_{t=1}^{T} f_{t}(\theta_{t}) - f_{t}(\theta^{*}) \leq \sum_{t=1}^{T} \langle g_{t}, \theta_{t} - \theta^{*} \rangle
\]

\[
\leq \sum_{t=1}^{T} \left\{ \frac{1}{2\alpha_{t}(1 - \beta_{1t})}\left[\left| s_{t}^{1/4}(\theta_{t} - \theta^{*}) \right|^2 - \left| s_{t}^{1/4}(\theta_{t+1} - \theta^{*}) \right|^2 \right] \\
+ \frac{\beta_{1t}}{2(1 - \beta_{1t})}\alpha_{t}\left| s_{t}^{-1/4}m_{t} \right|^2 + \frac{\beta_{1t}}{2\alpha_{t}(1 - \beta_{1t})}\left| s_{t}^{1/4}(\theta_{t} - \theta^{*}) \right|^2 \right\}
\]

(By formula (2))

\[
\leq \frac{1}{2(1 - \beta_{1})}\left| s_{t}^{1/4}(\theta_{1} - \theta^{*}) \right|^2 \\
+ \frac{1}{2(1 - \beta_{1})}\sum_{t=2}^{T} \left[ \left| s_{t}^{1/4}(\theta_{t} - \theta^{*}) \right|^2 - \left| s_{t-1}^{1/4}(\theta_{t} - \theta^{*}) \right|^2 \right] \\
+ \sum_{t=1}^{T} \frac{\beta_{1t}}{2(1 - \beta_{1})}\alpha_{t}\left| s_{t}^{-1/4}m_{t} \right|^2 \\
+ \sum_{t=1}^{T} \frac{\beta_{1t}}{2\alpha_{t}(1 - \beta_{1t})}\left| s_{t}^{1/4}(\theta_{t} - \theta^{*}) \right|^2
\]

($0 \leq s_{t-1} \leq s_{t}, 0 \leq \alpha_{t} \leq \alpha_{t-1}, 0 \leq \beta_{1t} \leq \beta_{1} < 1$)

\[
\leq \frac{1}{2(1 - \beta_{1})}\left| s_{t}^{1/4}(\theta_{1} - \theta^{*}) \right|^2 \\
+ \frac{1}{2(1 - \beta_{1})}\sum_{t=2}^{T} \left| \theta_{t} - \theta^{*} \right|^2 \left[ \frac{s_{t}^{1/2}}{\alpha_{t}} - \frac{s_{t-1}^{1/2}}{\alpha_{t-1}} \right]
\]
Now bound \( \sum_{t=1}^{T} \alpha_t \left| s_t^{-1/4} m_t \right|^2 \) in Formula (3), assuming \( 0 < c \leq s_t, \forall t \in [T] \).

\[
\sum_{t=1}^{T} \alpha_t \left| s_t^{-1/4} m_t \right|^2 = \sum_{t=1}^{T-1} \alpha_t \left| s_t^{-1/4} m_t \right|^2 + \alpha_T \left| s_T^{-1/4} m_T \right|^2
\]

\[
\leq \sum_{t=1}^{T-1} \alpha_t \left| s_t^{-1/4} m_t \right|^2 + \frac{\alpha_T}{\sqrt{c}} \left| m_T \right|^2
\]

\[
= \sum_{t=1}^{T-1} \alpha_t \left| s_t^{-1/4} m_t \right|^2 + \frac{\alpha_T}{\sqrt{cT}} \sum_{t=1}^{T} \left( \sum_{j=1}^{T-j} (1 - \beta_{1,j}) g_{j,i} \prod_{k=1}^{T-k+1} \beta_{1,T-k+1} \right)^2
\]

\[
\leq \sum_{t=1}^{T-1} \alpha_t \left| s_t^{-1/4} m_t \right|^2 + \frac{\alpha_T}{\sqrt{cT}} \sum_{t=1}^{T} \left( \sum_{j=1}^{T-j} g_{j,i} \prod_{k=1}^{T-k+1} \beta_{1} \right)^2
\]

\[
\leq \sum_{t=1}^{T-1} \alpha_t \left| s_t^{-1/4} m_t \right|^2 + \frac{\alpha_T}{\sqrt{cT}} \sum_{t=1}^{T} \left( \sum_{j=1}^{T-j} \beta_{1} \right)^2 \left( \sum_{j=1}^{T-j} g_{j,i}^2 \right)
\]

\[
\leq \frac{\alpha_T}{\sqrt{c(1 - \beta_1)}} \sum_{i=1}^{d} \sum_{j=1}^{T-j} \beta_{1}^{-1} g_{j,i}^2 \frac{1}{\sqrt{t}}
\]

(\text{since } 1 - \beta_1 \geq 1)
Apply formula (4) to (3), we have:

\[ \sum_{t=1}^{T} f_t(\theta_t) \leq \left( \text{Recursively bound each term in the sum } \sum_{t=1}^{T} \right) \]

\[ = \frac{\alpha}{\sqrt{c(1-\beta_1)}} \sum_{i=1}^{d} \sum_{t=1}^{T} g_{t,i} \sum_{j=t}^{T} \frac{\beta_1^{t-j}}{\sqrt{j}} \]

\[ \leq \frac{\alpha}{\sqrt{c(1-\beta_1)}} \sum_{i=1}^{d} \sum_{t=1}^{T} g_{t,i} \sum_{j=t}^{T} \frac{\beta_1^{t-j}}{\sqrt{j}} \]

\[ \leq \frac{\alpha}{\sqrt{c(1-\beta_1)^2}} \sum_{i=1}^{d} \sum_{t=1}^{T} g_{t,i} \cdot \frac{1}{\sqrt{t}} \]

\[ (\text{since } \sum_{j=t}^{T} \beta_1^{t-j} = \sum_{j=0}^{T-t} \beta_1^{j} = \frac{1-\beta_1^{T-t+1}}{1-\beta_1} \leq \frac{1}{1-\beta_1}) \]

\[ \leq \frac{\alpha}{\sqrt{c(1-\beta_1)^2}} \sum_{i=1}^{d} \left\| g_{t:T,i} \right\|_2 \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \]

\[ (\text{Cauchy – Schwartz}, \langle u, v \rangle \leq \left\| u \right\| \left\| v \right\|, u_t = g_{t,i}^2, v_t = \frac{1}{\sqrt{t}}) \]

\[ \leq \frac{\alpha \sqrt{1+\log T}}{\sqrt{c(1-\beta_1)^2}} \sum_{i=1}^{d} \left\| g_{t:T,i} \right\|_2 \left( \text{since } \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq 1 + \log T \right) \quad (4) \]

Apply formula (4) to (3), we have:

\[ \sum_{t=1}^{T} f_t(\theta_t) - f_t(\theta^*) \leq \frac{1}{2(1-\beta_1)} \left\| \frac{1}{1/4} (\theta_t - \theta^*) \right\|_2^2 + \frac{1}{2(1-\beta_1)} \sum_{i=1}^{d} \alpha_i \left\| s_t^{-1/4} m_t \right\|_2^2 \]

\[ + \frac{1}{2(1-\beta_1)} \sum_{t=1}^{T} \beta_1 t \left\| s_t^{-1/4} (\theta_t - \theta^*) \right\|_2^2 \]

\[ \leq \frac{1}{2(1-\beta_1)} \left\| \frac{1}{1/4} (\theta_t - \theta^*) \right\|_2^2 + \frac{1}{2(1-\beta_1)} \sum_{i=1}^{d} \alpha_i \left\| s_t^{-1/4} (\theta_t - \theta^*) \right\|_2^2 \]

\[ + \frac{(1+\beta_1) \alpha \sqrt{1+\log T}}{2 \sqrt{c(1-\beta_1)^3}} \sum_{i=1}^{d} \left\| g_{t:T,i} \right\|_2 \sum_{t=1}^{T} \beta_1 t \]

\[ + \frac{1}{2(1-\beta_1)} \sum_{t=1}^{T} \beta_1 t \left\| s_t^{-1/4} (\theta_t - \theta^*) \right\|_2^2 \]

\[ (\text{By formula (4)}) \]

\[ \leq \frac{1}{2(1-\beta_1)} \sum_{i=1}^{d} \alpha_i \left\| s_t^{-1/4} D_{i}^2 \right\|_2 + \frac{1}{2(1-\beta_1)} \sum_{t=2}^{T} \sum_{i=1}^{d} D_{i}^2 \left[ \frac{s_t^{1/2}}{\alpha_t} - \frac{s_{t-1}^{1/2}}{\alpha_{t-1}} \right] \]

\[ + \frac{(1+\beta_1) \alpha \sqrt{1+\log T}}{2 \sqrt{c(1-\beta_1)^3}} \sum_{i=1}^{d} \left\| g_{t:T,i} \right\|_2 \sum_{t=1}^{T} \beta_1 t \]

\[ + \frac{D_{i}^2}{2(1-\beta_1)} \sum_{t=1}^{T} \sum_{i=1}^{d} \frac{\beta_1 s_t^{1/2}}{\alpha_t} \]

16
Adam-type algorithm is given by

\[ \frac{s_{t, i}}{\alpha_t} \geq \frac{s_{t-1, i}}{\alpha_{t-1}} \text{ by assumption.} \]

Furthermore, let

\[ \mathbb{E} \left[ \sum_{i=1}^{d} \frac{1}{\alpha_t} s_{T, i}^2 \right] + \frac{D_2^2 \sqrt{T}}{2\alpha(1 - \beta_1)} \sum_{i=1}^{d} \beta_t \left( \frac{1 + \beta_1}{\alpha \sqrt{\log T}} \right)^{3/2} \sqrt{c(1 - \beta_1)^3} \sum_{i=1}^{d} \left\| g_{t, i}^2 \right\|_2 \]

where \( \beta_t \) is differentiable and has Lipschitz gradient, \( \| \nabla f(x) - \nabla f(y) \| \leq L \| x - y \| \), \( \forall x, y \). f is also lower bounded.

• A2, at time \( t \), the algorithm can access a bounded noisy gradient, the true gradient is also bounded. i.e. \( \| \nabla f(\theta_t) \| \leq H, \| g_t \| \leq H, \forall t > 1. \)

• A3, The noisy gradient is unbiased, and has independent noise. i.e. \( g_t = \nabla f(\theta_t) + \zeta_t, \mathbb{E} \zeta_t = 0, \zeta_t \perp \zeta_j, \forall j, t \in \mathbb{N}, t \neq j \)

\[ \sum_{t=1}^{T} \lambda^{t-1} \sqrt{t} \leq \sum_{t=1}^{T} \lambda^{t-1} \leq \frac{1}{(1 - \lambda)^2} \]

Plugging (7) into (5), we can derive the results above.

C. Convergence analysis for non-convex stochastic optimization (Theorem 2.2 in main paper)

Assumptions

- A1, \( f \) is differentiable and has \( L \)-Lipschitz gradient, \( \| \nabla f(x) - \nabla f(y) \| \leq L \| x - y \| \), \( \forall x, y \). f is also lower bounded.
- A2, at time \( t \), the algorithm can access a bounded noisy gradient, the true gradient is also bounded. i.e. \( \| \nabla f(\theta_t) \| \leq H, \| g_t \| \leq H, \forall t > 1. \)
- A3, The noisy gradient is unbiased, and has independent noise. i.e. \( g_t = \nabla f(\theta_t) + \zeta_t, \mathbb{E} \zeta_t = 0, \zeta_t \perp \zeta_j, \forall j, t \in \mathbb{N}, t \neq j \)

Theorem 3. [27] Suppose assumptions A1-A3 are satisfied, \( \beta_{t, 1} \) is chosen such that \( 0 \leq \beta_{t, 1} \leq \beta_{t, 2} < 1, 0 < \beta_{t, 2} < 1, \forall t > 0. \) For some constant \( G, \left\| \alpha_t \frac{\nabla f(\theta_t)}{\sqrt{s_t}} \right\| \leq G, \forall t. \) Then Adam-type algorithms yield

\[
\mathbb{E} \left[ \sum_{t=1}^{T} \alpha_t \left( \frac{\nabla f(\theta_t)}{\sqrt{s_t}} \right) \right] \leq \frac{D_2^2 \sqrt{T}}{2\alpha(1 - \beta_1)} \sum_{i=1}^{d} \beta_t \left( \frac{1 + \beta_1}{\alpha \sqrt{\log T}} \right)^{3/2} \sqrt{c(1 - \beta_1)^3} \sum_{i=1}^{d} \left\| g_{t, i}^2 \right\|_2 \]

where \( C_1, C_2, C_3 \) are constants independent of \( d \) and \( T \), \( C_4 \) is a constant independent of \( T \), the expectation is taken \( w.r.t. \) all randomness corresponding to \( \{ g_t \} \).

Furthermore, let \( \gamma_t := \min_{j \in [d]} \min_{i \in [d]} \alpha_t (\sqrt{s_{t-1}})_j \) denote the minimum possible value of effective stepsize at time \( t \) over all possible coordinate and past gradients \( \{ g_t \} \). The convergence rate of Adam-type algorithm is given by

\[
\min_{t \in [T]} \mathbb{E} \left[ \left\| \nabla f(\theta_t) \right\|^2 \right] = O \left( \frac{s_1(T)}{s_2(T)} \right) \]

where \( s_1(T) \) is defined through the upper bound of RHS of (8), and \( \sum_{t=1}^{T} \gamma_t = \Omega(s_2(T)) \)
Proof: We provide the proof from [27] in next section for completeness.

**Theorem 4.** Assume \( \min_{j \in [d]} (s_1)_j \geq c > 0 \), noise in gradient has bounded variance, \( \text{Var}(g_t) = \sigma_t^2 \leq \sigma^2, \forall t \in \mathbb{N} \), then the AdaBelief algorithm satisfies:

\[
\min_{t \in [T]} \mathbb{E} \left[ \left\| \nabla f(\theta_t) \right\|^2 \right] \leq \frac{H}{\sqrt{T} \alpha} \left[ \frac{C_1 \alpha^2 (H^2 + \sigma^2)(1 + \log T)}{c} + C_2 \frac{d \alpha}{\sqrt{c}} + C_3 \frac{\alpha^2}{c} + C_4 \right] \\
= \frac{1}{\sqrt{T}} (Q_1 + Q_2 \log T)
\]

where

\[
Q_1 = \frac{H}{\alpha} \left[ \frac{C_1 \alpha^2 (H^2 + \sigma^2)}{c} + C_2 \frac{d \alpha}{\sqrt{c}} + C_3 \frac{\alpha^2}{c} + C_4 \right] \\
Q_2 = \frac{HC_1 \alpha (H^2 + \sigma^2)}{c}
\]

Proof: We first derive an upper bound of the RHS of formula (8), then derive a lower bound of the LHS of (8).

\[
\mathbb{E} \left[ \sum_{t=1}^{T} \left\| \frac{\alpha_t g_t}{\sqrt{s_t}} \right\|^2 \right] \leq \frac{1}{c} \mathbb{E} \left[ \sum_{t=1}^{T} \sum_{i=1}^{d} (\alpha_{t,i} g_{t,i})^2 \right] \quad (\text{since } 0 < c \leq s_t, \forall t \in [T])
\]

\[
= \frac{1}{c} \sum_{i=1}^{d} \sum_{t=1}^{T} \alpha_{t,i}^2 \mathbb{E} (g_{t,i})^2 \\
= \frac{1}{c} \sum_{t=1}^{T} \alpha_t^2 \mathbb{E} \left[ \left\| \nabla f(\theta_t) \right\|^2 + \left\| \sigma_t \right\|^2 \right] \tag{10}
\]

\[
\mathbb{E} \left[ \sum_{t=1}^{T} \left\| \frac{\alpha_t}{\sqrt{s_t}} - \frac{\alpha_{t-1}}{\sqrt{s_{t-1}}} \right\|_1 \right] = \mathbb{E} \left[ \sum_{t=1}^{T} \sum_{i=1}^{d} \frac{\alpha_{t-1}}{\sqrt{s_{t-1,i}}} - \frac{\alpha_t}{\sqrt{s_{t,i}}} \right] \\
(\text{since } \alpha_t \leq \alpha_{t-1}, s_{t,i} \geq s_{t-1,i})
\]

\[
= \mathbb{E} \left[ \sum_{i=1}^{d} \frac{\alpha_1}{\sqrt{s_{1,i}}} - \frac{\alpha_T}{\sqrt{s_{T,i}}} \right] \\
\leq \mathbb{E} \left[ \sum_{i=1}^{d} \frac{\alpha_1}{\sqrt{s_{1,i}}} \right] \\
\leq \frac{d \alpha}{\sqrt{c}} \quad (\text{since } 0 < c \leq s_t, 0 \leq \alpha_t \leq \alpha_1 = \alpha, \forall t) \tag{11}
\]

\[
\mathbb{E} \left[ \sum_{t=1}^{T} \left\| \frac{\alpha_t}{\sqrt{s_t}} - \frac{\alpha_{t-1}}{\sqrt{s_{t-1}}} \right\|^2 \right] = \mathbb{E} \left[ \sum_{t=1}^{T} \sum_{i=1}^{d} \left( \frac{\alpha_t}{\sqrt{s_t}} - \frac{\alpha_{t-1}}{\sqrt{s_{t-1}}} \right)^2 \right] \\
\leq \mathbb{E} \left[ \sum_{t=1}^{T} \sum_{i=1}^{d} \left( \frac{\alpha_t}{\sqrt{s_t}} - \frac{\alpha_{t-1}}{\sqrt{s_{t-1}}} \right) \frac{\alpha}{\sqrt{c}} \right] \\
(\text{Since } \left| \frac{\alpha_T}{\sqrt{s_T}} - \frac{\alpha_{T-1}}{\sqrt{s_{T-1}}} \right| = \frac{\alpha_{T-1}}{\sqrt{s_{T-1}}} - \frac{\alpha_T}{\sqrt{s_T}} \leq \frac{\alpha_{T-1}}{\sqrt{s_{T-1}}} \leq \frac{\alpha}{\sqrt{c}}) \\
\leq \frac{d \alpha^2}{c} \quad (\text{By } (11)) \tag{12}
\]
Next we derive the lower bound of LHS of (8).

\[ \mathbb{E} \left[ \sum_{t=1}^{T} \alpha_t \langle \nabla f(\theta_t), \frac{\nabla f(\theta_t)}{\sqrt{s_t}} \rangle \right] \geq \frac{1}{H} \mathbb{E} \left[ \sum_{t=1}^{T} \alpha_t \left\| \nabla f(\theta_t) \right\|^2 \right] \geq \frac{\alpha \sqrt{T}}{H} \min_{t \in [T]} \mathbb{E} \left\| \nabla f(\theta_t) \right\|^2 \]  

(13)

Combining (10), (11), (12) and (13) to (8), we have:

\[ \mathbb{E} \left[ \sum_{t=1}^{T} \alpha_t \langle \nabla f(\theta_t), \frac{\nabla f(\theta_t)}{\sqrt{s_t}} \rangle \right] \leq \mathbb{E} \left[ \sum_{t=1}^{T} \alpha_t \left\| \nabla f(\theta_t) \right\|^2 \right] \leq \frac{\alpha \sqrt{T}}{H} \min_{t \in [T]} \mathbb{E} \left\| \nabla f(\theta_t) \right\|^2 \]

(14)

Re-arranging above inequality, we have

\[ \min_{t \in [T]} \mathbb{E} \left\| \nabla f(\theta_t) \right\|^2 \leq \frac{H}{\sqrt{T} \alpha} \left[ C_1 \alpha^2 (H^2 + \sigma^2) \left( 1 + \log T \right) + C_2 \frac{\alpha \sigma^2}{c} + C_3 \frac{\alpha^2 \sigma^2}{c} + C_4 \right] \]

\[ = \frac{1}{\sqrt{T}} (Q_1 + Q_2 \log T) \]  

(16)

where

\[ Q_1 = \frac{H}{\alpha} \left[ C_1 \alpha^2 (H^2 + \sigma^2) + C_2 \frac{\alpha \sigma^2}{c} + C_3 \frac{\alpha^2 \sigma^2}{c} + C_4 \right] \]  

(17)

\[ Q_2 = \frac{HC_1 \alpha (H^2 + \sigma^2)}{c} \]  

(18)

**Corollary 4.1.** If \( c > C_1 H \) and assumptions for Theorem 3 are satisfied, we have:

\[ \frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \left[ \alpha_t^2 \left\| \nabla f(\theta_t) \right\|^2 \right] \leq \frac{1}{T} \frac{1}{\sqrt{T} - C_1 c} \left[ C_1 \alpha^2 \sigma^2 \left( 1 + \log T \right) + C_2 \frac{\alpha \sigma^2}{c} + C_3 \frac{\alpha^2 \sigma^2}{c} + C_4 \right] \]  

(19)

**Proof:** From (13) and (14), we have

\[ \frac{1}{H} \mathbb{E} \left[ \sum_{t=1}^{T} \alpha_t \left\| \nabla f(\theta_t) \right\|^2 \right] \leq \mathbb{E} \left[ \sum_{t=1}^{T} \alpha_t \left\| \nabla f(\theta_t) \right\|^2 \right] \leq \frac{C_1}{c} \sum_{t=1}^{T} \mathbb{E} \left[ \alpha_t^2 \left\| \nabla f(\theta_t) \right\|^2 + \alpha_t^2 \left\| \sigma_t \right\|^2 \right] + C_2 \frac{\alpha \sigma^2}{c} + C_3 \frac{\alpha^2 \sigma^2}{c} + C_4 \]

(20)

By re-arranging, we have

\[ \left( \frac{1}{H} - \frac{C_1}{c} \right) \sum_{t=1}^{T} \mathbb{E} \left[ \alpha_t^2 \left\| \nabla f(\theta_t) \right\|^2 \right] \leq \frac{C_1}{c} \sum_{t=1}^{T} \mathbb{E} \left[ \alpha_t^2 \left\| \sigma_t \right\|^2 \right] + C_2 \frac{\alpha \sigma^2}{c} + C_3 \frac{\alpha^2 \sigma^2}{c} + C_4 \]
The following holds true:

\[ \sup_{\mathbf{T}} \left\{ \mathbb{E} \left[ \alpha_t^2 \left\| \nabla f(\theta_t) \right\|^2 \right] \right\} \leq \frac{C_1 \alpha^2 \sigma^2}{c} (1 + \log T) + C_2 \frac{\alpha \sqrt{c}}{\sqrt{c}} + C_3 \frac{\alpha^2}{c} + C_4 \] (21)

By assumption, \( \frac{1}{T} - \frac{C_3}{c} > 0 \), then we have

\[ \sum_{t=1}^T \mathbb{E} \left[ \alpha_t^2 \left\| \nabla f(\theta_t) \right\|^2 \right] \leq \frac{1}{T} - \frac{C_3}{c} \left[ \frac{C_1 \alpha^2 \sigma^2}{c} (1 + \log T) + C_2 \frac{\alpha \sqrt{c}}{\sqrt{c}} + C_3 \frac{\alpha^2}{c} + C_4 \right] \] (22)

\[ \square \]

D. Proof of Theorem 3

Lemma 5. Let \( \theta_0 \triangleq \theta_1 \) in the Algorithm, consider the sequence

\[ z_t = \theta_t + \frac{\beta_{1,t}}{1 - \beta_{1,t}} (\theta_t - \theta_{t-1}), \forall t \geq 2 \]

The following holds true:

\[ z_{t+1} - z_t = -\left( \frac{\beta_{1,t+1}}{1 - \beta_{1,t+1}} - \frac{\beta_{1,t}}{1 - \beta_{1,t}} \right) \frac{\alpha_t m_t}{\sqrt{s_t}} \]

\[ - \frac{\beta_{1,t}}{1 - \beta_{1,t}} \left( \frac{\alpha_t}{\sqrt{s_t}} - \frac{\alpha_{t-1}}{\sqrt{s_{t-1}}} \right) m_{t-1} \]

and

\[ z_2 - z_1 = -\left( \frac{\beta_{1,2}}{1 - \beta_{1,2}} - \frac{\beta_{1,1}}{1 - \beta_{1,1}} \right) \frac{\alpha_1 m_1}{\sqrt{v_1}} \]

Lemma 6. Suppose that the conditions in Theorem 3 hold, then

\[ \mathbb{E} \left[ f(z_{t+1} - f(z_t)) \right] \leq \sum_{i=1}^6 T_i \] (25)

where

\[ T_1 = -\mathbb{E} \left[ \sum_{i=1}^t \langle \nabla f(z_i), \frac{\beta_{1,i}}{1 - \beta_{1,i}} \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right) m_{i-1} \rangle \right] \] (26)

\[ T_2 = -\mathbb{E} \left[ \sum_{i=1}^t \alpha_i \langle \nabla f(z_i), \frac{g_i}{\sqrt{v_i}} \rangle \right] \] (27)

\[ T_3 = -\mathbb{E} \left[ \sum_{i=1}^t \langle \nabla f(z_i), \left( \frac{\beta_{1,i+1}}{1 - \beta_{1,i+1}} - \frac{\beta_{1,i}}{1 - \beta_{1,i}} \right) \frac{\alpha_t m_t}{\sqrt{v_t}} \rangle \right] \] (28)

\[ T_4 = \mathbb{E} \left[ \sum_{i=1}^t \frac{3L}{2} \left\| \left( \frac{\beta_{1,i+1}}{1 - \beta_{1,i+1}} - \frac{\beta_{1,i}}{1 - \beta_{1,i}} \right) \frac{\alpha_t m_t}{\sqrt{v_t}} \right\|^2 \right] \] (29)

\[ T_5 = \mathbb{E} \left[ \sum_{i=1}^t \frac{3L}{2} \left\| \frac{\beta_{1,i}}{1 - \beta_{1,i}} \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right) m_{i-1} \right\|^2 \right] \] (30)

\[ T_6 = \mathbb{E} \left[ \sum_{i=1}^t \frac{3L}{2} \left\| \frac{\alpha_t g_t}{\sqrt{v_t}} \right\|^2 \right] \] (31)

Lemma 7. Suppose that the condition in Theorem 3 hold, \( T_1 \) in (26) can be bounded as:

\[ T_1 = -\mathbb{E} \left[ \sum_{i=1}^t \langle \nabla f(z_i), \frac{\beta_{1,i}}{1 - \beta_{1,i}} \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right) m_{i-1} \rangle \right] \]

\[ \leq H^2 \frac{\beta_{1}}{1 - \beta_{1}} \mathbb{E} \left[ \sum_{i=2}^t \sum_{j=1}^d \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right),_j \right] \] (32)
Lemma 8. Suppose the conditions in Theorem 3 are satisfied, then $T_3$ in (28) can be bounded as:

$$T_3 = -E \left[ \sum_{i=1}^{t} (\nabla f(z_i), (\beta_{1,i+1} - \beta_{1,i}) \frac{\alpha_i m_i}{\sqrt{v_i}}) \right]$$

$$\leq \left( \frac{\beta_{1}}{1 - \beta_{1}} - \frac{\beta_{1,t+1}}{1 - \beta_{1,t+1}} \right) (H^2 + G^2)$$

(33)

Lemma 9. Suppose assumptions in Theorem 3 are satisfied, then $T_4$ in (29) can be bounded as:

$$T_4 = E \left[ \sum_{i=1}^{t} \frac{3L}{2} \left\| \left( \frac{\beta_{1,i+1}}{1 - \beta_{1,i+1}} - \frac{\beta_{1,i}}{1 - \beta_{1,i}} \right) \frac{\alpha_i m_i}{\sqrt{v_i}} \right\|^2 \right]$$

$$\leq \frac{3L}{2} \left( \frac{\beta_{1}}{1 - \beta_{1}} - \frac{\beta_{1,t+1}}{1 - \beta_{1,t+1}} \right)^2 G^2$$

(34)

Lemma 10. Suppose the assumptions in Theorem 3 are satisfied, then $T_5$ in (30) can be bounded as:

$$T_5 = E \left[ \sum_{i=1}^{t} \frac{3L}{2} \left\| \frac{\beta_{1,i}}{1 - \beta_{1,i}} \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right) m_{i-1} \right\|^2 \right]$$

$$\leq \frac{3L}{2} \left( \frac{\beta_{1}}{1 - \beta_{1}} \right)^2 H^2 E \left[ \sum_{i=2}^{t} \sum_{j=1}^{d} \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right) \right]$$

(35)

Lemma 11. Suppose the assumptions in Theorem 3 are satisfied, then $T_2$ in (28) are bounded as:

$$T_2 = -E \left[ \sum_{i=1}^{t} \alpha_i (\nabla f(z_i), \frac{g_i}{\sqrt{v_i}}) \right]$$

$$\leq E \sum_{i=2}^{t} \left\| \frac{\alpha_i g_i}{\sqrt{v_i}} \right\|^2 + L^2 \left( \frac{\beta_{1}}{1 - \beta_{1}} \right)^2 \left( \frac{1}{1 - \beta_{1}} \right)^2 E \left[ \sum_{j=2}^{t} \sum_{i=2}^{d} \left( \frac{\alpha_i g_i}{\sqrt{v_i}} \right) \right]$$

$$+ L^2 H^2 \left( \frac{\beta_{1}}{1 - \beta_{1}} \right)^4 \left( \frac{1}{1 - \beta_{1}} \right)^4 E \left[ \sum_{j=2}^{t} \sum_{i=2}^{d} \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right) \right]$$

$$+ 2H^2 E \left[ \sum_{j=2}^{t} \sum_{i=2}^{d} \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right) \right]$$

$$+ 2H^2 E \left[ \sum_{j=2}^{t} \left( \frac{\alpha_{i}}{\sqrt{v_{i}}} \right) \right]$$

$$- E \left[ \sum_{i=1}^{t} \alpha_i (\nabla f(x_i), \nabla f(x_i) / \sqrt{v_i}) \right]$$

(36)

Proof of Theorem 3

We provide the proof from (27) for completeness. We combine Lemma 3, 5, 6, 7, 8, 9, 10 and 11 to bound the objective.

$$E \left[ f(z_{t+1}) - f(z_t) \right] \leq \sum_{i=1}^{6} T_i$$

$$\leq H^2 \frac{\beta_{1}}{1 - \beta_{1}} E \left[ \sum_{i=2}^{t} \sum_{j=1}^{d} \left( \frac{\alpha_i}{\sqrt{v_i}} - \frac{\alpha_{i-1}}{\sqrt{v_{i-1}}} \right) \right]$$

21
We skip the proof, which is a direct application of the Bayes rule in the Gaussian distribution case as

\[\text{We analyze AdaBelief from a Bayesian perspective.}\]

According to Theorem .12, the gradient descent direction with maximum expected gain is:

\[\text{Assume the gradient follows a Gaussian prior with uniform diagonal covariance,} \quad \tilde{g} \sim \mathcal{N}(0, \sigma^2 I); \text{ assume the observed gradient follows a Gaussian distribution,} \quad g \sim \mathcal{N}(\tilde{g}, C), \text{ where} \quad C \text{ is some covariance matrix. Then the posterior is:} \quad \tilde{g}|g, C \sim \mathcal{N}\left((I + \frac{C}{\sigma^2})^{-1}g, \left(\frac{L}{\sigma^2} + C^{-1}\right)^{-1}\right)\]

\[\text{We skip the proof, which is a direct application of the Bayes rule in the Gaussian distribution case as in [?]. If} \quad g \text{ is averaged across a batch of size} \quad n, \text{ we can replace} \quad C \text{ with} \quad \frac{C}{n}.\]

\[\text{According to Theorem.12} \quad \text{the gradient descent direction with maximum expected gain is:} \quad \mathbb{E}[\tilde{g}|g, C] = (I + \frac{C}{\sigma^2})^{-1}g = \sigma^2(\sigma^2 I + C)^{-1}g \propto (\sigma^2 I + C)^{-1}g \quad (42)\]
Denote $\epsilon = \sigma^2$, then adaptive optimizers update in the direction $(\epsilon I + C)^{-1}g$; considering the noise in $g_t$, in practice most optimizers replace $g_t$ with its EMA $m_t$, hence the update direction is $(\epsilon I + C)^{-1}m_t$. In practice, adaptive methods such as Adam and AdaGrad replace $(\epsilon I + C)^{-1/2}(\epsilon I + C)^{-1/2}m_t$ with $\alpha I(\epsilon I + C)^{-1/2}m_t$ for numerical stability, where $\alpha$ is some predefined learning rate. Both Adam and AdaBelief take this form; their difference is in the estimate of $C$: Adam uses an uncentered approximation $C_{\text{Adam}} \approx \text{EMA diag}(g_t g_t^\top)$, while AdaBelief uses a centered approximation $C_{\text{AdaBelief}} \approx \text{EMA diag}[(g_t - \mathbb{E}g_t)(g_t - \mathbb{E}g_t)^\top]$. Note that the definition of $C$ is the covariance hence it is centered. Note that for the $i$th parameter, $\mathbb{E}(g_i^2) = (\mathbb{E}g_i)^2 + \text{Var}(g_i)$, so when $\text{Var}(g_i) \ll \|\mathbb{E}g_i\|$, we have $C_{\text{AdaBelief}}^i < C_{\text{Adam}}^i$, and AdaBelief behaves closer to the ideal and takes a larger step than Adam because $C$ is in the denominator.

From a practical perspective, $\epsilon$ can be interpreted as a numerical term to avoid division by 0; from the Bayesian perspective, $\epsilon$ represents our prior on $g_t$, with a larger $\epsilon$ indicating a larger $\sigma^2$. Note that as the network evolves with training, the distribution of the gradient is distorted (an example with Adam is shown in Fig. 2 of [16]), hence the Gaussian prior might not match the true distribution. To solve the mismatch between prior and the true distribution, it might be reasonable to use a weak prior during late stages of training (e.g., let $\sigma^2$ grow at late training phases, and when $\sigma^2 \rightarrow \infty$ reduces to a uniform prior). We only provide a Bayesian perspective here, and leave the detailed discussion to future works.
Figure 1: Training (top row) and test (bottom row) accuracy of CNNs on Cifar10 dataset. We report confidence interval $[\mu \pm \sigma]$ of 3 independent runs.

F. Experimental Details

1. Image classification with CNNs on Cifar

We performed experiments based on the official implementation\(^2\) of AdaBound [12], and exactly replicated the results of AdaBound as reported in [12]. We then experimented with different optimizers under the same setting: for all experiments, the model is trained for 200 epochs with a batch size of 128, and the learning rate is multiplied by 0.1 at epoch 150. We performed extensive hyperparameter search as described in the main paper. In the main paper we only report test accuracy; here we report both training and test accuracy in Fig. 1 and Fig. 2. AdaBelief not only achieves the highest test accuracy, but also a smaller gap between training and test accuracy compared with other optimizers such as Yogi.

2. Image Classification on ImageNet

We experimented with a ResNet18 on ImageNet classification task. For SGD, we use the same learning rate schedule as [30], with an initial learning rate of 0.1, and multiplied by 0.1 at epoch 30 and 60; for AdaBelief, we use an initial learning rate of 0.001, and decayed it at epoch 70 and 80. Weight decay is set as $10^{-4}$ for both cases. To match the settings in [7] and [16], we use decoupled weight decay. As shown in Fig. 3, AdaBelief achieves an accuracy very close to SGD, closing the generalization gap between adaptive methods and SGD. Meanwhile, when trained with a large learning rate (0.1 for SGD, 0.001 for AdaBelief), AdaBelief achieves faster convergence than SGD in the initial phase.

3. Robustness to hyperparameters

**Robustness to $\epsilon$**  We test the performances of AdaBelief and Adam with different values of $\epsilon$ varying from $10^{-4}$ to $10^{-9}$ in a log-scale grid. We perform experiments with a ResNet34 on Cifar10 dataset, and summarize the results in Fig. 4. Compared with Adam, AdaBelief is slightly more sensitive to the choice of $\epsilon$, and achieves the highest accuracy at the default value $\epsilon = 10^{-8}$; AdaBelief achieves accuracy higher than 94% for all $\epsilon$ values, consistently outperforming Adam which achieves an accuracy around 93%.

\(^2\)https://github.com/Luolc/AdaBound
Robustness to learning rate We test the performance of AdaBelief with different learning rates. We experiment with a VGG11 network on Cifar10, and display the results in Fig. 5. For a large range of learning rates from $5 \times 10^{-4}$ to $3 \times 10^{-3}$, compared with Adam, AdaBelief generates higher test accuracy curve, and is more robust to the change of learning rate.

4. Experiments with LSTM on language modeling

We experiment with LSTM models on Penn-TreeBank dataset, and report the results in Fig. 6. Our experiments are based on this implementation\(^3\). Results $[\mu \pm \sigma]$ are measured across 3 runs with independent initialization. For completeness, we plot both the training and test curves.

We use the default parameters $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ for 2-layer and 3-layer models; for 1-layer model we set $\epsilon = 10^{-12}$ and set other parameters as default. For simple models (1-layer LSTM), AdaBelief’s perplexity is very close to other optimizers; on complicated models, AdaBelief achieves a significantly lower perplexity on the test set.

---

3. https://github.com/salesforce/awd-lstm-lm
Figure 4: Training (top row) and test (bottom row) accuracy of ResNet34 on Cifar10, trained with AdaBelief (left column) and Adam (right column) using different values of $\epsilon$. Note that AdaBelief achieves an accuracy above 94% for all $\epsilon$ values, while Adam’s accuracy is consistently below 94%.

Figure 5: Training (top row) and test (bottom row) accuracy of VGG on Cifar10, trained with AdaBelief (left column) and Adam (right column) using different values of learning rate.
5. Experiments with GAN

We experimented with a WGAN [21] and WGAN-GP [39]. The code is based on several public github repositories [4, 5]. We summarize network structure in Table 1. For WGAN, the weight of discriminator is clipped within $[-0.01, 0.01]$; for WGAN-GP, the weight for gradient-penalty is set as 10.0, as recommended by the original implementation. For each optimizer, we perform 5 independent runs. We train the model for 100 epochs, generate 64,000 fake samples (60,000 real images in Cifar10), and measure the Frechet Inception Distance (FID) [40] between generated samples and real samples. Our implementation on FID heavily relies on an open-source implementation [6]. We report the FID scores in the main paper, and demonstrate fake samples in Fig. 7 and Fig. 8 for WGAN and WGAN-GP respectively.

We also experimented with Spectral Normalization GAN based on a public repository [7]. For this experiment, we set $\epsilon = 10^{-16}$ and use the rectification technique as in RAdam. Other hyperparameters and training schemes are the same as in the repository.

Table 1: Structure of GAN

| Generator | Discriminator |
|-----------|---------------|
| Conv Transpose ([inchannel = 100, outchannel = 512, kernel = 4×4, stride = 1]) | Conv2D([inchannel=3, outchannel=64, kernel = 4×4, stride=2]) BN-Relu |
| Conv Transpose ([inchannel = 512, outchannel = 256, kernel = 4×4, stride = 2]) | Conv2D([inchannel=64, outchannel=128, kernel = 4×4, stride=2]) BN-LeakyReLU |
| Conv Transpose ([inchannel = 256, outchannel = 128, kernel = 4×4, stride = 2]) | Conv2D([inchannel=128, outchannel=256, kernel = 4×4, stride=2]) BN-LeakyReLU |
| Conv Transpose ([inchannel = 128, outchannel = 64, kernel = 4×4, stride = 2]) | Conv2D([inchannel=256, outchannel=512, kernel = 4×4, stride=2]) BN-LeakyReLU |
| Conv Transpose ([inchannel = 64, outchannel = 3, kernel = 4×4, stride = 2]) | Linear([1, 1]) |

References:

4. https://github.com/pytorch/examples
5. https://github.com/eriklindernoren/PyTorch-GAN
6. https://github.com/mseitzer/pytorch-fid
7. https://github.com/POSTECH-CVLab/PyTorch-StudioGAN
Figure 7: Fake samples from WGAN trained with different optimizers.
Figure 8: Fake samples from WGAN-GP trained with different optimizers.