FastAdaBelief: Improving Convergence Rate for Belief-Based Adaptive Optimizers by Exploiting Strong Convexity

Yangfan Zhou, Kaizhu Huang, Cheng Cheng, Xuguang Wang, Amir Hussain, and Xin Liu, Member, IEEE

Abstract—AdaBelief, one of the current best optimizers, demonstrates superior generalization ability over the popular Adam algorithm by viewing the exponential moving average of observed gradients. AdaBelief is theoretically appealing in which it has a data-dependent $O\left(\sqrt{T}\right)$ regret bound when objective functions are convex, where $T$ is a time horizon. It remains, however, an open problem whether the convergence rate can be further improved without sacrificing its generalization ability. To this end, we make the first attempt in this work and design a novel optimization algorithm called FastAdaBelief that aims to exploit its strong convexity in order to achieve an even faster convergence rate. In particular, by adjusting the step size that better considers strong convexity and prevents fluctuation, our proposed FastAdaBelief demonstrates excellent generalization ability and superior convergence. As an important theoretical contribution, we prove that FastAdaBelief attains a data-dependent $O\left(\log T\right)$ regret bound, which is substantially lower than AdaBelief in strongly convex cases. On the empirical side, we validate our theoretical analysis with extensive experiments in scenarios of strong convexity and nonconvexity using three popular baseline models. Experimental results are very encouraging: FastAdaBelief converges the quickest in comparison to all mainstream algorithms while maintaining an excellent generalization ability, in cases of both strong convexity or nonconvexity. FastAdaBelief is, thus, posited as a new benchmark model for the research community.

Index Terms—Adaptive learning rate, image classification, online learning, optimization algorithm, stochastic gradient descent, strong convexity.

I. INTRODUCTION

THE training process is a significant stage in many fields of artificial neural networks, such as deep learning [1], transfer learning [2], and metalearning [3]. From an optimization perspective, the purpose of the training process is to minimize (or maximize) the loss value (or reward value) and, thus, can be considered as an optimization process [4]. As a popular paradigm, the training process can be conducted in a supervised way that requires a large number of labeled samples in order to achieve satisfactory performance [5], while, on the one hand, this can be very difficult to deploy in practice due to the high cost involved in annotating samples manually (or even automatically) [6]; on the other hand, even with sufficient labeled data, it is still a formidable challenge on how to design both fast and accurate training or optimization algorithms. To tackle this problem, many researchers have made attempts to improve the convergence speed of optimization algorithms, so as to both reduce the need for labeled samples and speed up the training process with available data at hand [7], [8]. Specifically, online learning is often used to accomplish such training tasks since it does not require information to be collected in batches at the same time [9].

One classic online optimization algorithm is online stochastic gradient descent (SGD) [10]. SGD has been extensively applied over the last few decades in many training tasks of deep learning due to its simple logic and good generalization ability [11], [12]. However, SGD suffers from the limitation of slow convergence. This disadvantage hinders its application, especially in large-scale problems, which may take extremely long to converge. To address this issue, researchers have developed various methods to speed up the SGD convergence rate. For example, one type of method focuses on exploring first-order momentum to accelerate SGD; such methods include SGD with momentum [13] and Nesterov momentum [14]. Typically, adopting a fixed step size, these methods may not be conducive to accelerate the convergence rate. To alleviate this problem, recent studies, including the popular Adam [15] and AMSGrad approaches [16], attempt to apply...
second-order momentum and prefer an adaptive step size while maintaining the first-order momentum.

As one of the most successful adaptive online algorithms, Adam enjoys a fast convergence, which is guaranteed with the regret bound of $O(\sqrt{T})$. Despite its outstanding performance, Reddi et al. [16] indicated that Adam has the issue of nonconvergence, which is caused by not satisfying $\Gamma_i > 0$ for all $t \in \{1, \ldots, T\}$, where $\Gamma_i = (\sqrt{\gamma_i} - a_i) - ((\gamma_{i-1})^{1/2} - a_{i-1})$. Moreover, another limitation with Adam is that it can lead to poorer generalization ability compared to SGD. To tackle this issue, many variants of Adam have been further proposed. For instance, Luo et al. [17] proposed AdaBound with a dynamic bound on the learning rate; Zaheer et al. [18] considered the effect of increasing minibatch size and proposed Yogi; Liu et al. [19] developed RADam to rectify the variance of the learning rate; Balles and Hennig [20] dissected Adam in practice. On closer examination, FastAdaBelief adopts the new second-order form that is significantly different from the form of SAdam; this posits a new challenge for the convergence analysis of FastAdaBelief. In addition, in order to fit strongly convex conditions, FastAdaBelief designs a tailored diagonal matrix of the second-order momentum, which also leads to a nontrivial challenge in the convergence analysis compared to Adam. In summary, the performance based on the convergence and generalization ability of FastAdaBelief and the current mainstream optimizers can be seen in Table I.

| Optimizer                  | Loss Function | Regret Bound | Convergence | Generalization |
|----------------------------|---------------|--------------|-------------|---------------|
| SGD ([10])                 | convex        | $O(\sqrt{T})$ | slow        | excellent     |
| Adam ([15])                | convex        | $O(\sqrt{T})$ | medium      | poor          |
| SAdam ([23])               | strongly convex | $O(\log T)$ | fast        | poor          |
| AdaBelief ([22])           | convex        | $O(\sqrt{T})$ | medium      | excellent     |
| FastAdaBelief (Ours)       | strongly convex | $O(\log T)$ | fast        | excellent     |

Our major contributions are summarized as follows.

1) We propose a fast variant of AdaBelief, named FastAdaBelief, to further improve the convergence rate under strongly convex conditions. We show that FastAdaBelief can lead to an adaptive step size that is more in line with an ideal optimizer.

2) We provide a convergence analysis for FastAdaBelief that presents a data-dependent $O(\sum_{i=1}^{n} \log(\|g_{1:T,i}\|^2))$ guaranteed regret bound, which is substantially better than AdaBelief.

3) We conduct extensive experiments to demonstrate that FastAdaBelief outperforms other state-of-the-art mainstream optimization algorithms in a variety of tasks. Interestingly, even in the case of nonconvexity, FastAdaBelief shows consistently superior performance over other state-of-the-art algorithms with all benchmark datasets.

II. NOTATION AND PRELIMINARIES

A. Notation

Since this article uses a lot of symbols, for brevity, we summarize the notations in Table II.

B. Online Learning

Machine learning (ML) plays an important role in the field of artificial intelligence. Moreover, offline learning in ML is usually expected to enable a batch of tasks at the same time, but this situation is difficult to meet. In contrast, online learning based on regret considers a sequential setting in which tasks are revealed one by one. Online learning can better adapt to complex and changeable practical applications and has become a prominent paradigm for ML, which is attractive in both theory and practice [24]. Within this paradigm, a learner
where $g_i$ iteratively generates a decision $x_t$, from a convex and compact domain $\mathcal{F} \subset \mathbb{R}^n$ in each round $t \in \{1, \ldots, T\}$. In response, an adversary produces a convex loss function $f_t(x)$ in round $t$, which causes the learner to suffer the loss $f_t(x_t)$. The goal of the learner is to generate a decision $x_t$ so that the regret can decrease quickly as $T$. Moreover, regret is defined as follows:

$$ R(T) = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{F}} \sum_{t=1}^{T} f_t(x). \quad (1) $$

To improve the generalization ability of the Adam optimizer family, AdaBelief fully considers the curvature information of loss functions, which will be introduced as part of the preliminary background in Section II-C.

### C. AdaBelief

The algorithm design of AdaBelief is shown in Algorithm 1. Review that Adam designs its second-order momentum as the gradient $\nabla_x f_t(x_t)$, which will be introduced as part of the preliminary background in Section II-C.

**Algorithm 1 AdaBelief**

**Input:** $\beta_1, \beta_2$

**Output:** $x_{t+1}$

1. Initialize: $x_0, m_0, s_0$
2. for $t = 1 \ldots T$
   1. $t \leftarrow t + 1$
   2. $\alpha_t \leftarrow \frac{\alpha}{t}$
   3. $g_t \leftarrow \nabla f_t(x_t)$
   4. $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)g_t$
   5. $s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2)(g_t - m_t)^2$
   6. $\hat{s}_t \leftarrow \max(s_{t-1}, s_t)$
   7. $\hat{S}_t \leftarrow \text{diag}([\hat{s}_t])$
   8. $x_{t+1} \leftarrow \prod_{t \in \mathcal{F}} \sqrt{\frac{g_t}{\sqrt{\hat{S}_t} + \epsilon}}$
11. return $x_{t+1}$

**Table II**

| Symbol | Meaning |
|--------|---------|
| x | lowercase bold letters represent vectors |
| $x_t$ | the value of vector $x$ at time $t$ |
| $x_{t,i}$ | the $i$-th coordinate of vector $x_t$ |
| $M$ | capital letters represent matrices |
| $M^n_+$ | the set of $n$ dimensional positive definite matrices |
| $\| \cdot \|$ | the $\ell_2$-norm |
| $\| \cdot \|_\infty$ | the $\ell_\infty$-norm |
| $\|x\|_M^2 = x^T M x$ | the $M$-weighted $\ell_2$-norm |
| $f_t(\cdot)$ | the loss function at time $t$ |
| $g_t$ | the gradient of the loss function $f_t(\cdot)$ |
| $g_t: T \mapsto [g_{t,1}, \ldots, g_{t,T}]$ | the sequence consisting of the $t$-th element of the gradient sequence $[g_{1,1}, \ldots, g_{T,T}]$ |
| $\prod_{s \in \mathcal{F}} R(x) = \arg \min_{x \in \mathcal{F}} \|y - x\|_M^2$ | the $M$-weighted projection operation of $x$ on $\mathcal{F}$, where $x \in \mathbb{R}^n$ |
| $x^\partial$ | the element-wise square, where $x \in \mathbb{R}^n$ |
| $\frac{x}{y}$ | the element-wise division, where $x, y \in \mathbb{R}^n$ |
| $\sqrt{x}$ | the element-wise square root, where $x \in \mathbb{R}^n$ |
| $A = \text{diag}(x)$ | the fact that $A$ is a diagonal matrix composed of the elements of vector $x$ |
| $I_d$ | a $d \times d$ identity matrix |
| $x^\ast$ | the best decision in hindsight, i.e., $x^\ast = \min_{x \in \mathcal{F}} \sum_{t=1}^{T} f_t(x)$ |

### FastAdaBelief: Improving Convergence Rate for Belief-Based Adaptive Optimizers by Exploiting Strong Convexity

An ideal optimizer considers the curvature of the loss function and prefers adaptive step size. The algorithm design of AdaBelief is shown in Algorithm 1. In region 1, where the loss function is flat, the gradient $g_t$ and $|g_t(x_1) - g_t(x_2)|$ are both very large. In fact, a large step size should be taken, in this case, for the efficiency of the optimizer. In this case, AdaBelief and Adam both take large step sizes, but SGD takes a small one.

In region 2, the loss function is “steep.” Hence, the gradient $g_t$ and $|g_t(x_8) - g_t(x_9)|$ are both very large. For this reason, an ideal optimizer should take a small step size. Moreover, by the design of the second-order momentum in AdaBelief and Adam, they take a small step size in this case, while SGD exploits a large step size.

momentum is designed as

$$ s_t = \beta_2 s_{t-1} + (1 - \beta_2)(g_t - m_t)^2 $$

where $m_t$ is the first-order momentum.
Algorithm 2 FastAdaBelief

\begin{algorithm}
\caption{FastAdaBelief}
\begin{algorithmic}[1]
\State \textbf{Input:} \{\(\beta_{1t}\)\}_{t=1}^{T}, \{\(\beta_{2t}\)\}_{t=1}^{T}, \delta
\State \textbf{Output:} \(x_{t+1}\)
\State \textbf{Initialize:} \(x_0, m_0, s_0\)
\For {\(t = 1 \ldots T\)}
\State \(t \leftarrow t + 1\)
\State \(\alpha_t \leftarrow \frac{\delta}{t}\)
\State \(g_t \leftarrow \nabla f_t(x_t)\)
\State \(m_t \leftarrow \beta_{1t} m_{t-1} + (1 - \beta_{1t}) g_t\)
\State \(s_t \leftarrow \beta_{2t} s_{t-1} + (1 - \beta_{2t}) (g_t - m_t)^2\)
\State \(\hat{s}_t \leftarrow \max(\hat{s}_{t-1}, s_t)\)
\State \(\hat{S}_t \leftarrow \text{diag}(\hat{s}_t) + \frac{\delta}{t} I_n\)
\State \(x_{t+1} \leftarrow \prod_{\mathcal{F}, \hat{S}_t} (x_t - \alpha_t \hat{S}_t^{-1} m_t)\)
\EndFor
\State \textbf{return} \(x_{t+1}\)
\end{algorithmic}
\end{algorithm}

In summary, AdaBelief fully considers all the above curvature situations and adopts a good step size selection strategy in each situation. For this reason, AdaBelief has the same generalization ability as the SGD optimizer family. Though AdaBelief retains the same regret bound guarantee \(O(\sqrt{T})\) as Adam, it is interesting to explore if it can be further sped up. To this end, we propose to utilize the strong convexity and develop a new model that is able to advance the regret bound of Adabelief to logarithmic convergence in this article.

III. FASTADABELIEF

In this section, we first present the detailed design of the proposed algorithm and then analyze the theoretical guarantee of its regret bound.

A. Algorithm Design

Before presenting the proposed algorithm, we introduce some standard definitions and general assumptions, which follows previous works, including [15], [16], [22], [23], [25].

**Definition 1:** A function \(f(\cdot): \mathcal{F} \to \mathbb{R}\) is \(\sigma\)-strongly convex, where \(\sigma\) is a positive constant, if, for all \(x, y \in \mathcal{F}\), the following equation is satisfied:

\[ f(x) - f(y) \geq (\nabla f(y))^\top (x - y) + \frac{\sigma}{2} \|x - y\|^2. \]  

**Assumption 1:** The feasible region \(\mathcal{F} \subseteq \mathbb{R}^n\) is bounded, that is, for all \(x, y \in \mathcal{F}\), \(\max_{x, y \in \mathcal{F}} \|x - y\| \leq D_\infty\), where \(D_\infty > 0\) is a constant.

**Assumption 2:** For all \(t \in \{1, \ldots, T\}\), the gradients of all loss functions, \(\{\nabla f_t(x)\}_{t=1}^{T}\), are bounded. Specifically, there exists a constant \(G_\infty > 0\) such that \(\max_{x \in \mathcal{F}} \|\nabla f_t(x)\| \leq G_\infty\).

Next, we present an accelerated and accurate belief-based optimization algorithm for strongly convex functions based on the above standard definitions and assumptions, called FastAdaBelief.

The detailed design of the proposed algorithm is shown in Algorithm 2, which follows the general design of [22]. In the proposed algorithm, \(\beta_{1t}\) and \(\beta_{2t}\) are time-variant nonincreasing hyperparameters, and \(\delta\) is a positive constant. Moreover, the parameter of step size, \(\alpha_t\), is assigned as \(\alpha_t = (\alpha/1)\), where \(\alpha\) is a constant. Furthermore, the gradient of loss function at time \(t\), \(g_t\), is calculated by \(g_t = \nabla f_t(x_t)\). Next, the proposed algorithm computes the first-order momentum, \(m_t\), through the exponential moving average (EMA) of \(g_t\), which is shown as follows:

\[ m_t = \beta_{1t} m_{t-1} + (1 - \beta_{1t}) g_t. \]  

Then, the second-order momentum, \(s_t\), in the proposed algorithm is calculated by EMA of the square of the observed gradient belief \((g_t - m_t)^2\), that is,

\[ s_t = \beta_{2t} s_{t-1} + (1 - \beta_{2t}) (g_t - m_t)^2. \]  

Moreover, to satisfy the condition of convergence, i.e., \(\Gamma_t = (\sqrt{\hat{s}_t}/\alpha_t - ((\hat{s}_{t-1})^{1/2}/\alpha_{t-1}) \geq 0\), the proposed algorithm further provides the following operation on the second-order momentum:

\[ \hat{s}_t = \max(\hat{s}_{t-1}, s_t). \]

Furthermore, to avoid step size explosion caused by too small gradients, the proposed algorithm adds a vanishing factor \((\delta/1)\) to the second-order momentum and obtains the following diagonal matrix:

\[ \hat{S}_t = \text{diag}(\hat{s}_t) + \frac{\delta}{1} I_n. \]

Finally, the proposed algorithm updates the decision point, \(x_{t+1}\), conditional on the projection to the feasible region and attains the following:

\[ x_{t+1} = \prod_{\mathcal{F}, \hat{S}_t} (x_t - \alpha_t \hat{S}_t^{-1} m_t). \]

In general, the proposed algorithm is designed to incorporate two key enhancements compared to AdaBelief. The first relates to the step size, which is modified to \((\alpha/t)\). The motivation behind this is to satisfy the property of strongly convex optimization. Moreover, the second enhancement is to change \(\beta_2\) of AdaBelief to \(\beta_{2t}\). The time-varying parameter, \(\beta_{2t}\), is set to constant \(\beta_2\) in AdaBelief, which simplifies application and convergence proof but can lead to step size fluctuations. In this work, we apply \(\beta_{2t}\) in its original form that achieves good convergence [16].

The detailed design of our proposed algorithm has been introduced in this section. Next, we interpret why the proposed FastAdaBelief can choose a better step size and leads to faster convergence. Following this, we theoretically prove that, when strong convexity of loss functions holds, our proposed algorithm has a guaranteed regret bound, which is much better than AdaBelief.

B. Why FastAdaBelief Can Choose a Better Step Size?

From the design of \(\hat{s}_t\) in FastAdaBelief, our algorithm adds a vanishing factor to the step size, which was originally considered to meet the strongly convex condition, but unexpectedly brings significant benefits to the choice of the step size. If we let \(\Delta\) denote the step size, then step sizes of SGD, Adam, SAdam, AdaBelief, and FastAdaBelief are shown in the following:

\[ \Delta_l(SGD) = \alpha m_t \]
\[ \Delta_l(Adam) = \alpha m_t / (\sqrt{\nu_t}) \]
\[ \Delta_l(SAdam) = \alpha m_t / (\nu_t + \delta) \]
\[ \Delta_l(AdaBelief) = \alpha m_t / (\sqrt{\nu_t}) \]
\[ \Delta_l(FastAdaBelief) = \alpha m_t / (\nu_t + \delta). \]
It can also be seen from Fig. 1 that, in regions 1 and 2, although the step size of FastAdaBelief is slightly smaller than that of AdaBelief, it is still consistent with the optimal choice. FastAdaBelief has large step sizes in both regions 1 and 2, but SAdam takes small step sizes. Thus FastAdaBelief outperforms SAdam with respect to generalization ability. Importantly, the step size of FastAdaBelief decays in general on the order of $O(1/t)$ that allows the optimal solution to be approximated at a smaller step size later in the training process without unnecessary oscillations. In addition, in region 3, the ideal optimizer would prefer a small step size. FastAdaBelief takes a smaller step size than AdaBelief in this region, which is due to the addition of vanishing factors.

In summary, a comparison of step size selection by FastAdaBelief, AdaBelief, SAdam, Adam, and SGD can be seen in Table III. This analysis shows that FastAdaBelief, such as AdaBelief, is in line with the choice of the ideal optimizer and, therefore, can lead to better performance than other mainstream optimizers. Moreover, FastAdaBelief has a smaller step size than that of AdaBelief in the later stage of training, which allows the optimizer to approximate the optimal solution more steadily. Therefore, FastAdaBelief, such as AdaBelief, has a better generalization ability than other algorithms when training deep models.

### C. Theoretical Guarantee

In this section, we first review some convergence conditions as developed by Reddi et al. [16], which solves the convergence issue for Adam [15]. Let $\{\beta_t\}$ satisfy the following conditions.

1) **Condition 1:** For some $\zeta > 0$ and all $t \in \{1, \ldots, T\}$, $j \in \{1, \ldots, n\}$, we have that

$$\sqrt{\frac{t}{\alpha}} \sqrt{\frac{t}{\alpha}} \sum_{i=1}^{T} \sum_{k=1}^{i} \beta_2(t-k+1)(1-\beta_2)g_{t,i}^2 \geq \frac{1}{\zeta} \sum_{i=1}^{T} \sum_{j=1}^{n} g_{j,i}^2.$$

2) **Condition 2:** For all $t \in \{1, \ldots, T\}$ and $i \in \{1, \ldots, n\}$, we have that

$$\sqrt{\frac{t}{\alpha}} \sqrt{\frac{t}{\alpha}} \sum_{i=1}^{T} \sum_{k=1}^{i} \beta_2(t-k+1)(1-\beta_2)g_{t,i}^2 \geq \frac{1}{\zeta} \sum_{i=1}^{T} \sum_{j=1}^{n} g_{j,i}^2.$$

As a matter of fact, Condition 1 is an important and standard condition for the convergence analysis of adaptive momentum algorithms, such as Adam and AdaBelief. Furthermore, the intrinsic motivation for Condition 2 is to follow the key condition of SGD, where its step size ($\alpha/\sqrt{t}$) satisfies that $(\sqrt{t}/\alpha) - ((t-1)^{1/2}/\alpha) \geq 0, \forall t \in \{T\}$. For this reason, we also follow this motivation and propose the following conditions with minor modifications.

3) **Condition 3:** For some $\zeta > 0$ and all $t \in \{1, \ldots, T\}, j \in \{1, \ldots, n\}$, we have that

$$t \sum_{i=1}^{T} \sum_{k=1}^{i} \beta_2(t-k+1)(1-\beta_2)g_{t,i}^2 \geq \frac{1}{\zeta} \sum_{i=1}^{T} \sum_{j=1}^{n} g_{j,i}^2.$$  \hspace{1cm} (8)

4) **Condition 4:** For all $t \in \{1, \ldots, T\}$ and $i \in \{1, \ldots, n\}$, we have that

$$0 \leq \frac{t}{\alpha} s_{t,i}^{1/2} - \frac{t-1}{\alpha} s_{t-1,i}^{1/2} \leq \sigma (1-\beta_t).$$  \hspace{1cm} (9)

The details on Condition 3 and Condition 4 are shown in Appendix B.

Now, we present the main results in the following for the convergence analysis when Conditions 3 and 4 are satisfied.

**Theorem 1:** Suppose that Assumptions 1 and 2 are satisfied, Conditions 3 and 4 hold, and loss functions $f_t(\cdot)$ are $\sigma$-strongly convex. Moreover, let parameter sequences $\{\beta_t\}$, $\{\beta_2\}$, and $\{\alpha_t\}$ be generated by the proposed algorithm, where $\beta_{t+1} = \beta_t \lambda_t$, $\beta_1 \in [0, 1)$, $\lambda_t \in [0, 1)$, $\beta_{t+1} \in [0, 1)$, $\delta > 0, \forall t \in \{1, \ldots, T\}$. For decision point $x$, generated by the proposed algorithm, we have the following upper bound of the regret:

$$R(T) \leq \frac{n \delta D_{\infty}^2}{2\alpha(1-\beta_1)} + \frac{D_{\infty}^2(G_\infty + \delta)}{2\alpha} \sum_{i=1}^{n} \sum_{i=1}^{T} \beta_{t}t^{s_{t,i}^{1/2}}$$

$$= \frac{\alpha \zeta}{\sigma^2(1-\beta_1)^3} \sum_{i=1}^{n} \log \left( \frac{1}{\delta} \|g_{1:T,i}\|_2^2 + 1 \right).$$

The proof of Theorem 1 is provided in Appendix A. Accordingly, Theorem 1 implies that our proposed algorithm converges with $O(\sum_{i=1}^{n} \log(\|g_{1:T,i}\|_2^2))$ regret bound in the case of strong convexity. Moreover, the regret bound of the worst case is $O(n \log T)$. In addition, the bound of the regret can be more tighter if the gradients are sparse or small such that $\|g_{1:T,i}\|_2^2 < T G_{\infty}^2$.

**Corollary 1:** Letting $\beta_{t+1} = \beta_t \lambda_t$, where $\lambda \in (0, 1)$ in Theorem 1, we have the following upper bound of the regret:

$$R(T) \leq \frac{n \delta D_{\infty}^2}{2\alpha(1-\beta_1)} + \frac{n \delta \beta_1 \lambda T^2(G_{\infty} + \delta)}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

$$= \frac{\alpha \zeta}{\sigma^2(1-\beta_1)^3} \sum_{i=1}^{n} \log \left( \frac{1}{\delta} \|g_{1:T,i}\|_2^2 + 1 \right).$$

Corollary 1 also implies that our proposed algorithm has a convergence guarantee $O(n \log T)$ for condition $\beta_{t+1} = \beta_t \lambda_t$, $\lambda \in (0, 1), t \in \{1, \ldots, T\}$. Then, our proposed...
algorithm executes with $\lim_{T \to +\infty} (R(T)/T) = 0$. Therefore, our proposed algorithm converges when loss functions are strongly convex, and its theoretical proof is provided in Appendix A. In order to verify the performance of our algorithm in specific applications, we present a series of experiments on benchmark public datasets in the next section.

**IV. EXPERIMENTS**

In this section, we conduct two groups of experiments to verify that our proposed algorithm works excellently for benchmark optimization problems in cases of strong convexity and nonconvexity. In the first group, we consider a strongly convex optimization problem of minibatch $\ell_2$-regularized softmax regression; in the second group, we apply our algorithm to nonconvex cases of deep training tasks with the traditional softmax function. To be specific, the traditional softmax function is generally convex but not strongly convex when the data is uniformly distributed. For this reason, the softmax function does not often satisfy convexity in deep neural network applications due to the sparsity of data and the nonlinearity of deep neural networks [27], [28]. Therefore, deep learning tasks are generally nonconvex optimization problems [29]. To examine the effectiveness of FastAdaBelief in real scenarios, we intentionally conduct the second group of experiments on image classification with CNN and language modeling with LSTM, respectively, both of which are commonly seen in practice. In all of our experiments, the source codes are implemented in the torch 1.1.0 module of Python 3.6 and executed on $4 \times 1080ti$ GPUs. Furthermore, we compare FastAdaBelief with the other algorithms in both experiments, including SGD [13], Adam [15], Yogi [18], AdaBound [17], AdaBelief [22], and SAdam [23]. We independently execute the experiments five times and, finally, report the top-one of them, which follows [18], [22].

**A. Hyperparameter Tuning**

We perform the following hyperparameter tuning in experiments of image classification and language modeling. To be fair, we initialize the decision variables and momentum of each algorithm to $x_0 = 0$, $m_0 = 0$, $v_0 = 0$, and $s_0 = 0$. Moreover, we choose the parameters of each algorithm exactly as suggested in the original articles. Specifically, the parameter settings for each algorithm are given as follows.

**SGD:** We follow the standard settings of ResNet [30] and DenseNet [31], and set the momentum as 0.9. We choose the learning rate from $\{10.0, 1.0, 0.1, 0.01, 0.001\}$.

**Adam:** We adopt the same parameter setting as the original article [15], where the first-order momentum $\beta_1$ is set to 0.9, and the second-order momentum $\beta_2$ is set to 0.999. Moreover, the step size $\alpha_t$ is set to $\alpha/\sqrt{T}$, where $\alpha$ is chosen from $\{0.1, 0.01, 0.001, 0.0001\}$.

**Yogi:** Following [18], we set the first-order momentum $\beta_1$ to 0.9 and set the second-order momentum $\beta_2$ to 0.999.

In addition, the step size $\alpha_t$ is set to $\alpha/\sqrt{T}$, where $\alpha$ is chosen from $\{0.1, 0.01, 0.001, 0.0001\}$.

**AdaBound:** We directly apply the default hyperparameters following [17] for AdaBound (i.e., $\beta_1 = 0.9$ and $\beta_2 = 0.999$). Moreover, the step size $\alpha_t$ is set to $\alpha/\sqrt{T}$, where $\alpha$ is chosen from $\{0.1, 0.01, 0.001, 0.0001\}$.

**AdaBelief:** We use the default hyperparameters as suggested in [22], i.e., $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. In addition, the step size $\alpha_t$ is set to $\alpha/\sqrt{T}$, where $\alpha$ is chosen from $\{0.1, 0.01, 0.001, 0.0001\}$.

**FastAdaBelief:** We adopt the same hyperparameters as SAdam: $\beta_1 = 0.9$ and $\beta_2 = 1 - (0.9/t)$. Moreover, the step size is set to $\alpha/\sqrt{T}$, where $\alpha$ is chosen from $\{0.1, 0.01, 0.001, 0.0001\}$.

It can be seen that, for a fair comparison, all the above algorithms basically follow a similar parameter setting and maintain the parameter suggestions provided in the original algorithms.

**B. Datasets**

In the experiments of CNN-based image classification, we perform evaluations on the benchmark CIFAR-10 dataset. Moreover, we apply the algorithms on three standard baseline models, i.e., DenseNet-121, ResNet-34, and VGG-11. DenseNet-121 is a dense convolutional network, which connects each layer to all other layers feedforwardly; ResNet-34 is a residual learning framework; and VGG-11 is a deep network using an architecture with small convolution filters. In the LSTM-based language modeling experiments, we test the various algorithms on the Penn Treebank dataset. Furthermore, we compare the algorithms in one-, two-, and three-layer LSTM models. For clarity, we show the summary of datasets and architectures used in our experiments in Table IV.

**C. Optimization With Strong Convexity**

In this group of experiments, we consider a minibatch task. In round $t$ of this task, the optimizer receives a minibatch of training samples denoted by $\{x_m, y_m\}_{m=1}^m$, where $m$ is the batch size, $K$ is the number of classes, and $y_i \in [K]$ $\forall i \in [m]$. Then, the optimizer generates decision vectors denoted by $\{w_i, b_i\}_{i=1}^K$. Finally, the generated result suffers a loss. The loss function is then given as

$$J(w) = -\frac{1}{m} \sum_{i=1}^m \log \left( \frac{e^{w_i x_i + b_i}}{\sum_{j=1}^K e^{w_j x_i + b_j}} \right) + \sigma_1 \sum_{k=1}^K \|w_k\|^2$$

$$+ \sigma_2 \sum_{k=1}^K b_k^2.$$  (10)
In our experiments, we set parameters $\sigma_1$ and $\sigma_2$ both to 0.01. In addition, we conduct the experiment on loss versus iterations. The results of this experiment are shown in Fig. 2. As clearly observed, the loss of our proposed algorithm decreases the quickest, and FastAdaBelief leads to the best convergence in all mainstream algorithms. As guaranteed theoretically, the strongly convex optimization algorithms (such as FastAdaBelief and SAdam) outperform convex optimization algorithms (such as Adam and AdaBelief) in the strongly convex case. When we inspect the difference between the two strongly convex optimization algorithms, FastAdaBelief generates much lower losses (particularly on CIFAR10 and CIFAR100) than SAdam, which echoes the advantages of FastAdaBelief over SAdam in relation to the generalization ability.

D. Training DNN With Nonconvexity

FastAdaBelief enjoys theoretical superiority over other mainstream optimizers when strong convexity holds. On the empirical side, the current DNNs may, however, adopt loss functions that are typically not strongly convex (e.g., only convex). Thus, it is both interesting and important to investigate if the proposed fast algorithm can still work well. For this purpose, we next conduct a series of empirical studies on tasks of image classification and language modeling.

1) Image Classification: In the experiments of image classification, we take CIFAR-10 as one typical example and compare the various algorithms with DenseNet-121, ResNet-34, and VGG-11. First, we compare the convergence rate for all the algorithms used in our experiment. Such results are reported in Fig. 3. As clearly observed, though the strong convexity may not hold, FastAdaBelief still leads to remarkable convergence, which is consistently faster than all the other algorithms. In comparison, SAdam also converges well, which empirically demonstrates the power of strongly convex algorithms. Furthermore, SGD converges the slowest; Adam and AdaBelief are also much slower than both SAdam and FastAdaBelief. All these empirical results are consistent with the theoretical analysis, as discussed earlier in Sections II and III, though the loss functions are not strongly convex.

Second, we record the training and test accuracy curves of all algorithms executed in our experiments, which are shown in Figs. 4 and 5. We can see that FastAdaBelief outperforms other comparison algorithms in 200 epochs on DenseNet-121, ResNet-34, and VGG-11. Specifically, FastAdaBelief demonstrates much faster convergence and the highest accuracy within 200 epochs compared to all other algorithms on the three baseline DNN models. In addition, it is evident that AdaBelief and FastAdaBelief generally lead to the best accuracy in the 200 epoch, which verifies the excellent generalization ability of the belief-based adaptive algorithms. Note that SGD did not converge due to its slow convergence rate though it was able to catch up with the accuracy of AdaBelief and FastAdaBelief in the long run.

To sum up, the experiments of image classification with DenseNet-121, ResNet-34, and VGG-11 on CIFAR-10 validate the fast convergence rate and excellent accuracy performance of FastAdaBelief even when strong convexity does not hold in the loss functions.
2) Language Modeling: We also conduct a group of experiments on the language modeling task. In this group of experiments, we use a classic recurrent network (i.e., LSTM) and an open dataset (i.e., Penn Treebank). In line with previous works [22], [23], [32], [33], we take the perplexity to measure the performance of all algorithms under comparison. Note that a lower perplexity is better.

The perplexity curves of all the algorithms are shown in Fig. 6. In the figure, once again, we can see that FastAdaBelief performs similar to SAdam on 1-layer LSTM, and these two algorithms both perform better than the rest of the algorithms. However, on the two- and three-layer LSTMs, FastAdaBelief performs better than SAdam and the other algorithms. Moreover, the perplexity of FastAdaBelief decreases the fastest among all the algorithms, which further validates the convergence analysis of FastAdaBelief.

We also summarize the test perplexities of all algorithms compared in this group of experiments, which are shown in Table V. From the table, we can see that FastAdaBelief attains superior performance to the other algorithms on all three models (i.e., one-, two-, and three-layer LSTMs). In summary, FastAdaBelief retains both excellent generalization ability and a fast convergence rate for language modeling tasks even when loss functions are not strongly convex. These empirical results are very encouraging, suggesting that FastAdaBelief has a high potential to be widely applied in real scenarios.

E. Discussion on Convexity

In the above, we focus on the convergence benefits of algorithms under strongly convex conditions and propose a strongly convex optimization algorithm. We prove that it converges faster than convex algorithms, such as Adam and
AdaBelief, in strongly convex cases, which is also verified in the above experimental part. Moreover, the proposed algorithm empirically exhibits much faster convergence than Adam and AdaBelief in non-convex cases as well.

In the case of convexity, we provide the regret bound proof of FastAdaBelief to fully understand its advantages and disadvantages. The proof presented in Appendix D shows that FastAdaBelief actually converges slower than certain special convex algorithms, such as AdaBelief and Adam, in convex cases. Therefore, we suggest using the proposed FastAdaBelief in strongly convex and non-convex cases, while convex algorithms, e.g., Adabelief, can be used in convex cases.

V. CONCLUSION

In this article, we made the first attempt and presented an affirmative answer to the open question of whether AdaBelief can be further improved with respect to its convergence rate under the strongly convex condition. Specifically, we exploited strong convexity and proposed a novel algorithm named FastAdaBelief, which exhibits an even faster data-dependent regret bound of $O(\log T)$ while maintaining excellent generalization ability. In light of our theoretical findings, we carried out a series of empirical studies, which validated the superiority of our proposed algorithm. Importantly, we showed that FastAdaBelief converged the fastest in both strong convexity and nonconvexity cases, hence demonstrating its significant potential as a new benchmark model that can be widely utilized in various scenarios.

In our current work, we exploited the strong convexity of FastAdaBelief and empirically demonstrated its excellent generalization and fast convergence even when the loss functions are nonconvex. We believe that this may be partially attributable to the vanishing factor $\delta/t$, as also engaged in the second-order moment that enables a closer approximation to ideal step size. However, it remains unclear why this may happen strictly in theory. We will explore this in future work. In addition, while research on the sparsity of samples can improve the convergence rate of SGD, as demonstrated in [34]–[36], it remains unclear whether sparse samples will further improve FastAdaBelief’s convergence rate. We also leave this investigation as future work.

APPENDIX A

CONVERGENCE ANALYSIS IN STRONGLY

CONVEX ONLINE OPTIMIZATION

Before presenting the proof of Theorem 1, we first review the following Lemma 1.

Lemma 1 [26]: For all $M \in M^+_n$ and convex feasible region $\mathcal{F} \subseteq \mathbb{R}^n$, let

$$y_1 = \min_{x \in \mathcal{F}} \| M^{1/2}(x - z_1) \|$$

and

$$y_2 = \min_{x \in \mathcal{F}} \| M^{1/2}(x - z_2) \|.$$ 

Then, we obtain the following:

$$\| M^{1/2}(y_1 - y_2) \| \leq \| M^{1/2}(z_1 - z_2) \|.$$ 

**Theorem 1**: Suppose that Assumptions 1 and 2 are satisfied, Conditions 3 and 4 hold, and loss functions $f_i(\cdot)$ are $\sigma$-strongly convex. Moreover, let parameter sequences $\{\beta_{1t}\}$, $\{\beta_{2t}\}$, and $\{\alpha_t\}$ are generated by the proposed algorithm, where $\beta_{1t} = \beta_{1t}'$, $\beta_{1t} \in [0, 1]$, $\beta_{2t} \in [0, 1]$, $\delta > 0$, $t \in \{1, \ldots, T\}$. For decision point $x_t$ generated by the proposed algorithm, we have the following upper bound of the regret:

$$R(T) \leq \frac{n \delta D^2}{2(1 - \beta_1)} + \frac{D^2}{2(1 - \beta_1)} \sum_{i=1}^{n} \sum_{t=1}^{T} \beta_{1t} g_{1t}^2 + \frac{2\alpha \delta}{\sigma^2 (1 - \beta_1)^2} \sum_{t=1}^{n} \sum_{i=1}^{T} g_{i,t}^2 (1 + \| g_{i,t} \|).$$

**Proof**: By the updating method of decision variable, i.e., (7), we have

$$x_{t+1} = \min_{x \in \mathcal{F}} \| S^{1/2}(x - (x_t - \alpha_t S_t^{-1} m_t)) \|.$$ 

From the definitions of $x^*$ and projection $\prod(\cdot)$, we have that $x^* = \prod_{x \in \mathcal{F}}(x^*) = \min_{x \in \mathcal{F}}(x - x^*)$. In addition, if we apply Lemma 1, let $y_1 = x_{t+1}$, $y_2 = x^*$, by the update rules of $x_t$ and $m_t$, we obtain the following:

$$\| S_t^{1/2}(x_{t+1} - x^*) \|^2 \leq \| S_t^{1/2}(x_t - \alpha_t S_t^{-1} m_t - x^*) \|^2$$

Next, rearranging (12), we have that

$$\| g_t, x_t - x^* \| \leq \frac{2\alpha \delta}{(1 - \beta_{1t})} \| m_{t+1} - (1 - \beta_{1t}) g_t \|.$$ 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Applying Young’s inequality (i.e., \((a, b) \leq (a^2/2) + (b^2/2e), \forall e > 0\)) into the term \((a)\) of (13), and considering \(\alpha_t > 0, \beta_{tt} \in [0, 1]\), we can attain
\[
(a) = \frac{-\beta_{tt}}{1-\beta_{tt}}(m_{t-1}, x_t - x^*) \leq \frac{\alpha_t \beta_{tt}}{2(1-\beta_{tt})} \left( S_t^{1/2} m_{t-1} \right)^2 \\
+ \frac{\beta_{tt}}{2\alpha_t(1-\beta_{tt})} \left( S_t^{1/2} (x_t - x^*) \right)^2. \tag{14}
\]
Furthermore, applying the Cauchy–Schwarz inequality into (14), we have
\[
(a) \leq \frac{\alpha_t \beta_{tt}}{2(1-\beta_{tt})} \left( S_t^{-1/2} m_{t-1} \right)^2 \\
+ \frac{\beta_{tt}}{2\alpha_t(1-\beta_{tt})} \left( S_t^{1/2} (x_t - x^*) \right)^2. \tag{15}
\]
Then, plugging (15) into (13), we obtain the following:
\[
\{g_t, x_t - x^*\} \leq \frac{\alpha_t \beta_{tt}}{2(1-\beta_{tt})} \left( S_t^{-1/2} (x_{t-1} - x^*) \right)^2 \\
+ \frac{\beta_{tt}}{2\alpha_t(1-\beta_{tt})} \left( S_t^{1/2} m_{t-1} \right)^2 \\
+ \frac{\alpha_t \beta_{tt}}{2(1-\beta_{tt})} \left( S_t^{1/2} m_{t-1} \right)^2 \\
+ \frac{\beta_{tt}}{2\alpha_t(1-\beta_{tt})} \left( S_t^{1/2} (x_t - x^*) \right)^2. \tag{16}
\]
On the other hand, let \(x = x^*, y = x_t \in (2)\). With the strong convexity of \(f_t(\cdot)\), we attain
\[
f_t(x_t) - f_t(x^*) \leq \{g_t, x_t - x^*\} - \frac{\sigma}{2} \|x_t - x^*\|^2. \tag{17}
\]
Therefore, from definition of the regret [i.e., (1)], and (17), we obtain the following:
\[
R(T) = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in F} f_t(x) = \sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x^*) \\
\leq \sum_{t=1}^{T} \{g_t, x_t - x^*\} - \frac{\sigma}{2} \sum_{t=1}^{T} \|x_t - x^*\|^2. \tag{18}
\]
In addition, plugging (16) into (18), we can attain the following:
\[
R(T) \leq \sum_{t=1}^{T} \left[ \frac{\|S_{t}^{1/2}(x_t - x^*)\|^2 - \|S_{t}^{1/2}(x_{t-1} - x^*)\|^2}{2\alpha_t(1-\beta_{tt})} \right] \\
+ \sum_{t=1}^{T} \frac{\alpha_t}{2(1-\beta_{tt})} \|S_{t}^{-1/2} m_{t-1}\|^2 \\
+ \sum_{t=1}^{T} \frac{\alpha_t \beta_{tt}}{2(1-\beta_{tt})} \|S_{t}^{1/2} m_{t-1}\|^2 \\
\times \left\{ \left( \frac{\beta_{tt}}{2\alpha_t(1-\beta_{tt})} \right) \|S_{t}^{1/2} (x_t - x^*)\|^2 \right\} \\
- \frac{\sigma}{2} \sum_{t=1}^{T} \|x_t - x^*\|^2. \tag{19}
\]
Furthermore, since \(0 \leq s_{t-1,i} \leq s_{t,i}, 0 \leq a_t \leq a_{t-1}, \) and \(0 \leq \beta_{tt} \leq \beta_1 < 1,\) by (19), we have
\[
R(T) \leq \sum_{t=1}^{T} \left[ \frac{\|S_{t}^{1/2}(x_t - x^*)\|^2 - \|S_{t}^{1/2}(x_{t-1} - x^*)\|^2}{2\alpha_t(1-\beta_{tt})} \right] \\
- \frac{\sigma}{2} \sum_{t=1}^{T} \|x_t - x^*\|^2 \\
+ \sum_{t=1}^{T} \frac{\alpha_t}{2(1-\beta_{tt})} \|S_{t}^{-1/2} m_{t}\|^2 \\
+ \sum_{t=2}^{T} \frac{\alpha_t \beta_{tt}}{2(1-\beta_{tt})} \|S_{t}^{1/2} m_{t-1}\|^2 \\
- \frac{\sigma}{2} \sum_{t=1}^{T} \|x_t - x^*\|^2 \\
+ \sum_{t=1}^{T} \frac{\beta_{tt}}{2\alpha_t(1-\beta_{tt})} \|S_{t}^{1/2} (x_t - x^*)\|^2. \tag{20}
\]
Next, we consider the upper bounds of three parts \((E_1, E_2, \) and \(E_3)\) in (20), respectively. For part \(E_1,\) we attain the following:
\[
E_1 = \frac{1}{2\alpha_t(1-\beta_{tt})} \|S_{t}^{1/2}(x_1 - x^*)\|^2 \\
+ \sum_{t=1}^{T} \frac{1}{2\alpha_t(1-\beta_{tt})} \|S_{t}^{1/2}(x_t - x^*)\|^2 \\
- \frac{1}{2\alpha_t(1-\beta_{tt})} \|S_{t}^{1/2}(x_{t-1} - x^*)\|^2 \\
- \frac{\sigma}{2} \sum_{t=1}^{T} \|x_t - x^*\|^2 \\
\leq \sum_{t=2}^{T} \frac{1}{2\alpha_t(1-\beta_{tt})} \|S_{t}^{1/2}(x_t - x^*)\|^2 \\
+ \frac{1}{2\alpha_t(1-\beta_{tt})} \|S_{t}^{1/2}(x_{t-1} - x^*)\|^2 \\
- \frac{\sigma}{2} \sum_{t=1}^{T} \|x_t - x^*\|^2. \tag{21}
\]
Since \(a_t = (a/t)\) and from (21), we have the following:
\[
E_1 \leq \sum_{t=2}^{T} \frac{1}{2\alpha_t(1-\beta_{tt})} \|S_{t}^{1/2}(x_t - x^*)\|^2 \\
- \frac{\sigma}{2} \sum_{t=1}^{T} \|x_t - x^*\|^2 \\
= \sum_{t=2}^{T} \frac{1}{2\alpha_t(1-\beta_{tt})} \|S_{t}^{1/2}(x_t - x^*)\|^2. \tag{22}
\]
Hence,

\[ E_1' = \sum_{i=1}^{n} (x_{i,t} - x^*_i)^2 \left[ t (\tilde{\delta}_{i,t} - \alpha (1 - \beta_{1t})) - \frac{\delta}{t - 1} - \sigma \alpha (1 - \beta_{1t}) \right] \]

where \( i \in \{1, \ldots, n\} \), and \( n \) is the dimension of decision vectors. Moreover, since \( (a+b)^{1/2} \leq a^{1/2} + b^{1/2} \) and \( 1 - \beta \leq 1 - \beta_{1t} \), by (9), we can have the following:

\[ E_1' \leq \sum_{i=1}^{n} (x_{i,t} - x^*_i)^2 \left[ t \tilde{\delta}_{i,t}^{1/2} - (t-1) \tilde{\delta}_{i,t-1}^{1/2} - \sqrt{\delta(t-1)} - \sigma \alpha (1 - \beta_{1t}) \right] \]

\[ \leq \sum_{i=1}^{n} (x_{i,t} - x^*_i)^2 \left[ t \tilde{\delta}_{i,t}^{1/2} - (t-1) \tilde{\delta}_{i,t-1}^{1/2} - \delta \right] - \sigma \alpha (1 - \beta_{1t}) \]  

\[ \leq \sum_{i=1}^{n} (x_{i,t} - x^*_i)^2 \left[ \sigma \alpha (1 - \beta_{1t}) - \sqrt{\delta(t-1)} - \sigma \alpha (1 - \beta_{1t}) \right] \]

\[ \leq 0. \]

Next, for term \( E_1'' \) of (22), we have

\[ E_1'' = \frac{1}{2\alpha(1 - \beta_{1t})} \| \tilde{\delta}_{i,t}^{1/2}(x_{i,t} - x^*_i) \|^2 - \frac{\sigma}{2} \| x_{i,t} - x^*_i \|^2 \]

By (9), \( s_i - \sigma \alpha (1 - \beta_{1t}) \leq 0 \) when \( t = 1 \), and from (25), we obtain the following:

\[ E_1'' \leq \frac{\delta}{2\alpha(1 - \beta_{1t})} (x_{i,t} - x^*_i)^2. \]

Moreover, from Assumption 1, we have that \( x_{1,1} - x^*_1 \leq D \). Then, combining (22), (24), and (26), we finally attain

\[ E_1 \leq \sum_{i=1}^{n} \frac{\delta D^2}{2\alpha(1 - \beta_{1t})} = \frac{n\delta D^2}{2\alpha(1 - \beta_{1t})}. \]

Since \( \beta_{1t} \leq \beta_1 \), we have the following:

\[ E_2 \leq \sum_{i=1}^{T} \frac{\alpha}{2(1 - \beta_{1t})} \| \tilde{\delta}_{i,t}^{1/2}m_{i} \|^2 + \sum_{i=2}^{T} \frac{\beta_1 \alpha_{i-1}}{2(1 - \beta_{1t})} \times \| \tilde{\delta}_{i,t}^{1/2}m_{i} \|^2. \]

For (28), we first consider the term \( \sum_{i=1}^{T} \alpha_i \| \tilde{\delta}_{i,t}^{1/2}m_{i} \|^2 \); we obtain

\[ \sum_{i=1}^{T} \alpha_i \| \tilde{\delta}_{i,t}^{1/2}m_{i} \|^2 = \sum_{i=1}^{T} \alpha_i \| \tilde{\delta}_{i,t}^{1/2}m_{i} \|^2 + \alpha \| \tilde{\delta}_{T}^{1/2}m_{T} \|^2 \]

\[ \leq \sum_{i=1}^{T} \alpha_i \| \tilde{\delta}_{i,t}^{1/2}m_{i} \|^2 + \alpha \| m_{T} \|^2. \]

Moreover, assuming that \( \frac{(g_{1,1} - m_{1,1})}{\| g_{1,1} \|} = \omega \), where \( \omega \in (0, 1) \) and \( i \in \{1, \ldots, n\} \), let \( \omega = \min \{ \omega_{1, \ldots, n} \} \). Furthermore, applying the recursive algorithm to (3) and (4), we have that

\[ E_2' \]

\[ = \frac{\alpha}{\omega} \sum_{i=1}^{n} \left( \frac{\sum_{j=1}^{T} \beta_{1j}(T-k-1)}{\sum_{j=1}^{T} \beta_{1j}(T-k-1)} \right) + \delta \]

\[ \leq \frac{\alpha}{\omega} \sum_{i=1}^{n} \left( \frac{\sum_{j=1}^{T} \beta_{1j}(T-k-1)}{\sum_{j=1}^{T} \beta_{1j}(T-k-1)} \right) + \delta. \]

From (30) and \( \beta_{1t} \in (0, 1) \), \( E'_{2} \) can be further bounded as

\[ E_2' \leq \frac{\alpha}{\omega} \sum_{i=1}^{n} \left( \frac{\sum_{j=1}^{T} \beta_{1j}(T-k-1)}{\sum_{j=1}^{T} \beta_{1j}(T-k-1)} \right) + \delta. \]

Furthermore, according to the Cauchy–Schwarz inequality, i.e., \( \sum_{i=1}^{T} (a_i \cdot b_i) \leq \sum_{i=1}^{T} (a_i^2 \cdot b_i^2) \), and from (31), we attain the following:

\[ E_2' \]

\[ \leq \frac{\alpha}{\omega} \sum_{i=1}^{n} \left( \frac{\sum_{j=1}^{T} \beta_{1j}(T-k-1)}{\sum_{j=1}^{T} \beta_{1j}(T-k-1)} \right) + \delta. \]

Since \( \beta_{1t} \leq \beta_1 \) and from (32), we can further attain the following bound for \( E'_2 \):

\[ E_2' \leq \frac{\alpha}{\omega} \sum_{i=1}^{n} \left( \frac{\sum_{j=1}^{T} \beta_{1j}(T-k-1)}{\sum_{j=1}^{T} \beta_{1j}(T-k-1)} \right) + \delta. \]
By (8), we have the following:

\[
E'_2 \leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)} \sum_{i=1}^{n} \sum_{j=1}^{T_i} \beta_{i-j}^T g_{i,j} \sum_{k=1}^{T} g_{k,i}^2 + \zeta \delta
\]

\[
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)} \sum_{i=1}^{n} \sum_{j=1}^{T} \beta_{i-j}^T g_{i,j} \sum_{k=1}^{T} g_{k,i}^2 + \zeta \delta
\]

(34)

Moreover, plugging (34) into (29), and applying the recursive algorithm, we attain

\[
\sum_{i=1}^{T} a_{i} \|\hat{S}_{i}^{1/2} m_{i}\|^2 \\
\leq \sum_{i=1}^{T-1} \|\hat{S}_{i}^{1/2} m_{i}\|^2 \\
+ \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)} \sum_{i=1}^{n} \sum_{j=1}^{T} \beta_{i-j}^T g_{i,j} \sum_{k=1}^{T} g_{k,i}^2 + \zeta \delta
\]

\[
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)} \sum_{i=1}^{n} \sum_{j=1}^{T} \beta_{i-j}^T g_{i,j} \sum_{k=1}^{T} g_{k,i}^2 + \zeta \delta
\]

\[
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)} \sum_{i=1}^{n} \sum_{j=1}^{T} \beta_{i-j}^T g_{i,j} \sum_{k=1}^{T} g_{k,i}^2 + \zeta \delta
\]

\[
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \sum_{j=1}^{T} \beta_{i-j}^T g_{i,j} \sum_{k=1}^{T} g_{k,i}^2 + \zeta \delta
\]

(35)

If, let \( \Psi = (g_{i,j}^2/(\sum_{k=1}^{T} g_{k,i}^2 + \zeta \delta)), \) then \( \omega_j = \sum_{k=1}^{T} g_{k,i}^2 + \zeta \delta, \) and \( \omega_0 = \zeta \delta, \) we obtain

\[
\Psi = \frac{\omega_j - \omega_{j-1}}{\omega_j}.
\]

(36)

In addition, for any \( a \geq b > 0, \) the inequality \( 1 + x \leq e^x \) implies that

\[
\frac{a-b}{a} \leq \log \frac{a}{b}.
\]

(37)

Therefore, by (37), (36) has the following bound:

\[
\Psi \leq \log \frac{\omega_j}{\omega_{j-1}}.
\]

(38)

Plugging (38) into (35), we have

\[
\sum_{i=1}^{T} a_{i} \|\hat{S}_{i}^{1/2} m_{i}\|^2 \\
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \sum_{j=1}^{T} \log \frac{\omega_j}{\omega_{j-1}}
\]

\[
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \log \frac{\omega_j}{\omega_0}
\]

\[
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \log \left(\frac{\sum_{k=1}^{T} g_{k,i}^2}{\zeta \delta} + 1\right).
\]

(39)

From (28) and (39), \( E_2 \) can be further bounded as

\[
E_2 \leq \frac{1}{1 - \beta_1} \sum_{i=1}^{T} a_{i} \|\hat{S}_{i}^{1/2} m_{i}\|^2
\]

\[
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \log \left(\frac{\sum_{k=1}^{T} g_{k,i}^2}{\zeta \delta} + 1\right)
\]

\[
\leq \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \log \left(\frac{1}{\zeta \delta} g_{1:T,i}^2 + 1\right).
\]

(40)

Next, we consider the last term \( E_3 \) in (20). From the definition of \( \hat{S}_t \) and Assumption 2, we obtain the following:

\[
E_3 \leq \sum_{i=1}^{T} \sum_{j=1}^{T} t_{i,j} \beta_{i,j} \left( s_{i,j} + \delta \right) ^{1/2} (x_i - x^*)^2
\]

\[
\leq \frac{n}{2a(1 - \beta_1)^2} \sum_{i=1}^{T} \sum_{j=1}^{T} t_{i,j} \beta_{i,j} \left( s_{i,j} + \delta \right) ^{1/2} (x_i - x^*)^2
\]

\[
\leq \frac{D_{\infty}(G_{\infty} + \delta)}{2a(1 - \beta_1)} \sum_{i=1}^{T} \sum_{j=1}^{T} \beta_{i,j} \left( s_{i,j} + \delta \right) ^{1/2} (x_i - x^*)^2
\]

\[
\leq \frac{D_{\infty}(G_{\infty} + \delta)}{2a(1 - \beta_1)} \sum_{i=1}^{T} \sum_{j=1}^{T} \beta_{i,j} \left( s_{i,j} + \delta \right) ^{1/2} (x_i - x^*)^2
\]

(41)

Finally, combining (20), (27), (40), and (41), we obtain the upper bound of \( R(T) \) as follows:

\[
R(T) \leq \frac{n\delta D_{\infty}^2}{2a(1 - \beta_1)} + \frac{D_{\infty}^2(G_{\infty} + \delta)}{2a(1 - \beta_1)} \sum_{i=1}^{T} \sum_{j=1}^{T} \beta_{i,j} \left( s_{i,j} + \delta \right) ^{1/2} (x_i - x^*)^2
\]

\[
+ \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \log \left(\frac{1}{\zeta \delta} g_{1:T,i}^2 + 1\right).
\]

(42)

Therefore, the proof of Theorem 1 is completed.

\[\blacksquare\]

Corollary 1: Let \( \beta_{i,t} = \beta_i \lambda^t \), where \( \lambda \in (0, 1) \) in Theorem 1. Then, we have the following upper bound of the regret:

\[
R(T) \leq \frac{n\delta D_{\infty}^2(G_{\infty} + \delta)}{2a(1 - \beta_1)} + \frac{n\beta_1 \lambda D_{\infty}^2(G_{\infty} + \delta)}{2a(1 - \beta_1)(1 - \lambda)^2} + \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \log \left(\frac{1}{\zeta \delta} g_{1:T,i}^2 + 1\right).
\]

Proof: Since \( \beta_{i,t} = \beta_i \lambda^t \), (41) can be further bounded as follows:

\[
E_3 \leq \beta_1 D_{\infty}^2(G_{\infty} + \delta) \sum_{i=1}^{T} \sum_{j=1}^{T} \beta_{i,j} \left( s_{i,j} + \delta \right) ^{1/2} (x_i - x^*)^2
\]

\[
\leq \beta_1 D_{\infty}^2(G_{\infty} + \delta) \sum_{i=1}^{T} \sum_{j=1}^{T} \beta_{i,j} \left( s_{i,j} + \delta \right) ^{1/2} (x_i - x^*)^2
\]

\[
\leq \beta_1 D_{\infty}^2(G_{\infty} + \delta) \sum_{i=1}^{T} \sum_{j=1}^{T} \beta_{i,j} \left( s_{i,j} + \delta \right) ^{1/2} (x_i - x^*)^2
\]

(43)

In addition, plugging (43) into (42), we further attain the upper bound of \( R(T) \) as follows:

\[
R(T) \leq \frac{n\delta D_{\infty}^2(G_{\infty} + \delta)}{2a(1 - \beta_1)} + \frac{n\beta_1 \lambda D_{\infty}^2(G_{\infty} + \delta)}{2a(1 - \beta_1)(1 - \lambda)^2} + \frac{a_{\zeta}}{\sigma^2(1 - \beta_1)^2} \sum_{i=1}^{n} \log \left(\frac{1}{\zeta \delta} g_{1:T,i}^2 + 1\right).
\]

(44)

Therefore, the proof of Corollary 1 is completed.

\[\blacksquare\]
APPENDIX B

DETAILS ON CONDITIONS 3 AND 4

The contribution of conditions 3 and 4 is to ensure the convergence of the proposed algorithm, which are common conditions in many second-order momentum algorithms, such as AMSGrad and SAdam. However, in the actual training task, the parameters in these conditions are not explicitly involved, so they are only used in the proof process.

For condition 3, the parameter $\zeta$ only needs to satisfy that

$$ t \sum_{j=1}^{t-j} \prod_{j=1}^{k=1} \beta_{2(t-k+1)}(1 - \beta_2) g_{j,i}^2 \geq \frac{1}{\zeta} \sum_{j=1}^{t} g_{j,i}^2. $$

From the above in equation, we have that

$$ \frac{1}{\zeta} \geq t \prod_{j=1}^{k=1} \beta_{3(t-k+1)}(1 - \beta_2) > t. $$

Therefore, the value of parameter $\zeta$ is easy to choose.

For condition 4, from the definition of $s_t$

$$ s_t = \beta_2 s_{t-1} + (1 - \beta_2) (g_t - m_t)^2 $$

and applying the recursive formula for the above equation, we have

$$ s_t = (1 - \beta_2)(g_t - m_t)^2 + \beta_2 (1 - \beta_2) (g_{t-1} - m_{t-1})^2 + \cdots + \beta_2^{T-1} (1 - \beta_2) (g_0 - m_0)^2 $$

and

$$ s_{t-1} = (1 - \beta_2)(g_t - m_t)^2 + \beta_2 (1 - \beta_2) (g_{t-1} - m_{t-1}) + \cdots + \beta_2^{T-1} (1 - \beta_2) (g_0 - m_0)^2. $$

Thus, we further obtain

$$ \frac{t}{\alpha} s_t - \frac{1}{\alpha} s_{t-1} \leq \frac{t}{\alpha} (1 - \beta_2)^2 (g_t - m_t)^2 + \frac{1}{\alpha} s_{t-1} \leq \frac{T}{\alpha} G_\infty^2 + \frac{1}{\alpha} (1 - \beta_2) G_\infty^2 = \frac{1}{\alpha} (T + 1 - \beta_2) G_\infty^2. $$

Therefore, the parameter $\sigma$ only needs to be selected in the following way to make condition 3 true:

$$ \sigma \geq \frac{1}{\alpha(1 - \beta_1)} (T + 1 - \beta_2) G_\infty^2. $$

In summary, the selection of parameters in conditions 3 and 4 is not difficult.

APPENDIX C

EXPERIMENTS ON CIFAR-100

We experiment five times with a DenseNet-121 model on the CIFAR-100 classification task. The parameters are set as the same as in Section IV. The results shown in Fig. 7 demonstrate that the proposed algorithm converges faster than other algorithms. In Fig. 8, the top-one training accuracy of the algorithms. In addition, Fig. 9 shows that the top-one test accuracy of the proposed algorithm is better than other algorithms.

APPENDIX D

CONVERGENCE ANALYSIS OF FASTADABELIEF IN CONVEX CONDITIONS

The regret bound analysis for the proposed algorithm when loss functions are convex is present as follows.

**Proof:** Review the forms of the proposed algorithm and AdaBelief that the difference is their step sizes. The step sizes of the proposed algorithm and AdaBelief are shown as follows:

$$ \Delta_t(\text{FastAdaBelief}) = \frac{a_t m_t}{\delta_t} - \frac{a_t m_t}{\sqrt{\delta_t}} , (a_t = 1/t) \quad (46) $$

and

$$ \Delta_t(\text{AdaBelief}) = \frac{a_t m_t}{\sqrt{\delta_t}} = \frac{a_t m_t}{\sqrt{T(\delta_t + \epsilon)}} , (a_t = 1/\sqrt{T}) \quad (47) $$

where $s_t = \beta_2 s_{t-1} + (1 - \beta_2)(g_t - s_t)^2$. Note that $\epsilon$ is a very small positive term to keep the denominator from going to zero, which is ignored in the convergence analysis of AdaBelief. Therefore, $\delta$, which has the same effect as $\epsilon$, will also be ignored in the convergence analysis of the proposed algorithm.

To attain the regret bound for the proposed algorithm when the loss functions are convex, we first consider the bound of the following term:

$$ \|s_t^T (x_{t+1} - x^*)\| \leq \|s_t^T (x_t - x^*)\|^2 + a_t^2 \|s_t^T m_t\|^2 - 2a_t (\beta_{1t} m_{t-1} + (1 - \beta_{1t}) g_t, x_t - x^*). $$

$$ (48) $$
Rearranging in (48), we have
\[
\begin{align*}
\{g_t, x_t - x^*\} & \leq \frac{1}{2\alpha_t(1 - \beta_t)} \left( \left\| s_t^{1/2}(x_t - x^*) \right\|^2 - \left\| s_t^{1/2}(x_{t+1} - x^*) \right\|^2 \right) \\
+ & \frac{1}{2(1 - \beta_t)} \left\| s_t^{-1/2}m_t \right\|^2 + \frac{\beta_t \alpha_t}{2(1 - \beta_t)} \left\| s_t^{-1/2}m_{t-1} \right\|^2 \\
+ & \frac{\beta_t}{2\alpha_t(1 - \beta_t)} \left\| s_t^{1/2}(x_t - x^*) \right\|^2.
\end{align*}
\] (49)

By the convexity of function, we attain
\[
\sum_{t=1}^{T} \{f_t(x_t) - f_t(x^*)\} \leq \frac{1}{2(1 - \beta_t)} \left\| x_1 - x^* \right\|^2
+ \frac{1}{2(1 - \beta_t)} \sum_{t=2}^{T} \left\| x_t - x^* \right\|^2 \left( \frac{s_t^{1/2}}{\alpha_t} - \frac{s_{t-1}^{1/2}}{\alpha_{t-1}} \right)
+ \frac{1 + \beta_t}{2(1 - \beta_t)} \sum_{t=1}^{T} \alpha_t \left\| s_t^{-1/2}m_t \right\|^2
+ \frac{1}{2(1 - \beta_t)} \sum_{t=1}^{T} \beta_t \alpha_t \left\| s_t^{1/2}(x_t - x^*) \right\|^2.
\] (50)

Assuming that \(0 < c < \|s_t\|\), the term \(\sum_{t=1}^{T} \alpha_t \left\| s_t^{-1/2}m_t \right\|^2\) in (50) can be bounded as follows:
\[
\begin{align*}
\sum_{t=1}^{T} \alpha_t \left\| s_t^{-1/2}m_t \right\|^2 & \leq \sum_{t=1}^{T-1} \alpha_t \left\| s_t^{-1/2}m_t \right\|^2 + \frac{\alpha}{cT} \|m_T\|^2 \\
& \leq \sum_{t=1}^{T-1} \alpha_t \left\| s_t^{-1/2}m_t \right\|^2 \\
& + \frac{\alpha}{cT} \sum_{i=1}^{d} \left( \sum_{j=1}^{T} (1 - \beta_{1,i}) g_{j,i} \prod_{k=1}^{T-j} \beta_{1,k} \right)^2 \\
& \leq \frac{\alpha}{c(1 - \beta_1)^2} \sum_{i=1}^{d} \left\| s_{1,T,i} \right\|^2 \left\| 1 \right\| T \leq \frac{\alpha(1 + \log T)}{c(1 - \beta_1)^2} \sum_{i=1}^{d} \left\| s_{1,T,i} \right\|^2 \\
\end{align*}
\] (51)

Applying (51) into (50), we obtain
\[
\begin{align*}
R(T) = \sum_{t=1}^{T} \{f_t(x_t) - f_t(x^*)\} & \leq \frac{D_{\infty}^2 T}{2\alpha(1 - \beta_1)} \sum_{i=1}^{d} \left\| s_{1,T,i} \right\|^2 \\
+ & \frac{(1 - \beta)\alpha \log T}{2c(1 - \beta_1)^3} \sum_{i=1}^{d} \left\| s_{1,T,i} \right\|^2 \\
+ & \frac{D_{\infty}^2}{2(1 - \beta_1)} \sum_{t=1}^{T} \beta_t s_t^{1/2}/\alpha_t.
\end{align*}
\] (52)

Therefore, the proof of the regret bound of the proposed algorithm when loss functions are convex is completed.
Yangfan Zhou is currently pursuing the Ph.D. degree with the School of Nano-Tech and Nano-Bionics, University of Science and Technology of China, Hefei, China.

His current research interests are focused on theoretical and algorithmic issues related to large-scale optimization, stochastic optimization, convex online optimization, and their applications to deep learning, metalearning, and networking.

Dr. Zhou is currently a Reviewer for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Kaizhu Huang received the Ph.D. degree from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2004.

He worked at the Fujitsu Research Centre, CUHK, the University of Bristol, the National Laboratory of Pattern Recognition, and the Chinese Academy of Sciences, from 2004 to 2012. He is currently a Full Professor at the Institute of Applied Physical Sciences and Engineering, Duke Kunshan University (DKU). Before joining DKU, he was a Full Professor at the Department of Intelligent Science, Xi’an Jiaotong-Liverpool University (XJTLU), and an Associate Dean of research with the School of Advanced Technology, XJTLU. He founded and directed the Suzhou Municipal Key Laboratory of Cognitive Computation and Applied Technology. His research interests include machine learning, neural information processing, and pattern recognition.

Dr. Huang was a recipient of the 2011 Asia Pacific Neural Network Society Young Researcher Award. He received best paper or book award six times. He serves as an associated editor/advisory board members in a number of journals and book series. He was invited as a keynote speaker in more than 30 international conferences or workshops.

Cheng Cheng received the B.S. and M.S. degrees in computer science and technology from Guizhou University, Guiyang, China, in 2004 and 2009, respectively, and the Ph.D. degree in information engineering from the Tokyo University of Agriculture and Technology (TUAT), Tokyo, Japan, in 2013. He is currently an Associate Professor. His current research interest focuses on 3-D vision.

Xuguang Wang received the bachelor’s degree from the Department of Materials, Tsinghua University, Beijing, China, in 1999, the master’s degree from the Department of Electronic Engineering, Rice University, Houston, TX, USA, in 2002, and the Ph.D. degree from the Department of Electronic Engineering, University of Texas, Austin, TX, USA, in 2005. He has been engaged in the research of semiconductor storage technology for ten years, and has undertaken the research of semiconductor memory in many scientific research institutions, such as the National Natural Science Foundation of the United States, MARCO, and SRC.

Amir Hussain received the B.Eng. (Hons.) and Ph.D. degrees from the University of Strathclyde, Glasgow, U.K., in 1992 and 1997, respectively.

He is the Founding Director of the Centre of AI and Data Science, Edinburgh Napier University, U.K. He has (co)authored three international patents and around 500 publications. He is the Founding Chief Editor of Springer’s Cognitive Computation Journal and Springer Book Series on Socio-Affective Computing. His research interests are cross-disciplinary and industry-led, aimed at developing cognitive data science and trustworthy AI technologies to engineer the smart industrial and healthcare systems of tomorrow.

Dr. Hussain has been invited Associate Editor/Editorial Board member for various other top journals, including the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, Information Fusion, the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS: SYSTEMS, and the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE. Among other distinguished roles, he is an elected Executive Committee Member of the U.K. Computing Research Committee, a General Chair of the IEEE WCCI 2020, and the Chair of the IEEE U.K. and Ireland Chapter of the IEEE Industry Applications Society.

Xin Liu (Member, IEEE) received the B.Eng. degree in electrical engineering from Tianjin University, Tianjin, China, and the Ph.D. degree in electrical engineering from Nanyang Technological University (NTU), Singapore, in 2000 and 2007, respectively.

From 2007 to 2018, he was a Principal Investigator and the Head of the Intelligent Computing Chips Department with the Institute of Microelectronics, A*STAR, Singapore. He joined the Suzhou Institute of Nano-Tech and Nano-Bionics (SINANO), Chinese Academy of Sciences, as a Professor, in 2018. His research interests include artificial intelligence signal processing algorithms, high-performance massively parallel processing chip architecture design, ultralow-power digital processor design, and embedded nonvolatile memory circuit design.