Towards Faster Training Algorithms Exploiting Bandit Sampling from Convex to Strongly Convex Conditions

Yangfan Zhou, Kaizhu Huang, Cheng Cheng, Xuguang Wang, Amir Hussain, and Xin Liu*.

***Abstract*—The training process for deep learning and pattern recognition normally involves the use of convex and strongly convex optimization algorithms such as AdaBelief and SAdam to handle lots of "uninformative" samples that should be ignored, thus incurring extra calculations. To solve this open problem, we propose to design bandit sampling method to make these algorithms focus on "informative" samples during training process. Our contribution is twofold: first, we propose a convex optimization algorithm with bandit sampling, termed AdaBeliefBS, and prove that it converges faster than its original version; second, we prove that bandit sampling works well for strongly convex algorithms, and propose a generalized SAdam, called SAdamBS, that converges faster than SAdam. Finally, we conduct a series of experiments on various benchmark datasets to verify the fast convergence rate of our proposed algorithms.**

***Index Terms*—Bandit Sampling, Convex Optimization, Image Processing, Training Algorithm.**

I. Introduction

TRAINING involves a large number of parameters and a complex structure in pattern recognition and deep learning. Such a process is typically difficult and time-consuming especially in cases of a large amount of data samples [1]. To accomplish training in deep learning, efficient optimization algorithms are crucially demanded [2]. To examine the performance of optimization algorithms, two indicators, i.e., convergence rate and generalization ability are usually adopted. Additionally, current optimization algorithms often assume that their loss function and feasible region are both convex to obtain better algorithm benefits from the good properties of convexity.

With a strong generalization ability, stochastic gradient descent (SGD) is one of the most popular algorithms that has been widely used in many applications. Unfortunately, SGD is limited due to its slow convergence rate. Thereby a large number of labeled samples are necessary to complete the SGD training. To tackle this limitation, many faster optimization algorithms have been proposed including Adam [3], AMSGrad [4], and AdaBound [5]. These algorithms focus on adjusting adaptive step sizes and historical gradients to improve the convergence rate. For example, as one most successful algorithm, Adam considers the historical gradients by the first-order momentum ($\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$) and adapts the step sizes by the second-order momentum ($\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2$) (where β_1, β_2 denote the exponential decay rates, $\mathbf{g}_t$ represents the original gradient, $\mathbf{m}_t$ denotes the first-order momentum, and $\mathbf{v}_t$ represents the second-order momentum). In fact, Adam has been extensively used in the past few years and its two momentum strategies have inspired many other improved adaptive algorithms.

Although Adam converges faster than SGD, it has a weak generalization ability. Such drawback limits its application in deep learning, requiring both efficiency and precision. To promote the generalization ability of Adam, [6] proposed an effective method named AdaBelief, which resets the second-order momentum as a new form $\mathbf{s}_t = \beta_2 \mathbf{s}_{t-1} + (1-\beta_2)(\mathbf{g}_t - \mathbf{m}_t)^2$. Its main motivation is to adjust the step size according to the difference between the observed gradient and the predicted gradient (called the "belief" in the current gradient direction (belief-gradient)). Under convex conditions, AdaBelief guarantees a theoretical superiority to Adam on generalization ability with the same convergence bound $O\Big(d\sqrt{T}\Big) + O\Big(\frac{\sqrt{n\log d}}{\sqrt{K}}\frac{\sqrt{n\log n}}{n}\sqrt{T}\Big)$, where T is a time horizon, n represents the total number of samples, d is the dimension of the decision vector, and K denotes the mini-batch size.

On the other hand, in addition to improving further the generalization ability, investigations have also been made to explore if the convergence bound of various algorithms including Adam can be further lifted up particularly in convex cases. Indeed, many prior works indicate that sampling training examples is one effective method to improve the convergence rate without any additional optimization constraints or assumptions. For instance, SGD's convergence rate has been accelerated by different sampling methods [7]–[10]. Moreover, to improve Adam's convergence rate, bandit sampling is used to select a mini-batch examples from the training set for each iteration [11]. This work focuses on the relationship between sample distribution and model parameters during

* Corresponding author: Xin Liu (xliu2018@sinano.ac.cn).

Yangfan Zhou and Xin Liu are with School of Nano-Tech and Nano-Bionics, University of Science and Technology of China, 96 Jinzhai Road, Hefei City, Anhui Province, 230026, China.

Kaizhu Huang is with Data Science Research Center, Duke Kunshan University, No. 8 Duke Avenue, Kunshan, 215316, China.

Yangfan Zhou, Cheng Cheng, Xuguang Wang, and Xin Liu are with Suzhou Institute of Nano-Tech and Nano-Bionics (SINANO), Chinese Academy of Sciences, 398 Ruoshui Road, Suzhou Industrial Park, Suzhou City, Jiangsu Province, 215123, China.

Cheng Cheng is also with Gusu Laboratory of Materials, 388 Ruoshui Road, Suzhou, 215123, China.

Amir Hussain is with School of Computing, Edinburgh Napier University, Edinburgh, EH11 4BN, UK.

E-mails: yfzhou2020@sinano.ac.cn, kaizhu.huang@dukekunshan.edu.cn, ccheng2017@sinano.ac.cn, xgwang2009@sinano.ac.cn, A.Hussain@napier.ac.uk, and xliu2018@sinano.ac.cn.

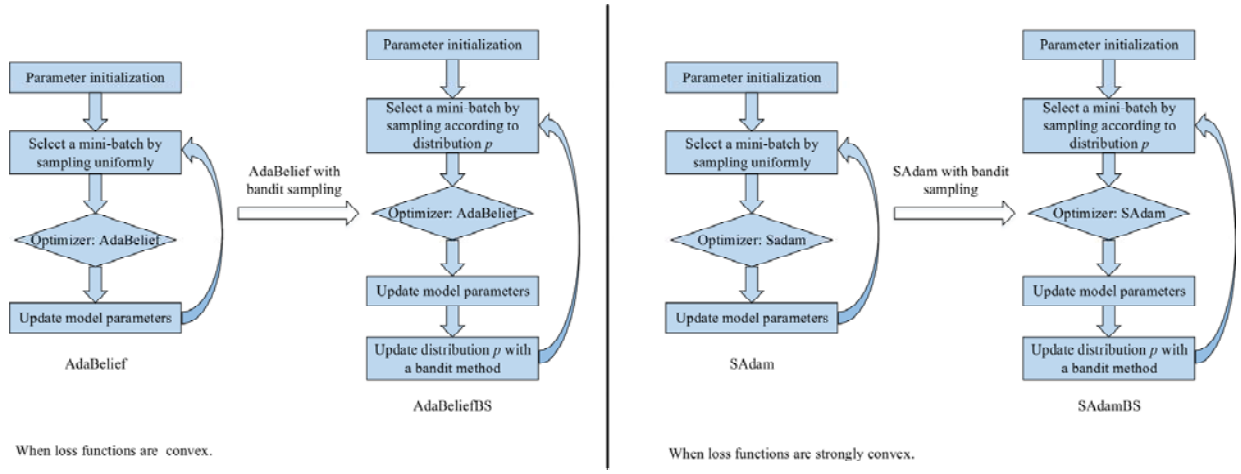

Fig. 1. Left is the AdaBelief with bandit sampling when loss functions are convex; Right is the SAdam with bandit sampling when loss functions are strongly convex.

training process. Importantly, with bandit sampling, AdamBS certificates theoretically a faster convergence than the original Adam.

Interestingly, though efforts have been made to improve these popular algorithms from the perspective of either generalization ability (e.g. AdaBelief) or convergence rate (e.g. AdamBS), there are rarely works exploring if an optimization algorithm can be developed improving both the generalization ability and convergence rate, particularly for the recent strong methods such as Adam and its variants. For instance, though bandit sampling has been shown successful in boosting the convergence for Adam, it remains an open question whether it also works when second order momentum is introduced as typically used in improving the generalization ability.

To this end, this paper presents a novel design that successfully bridges various Adam-based algorithms including AdaBelief and SAdam with bandit sampling. As a major contribution, we prove that the proposed algorithm named AdabeliefBS has a more compact convergence bound than the extended Adam, i.e., AdaBelief for the first time though some computational overhead incurred by maintaining and updating distribution of bandit sampling. On the other hand, inherited from AdaBelief, AdabeliefBS has also stronger generalization ability as empirically verified in the experiments. Going further, we also examine the strongly convex optimization that converges faster than convex optimization algorithms. We propose a new model called SAdamBS which integrates bandit sampling to attain a guaranteed bound more compact than SAdam.

More specifically, we incorporate the bandit sampling method into AdaBelief with $\mathbf{s}_t = \beta_2 \mathbf{s}_{t-1} + (1-\beta_2)(\hat{G}_t - \mathbf{m}_t)^2$ which is different from that of Adam and SGD. The difference in the second-order momentum leads to a challenge on the convergence analysis. Moreover, we propose the strongly convex algorithm with bandit sampling and prove its convergence property theoretically. This is also distinct with the existing works that are usually conducted under convex conditions.

Our ideas are shown in Figure 1 that are inspired by [11]. Our main contributions can be summarized as follows:

- We propose AdaBeliefBS that integrates bandit sampling to improve the convergence rate for the original AdaBelief, and prove that AdaBeliefBS converges faster than AdaBelief under convex conditions.
- We further study whether bandit sampling could accelerate the convergence rate for strongly optimization algorithms, and propose the corresponding algorithm named SAdamBS. Moreover, we also prove that SAdamBS converges faster than SAdam under strongly convex conditions.
- We prove that the convergence of the proposed algorithms can be further bounded when the samples follow doubly heavy-tailed distribution.
- We conduct a series of experiments to verify our proposed algorithms AdaBeliefBS and SAdamBS, which in practice also converge faster than the other algorithms under convexity and strongly convexity, respectively. Moreover, these experimental results are consistent with our theoretical proofs.

Notations. In this paper, a bold symbol denotes a vector, and $\mathbf{x}^2$, $\sqrt{\mathbf{x}}$, and $\frac{\mathbf{x}}{\mathbf{y}}$ represent the element-wise square, square root, and division, respectively. In addition, $\prod$ indicates a projection operator. For a deep model training task, its dataset contains N labeled samples in total. We use $p_t = \{p_t^1, p_t^2, \ldots, p_t^N\}$ to denote the sample distribution at time t. Additionally, p_t represents the related importance of samples in the training process. At each time, the optimizer chooses a mini-batch of K samples. Moreover, the gradient of the k-th sample at time t is denoted as $\mathbf{g}_t^k$.

II. Related Work

A. Adaptive Gradient Optimization Algorithms

Adaptive gradient optimization algorithms, represented by Adam, are widely used in pattern recognition and deep learning because of their fast convergence speed. For Adam, [4] pointed out that the quantity of the adaptive learning rate with respect to time, $\Gamma_{t+1} = \left(\frac{\sqrt{V_{t+1}}}{\alpha_{t+1}} - \frac{\sqrt{V_t}}{\alpha_t}\right)$, could be negative, leading that Adam fails to converge to an optimal solution. To

solve this problem, [4] modified the second-order momentum as $\hat{\mathbf{v}}_t = \max\{\hat{\mathbf{v}}_t, \mathbf{v}_t\}$.

Though the convergence problem of Adam has been solved, the gap of its generalization ability in deep model training processes still haunts its applications. To reduce this gap, [5] demonstrated that some extreme learning rate in Adam and AMSGrad can lead to poor performance, and proposed a new adaptive gradient algorithm named AdaBound that adds a dynamic bound on its learning rate to improve generalization ability. Another method to improve generalization is called Lookahead [12], which tries to update iteratively the weights of SGD and Adam. In addition to AdaBound and Lookahead, there are many other different methods to ameliorate the performance for Adam including [13]–[15].

In regard to the essential reason of Adam's poor performance, [6] analyzed three different cases about curvature of the loss function in which Adam always takes improper stepsize. To ensure the optimizer make proper decision in all the three cases, [6] completely modified the second-order momentum to $\mathbf{s}_t = \beta_2 \mathbf{s}_{t-1} + (1-\beta_2)(\mathbf{g}_t - \mathbf{m}_t)^2$ and proposed a new algorithm called AdaBelief. Nowadays, AdaBelief showed a new idea in improving the performance of Adam and achieved good results. Nevertheless, its convergence bound still follows Adam and AMSGrad with $O\left(d\sqrt{T}\right) + O\left(\frac{\sqrt{n\log d}}{\sqrt{K}}\frac{\sqrt{n\log n}}{n}\sqrt{T}\right)$. In this paper, we try to improve the convergence rate of these popular algorithm further while maintaining their excellent generalization ability.

As we know, optimization algorithms adapted to strongly convex functions can converge faster than those adapted to convex functions. For example, Adam [3] has a guaranteed convergence bound $O(\sqrt{T})$ when loss functions are convex, while its strongly convex variant SAdam [16] can achieve a much faster convergence bound of $O(\log T)$. Therefore, strongly convex algorithms enjoy the advantage of faster convergence rate when loss functions are strong convex. However, it keeps uncertain if further improvement can be made towards an even fast convergence rate for the strong convex algorithms. To fill this gap, we exploit bandit sampling and propose a mini-batch gradient descent optimization algorithm under strongly convex conditions, called SAdamBS.

B. Sampling Methods

Sampling methods play an important role in deep model training which enable optimization algorithms even faster. Bandit sampling is a standard online learning problem which applies β-distribution to describe the importance of sample and updates its parameters based on the regret of each decision. Many optimization algorithms have introduced bandit sampling to improve their performance. To speed up the convergence rate of SGD, [17] proposed a novel and efficient algorithm for generalized linear bandit with a regret bound of order $\tilde{O}(\sqrt{T})$. Moreover, to reduce the variance caused by uniform sampling of mini-batch of samples by SGD during each training, [18] exploited bandit sampling to non-uniformly select a mini-batch of samples for each iteration. In addition, bandit sampling is also utilized for Adam in [11], in which the mini-batch of samples of each iteration is selected by bandit sampling. As a result, the subterm of regret bound of AdamBS proposed by [11] is improved from $O(\sqrt{\frac{n}{T}})$ to $O\left(\sqrt{\frac{\log n}{T}}\right)$.

Important sampling is a widely used method that could both improve the computation rate and stability of numerical results. Important sampling estimates the original function with a simple probability distribution function and performs sampling on it, and then updates the weight of each part of this simple function according to the sampling result. The block coordinate descent (BCD) optimization algorithm randomly selects a block of coordinates to compute gradient for each iteration to avoid the costly computation of full coordinate gradient. However, improving the efficiency of BCD's coordinate block selection is always a difficult problem. To this end, [19], [20] utilizes the importance sampling to choose coordinate blocks to improve convergence rate of BCD algorithms. However, it is not clear how much importance sampling has accelerated these methods due to the difficulty of theoretical analysis.

Up to now, lots of optimization algorithms are based on mini-batch gradient descent which uses a mini-batch of samples for each iteration. However, not every batch of samples is equally important to the optimizer. Therefore, some optimization algorithms take the boosting sampling to choose the optical mini-batch of samples for each time. For instance, [21] studied how boosting sampling help accelerate SGD' convergence rate and proposed a new explanation for this process; [22] proposed a clustered sampling to reduce the variance of the clients stochastic aggregation weights in federated learning. In summary, boosting sampling learns multiple weak classifiers by changing the weight distribution of samples, and then combines them linearly to form a strong classifier. However, this method requires each sample to be calculated once, therefore incurring extra computational overhead.

III. Preliminaries

A. Online Learning

In this work, we consider an online learning problem that is formed as $\min_{\mathbf{x}_t \in \mathcal{R}^d} \sum_{t=1}^{T} f_t(\mathbf{x}_t)$, where T is a time horizon. In an online learning problem, the optimizer deals with tasks one by one in order. For the task of time t, the optimizer makes a decision $\mathbf{x}_t$ without knowing whether its outcome is good or bad. In contrast, the adversary returns a loss $f_t(\mathbf{x}_t)$ based on the above decision. Consequently, the target of the online learning task is to minimize the loss of the optimizer suffered from the adversary. To measure this problem intuitively, the regret was introduced and its mathematical form is as follows:

$$R(T) = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{R}^d} \sum_{t=1}^{T} f_t(\mathbf{x}), \tag{1}$$

where $R(T)$ represents the cumulative regret at time T, $\mathcal{R}^d$ is the set of real number with dimension d and T denotes a time horizon. For simplicity, we use $\mathbf{x}^*$ to denote the optimal solution of f_t with $t \in [T]$ on $\mathcal{R}^d$. Therefore, we have that $\sum_{t=1}^{T} f_t(\mathbf{x}^*) = \min_{\mathbf{x}\in\mathcal{R}^d} \sum_{t=1}^{T} f_t(\mathbf{x})$.

B. Mini-batch Gradient Descent

Since the deep model training process typically applies gradient descent that requires massive labeled samples, it will take a lot of computation and memory cost if all the samples are used at each iteration. Alternatively, if one single sample is used for each iteration, the gradient direction cannot be accurately estimated. As a trade-off, the most commonly used method is to select a mini-batch of samples at each iteration. We assume that a mini-batch is uniformly sampled from all the training samples. Therefore, at time t, the unbiased gradient estimate $\hat{G}$ of the mini-batch I_t is defined as

$$\hat{G}_t = \frac{1}{K}\sum_{k=1}^{K} \hat{\mathbf{g}}_t^k, \tag{2}$$

where $\hat{\mathbf{g}}_t^k$ denotes the k-th sample gradient at time t. The definition of the unbiased gradient estimate $\hat{G}$ follows [11] in which more detailed explanations are provided.

IV. Bandit Sampling for Convex Optimization

In this section, we first propose a belief-gradient based algorithm with bandit sampling in convex online learning cases, named AdaBeliefBS. Second, we present the convergence analysis for AdaBeliefBS.

As described in Section III-B, mini-batch gradient descent algorithms such as SGD, Adam, and AdaBelief are uniformly sampled at each iteration. These algorithms do not adequately account for differences between samples. For this reason, we propose a gradient descent algorithm that adaptively selects mini-batch samples under convex conditions. The proposed algorithm selects a mini-batch of K samples from the training set containing N samples for each iteration. Assume that the distribution of the samples is $p_t = \{p_t^1, p_t^2, \ldots, p_t^N\}$, and the gradient of a single sample is $\mathbf{g}_t^k$ for $t \in [T], k \in [K]$. Therefore, the unbiased estimate of $\mathbf{g}_t^k$ is $\hat{\mathbf{g}}_t^k = \mathbf{g}_t^k/(Np_t^k)$. Moreover, since $\mathbb{E}_{p_t}[\hat{\mathbf{g}}_t^k] = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_t^i = G$, $\hat{\mathbf{g}}_t^k$ is unbiased. Thus, the unbiased gradient estimate $\hat{G}_t$ of a mini-batch with K samples at iteration t is

$$\hat{G}_t = \frac{1}{K}\sum_{k=1}^{K} \hat{\mathbf{g}}_t^k. \tag{3}$$

From equation (3), we have $\mathbb{E}_{p_t}[\hat{\mathbf{G}}_t] = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_t^i = G$. Therefore, the unbiased estimate $\hat{G}_t$ can be used in our proposed algorithm.

Details of our proposed algorithm are shown in Algorithm 1 in which the modifications from its original version are highlighted in blue font. As we can see, there are not many steps highlighted in blue, but the convergence of the proposed algorithm should be re-proof due to these modifications, which is a challenge. In the algorithm, $f_t(x) \in \mathcal{R}$ is the loss function at time t that satisfies online learning cases. At time t, $\mathbf{x}_t \in \mathcal{R}^d$ is the decision vector, where d is the number of dimension. Moreover, $\mathbf{g}_t^k$ is the original gradient of the k-th sample at time t. And $\hat{G}_t$ is unbiased gradient estimate of one mini-batch with K samples. In addition, $\mathbf{m}_t$ and $\mathbf{s}_t$ are exponential moving average of $\hat{G}_t$ and $(\hat{G}_t - \mathbf{m}_t)^2$, respectively. The typical values of hyper-parameters are $\beta_1 = 0.9, \beta_2 = 0.999$. Specifically, learning rate α is set to 10^{-3}, and normal term ϵ is set to 10^{-8}.

Algorithm 1 AdaBeliefBS

Input: $\beta_1, \beta_2, \epsilon$

Initialize: $\mathbf{x}_0, \mathbf{m}_0 \leftarrow \mathbf{0}, \mathbf{v}_0 \leftarrow \mathbf{0}, p_0^k \leftarrow 1/n, \forall 1 \le k \le N$, and set time start with $t = 0$.

while $\mathbf{x}_t$ not converged **do**
  $t \leftarrow t + 1$
  select a mini-batch of K samples by sampling with replacement from p_{t-1}
  compute original gradient of one sample:
    $\mathbf{g}_t \leftarrow \nabla f_t(\mathbf{x}_{t-1})$
  compute unbiased gradient estimate of one mini-batch:
    $\hat{G}_t = \frac{1}{K}\sum_{k=1}^{K}\frac{\mathbf{g}_t^k}{np_{t-1}^k}$
  compute the first-order momentum:
    $\mathbf{m}_t \leftarrow \beta_1\mathbf{m}_{t-1} + (1-\beta_1)\hat{G}_t$
  compute the second-order momentum based on gradient belief:
    $\mathbf{s}_t \leftarrow \beta_2\mathbf{s}_{t-1} + (1-\beta_2)(\hat{G}_t - \mathbf{m}_t)^2$,
    $\mathbf{s}_t \leftarrow \max\{\mathbf{s}_t, \mathbf{s}_{t-1}\}$
  compute bias correction of $\mathbf{m}_t$:
    $\hat{\mathbf{m}}_t \leftarrow \mathbf{m}_t/(1-\beta_1^t)$
  compute bias correction of $\mathbf{s}_t$:
    $\hat{\mathbf{s}}_t \leftarrow \mathbf{s}_t/(1-\beta_2^t)$
  update the decision vector $\mathbf{x}_t$:
    $\mathbf{x}_t \leftarrow \prod_{\mathcal{F},\sqrt{\hat{\mathbf{s}}_t}}\left(\mathbf{x}_{t-1} - \frac{\alpha_t\mathbf{m}_t}{\sqrt{\hat{\mathbf{s}}_t}+\epsilon}\right)$
  update the sample distribution:
    $p_t \leftarrow update(p_{t-1}, I_t, \{\mathbf{g}_t^k\}_{k=1}^K)$
end while
Return: $\mathbf{x}_t$

Next, we set out the following theorem under online learning framework to prove that our proposed algorithm has a guaranteed regret bound, which indicates the convergence of AdaBeliefBS.

Theorem 1: Assume that the gradient of the loss function is bounded, i.e, $\|\nabla f_t(x_i) - \nabla f_t(x_j)\|_\infty \le G_\infty$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{R}^d$, and the convex feasiable set $\mathcal{R}^d$ is also bounded, i.e., $\|\mathbf{x}_i - \mathbf{x}_j\| \le D_\infty$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{R}^d$, where G_∞ and D_∞ are constants. Moreover, suppose that $\gamma = \frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\beta_{1t} = \beta_1\lambda^{t-1}, \lambda \in (0,1)$, and $\alpha_t = \frac{\alpha}{\sqrt{t}}$. Then, the regret of AdaBeliefBS has the following bound:

$$R(T) \le \frac{\sqrt{d}\alpha(1+\beta_1)\sqrt{1+\log T}}{\sqrt{2c}(1-\beta_1)^3}\sqrt{\frac{1}{KN^2}\sum_{t=1}^{T}\mathbb{E}\left[\sum_{j=1}^{N}\frac{\|\mathbf{g}_t^j\|^2}{p_t^j}\right]} + \frac{dG_\infty D_\infty^2\sqrt{T}}{2\alpha(1-\beta_1)} + \frac{D_\infty^2\beta_1 G_\infty}{2\alpha(1-\beta_1)(1-\lambda)^2}. \tag{4}$$

Proof. [6] proved that the regret of AdaBelief is with the following upper bound:

$$\sum_{t=1}^{T}\left[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\right] \le \frac{D_\infty^2\sqrt{T}}{2\alpha(1-\beta_1)}\sum_{i=1}^{d}\sqrt{\hat{s}_{T,i}} + \frac{\alpha(1+\beta_1)\sqrt{1+\log T}}{2\sqrt{c}(1-\beta_1)^3}\sum_{i=1}^{d}\left\|g_{1:T,i}^2\right\|_2 + \frac{D_\infty^2\beta_1 G_\infty}{2\alpha(1-\beta_1)(1-\lambda)^2}. \tag{5}$$

We first consider the term $\sum_{i=1}^{d}\sqrt{\hat{s}_{T,i}}$ in inequation (5). Note that the second-order momentum in [6] is defined as follows:

$$\begin{aligned}\mathbf{s}_t &= \beta_2\mathbf{s}_{t-1} + (1-\beta_2)(\mathbf{g}_t - \mathbf{m}_t)^2,\\ \hat{\mathbf{s}}_t &= \frac{\mathbf{s}_t + \epsilon}{1-\beta_2^t}.\end{aligned} \tag{6}$$

Moreover, the first-order momentum $\mathbf{m}_t$ in [6] is:

$$\mathbf{m}_t = \beta_1\mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t. \tag{7}$$

Using the recursive algorithm for equation (7), we have the following upper bound:

$$m_{t,i} \le (1-\beta_1^T)G_\infty. \tag{8}$$

From (8), we obtain:

$$g_{t,i} - m_{t,i} = \beta_1 g_{t,i} - \beta_1 m_{t-1,i} \le \beta_1^T G_\infty. \tag{9}$$

Meanwhile, applying the recursive algorithm for equation (6), and by (8), we have:

$$\begin{aligned}s_{t,i} &= \beta_2 s_{t-1,i} + (1-\beta_2)(g_{t,i} - m_{t,i})^2\\ &\le (1-\beta_2)(g_{t,i} - m_{t,i})^2 + \beta_2(1-\beta_2)(g_{t-1,i} - m_{t-1,i})^2\\ &\quad + \ldots + \beta_2^{T-1}(1-\beta_2)(g_{0,i} - m_{0,i})^2\\ &\le (1-\beta_2)\frac{1-\beta_2^T}{1-\beta_2}\left(\beta_1^T G_\infty\right)^2\\ &\le (1-\beta_2^T)G_\infty^2.\end{aligned} \tag{10}$$

Therefore, from inequation (10) and the 9-th step of Algorithm 1, we attain:

$$\sum_{i=1}^{d}\sqrt{\hat{s}_{T,i}} \le \sum_{i=1}^{d}\frac{s_{T,i}}{1-\beta_2^T} \le dG_\infty. \tag{11}$$

Note that the regularization ϵ is ignored for brevity as in other articles, such as [4], [6], [11]. Next, we consider the term $\sum_{i=1}^{d}\left\|g_{1:T,i}^2\right\|_2$ in (5).

$$\sum_{i=1}^{d}\left\|g_{1:T,i}^2\right\|_2 = \sum_{i=1}^{d}\sqrt{\sum_{t=1}^{T}\hat{G}_{t,i}^2} \le \sqrt{2\sum_{i=1}^{d}\sum_{t=1}^{T}\hat{G}_{t,i}^2} = \sqrt{2d\sum_{t=1}^{T}\left\|\hat{G}_t\right\|^2}. \tag{12}$$

The inequation in (12) holds because $f(x) = \sqrt{x}$ is concave, i.e., $\frac{f(x)+f(y)}{2} = \frac{\sqrt{x}+\sqrt{y}}{2} \le f(\frac{x+y}{2}) = \sqrt{\frac{x+y}{2}}$. Therefore, from inequations (11) and (12), inequation (5) can be rewritten as:

$$\begin{aligned}&\sum_{t=1}^{T}\left[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\right]\\ &\le \frac{\sqrt{d}\alpha(1+\beta_1)\sqrt{1+\log T}}{\sqrt{2c}(1-\beta_1)^3}\sqrt{\sum_{t=1}^{T}\left\|\hat{G}_t\right\|^2}\\ &+ \frac{D_\infty^2\beta_1 G_\infty}{2\alpha(1-\beta_1)(1-\lambda)^2} + \frac{dG_\infty D_\infty^2\sqrt{T}}{2\alpha(1-\beta_1)}.\end{aligned} \tag{13}$$

Note that the bound in inequation (13) is AdaBelief's regret bound. Since our proposed algorithm, AdaBeliefBS, uses bandit sampling to choose a mini-batch samples with K size, we have the following bound:

$$\begin{aligned}&\sum_{t=1}^{T}\left[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\right]\\ &\le \frac{\sqrt{d}\alpha(1+\beta_1)\sqrt{1+\log T}}{\sqrt{2c}(1-\beta_1)^3}\sqrt{\sum_{t=1}^{T}\mathbb{E}\left[\left\|\hat{G}_t\right\|^2\right]}\\ &+ \frac{D_\infty^2\beta_1 G_\infty}{2\alpha(1-\beta_1)(1-\lambda)^2} + \frac{dG_\infty D_\infty^2\sqrt{T}}{2\alpha(1-\beta_1)}.\end{aligned} \tag{14}$$

We next consider the upper bound of term $\mathbb{E}[\|\hat{G}_t\|^2]$ in the above inequation. Since that $\hat{G}_t = \frac{1}{K}\sum_{k=1}^{K}\hat{g}_t^k$ and $\hat{g}_t^k = \frac{\mathbf{g}_t^k}{np_t^k}$, we attain:

$$\begin{aligned}\mathbb{E}\left[\left\|\hat{G}_t\right\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{K}\sum_{k=1}^{K}\frac{\mathbf{g}_t^k}{Np_t^k}\right\|^2\right]\\ &\le \frac{1}{N^2K^2}\sum_{k=1}^{K}\mathbb{E}\left[\frac{\|\mathbf{g}_t^k\|^2}{(p_t^k)^2}\right] = \frac{1}{N^2K^2}\sum_{k=1}^{K}\mathbb{E}\left[\sum_{j=1}^{N}\frac{\|\mathbf{g}_t^j\|^2}{(p_t^j)^2}p_t^j\right]\\ &= \frac{1}{KN^2}\mathbb{E}\left[\sum_{j=1}^{N}\frac{\|\mathbf{g}_t^j\|^2}{p_t^j}\right].\end{aligned} \tag{15}$$

Therefore, plugging inequation (15) into (14), we obtain the following bound for the regret:

$$\begin{aligned}R(T) &= \sum_{t=1}^{T}\left[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\right]\\ &\le \frac{\sqrt{d}\alpha(1+\beta_1)\sqrt{1+\log T}}{\sqrt{2c}(1-\beta_1)^3}\sqrt{\frac{1}{KN^2}\sum_{t=1}^{T}\mathbb{E}\left[\sum_{j=1}^{N}\frac{\|\mathbf{g}_t^j\|^2}{p_t^j}\right]}\\ &+ \frac{dG_\infty D_\infty^2\sqrt{T}}{2\alpha(1-\beta_1)} + \frac{D_\infty^2\beta_1 G_\infty}{2\alpha(1-\beta_1)(1-\lambda)^2}.\end{aligned} \tag{16}$$

Therefore, the proof of Theorem 1 is completed. ■

The update strategy $p_t \leftarrow update(p_{t-1}, I_t, \{\mathbf{g}_t^k\}_{k=1}^K)$ in Algorithm 1 follows Algorithm 2 in [11]. For the sake of algorithm self-containess, we present the update strategy in Algorithm 2. Note that the number of arms is N, L denotes the upper bound, I_t represents the mini-batch of training samples at time t, and $D_{KL}(q\|w_t)$ indicates the KL divergence between q and w_t. Moreover, the set $\mathcal{P} := \{p \in \mathcal{R}^N : \sum_{j=1}^{N} p_j = 1, p_j \ge p_{\min}\forall j\}$, where $p_{\min}$ is a constant.

Algorithm 2 The distribution update strategy for p_t [11].

Function: $p_t \leftarrow update(p_{t-1}, I_t, \{\mathbf{g}_t^k\}_{k=1}^K)$
Compute p_t^j:
  $p_t^j = \frac{\|\mathbf{g}_t^j\|^2}{\sum_{i=1}^{N}\|\mathbf{g}_t^i\|}$
Compute the loss:
  $l_t^j = -\frac{\|\mathbf{g}_t^j\|^2}{(p_t^j)^2} + \frac{L^2}{p_{\min}^2}$, if $j \in I_t$; otherwise, $l_t^j = 0$
Compute an unbiased gradient estimate:
  $\hat{h}_t^j = \frac{l_t^j\sum_{k=1}^{K}\mathbb{1}(j=I_t^k)}{Kp_t^j}, \forall 1 \le j \le N$
  $w_t^j = p_{t-1}^j\exp(-\alpha_p\hat{h}_t^j), \forall 1 \le j \le N$
  $p_t = \arg\min_{q\in\mathcal{P}} D_{KL}(q\|w_t)$
Return: p_t

Obviously, inequality (4) in Theorem 1 shows that p_t affects the regret bound, thereby the optimal p_t that promotes the convergence rate is desirable. Although the optimal solution is $p_t^j = \frac{\|\mathbf{g}_t^j\|^2}{\sum_{i=1}^N \|\mathbf{g}_t^i\|}$ for the problem $\arg\min_{\sum_{i=1}^N p_t^i=1} \sum_{j=1}^N \frac{\|\mathbf{g}_t^j\|^2}{p_t^j}$, its computation is prohibitive due to the gradient calculation of all samples at each iteration. For this reason, Algorithm 2 proposed by [11] uses a multi-armed bandit method based on EXP3 [23] to overcome this problem. To further obtain the upper bound of the regret, we introduce the following lemma from [11].

Lemma 1: If let $B_\phi(x,y) = \phi(x) - \langle \nabla\phi(y), x-y\rangle - \phi(y)$ be the Bregman divergence associated with ϕ, where $\phi(p) = \sum_{j=1}^N p^j \log p^j$, assume that $\|\mathbf{g}_t^j\| \le L$, $p_t^j \ge p_{\min}$ for all $t \in [T]$ and $j \in [N]$, and $B_\phi(p,p') \le R^2$ for any $p, p' \in \mathcal{P}$, set $\alpha_p = \sqrt{\frac{2R^2 p_{\min}^4}{NTL^4}}$, the *update* strategy in Algorithm 2 implies:

$$\mathbb{E}\left[\sum_{t=1}^T\sum_{j=1}^N \frac{\|\mathbf{g}_t^j\|^2}{p_t^j}\right] - \min_{p\in\mathcal{P}}\left[\sum_{t=1}^T\sum_{j=1}^N \frac{\|\mathbf{g}_t^j\|^2}{p_t^j}\right] \le \frac{RL^2}{p_{\min}^2}\sqrt{2NT}. \tag{17}$$

The proof of Lemma 1 has been provided in Appendix B in [11]. Furthermore, from Theorem 1 and Lemma 1, we have the following corollary.

Corollary 1: Suppose that assumptions in Theorem 1 and Lemma 1 are satisfied, AdaBeliefBS with the distribution *update* strategy attain the following regret bound:

$$\begin{aligned} R(T) \le & \frac{\sqrt{d}\alpha(1+\beta_1)\sqrt{1+\log T}}{N\sqrt{2cK}(1-\beta_1)^3}\frac{L\sqrt{R}}{p_{\min}}(2NT)^{1/4} \\ & + \frac{\sqrt{d}\alpha(1+\beta_1)\sqrt{1+\log T}}{N\sqrt{K}\sqrt{2c}(1-\beta_1)^3}\sqrt{\min\sum_{t=1}^T \mathbb{E}\left[\sum_{j=1}^N \frac{\|\mathbf{g}_t^j\|^2}{p_t^j}\right]} \\ & + \frac{dG_\infty D_\infty^2\sqrt{T}}{2\alpha(1-\beta_1)} + \frac{D_\infty^2\beta_1 G_\infty}{2\alpha(1-\beta_1)(1-\lambda)^2}. \end{aligned} \tag{18}$$

Obviously, Theorem 1 and Corollary 1 prove that AdaBeliefBS converges under online convex conditions.

Indeed, the regret bound $R(T)$ can be further bounded under the doubly heavy-tailed distribution, which is tighter than AdaBelief with uniform sampling. The doubly heavy-tailed distribution is a more extensive distribution of random variables than the normal distribution, and it mainly shows that a small number of samples take up a large number of resources. Specifically, let $\mathbf{g}_t^k$ represent the gradient of k-th sample at time t, $\mathbf{z}_t^k$ denote the feature vector for k-th sample at time t, and $z_{t,i}^k$ is the i-th dimension of $\mathbf{z}_t^k$. Then the case of doubly heavy-tailed distribution implies that $\|\mathbf{g}_t^k\|^2 \le \|\mathbf{z}_t^k\|^2 = \sum_{i=1}^d (z_{t,i}^k)^2 = \beta_3 i^{-\gamma} k^{-\gamma}$, where d is the dimensional number, β_3 and γ are constants with $\gamma \ge 2$. In this case, the regret bounds of AdaBelief and AdaBeliefBS are shown in the following theorems.

Theorem 2: If feature vector follows doubly heavy-tailed distribution, for a neural network with ReLU hidden layer and sigmoid output layer, AdaBelief achieves the following regret bound:

$$R(T) \le O(d\sqrt{T}) + O\left(\underbrace{\frac{\sqrt{N\log N}}{N}}_{(A)}\frac{\sqrt{d\log d}}{\sqrt{K}}\sqrt{T(1+\log T)}\right). \tag{19}$$

The proof of Theorem 2 can be easily obtained by following Theorem 3 of [11].

Theorem 3: If feature vector follows doubly heavy-tailed distribution, for a neural network with ReLU hidden layer and sigmoid output layer, AdaBeliefBS achieves the following regret bound:

$$R(T) \le O(d\sqrt{T}) + O\left(\underbrace{\frac{\sqrt{\log^2 N}}{N}}_{(B)}\frac{\sqrt{d\log d}}{\sqrt{K}}\sqrt{T(1+\log T)}\right). \tag{20}$$

The proof of Theorem 3 is shown in Appendix A.

From inequalities (19) and (20), their different terms, (A) and (B), distinctly indicate that the proposed algorithm converges faster than AdaBelief when the feature vector follows doubly heavy-tailed distribution under convex conditions. In short, this work first proves that bandit sampling benefits the convergence rate for AdaBelief in convex optimization. Additionally, we also make a further step: we are interested in whether bandit sampling improves the convergence rate in strongly convex optimization. To this end, we develop the following strongly convex optimization algorithm with bandit sampling.

V. Bandit Sampling for Strongly Convex Optimization

For strongly convex optimization, we present the following optimization algorithm which is improved by bandit sampling. This algorithm named SAdamBS is based on SAdam [16], a variant of Adam under strongly convex conditions. Moreover, its pseudo code is shown in Algorithm 3 in which the modifications from its original version are highlighted in blue font. Although there are not many modifications in terms of algorithm steps, these modifications will bring challenges to the convergence proof of the proposed algorithm.

Same as AdamBS, we take the unbiased gradient estimate $\hat{G}_t$ to compute the first-order and second-order momentums for SAdamBS. Under the conditions and assumptions of SAdam in [16], we can obtain the following regret bound for SAdamBS.

Theorem 4: Assume that the gradient of the loss function is bounded, i.e, $\|\nabla f_t(x_i) - \nabla f_t(x_j)\|_\infty \le G_\infty$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{R}^d$, and the convex feasiable set $\mathcal{R}^d$ is also bounded, i.e., $\|\mathbf{x}_i - \mathbf{x}_j\| \le D_\infty$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{R}^d$, where G_∞ and D_∞ are constants. Moreover, suppose that all loss functions $\{f_1(\cdot), f_2(\cdot), \ldots, f_T(\cdot)\}$ are λ-strongly convex. Set $\alpha \ge \frac{C}{\lambda}$. Let $\delta > 0, \beta_{1t} = \beta_1\nu^{t-1}$ where $\beta_1 \in [0,1)$ and $\{\beta_{2t}\}_{t=1}^T \in [0,1]^T$ satisfy the Conditions 3 and 4 in [16]. We have the following bound for the regret of SAdamBS:

$$\begin{aligned} R(T) \le & \frac{\alpha\zeta}{(1-\beta_1)^3}\sum_{i=1}^d \log\left(\frac{1}{\zeta\delta KN^2}\sum_{t=1}^T \mathbb{E}\left[\sum_{j=1}^N \frac{(g_{t,i}^j)^2}{p_t^j}\right] + 1\right) \\ & + \frac{dD_\infty^2\delta}{2\alpha(1-\beta_1)} + \frac{d\beta_1 D_\infty^2(G_\infty^2+\delta)}{2\alpha(1-\beta_1)(\nu-1)^2}. \end{aligned} \tag{21}$$

The proof of Theorem 4 is provided in Appendix B. By applying Lemma 1 into Theorem 4, we have the following corollary.

Algorithm 3 SAdamBS

Input: $\{\beta_{1t}\}_{t=1}^{T}, \{\beta_{2t}\}_{t=1}^{T}, \delta$
Initialize: $\mathbf{x}_0, \mathbf{m}_0 \leftarrow \mathbf{0}, \mathbf{v}_0 \leftarrow \mathbf{0}, p_0^k \leftarrow 1/n, \forall 1 \leq k \leq N$, and set time start with $t = 0$.

while $\mathbf{x}_t$ not converged **do**
  $t \leftarrow t + 1$
  select a mini-batch of K samples by sampling with replacement from p_{t-1}
  compute original gradient of one sample:
    $\mathbf{g}_t \leftarrow \nabla f_t(\mathbf{x}_{t-1})$
  compute the unbiased gradient estimate of one mini-batch:
    $\hat{G}_t = \frac{1}{K}\sum_{k=1}^{K} \frac{\mathbf{g}_t^k}{n p_{t-1}^k}$
  compute the first-order momentum:
    $\mathbf{m}_t \leftarrow \beta_{1t}\mathbf{m}_{t-1} + (1-\beta_{1t})\hat{G}_t$
  compute the second-order momentum based on gradient belief:
    $\mathbf{v}_t \leftarrow \beta_{2t}\mathbf{v}_{t-1} + (1-\beta_{2t})\hat{G}_t^2$
  compute the diagonal matrix:
    $V_t = diag\{\mathbf{v}_t\}$, $\hat{V}_t = V_t + \frac{\delta}{t} I_d$
  update the decision vector $\mathbf{x}_t$:
    $\mathbf{x}_t \leftarrow \prod_{\mathcal{F}, \hat{V}_t}\left(\mathbf{x}_t - \frac{\alpha}{t}\hat{V}_t^{-1}\mathbf{m}_t\right)$
  update the sample distribution:
    $p_t \leftarrow update\left(p_{t-1}, I_t, \{\mathbf{g}_t^k\}_{k=1}^{K}\right)$
end while
Return: $\mathbf{x}_t$

Corollary 2: Suppose that assumptions of both Lemma 1 and Theorem 4 are satisfied, SAdamBS obtains the following convergence rate

$$R(T) \leq \frac{\alpha\zeta}{(1-\beta_1)^3}\sum_{i=1}^{d}\log\left(\frac{1}{\zeta\delta K N^2}\left(M + \frac{RL^2}{p_{\min}^2}\right) + 1\right) + \frac{dD_\infty^2\delta}{2\alpha(1-\beta_1)} + \frac{d\beta_1 D_\infty^2(G_\infty^2+\delta)}{2\alpha(1-\beta_1)(\nu-1)^2}, \tag{22}$$

where $M = \min_p \sum_{t=1}^{T}\mathbb{E}\left[\sum_{j=1}^{N}\frac{(g_{t,i}^j)^2}{p_t^j}\right]$.

The proof of Corollary 2 can be obtained by plugging Lemma 1 into Theorem 4. Since SAdamBS uses the same *update* strategy as AdamBS, Lemma 1 also holds in SAdamBS. Obviously, this corollary proves that SAdamBS has an upper regret bound. Therefore, SAdamBS is proved to converge in strongly convex cases.

Next, we demonstrate that the regret bound of SAdamBS can be further bounded when the feature vector follows the doubly heavy-tailed distribution. SAdamBS's convergence rate is provably better than that of SAdam, which uses uniform sampling.

Theorem 5: If the feature vector follows doubly heavy-tailed distribution, for a neural networks with the ReLU hidden layer and sigmoid output layer, SAdam achieves the following regret bound:

$$R(T) \leq O(d\log(TN\log N\log d)). \tag{23}$$

The proof of Theorem 5 is presented in Appendix C.

On the other hand, SAdamBS achieves the following convergence rate for the same case.

Theorem 6: If the feature vector follows doubly heavy-tailed distribution, for a neural network with ReLU hidden layer and sigmoid output layer, SAdamBS achieves the following convergence rate:

$$R(T) \leq O\left(d\log\left(T\frac{\log^2 N}{KN^2}\log d\right)\right). \tag{24}$$

Proof. From Lemma 1, we have

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{N}\frac{(g_{t,i}^j)^2}{p_t^j}\right] \leq \min_{p\in\mathcal{P}}\left[\sum_{t=1}^{T}\sum_{j=1}^{N}\frac{(g_{t,i}^j)^2}{p_t^j}\right] + \frac{RL^2}{p_{\min}^2}\sqrt{2NT}. \tag{25}$$

Moreover, reviewing Lemma 3 of [11], we attain that

$$\min_{p_t^j \geq p_{\min}}\sum_{j=1}^{N}\sum_{i=1}^{d}\frac{(g_{t,i}^j)^2}{p_{t,i}^j} = O(\log d\log^2 N). \tag{26}$$

Plugging inequation (25) and equation (26) into Corollary 2, we obtain

$$R(T) \leq O\left(d\log\left(T\frac{\log^2 N}{KN^2}\log d\right)\right). \tag{27}$$

Therefore, the proof of Theorem 6 is completed. ■

Comparing the regret bounds of SAdam and SAdamBS, we find that our proposed SAdamBS achieves a faster convergence rate than SAdam. Therefore, bandit sampling is proved to also improve the convergence rate of optimization algorithms under strongly convex conditions. To further demonstrate that our proposed algorithm converges faster, we execute our algorithms and other classical algorithms on the public datasets in the next section.

VI. Experiments

In the above content, we prove that the two proposed algorithms, AdaBeliefBS and SAdamBS, have tighter convergence bounds than AdaBelief and SAdam in convex and strongly convex cases, respectively. In order to verify the above theoretical results in practice, we conduct a series of experiments in this section. In our experiments, we use 1) SGD, 2) SGD with momentum (SGD-M) with $O(\frac{1}{t})$ decaying step size, and 3) RMSProp [31] with parameter settings $\beta_t = 0.9, \gamma = 0.9, \delta = 10^{-8}$, as comparison algorithms.

The experiments are completed on four 1080-Ti GPUs. For the comparison algorithms, we use the same parameter initialization to ensure fairness. All the experiments are executed five times, and their mean values and the corresponding confidence intervals are taken as the final results. Moreover, we use four classical benchmark datasets in the experiments: MNIST [24], CIFAR-10 [25], CIFAR-100 [25], and Penn TreeBank [26]. MNIST is a famous public dataset of handwritten digits, which has a training set of 60,000 samples, and a test set of 10,000 samples. CIFAR-10 and CIFAR-100 are both consists of 60,000 color images of size 32×32, where CIFAR-10 has 10 classes with 6,000 images per class and CIFAR-100 has 100 classes with 600 images per class. Penn TreeBank is a well-known corpus dataset, widely used in nature language

TABLE I
SUMMARY OF MODELS AND DATASETS.

Our Algorithm	Task	Model	Dataset
AdaBeliefBS	Image Classification	VGG-11, ResNet-34, DenseNet-121	CIFAR-10, CIFAR-100
AdaBeliefBS	Language Modeling	1,2,3-layers LSTM	Penn TreeBank
SAdamBS	Softmax Regression	-	MNIST, CIFAR-10, CIFAR-100
SAdamBS	Image Classification	2,4-layers CNN	MNIST, CIFAR-10, CIFAR-100

processing tasks, which contains 1 million words come from 2,499 articles. In this section, we conduct two experiments: the first one is to validate that our proposed algorithm AdaBeliefBS outperforms the other algorithms on convergence under convex conditions; the second one is to verify that our proposed algorithm SAdamBS converges faster than the other algorithms in strongly convex cases.

A. AdaBeliefBS: Bandit Sampling for Convex Optimization

To examine the performance of AdaBeliefBS, we execute extensive experiments to train two classical deep neural networks: convolutional neural networks (CNN) and recurrent neural networks (RNN). The relationship between models and datsets are summarized in Table I. Moreover, experiments are implemented in Keras with TensorFlow based on the codes from [10][1] and [6][2]. Since the importance sampling method proposed by [10] could also be applied to AdaBelief, we use it as a comparison algorithm and call it AdaBelief-IS. To show the performance of our proposed algorithm on training acceleration, we plot the curves of training loss vs. epochs for the experimental results that follows [11].

To be fair, we set the same hyper-parameters as the original settings for all executed algorithms. In AdaBelief-IS and AdaBeliefBS, we set their hyper-parameters, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-14$, and $\alpha = 1e-3$, following their original method AdaBelief [6].

1) CNNs on Image Classification: We perform this group of experiments on CIFAR-10 and CIFAR-100 with VGG-11 [27], ResNet-34 [28] and DenseNet-121 [29], respectively. We report the average of five runs and the corresponding confidence intervals in Figure 2. As we can see, Figure 2 shows that the proposed algorithm converges faster than the others in a fixed number of epochs on CIFAR-10 and CIFAR-100. The curves in Figure 2 validate that the regret bound of our proposed algorithm AdaBeliefBS is more compact than AdaBelief. As the generalization ability is another important indicator of training algorithm, we need to verify whether our method has a negative effect on the generalization performance of the original algorithm. For this reason, we further examine the generalization results of all executed algorithms on CIFAR-10 and CIFAR-100. These results are reported in Figure 3, which plots the curves of test accuracy vs. epochs. Obviously, within 200 epochs on CIFAR-10 and CIFAR-100, our proposed algorithms achieve high test accuracy at the fastest speed. Our algorithms' highest accuracy in many cases is almost the same as or better than that of SGD and other excellent algorithms. Inherited from AdaBelief, the proposed algorithm uses more reasonable momentum to choose more appropriate step size. More importantly, the proposed algorithm can lead to good generalization ability by extracting more useful information at each iteration, which is empirically verified from the experiments.

2) RNNs on Language Modeling: In pattern recognition, natural language processing is another important application. For this reason, we further consider whether our proposed algorithm could bring benefit for language modeling tasks. To this end, we respectively adopt 1-layer, 2-layer and 3-layer LSTMs as the RNN model to conduct the experiments on Penn TreeBank. In this experiment, we report the curve of perplexity vs. epochs to validate our proposed algorithm's training acceleration. Note that the lower perplexity, the better. The experimental results are shown in Figure 4. As observed, our proposed algorithm is able to quickly obtain a lower perplexity than the other algorithms; this verifies the theoretical superiority of our proposed algorithm on the language modeling task.

B. SAdamBS: Bandit Sampling for Strongly Convex Optimization

In the original article [16], SAdam is proved to converge faster than the other algorithms under strongly convex and even non-convex conditions. Since the SAdamBS we proposed is derived from SAdam, SAdamBS is also a strongly convex training algorithm. For this reason, we follow the settings of SAdam and provide two groups of experiments to verify the performance of SAdamBS. The first is to solve the problem of mini-batch ℓ-regularized softmax regression, which is a typical strongly convex optimization problem. The second is to train artificial neural models for image classification, which is a classical non-convex optimization problem [16].

1) Regression Problem: In this experiment, we consider a classical online strongly convex optimization problem, i.e., the problem of mini-batch ℓ_2-regularized softmax regression. We exploit the following strongly convex function to measure the loss value for this problem:

$$J(\mathbf{w}, b) = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{e^{\mathbf{w}_{y_i}^\top \mathbf{x}_i + b_{y_i}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}_i + b_j}}\right) + \lambda_1 \sum_{k=1}^{K}\|\mathbf{w}_k\|^2 + \lambda_2 \sum_{k=1}^{K} b_k^2, \tag{28}$$

where $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ denotes a mini-batch of training samples, N is the batch size, $y_i \in [K], \forall i \in [N]$, K is the number of

[1]http://github.com/idiap/ importance-sampling
[2]http://github.com/juntang-zhuang/Adabelief-Optimizer

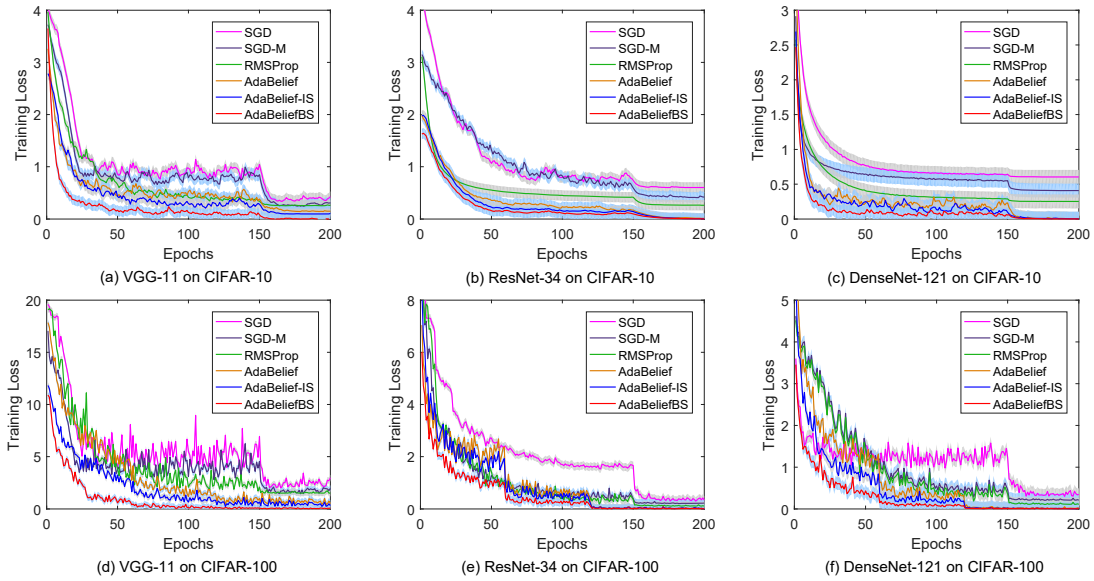

Fig. 2. Convex optimization: Training loss of image classification on CIFAR-10 and CIFAR-100 for 200 epochs.

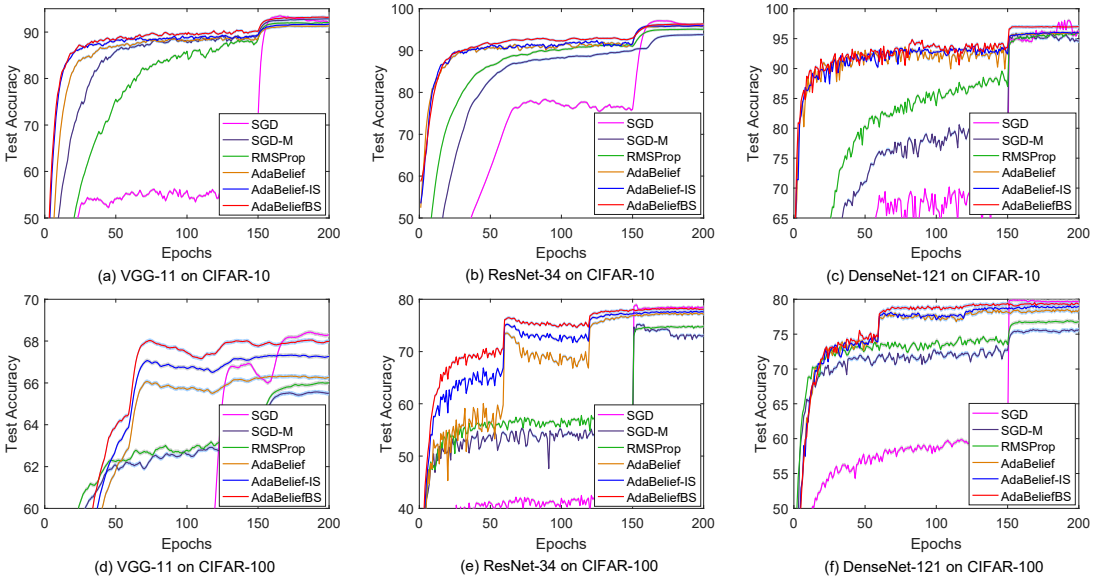

Fig. 3. Convex optimization: Test accuracy of image classification on CIFAR-10 and CIFAR-100 for 200 epochs.

classes, and $\{\mathbf{w}_i, b_i\}_{i=1}^K$ represents model parameters. Moreover, both λ_1 and λ_2 are set to 0.01.

The results are shown in Figure 5. It can be seen that our proposed algorithm quickly achieves lower loss than the others, which verifies the acceleration effect of the bandit sampling method on strongly convex algorithms. Therefore, the faster convergence of SAdamBS is verified in the strongly convex optimization task. Moreover, we further conduct a group of experiments to validate the performance of our proposed algorithm on test accuracy, which is plotted in Figure 6. It shows that our proposed algorithm again achieve a good performance on test accuracy in 100 epochs on the ℓ_2-regularized softmax regression problem.

2) Image Classification: In this experiment, we evaluate our proposed algorithm and the comparison algorithms in a non-convex optimization task. Specifically, we use the same experimental settings as [16]: for MNIST, the CNN model contains two layers of 3×3 filters, its max pooling is with size 2×2 applied to the second convolution layer that follows a fully connected layer of 128 hidden units; for CIFAR-10 and CIFAR-100, the CNN model has 4 layers of 3×3 filters and 2×2 max pooling, its dropping probability is 0.25 applied to the second and fourth convolution layers, and it is followed by a fully connected layer of 512 hidden units.

The experimental results for image classification are shown in Figure 7. As we know, recent studies have indicated that the original SAdam could achieve superior performance even in artificial neural network training tasks which are highly non-convex cases [16]. For this reason, we present this group of experiments to validate that our proposed algorithm also

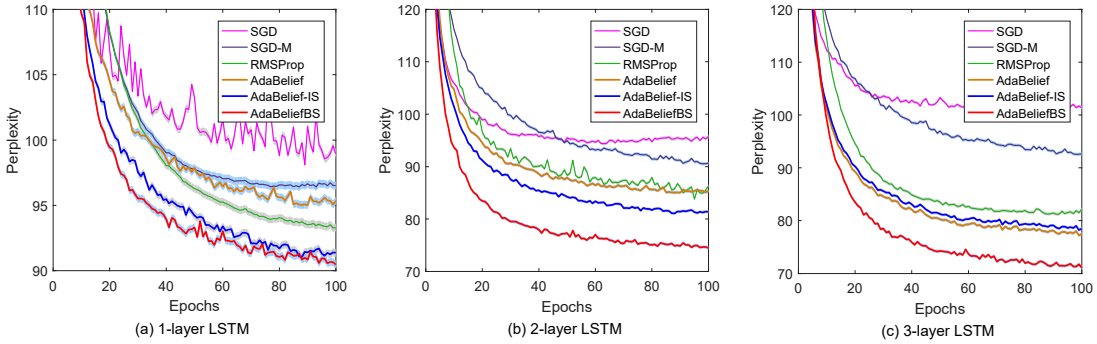

Fig. 4. Convex optimization: Perplexity of language modeling on Penn TreeBank for 100 epochs.

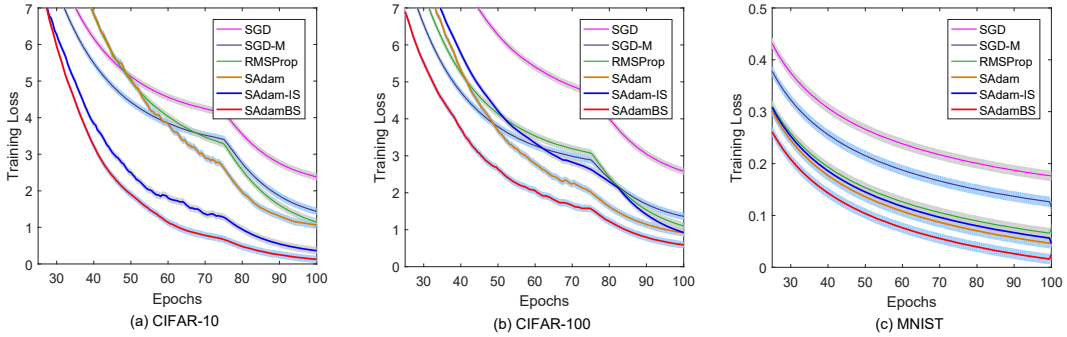

Fig. 5. Strongly convex optimization: Training loss for ℓ_2-regularized softmax regression with 100 epochs.

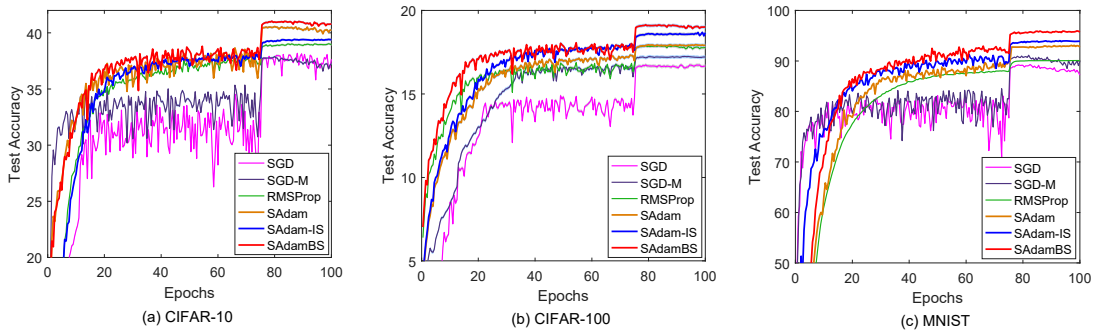

Fig. 6. Strongly convex optimization: Test accuracy on ℓ_2-regularized softmax regression with 100 epochs.

outperforms the others on quickly achieving lower training loss in machine learning cases. Furthermore, we observe the test accuracy performance of all executed algorithms in this experiment under strongly convex settings. Although our aim is to propose a faster convergence algorithm, in the image classification task using the 4 layers network in the non-convex case, the resulting curves in Figure 8 show that our proposed algorithm has test accuracy advantage within 100 epochs. We believe this is the additional benefit brought by the proposed algorithm for extracting informative samples at each iteration.

VII. Conclusion and Future Work

In this paper, we present two accelerated optimization algorithms exploiting bandit sampling. First, we focus on the convex optimization problem, and propose AdaBeliefBS that extends the general framework of AdaBelief. AdaBeliefBS utilizes the bandit sampling method to select informative samples from the full training sample set, thereby accelerating the convergence rate of the convex optimization algorithm. Second, we further consider whether bandit sampling could accelerate the training process of strongly convex optimization. The proposed algorithm SAdamBS follows the framework of SAdam and accelerates the strongly convex algorithm by exploiting bandit sampling. Moreover, we provide complete proofs for the convergence of our proposed algorithms. The theoretical conclusions indicate that the two proposed algorithms achieve tighter regret bounds compared to their original variants. Furthermore, to verify our algorithms empirically, we conduct a series of experiments on benchmark MNIST, CIFAR-10, CIFAR-100, and Penn TreeBank datasets for deep learning and machine learning tasks including image classification and language modeling. The results clearly show

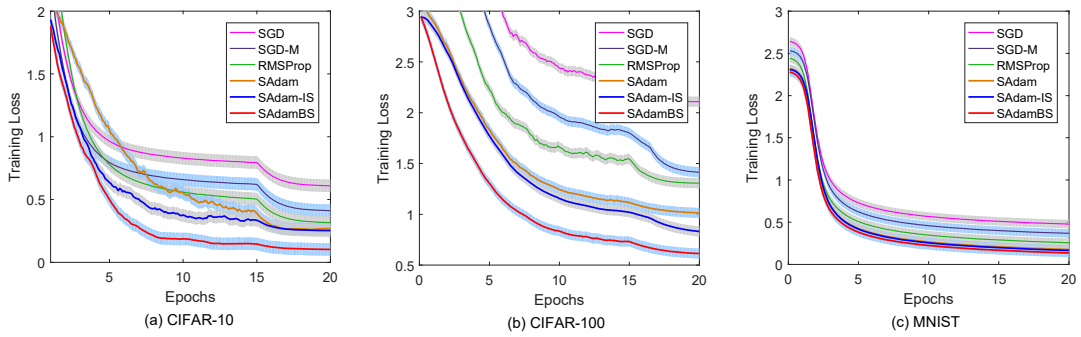

Fig. 7. Strongly convex optimization: Training loss on image classification (non-convex case) with 20 epochs.

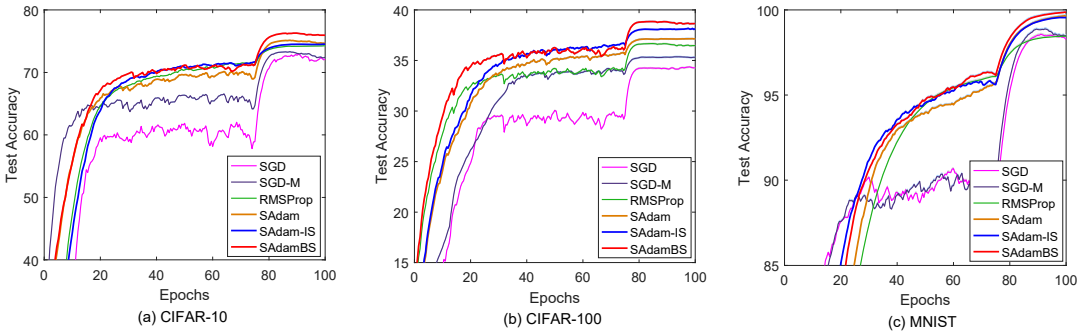

Fig. 8. Strongly convex optimization: Test accuracy image classification (non-convex case) with 100 epochs.

that our proposed algorithms are effective in accelerating the training processes for different deep learning and machine learning tasks whilst maintaining good generalization ability.

We have demonstrated that the bandit sampling method can accelerate AdaBelief and SAdam, and infer that bandit sampling can benefit most training algorithms. However, the theoretical proof to support this general inference has not yet been established. Therefore, we plan to address this outstanding challenge in future.

VIII. Broader Impact

Pattern recognition and deep learning have been widely applied in many fields, such as image recognition and nature language processing. However, the time-consuming training process brings difficulties for practical application of deep learning. To address this open problem, we innovatively exploit the bandit sampling method to reduce training time of both convex and strongly convex algorithms. Our proposed algorithms can select informational samples to train the models by utilising bandit sampling. We believe our methods: AdaBeliefBS and SAdamBS can be applied to all scenarios where their original versions can be used, and take less training time. In conclusion, our methods have the potential to deliver broad impact in many real-life applications that rely on these algorithms.

Acknowledgments

This work was partially supported by Suzhou Foreign Experts Project under grant No. E290010201, and Chinese Academy of Sciences Project under grant No. E21Z010101. This work was also supported by the innovation workstation of Suzhou Institute of Nano-Tech and Nano-Bionics (SINANO) under grant No. E010210101. Huang would like to acknowledge the support of National Natural Science Foundation of China under No. 61876155, and Jiangsu Science and Technology Programme under No. BE2020006-4. Hussain would like to acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) - Grants Ref. EP/M026981/1, EP/T021063/1, EP/T024917/1.

Appendix

A. Proof of Theorem 2

Proof. Lemma 3 in [11] has proved that

$$\min_{p_j \geq p_{\min}} \sum_{j=1}^{N} \frac{\|\mathbf{g}_j\|^2}{p_j} \leq \beta_3 \log d \min_{p \geq p_{\min}} \sum_{j=1}^{N} \frac{j^{-\gamma}}{p_j}, \tag{29}$$

where $\beta_3 \in (0, 1]$, and $\gamma > 2$. Plugging equation (29) into inequation (18), we have:

$$\begin{aligned} R(T) \leq & \frac{\sqrt{d}\alpha(1+\beta_1)\sqrt{1+\log T}}{N\sqrt{2cK}(1-\beta_1)^3} \frac{L\sqrt{R}}{p_{\min}} (2NT)^{1/4} \\ & + \frac{\sqrt{d}\alpha(1+\beta_1)\sqrt{1+\log T}}{N\sqrt{K}\sqrt{2c}(1-\beta_1)^3} \sqrt{\beta_3 T \log d \min_{p \geq p_{\min}} \sum_{j=1}^{N} \frac{j^{-\gamma}}{p_j}} \\ & + \frac{dG_\infty D_\infty^2 \sqrt{T}}{2\alpha(1-\beta_1)} + \frac{D_\infty^2 \beta_1 G_\infty}{2\alpha(1-\beta_1)(1-\lambda)^2}. \end{aligned} \tag{30}$$

From Proposition 5 in [30], we know

$$\beta_3 \log d \min_{p \geq p_{\min}} \sum_{j=1}^{N} \frac{j^{-\gamma}}{p_j} = O(\log^2 N). \tag{31}$$

By inequation (30) and equation (31), we attain:

$$R(T) \leq O(d\sqrt{T}) + O\left(\frac{\sqrt{\log^2 N}}{N}\frac{\sqrt{d\log d}}{\sqrt{K}}\sqrt{T(1+\log T)}\right). \quad (32)$$

Therefore, the proof of Theorem 2 is completed. ■

B. Proof of Theorem 4

Proof. According to Theorem 1 from [16], we know that the regret bound of SAdam is as follows:

$$\sum_{t=1}^{T}\left[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\right] \leq \frac{\alpha\zeta}{(1-\beta_1)^3}\sum_{i=1}^{d}\log\left(\frac{1}{\zeta\delta}\sum_{j=1}^{T} g_{j,i}^2 + 1\right) + \frac{dD_\infty^2\delta}{2\alpha(1-\beta_1)} + \frac{d\beta_1 D_\infty^2(G_\infty^2+\delta)}{2\alpha(1-\beta_1)(\nu-1)^2}. \quad (33)$$

Since we use bandit sampling in the SAdamBS optimization algorithm, the gradient expectation should be considered in its convergence analysis. For this reason, the regret bound of SAdamBS can be attained as follows from inequation (33):

$$\sum_{t=1}^{T}\left[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\right] \leq \frac{\alpha\zeta}{(1-\beta_1)^3}\sum_{i=1}^{d}\log\left(\frac{1}{\zeta\delta}\sum_{t=1}^{T}\mathbb{E}\left[\hat{G}_{t,i}\right]^2 + 1\right) + \frac{dD_\infty^2\delta}{2\alpha(1-\beta_1)} + \frac{d\beta_1 D_\infty^2(G_\infty^2+\delta)}{2\alpha(1-\beta_1)(\nu-1)^2}. \quad (34)$$

For a mini-batch with K samples at time t, its gradient of the i-th dimension is $\hat{G}_{t,i} = \frac{1}{K}\sum_{k=1}^{K}\hat{g}_{t,i}^k = \frac{1}{K}\sum_{k=1}^{K}\frac{g_{t,i}^k}{Np_t^k}$. Therefore, inequation (34) can be rewritten as:

$$\begin{aligned}
&\sum_{t=1}^{T}\left[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\right] \\
&\leq \frac{\alpha\zeta}{(1-\beta_1)^3}\sum_{i=1}^{d}\log\left(\frac{1}{\zeta\delta}\sum_{t=1}^{T}\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\frac{g_{t,i}^k}{Np_t^k}\right]^2 + 1\right) + \frac{dD_\infty^2\delta}{2\alpha(1-\beta_1)} + \frac{d\beta_1 D_\infty^2(G_\infty^2+\delta)}{2\alpha(1-\beta_1)(\nu-1)^2} \\
&\leq \frac{\alpha\zeta}{(1-\beta_1)^3}\sum_{i=1}^{d}\log\left(\frac{1}{\zeta\delta N^2K^2}\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\left[\frac{g_{t,i}^k}{p_t^k}\right]^2 + 1\right) + \frac{dD_\infty^2\delta}{2\alpha(1-\beta_1)} + \frac{d\beta_1 D_\infty^2(G_\infty^2+\delta)}{2\alpha(1-\beta_1)(\nu-1)^2} \\
&= \frac{\alpha\zeta}{(1-\beta_1)^3}\sum_{i=1}^{d}\log\left(\frac{1}{\zeta\delta N^2K^2}\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\left[\sum_{j=1}^{N}\frac{(g_{t,i}^j)^2}{(p_t^j)^2}p_t^j\right] + 1\right) + \frac{dD_\infty^2\delta}{2\alpha(1-\beta_1)} + \frac{d\beta_1 D_\infty^2(G_\infty^2+\delta)}{2\alpha(1-\beta_1)(\nu-1)^2} \\
&= \frac{\alpha\zeta}{(1-\beta_1)^3}\sum_{i=1}^{d}\log\left(\frac{1}{\zeta\delta KN^2}\sum_{t=1}^{T}\mathbb{E}\left[\sum_{j=1}^{N}\frac{(g_{t,i}^j)^2}{p_t^j}\right] + 1\right) + \frac{dD_\infty^2\delta}{2\alpha(1-\beta_1)} + \frac{d\beta_1 D_\infty^2(G_\infty^2+\delta)}{2\alpha(1-\beta_1)(\nu-1)^2}.
\end{aligned} \quad (35)$$

Therefore, the proof of Theorem 4 is completed. ■

C. Proof of Theorem 5

Proof. In SAdam, the examples are sampled with uniform distribution, thus $p_j = 1/N$ for all $j \in [1, N]$. From Lemma 2 of [11], we have

$$\mathbb{E}\left[\sum_{j=1}^{N}\frac{\|g_{t,i}^j\|^2}{p_t^j}\right] = \beta_3 N\log N\log d. \quad (36)$$

By plugging inequation (36) into Theorem 1 of [16], we directly attain inequation (23). Therefore, the proof of Theorem 5 is completed. ■

REFERENCES

[1] G. Zhong, W. Jiao, *et al.* "Automatic Design of Deep Networks with Neural Blocks," *Cognitive Computation*, vol. 12, pp. 1–12, 2020.

[2] Y. Zhou, K. Huang, *et al.* "LightAdam: Towards a Fast and Accurate Adaptive Momentum Online Algorithm," *Cognitive Computation*, 2022, https://doi.org/10.1007/s12559-021-09985-9.

[3] B. Jimmy and K. Diederik. "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference for Learning Representations*. 2015.

[4] S. J. Reddi, S. Kale, and S. Kumar. "On the Convergence of Adam and Beyond," in *International Conference on Learning Representations*, 2018.

[5] L. Luo, Y. Xiong, and Y. Liu. "Adaptive Gradient Methods with Dynamic Bound of Learning Rate," in *International Conference on Learning Representations*, 2019.

[6] J. Zhuang, T. Tang, *et al.* "AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients," in *Proceedings of Neural Information Processing Systems*, vol. 33, pp. 18795–18806, 2020.

[7] A. Bordes, S. Ertekin, *et al.* "Fast Kernel Classifiers with Online and Active Learning," *Journal of Machine Learning Research*, vol. 6, pp. 1579–1619, 205.

[8] O. Canevet, C. Jose, and F. Fleuret. "Importance Sampling Tree for Large-scale Empirical Expectation," in *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1454–1462, 2016.

[9] D. Csiba, Z. Qu, and P. Richtarik. "Stochastic Dual Coordinate Ascent with Adaptive Probabilities," in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 674–683, 2015.

[10] A. Katharopoulos and F. Fleuret. "Not All Samples Are Created Equal: Deep Learning with Importance Sampling," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 2525–2534, 2018.

[11] R. Liu, T. Wu, and B. Mozafari. "Adam with Bandit Sampling for Deep Learning," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 5393–5404, 2020.

[12] M. Zhang, J. Lucas, James, *et al.* "Lookahead Optimizer: k steps forward, 1 step back," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] L. Ilya and H. Frank. "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations*, 2019.

[14] L. Balles and P. Hennig. "Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 404–413, 2018.

[15] J. Chen, D. Zhou, *et al.* "Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 3267–3275, 2020.

[16] G. Wang, S. Lu, *et al.* "SAdam: A Variant of Adam for Strongly Convex Functions," in *International Conference on Learning Representations*, 2020.

[17] Q. Ding, C. Hsieh, and J. Sharpnack. "An Efficient Algorithm For Generalized Linear Bandit: Online Stochastic Gradient Descent and Thompson Sampling," [Online]. Available: https://arxiv.org/abs/2006.04012, CORR, 2020.

[18] S. Farnood, L. E. Celis, and T. Patrick. "Stochastic Optimization with Bandit Sampling," [Online]. Available: https://arxiv.org/abs/1708.02544, CORR, 2017.

[19] D. Needell, R. Ward, and N. Srebro. "Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz algorithm," in *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[20] P. Richtárik and M. Takáč. "On optimal probabilities in stochastic coordinate descent methods," *Optimization Letters*, vol. 10, pp. 1233–1243, 2016.

[21] X. Peng, J. Zhang, *et al.* "Drill the Cork of Information Bottleneck by Inputting the Most Important Data," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1-13, 2021.

[22] Y. Fraboni, R. Vidal, *et al.* "Clustered Sampling: Low-Variance and Improved Representativity for Clients Selection in Federated Learning," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, pp. 3407–3416, 2021.

[23] P. Auer, N. Cesa-Bianchi, and P. Fischer. "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, 2002.

[24] Y. LeCun, L. Bottou, *et al.* "Gradient-based Learning Applied to Document Recognition," in *Proceedings of the IEEE*, 1998.

[25] A. Krizhevsky. "Learning Multiple Layers of Features from Tiny Images," [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html, 2009.

[26] M. P. Marcus, S. Beatrice, and M. A. Marcinkiewicz. "Treebank-2 LDC95T7," *Web Download. Philadelphia: Linguistic Data Consortium*, [Online]. Available: https://doi.org/10.35111/wf9p-g717, 1995.
[27] S. Karen and Z. Andrew. "Very deep convolutional networks for large-scale image recognition," https://arxiv.org/abs/1409.1556, CORR, 2014.
[28] K. He, X. Zhang, *et al*. "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
[29] G. Huang, Z. Liu, *et al*. "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708, 2017.
[30] N. Hongseok, S. Aman, *et al*. "Adaptive Sampling Probabilities for Non-Smooth Optimization," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 2574–2583, 2017.
[31] G. Hinton, N. Srivastava, and K. Swersky. Lecture 6d - a separate, adaptive learning rate for each connection. Slides of Lecture Neural Networks for Machine Learning, 2012.

Yangfan Zhou is currently pursuing the Ph.D. degree in the School of Nano-Tech and Nano-Bionics, University of Science and Technology of China. His current research interests are focused theoretical and algorithmic issues related to on large-scale optimization, stochastic optimization, convex online optimization, and their applications to deep learning, meta learning, and networking. He is currently a reviewer for many well-known journals, such as the IEEE Transactions on Neural Networks and Learning Systems.

Kaizhu Huang 's research interests include machine learning, neural information processing, and pattern recognition. He is a now a full professor at Institute of Applied Physical Sciences and Engineering, Duke Kunshan University (DKU). Before joining DKU, he was a full professor at Department of Intelligent Science, Xi'an Jiaotong-Liverpool University (XJTLU) and Associate Dean of Research in School of Advanced Technology, XJTLU. He founded Suzhou Municipal Key Laboratory of Cognitive Computation and Applied Technology. Prof. Huang obtained his PhD degree from Chinese University of Hong Kong (CUHK) in 2004. He worked in Fujitsu Research Centre, CUHK, University of Bristol, National Laboratory of Pattern Recognition, Chinese Academy of Sciences from 2004 to 2012. He was the recipient of 2011 Asia Pacific Neural Network Society Young Researcher Award. He received best paper or book award six times. He serves as associated editors/advisory board members in a number of journals and book series. He was invited as keynote speaker in more than 30 international conferences or workshops.

Cheng Cheng is currently an associate professor. He received the B.S. degree and M.S. degree in Computer Science and Technology from Guizhou University, Guiyang, China, in 2004 and 2009, respectively, and the Ph.D. degree in Information Engineering from Tokyo University of Agriculture and Technology (TUAT), Japan, in 2013. His current research interests focus on 3D vision, particularly on 3D feature learning, 3D modeling, 3D object recognition, and 3D shape measurement, etc.

Xuguang Wang received the Ph.D. degree from the Department of Electronic Engineering, University of Texas, Austin, USA, in 2005, with a master's degree from the Department of Electronic Engineering, Rice University, USA, in 2002, and a bachelor's degree from the Department of Materials, Tsinghua University, Beijing, in 1999. Dr. Wang has been engaged in the research of semiconductor storage technology for 10 years, and has undertaken the research of semiconductor memory in many scientific research institutions such as National Natural Science Foundation of the United States, MARCO, SRC, etc. Dr. Wang has published many important papers as the first author in the top two journals and conferences of the international semiconductor device category, all of which belong to SCI and have been cited more than 100 times.

Amir Hussain received his B.Eng (highest 1st Class Honours with distinction) and Ph.D degrees, from the University of Strathclyde, Glasgow, U.K., in 1992 and 1997, respectively. He is founding Director of the Centre of AI and Data Science at Edinburgh Napier University, UK. His research interests are cross-disciplinary and industry-led, aimed at developing cognitive data science and trustworthy AI technologies to engineer smart industrial and healthcare systems of tomorrow. He has (co)authored three international patents and around 500 publications, including over 200 journal papers and 20 Books/monographs. He has led major national and international projects, and supervised over 35 PhD students. He is the founding Chief Editor of Springer's Cognitive Computation journal and Springer Book Series on Socio-Affective Computing. He 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 (Elsevier), the IEEE Transactions on Systems, Man and Cybernetics: Systems, and the IEEE Transactions on Emerging Topics in Computational Intelligence. Amongst other distinguished roles, he is an elected Executive Committee member of the UK Computing Research Committee (national expert panel of the IET and the BCS for UK computing research), General Chair of IEEE WCCI 2020 (the world's largest IEEE technical event in computational intelligence, comprising IJCNN, IEEE CEC and FUZZ-IEEE), and Chair of the IEEE UK and Ireland Chapter of the IEEE Industry Applications Society.

Xin Liu (M'12) 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, Dr. Liu worked as Principal Investigator and Head of Intelligent Computing Chips Department with Institute of Microelectronics, A*STAR, Singapore. Dr. Liu joined the Suzhou Institute of Nano-Tech and Nano-Bionics (SINANO), Chinese Academy of Sciences as professor in 2018. His research interests include artificial intelligence signal processing algorithms, high-performance massively parallel processing chip architecture design, ultra-low power digital processor design, embedded non-volatile memory circuit design, etc.