Randomized Block-Coordinate Adaptive Algorithms for Nonconvex Optimization Problems

Yangfan Zhou[a,b], Kaizhu Huang[c], Jiang Li[b,d], Cheng Cheng[b,d], Xuguang Wang[b], Amir Hussian[e] and Xin Liu[a,b,*]

[a]School of Nano-Tech and Nano-Bionics, University of Science and Technology of China, Hefei, 230026, China
[b]Suzhou Institute of Nano-Tech and Nano-Bionics (SINANO), Chinese Academy of Sciences, Suzhou, 215123, China
[c]Data Science Research Center, Duke Kunshan University, No. 8 Duke Avenue, Kunshan, 215316, China
[d]Gusu Laboratory of Materials Science, 388 Ruoshui Road, Suzhou, Jiangsu Province, 215123, China
[e]School of Computing, Edinburgh Napier University, Edinburgh, EH11 4BN, UK

ARTICLE INFO

Keywords:
Adaptive Algorithm
Deep Model Training
Nonconvex Optimization
Randomized Block-Coordinate

ABSTRACT

Nonconvex optimization problems have always been one focus in deep learning, in which many fast adaptive algorithms based on momentum are applied. However, the full gradient computation of high-dimensional feature vector in the above tasks become prohibitive. To reduce the computation cost for optimizers on nonconvex optimization problems typically seen in deep learning, this work proposes a randomized block-coordinate adaptive optimization algorithm, named RAda, which randomly picks a block from the full coordinates of the parameter vector and then sparsely computes its gradient. We prove that RAda converges to a δ-accurate solution with the stochastic first-order complexity of $O(1/\delta^2)$, where δ is the upper bound of the gradient's square, under nonconvex cases. Experiments on public datasets including CIFAR-10, CIFAR-100, and Penn TreeBank, verify that RAda outperforms the other compared algorithms in terms of the computational cost.

1. Introduction

Deep learning has achieved great success in many domains including image recognition [1, 2], natural language processing [3, 4], constrained learning [5], and intelligent decision [6, 7]. On the other hand, deep neural networks are typically of high complexity with ultra-high-dimensional features [8, 9, 10]. To solve optimization problems in deep learning tasks is thus becoming increasingly difficult due to prohibitive computations as well as the unpopularity of expensive high-performance devices.

One popular fast optimization algorithms for deep training is Stochastic Gradient Descent (SGD). Note that "stochastic" means that at each iteration one or a mini-batch of samples are stochastically drawn from the sample set, and numerical evaluation shows that the batch size has some influence on the complexity of the algorithm [11]. SGD has simple steps that are easy to implement, and its generalization ability is excellent. However, SGD converges slowly in deep training tasks which restricts its further applications. For this reason, many faster convergence algorithms with the adaptive learning rate have been proposed, such as Adam [19], LightAdam [20], and AdaBelief [21]. Most adaptive algorithms are theoretically appealing due to the guaranteed convergence in cases of convex loss functions. Practically, these algorithms treat deep learning tasks as convex optimization problems for which guaranteed convergence bounds can be obtained [22, 23, 24]. However, deep learning tasks are typically not convex optimization problems. Therefore, nonconvex optimization algorithms appear more suitable for deep learning tasks though their convergence is more difficult to prove. To this end, some nonconvex algorithms for deep training have been proposed including Yogi [25] and PAGE [26].

Albeit their success of the previous fast optimization algorithms, these algorithms mainly consider the computational challenge due to the huge amount of samples. They seldom address the ultra-high-dimensionality issue, which is typically seen in deep learning problems. In case of high non-convexity, such issue may even be more serious. To further enable efficient training in deep learning models, randomized algorithms have drawn much attention

*Corresponding authors: xliu2018@sinano.ac.cn
ORCID(s): 0000-0001-5311-1482 (Y. Zhou); 0000-0002-3034-9639 (K. Huang); 0000-0003-4083-4731 (X. Liu)
[1]E-mail addresses: yfzhou2020@sinano.ac.cn, kaizhu.huang@dukekunshan.edu.cn, jli2018@sinano.ac.cn, ccheng2017@sinano.ac.cn, xgwang2009@sinano.ac.cn, A.Hussain@napier.ac.uk, xliu2018@sinano.ac.cn

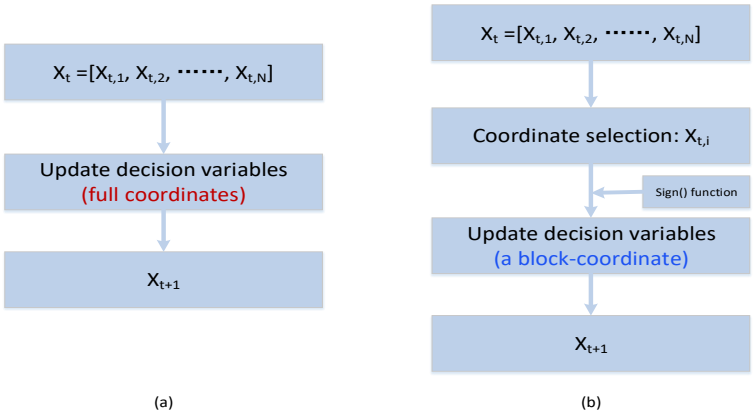

Figure 1: Comparison between RAda and the general adaptive optimization algorithms: (a) the general algorithms with adaptive learning rate; (b) The proposed algorithm, RAda.

recently. Concretely, an effective way to reduce the computational cost of high-dimensional vectors by performing Randomized Coordinate-Block (RBC) optimization has become one hot spot [14, 15]. By picking stochastically a block of coordinates to compute the updated information, RBC based algorithms have shown their capability in greatly reducing the cost of each calculation as opposed to their original optimization algorithms [16, 17, 18].

While existing RBC-based algorithms mainly consider convex optimization problems [27], in this work, we focus on designing optimization algorithms on non-convex problems in deep learning. We propose a novel RBC-based training algorithm called RAda with the adaptive learning rate. First, the proposed algorithm randomly picks a block-coordinate of the high dimensional parameter vector to compute its gradient per iteration; this can significantly lower the computational cost for the original cases where a gradient is calculated in the whole dimensions. Then, our algorithm exploits the momentum of the sparse gradient to achieve an adaptive learning rate, which can ensure fast convergence. Finally, the proposed algorithm engages the $sign(\cdot)$ function to control the variation of the first-order momentum and the second-order momentum within a reasonable range to prevent the variation from changing drastically, which can promote convergence even in nonconvex cases. Note that the proposed algorithm uses a similar approach for the adaptive learning rate as Yogi. However, the second-order momentum $\mathbf{v}_t$ of the proposed algorithm is based on its first-order momentum $\mathbf{m}_t$ as opposed to dependence on $\mathbf{g}_t$ in Yogi. As we shall see shortly, this approach adopted by RAda remains as advanced as Yogi in the nonconvex case. An illustration of this process can be seen in Figure 1.

To sum up, the main contributions of this paper can be three-fold:

- We develop a computationally efficient adaptive algorithm for nonconvex optimization problems. The proposed algorithm decreases its computational cost by the randomized block coordinate descent optimization method. Meanwhile, the proposed algorithm preserves the learning rate form of the previous adaptive optimization algorithms, so as to follow their performance.
- We provide the complexity analysis of the proposed algorithm. We prove that the proposed algorithm converges with the bound $O(1/T + 1/b)$, where T is the time horizon and b denotes the mini-batch size. Moreover, we analyze that RAda has the stochastic first-drder complexity $O(1/\delta^2)$ for achieving a δ-accurate solution under nonconvex conditions.
- We validate the excellent performance of the proposed algorithm in terms of the low computational cost on public datasets for nonconvex optimization problems.

The rest of this paper is structured as follows. Related work is introduced in Section 2. Section 3 provides the preliminaries for this work, including notations, problem setup, assumptions, and definitions. We present our algorithm design in Section 4. The convergence analysis of the proposed algorithm is shown in Section 5. Experimental details are shown in Section 6. Section 7 summarizes this work succinctly.

2. Related Work

Nonconvex optimization problem is always one of the challenges in the optimization field, in which a local optimum is typically found. Meanwhile, SGD optimization algorithm is one of the main methods for dealing with training tasks in deep learning. On the one hand, for the nonconvex optimization, Ghadimi and Lan [28] proposed an algorithm named the randomized stochastic gradient method that achieves a convergence rate of $O(1/\varepsilon^2)$, where ε is the gradient bound. Furthermore, Duchi *et al.* [29] utilized the adaptive regularization method to achieve an acceleration in terms of the dimension. On the other hand, many adaptive learning rate algorithms have been proposed for nonconvex optimization problems in recent years. For instance, Chen *et al.* [30] studied the convergence of Adam-type algorithms for nonconvex optimization. In this model, the convergence bound of order $O(\log T/\sqrt{T})$ (where T is the time horizon), is guaranteed with certain additional conditions. Moreover, Zaheer *et al.* [25] proposed an adaptive gradient algorithm, named Yogi, for nonconvex stochastic optimization problems, which converges to a δ-accurate solution with the stochastic first-order complexity of $O(1/\delta^2)$. Here δ is the bound of the squared gradient.

Recently, RBC has been successfully applied in gradient descent optimization methods to reduce the computational cost of computing gradients of high-dimensional vectors. Simon *et al.* [31] successfully combined RBC and the classical Frank-Wolfe algorithm for the dual structural support vector machine problem. Moreover, to reduce the expensive operations in huge-scale optimization problems, Nesterov [32] proposed the solution based on the RBC method. Furthermore, Zhou *et al.* [27] proposed a new variant of Adam based on RBC that obtains good performance on the computational cost. However, these RBC-based algorithms are investigated mainly under convex optimization assumptions. To extend the RBC optimization in nonconvex problems, Xie *et al.* [33] proposed an RBC ascent policy search algorithm for mean-variance optimization.

Although various RBC methods flourish in both convex and nonconvex optimization problems, there are rare studies to explore RBC in nonconvex optimization when the learning rate is adaptive. To fill this gap, we propose an RBC-based adaptive optimization algorithm under nonconvex conditions in this paper. Importantly, we prove that even with the adaptive learning rate, the proposed algorithm can converge under the more general nonconvex loss functions which are typically seen in deep neural networks.

3. Preliminaries

3.1. Notations

In this paper, the sets of all real numbers and the real Euclidean space with d dimensions are denoted by $\mathbb{R}$ and $\mathbb{R}^d$, respectively. All vectors are denoted in bold. $\mathbf{x}_t$ represents the vector $\mathbf{x}$ at time t, and $x_{t,i}$ denotes the i-th coordinate element of the vector $\mathbf{x}_t$. For vector operations, $\mathbf{x}^2$ and $\sqrt{\mathbf{x}}$ are used to represent the element-wise square and square root, respectively. Meanwhile, $\mathbf{x}/\mathbf{y}$ is used to represent the element-wise division of vectors $\mathbf{x}$ and $\mathbf{y}$. Throughout this paper, we utilize $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ to denote the standard Euclidean norm and the scalar inner product in Euclidean space, respectively.

3.2. Problem Setup

In this paper, we consider the nonconvex stochastic optimization problems formed as

$$\min_{\mathbf{x}_t\in\mathbb{R}^d} f(\mathbf{x}_t) := \mathbb{E}_{s_t\sim\mathbb{P}}[\ell(\mathbf{x}_t, s_t)], \tag{1}$$

where function $\ell(\cdot)$ is smooth (possibly nonconvex), $\mathbb{P}$ is a probability distribution on the domain $\mathcal{S}\subset\mathbb{R}^k$, and $\{\mathbf{x}_t\}$ are parameter vectors. The optimization problem, i.e, Equation (1), arise naturally in deep learning under nonconvex conditions where $\ell(\cdot)$ is the loss function, $\mathbf{x}_t$ is called the deep model parameter vector, and $\mathbb{P}$ is an unknown data distribution.

3.3. Assumptions and Definitions

In this section, we first state the assumptions used in this work.

Assumption 1 *Suppose that the loss function $\ell(\cdot)$ used in our algorithm is Lipschitz continuously differentiable with Lipschitz constant $L>0$, i.e., there exists a constant L satisfying that*

$$\|\nabla\ell(\mathbf{x},s)-\nabla\ell(\mathbf{y},s)\|\le L\|\mathbf{x}-\mathbf{y}\|, \forall\mathbf{x},\mathbf{y}\in\mathbb{R}^d. \tag{2}$$

Assumption 2 *Suppose that the gradient of the loss function $\ell(\cdot)$ is bounded, i.e., $\|\nabla\ell(\mathbf{x}_{t,i}, s)\| \leq G$ for all $\mathbf{x} \in \mathbb{R}^d$, $s \in \mathcal{S}$, and $i \in [d]$.*

Assumption 3 *Suppose that the gradient variance between the expected loss $f(\cdot)$ and the real loss $\ell(\cdot)$ is bounded, i.e.,*

$$\mathbb{E}\left\|\nabla\ell(\mathbf{x}_t, s) - \nabla f(\mathbf{x}_t)\right\|^2 \leq \sigma^2$$

for all $\mathbf{x}_t \in \mathbb{R}^d$.

It is noted that these assumptions are also typically used in the analysis of Stochastic First-Order (SFO) methods such as [27, 28, 25].

In the stochastic *convex* optimization problems, the convergence measure is the regret, i.e.,

$$R(T) = \sum_{t=1}^{T}\left(f(\mathbf{x}_t) - f(\mathbf{x}^*)\right),$$

where $\mathbf{x}^*$ is the optimal solution on domain $\mathbb{R}^d$. Regret is a popular paradigm in the field of stochastic convex optimization, such as [20, 19, 34, 35]. As mentioned above, our work focuses on the *nonconvex* condition that is the common setting for risk minimization in deep learning. However, the notion of regret does not apply to the stochastic *nonconvex* optimization setting. For this reason, we engage an applicable convergence measure for the *nonconvex* setting by the following definition.

Definition 1 *The algorithm's SFO complexity is defined as the gradient evaluation times of the objective function ℓ with respect to its first argument made by the algorithm.*

Definition 1 is a general framework for convergence analysis in nonconvex optimization problems that is common used in many previous works, such as [28, 25]. Following these previous work on nonconvex optimization problems, we also measure the "stationarity" of the iteration $\mathbf{x}$ by exploiting $\|\nabla f(\mathbf{x})\|^2 \leq \delta$. The solution of this case is called δ-accurate solution. In addition, we will prove that the proposed algorithm converges in nonconvex settings based on this definition in the following content.

4. Proposed Algorithm

In this section, we will elaborate the proposed algorithm where the pseudo code of RAda is listed in Algorithm 1. In the training process of deep learning, the parameter vector $\mathbf{x}_t$ presents all parameters including complex deep model nodes and high dimensional data features. Therefore, the gradient evaluations of the very high dimensional $\mathbf{x}_t$ make the training prohibitive in deep learning, especially when the computing resources are limited. To significantly reduce the gradient evaluation cost, we design a randomized gradient descent algorithm that only need calculate the gradient of a block of coordinates.

First, the proposed RAda algorithm randomly chooses a block i from the full coordinates of $\mathbf{x}_t$, where $i \in [d]$. Then, RAda evaluates the gradient for the i-th coordinate: $g_{t,i} = \nabla f(x_{t,i})$. Obviously, RAda's gradient estimation cost saves a lot of compuation compared to the full-coordinate gradient estimation which are often used in Adam, AMSGrad, and Yogi. Moreover, RAda follows the iteration form of the first order momentum of Adam, AMSGrad and Yogi, i.e., the step 3 in Algorithm 1. To adapt to the nonconvex optimization settings, we introduce the $sign(\cdot)$ function for the second order momentum: $v_{t,i} = v_{t-1,i} - (1-\beta_2)sign\left(v_{t-1,i} - m_{t,i}^2\right)m_{t,i}^2$, which follows Algorithm 2 in [25]. Finally, RAda updates the parameter vector for time $t+1$: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t\mathbf{m}_t/\sqrt{\mathbf{v}_t + \epsilon}$.

In terms of the update rule, $\mathbf{v}_t - \mathbf{v}_{t-1}$ is given as $-(1-\beta_2)sign\left(\mathbf{v}_{t-1} - \mathbf{m}_t^2\right)\mathbf{m}_t^2$ in our RAda, while it is $-(1-\beta_2)(\mathbf{v}_t - \mathbf{g}_t)^2$ in Adam. Therefore, the difference between $\mathbf{v}_t$ and $\mathbf{v}_{t-1}$ depends only on $\mathbf{m}_t^2$ in RAda as opposed to dependence on both $\mathbf{v}_{t-1}$ and $\mathbf{g}_t^2$ in Adam. For this reason, Adam can rapidly increase its learning rate when $\mathbf{v}_{t-1}$ is much larger than $\mathbf{g}_t^2$. Such strategy of increasing rapidly the learning rate can provide a fast convergence speed for the algorithm. However, once it converges to the local minimum or saddle point, the extremely small learning rate makes Adam unable to escape from the saddle point, partially leading to its poor generalization ability in some cases. Instead, RAda can increase its learning rate in a controlled fashion similar to Yogi, since $\mathbf{m}_t$ is based on the linear form of $\mathbf{g}_t$.

Intuitively, at each iteration, RAda updates the parameter vector $\mathbf{x}_t$ based on a block coordinate of the gradient, leading to a significant reduction of the computation cost. Therefore, the advantage of RAda would be more obvious especially in training ultra-high dimensional deep models. Theoretically,In the following, we provide the thorough convergence analysis for RAda showing that the block coordinate method can still converge. This makes our proposed RAda algorithm also appealing theoretically.

Algorithm 1 Pseudo Code of RAda

Input: $\mathbf{x}_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, decay parameters $0 \geq \beta_1, \beta_2 \geq 1$, $\epsilon > 0$.
Initially Set: $\mathbf{m}_0 = \mathbf{0}$, $\mathbf{v}_0 = \mathbf{0}$.
Output: $\mathbf{x}_{t+1}$

1: **for** $t = 1$ to T **do**
2: Choose randomly i via uniform distribution on $\{1, \dots, d\}$
3: Compute the gradient of the i-th coordinate:
$g_{t,i} = \nabla \ell(x_{t,i})$
4: Compute the first-order momentum of the i-th coordinate:
$m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}$
5: Compute the second-order momentum of the i-th coordinate:
$v_{t,i} = v_{t-1,i} - (1-\beta_2) sign\left(v_{t-1,i} - m_{t,i}^2\right) m_{t,i}^2$
6: Update the i-th coordinates of $\mathbf{m}_t$ and $\mathbf{v}_t$ respectively
7: Compute the parameter vector at time $t+1$:
$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{m}_t / \sqrt{\mathbf{v}_t + \epsilon}$
8: **end for**
9: **return** $\mathbf{x}_{t+1}$

5. Convergence Analysis

In this section, we prove that the SFO complexity of RAda has a guaranteed bound under nonconvex conditions. Obviously, the training process of deep learning is a typical time-slotted system. In each slot, the learner randomly selects a variable from its subset and updates that variable in the opposite direction of its gradient. For the sake of simplicity, let $q_i^{(t)}$ be randomly chosen from $\{0, 1\}$ and be i.i.d Bernoulli random variables with probability $\mathcal{P}(q_i^{(t)} = 1) = p$, where $i \in [d]$ and $t \in [T]$. Note that $q_i^{(t)}$ denotes the value of q_i at the t-th slot. Then, step 2 and 3 of RAda can be expressed mathematically as

$$g_{t,i} = q_i^{(t)} \nabla \ell(x_{t,i}) = Q^{(t)} \nabla \ell(x_{t,i}), \tag{3}$$

where $Q^{(t)} \in \{0,1\}^{d \times d}$ is a diagonal matrix with $Q_{ii}^{(t)} = q_i^{(t)}$. Moreover, we let $\mathcal{F}^t$ represent the history of RAda until time slot t, i.e.,

$$\mathcal{F}^{(t)} = \left\{Q^{(0)}, Q^{(1)}, \dots, Q^{(t-1)}\right\}. \tag{4}$$

To prove the convergence of RAda, we next present the following Lemmas 1-4 that are based on the proposed algorithm.

Lemma 1 *If function $f(\cdot)$ is Lipschitz continuous gradient with constant $L > 0$ for all $\mathbf{x}_{t+1}, \mathbf{x}_t \in \mathbb{R}^d$ and $t \in [T]$, we have*

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) + \left\langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle + \frac{L}{2} \left\| \mathbf{x}_{t+1} - \mathbf{x}_t \right\|^2. \tag{5}$$

The proof of Lemma 1 is presented in Appendix A.

Lemma 2 *If $g_{t,i}, v_{t,i}, \epsilon, \beta_2$ are generated by the proposed algorithm RAda, then we have the following inequality:*

$$\frac{|m_{t,i}|}{\sqrt{v_{t,i} + \epsilon} + \sqrt{v_{t-1,i} + \epsilon}} \leq \frac{1}{\sqrt{1-\beta_2}}. \tag{6}$$

The proof of Lemma 2 is provided in Appendix B.

Lemma 3 *If $v_{t,i}, v_{t-1,i}, \beta_2$ are generated by RAda, the we have that $v_{t,i} \geq \beta_2 v_{t-1,i}$.*

The proof of Lemma 3 is shown in Appendix C.

Lemma 4 *For the iterates $\{\mathbf{x}_t\}$ for all $t \in [T]$ in RAda that chooses a mini-batch with size b, the following inequality holds:*

$$\mathbb{E}\left[\|m_{t,i}\|^2\right] \leq \left[\nabla f(x_{t,i})\right]^2 + \frac{\sigma_i^2}{b}, \tag{7}$$

for all $i \in [d]$.

The proof of Lemma 4 can be found in Appendix D.

From the above Lemmas 1, 2, 3, 4, we can obtain the following Theorem 1 which provides a guaranteed convergence bound for RAda.

Theorem 1 *Suppose that Assumptions 1,2,3 are applied, let $\eta_t = \eta$ for all $t \in [T]$. The parameters $\eta, \beta_1, \beta_2, \epsilon$ are generated by RAda, which satisfy the following conditions: $1 - \beta_2 \leq \frac{\sigma^2}{16G^2}$ and $\eta \leq \frac{\sqrt{\epsilon\beta_2}}{2L}$. The mini-batch size chosen by RAda is b. Then we have the following bound*

$$\sum_{t=1}^{T} \mathbb{E}\left\|\nabla f(\mathbf{x}_t)\right\|^2 \leq O\left(\frac{f(\mathbf{x}_1) - f(\mathbf{x}^*)}{p\eta T} + \sigma^2\right). \tag{8}$$

Proof. *Indeed, RAda updates the parameter vector according to the gradient of a randomly picked coordinate, since that $g_{t,i} = \nabla \ell(x_{t,i})$ where the coordinate i is chosen randomly from $\{1, \ldots, d\}$ via uniform distribution. Therefore, the condition expectation should be taken in Equation (5), which is conditioned on $\mathcal{F}^{(t)}$. Then we take the condition expectation for each term in Equation (5) and have*

$$\mathbb{E}\left[f(\mathbf{x}_{t+1})|\mathcal{F}^{(t)}\right] \leq f(\mathbf{x}_t) + \mathbb{E}\left[\left\langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle |\mathcal{F}^{(t)}\right] + \mathbb{E}\left[\frac{L}{2}\left\|\mathbf{x}_{t+1} - \mathbf{x}_t\right\|^2 |\mathcal{F}^{(t)}\right]. \tag{9}$$

Since that $x_{t,i} \sim \mathcal{P}\left(x_{t,i} = q_i^{(t)} = 1\right)$, we attain

$$\begin{aligned}\mathbb{E}\left[f(x_{t+1,i})|\mathcal{F}^{(t)}\right] &\leq f(x_{t,i}) + \mathbb{E}\left[\left\langle \nabla f(x_{t,i}), x_{t+1,i} - x_{t,i} \right\rangle |\mathcal{F}^{(t)}\right] + \mathbb{E}\left[\frac{L}{2}\left\|x_{t+1,i} - x_{t,i}\right\|^2 |\mathcal{F}^{(t)}\right] \\ &= f(x_{t,i}) + \underbrace{p \cdot \left\langle \nabla f(x_{t,i}), x_{t+1,i} - x_{t,i} \right\rangle}_{M_1} + \underbrace{\frac{pL}{2} \cdot \left\|x_{t+1,i} - x_{t,i}\right\|^2}_{M_2}.\end{aligned} \tag{10}$$

Now, we consider the term M_1 in Equation (10):

$$\begin{aligned}M_1 &= p\nabla f(x_{t,i}) \cdot (x_{t+1,i} - x_{t,i}) \\ &= p\nabla f(x_{t,i}) \cdot \frac{-\eta_t m_{t,i}}{\sqrt{v_{t,i} + \epsilon}} \\ &= -p\eta_t \nabla f(x_{t,i}) \left[\frac{m_{t,i}}{\sqrt{v_{t,i} + \epsilon}} - \frac{m_{t,i}}{\sqrt{v_{t-1,i} + \epsilon}} + \frac{m_{t,i}}{\sqrt{v_{t-1,i} + \epsilon}}\right] \\ &= \frac{-p\eta_t \nabla f(x_{t,i}) m_{t,i}}{\sqrt{v_{t-1,i} + \epsilon}} - p\eta_t \nabla f(x_{t,i}) \underbrace{\left[\frac{m_{t,i}}{\sqrt{v_{t,i} + \epsilon}} - \frac{m_{t,i}}{\sqrt{v_{t-1,i} + \epsilon}}\right]}_{M_1'}.\end{aligned} \tag{11}$$

where the second equation follows from step 7 of RAda, the third equation is obtained by arranging the terms.

For clarity, we next consider the bound of term M_1' in Equation (11) separately.

$$M_1' \leq \left|m_{t,i}\right| \cdot \left|\frac{1}{\sqrt{v_{t,i} + \epsilon}} - \frac{1}{\sqrt{v_{t-1,i} + \epsilon}}\right|$$

$$
\begin{aligned}
&= \left|m_{t,i}\right| \cdot \left| \frac{\sqrt{v_{t-1,i}+\epsilon} - \sqrt{v_{t,i}+\epsilon}}{\sqrt{v_{t,i}+\epsilon} \cdot \sqrt{v_{t-1,i}+\epsilon}} \right| \\
&= \frac{\left|m_{t,i}\right|}{\sqrt{v_{t,i}+\epsilon} \cdot \sqrt{v_{t-1,i}+\epsilon}} \cdot \frac{\left|v_{t,i} - v_{t-1,i}\right|}{\sqrt{v_{t,i}+\epsilon} + \sqrt{v_{t-1,i}+\epsilon}} \\
&= \frac{\left|m_{t,i}\right|}{\sqrt{v_{t,i}+\epsilon} \cdot \sqrt{v_{t-1,i}+\epsilon}} \cdot \frac{(1-\beta_2) m_{t,i}^2}{\sqrt{v_{t,i}+\epsilon} + \sqrt{v_{t-1,i}+\epsilon}} \\
&\leq \frac{\sqrt{1-\beta_2} \cdot m_{t,i}^2}{\left(\sqrt{v_{t-1,i}+\epsilon}\right) \cdot \epsilon}.
\end{aligned} \tag{12}
$$

In the above inequality (12), the first inequality is attained by Cauchy–Schwarz inequality. The third equation follows from the 5-th step of the proposed algorithm which is

$$v_{t,i} = v_{t-1,i} - (1-\beta_2) sign\left(v_{t-1,i} - m_{t,i}^2\right) m_{t,i}^2.$$

Moreover, the last inequality of the inequality (12) is from Lemma 2 that is

$$\frac{\left|m_{t,i}\right|}{\sqrt{v_{t,i}+\epsilon} + \sqrt{v_{t-1,i}+\epsilon}} \leq \frac{1}{\sqrt{1-\beta_2}}.$$

Therefore, plugging inequality (12) into equation (11), we have the following bound of term M_1:

$$M_1 \leq \frac{-p\eta_t m_{t,i} \nabla f(x_{t,i})}{\sqrt{v_{t-1,i}+\epsilon}} - \frac{p\eta_t m_{t,i}^2 \nabla f(x_{t,i}) \cdot \sqrt{1-\beta_2}}{\left(\sqrt{v_{t-1,i}+\epsilon}\right) \cdot \epsilon}. \tag{13}$$

Going back to inequality (10) and considering the bound of term M_2, we obtain the following

$$
\begin{aligned}
M_2 &= \frac{pL}{2} \cdot \left\| x_{t+1,i} - x_{t,i} \right\|^2 \\
&= \frac{pL}{2} \cdot \left\| - \frac{\eta_t m_{t,i}}{\sqrt{v_{t,i}+\epsilon}} \right\|^2 \\
&\leq \frac{pL\eta_t^2}{2\sqrt{\beta_2 v_{t-1,i}+\epsilon}} \cdot \frac{m_{t,i}^2}{\sqrt{\beta_2 v_{t-1,i}+\epsilon}} \\
&\leq \frac{pL\eta_t^2}{2\sqrt{\beta_2 v_{t-1,i}+\epsilon}} \cdot \frac{m_{t,i}^2}{\sqrt{\beta_2 (v_{t-1,i}+\epsilon)}} \\
&\leq \frac{pL\eta_t^2}{2\sqrt{\epsilon\beta_2}} \cdot \frac{m_{t,i}^2}{\sqrt{v_{t-1,i}+\epsilon}},
\end{aligned} \tag{14}
$$

where the first equation is from the 7-th step of Algorithm 1, the first inequality follows from Lemma 3 which is $v_{t,i} \geq \beta_2 v_{t-1,i}$, and the second inequality obtained from the fact that $0 < \beta_2 < 1$.

Substituting inequalities (13) and (14) into inequality (10), we further attain the following bound

$$
\begin{aligned}
&\mathbb{E}\left[f(x_{t+1,i}) | \mathcal{F}^{(t)}\right] \\
&\leq f(x_{t,i}) - \frac{p\eta_t m_{t,i} \nabla f(x_{t,i})}{\sqrt{v_{t-1,i}+\epsilon}} - \frac{p\eta_t m_{t,i}^2 \nabla f(x_{t,i}) \cdot \sqrt{1-\beta_2}}{\left(\sqrt{v_{t-1,i}+\epsilon}\right) \cdot \epsilon} + \frac{pL\eta_t^2}{2\sqrt{\epsilon\beta_2}} \cdot \frac{m_{t,i}^2}{\sqrt{v_{t-1,i}+\epsilon}} \\
&\leq f(x_{t,i}) - \left(p\eta_t - \frac{p\eta_t G\sqrt{1-\beta_2}}{\epsilon} - \frac{pL\eta_t^2}{2\sqrt{\epsilon\beta_2}}\right) \frac{[\nabla f(x_{t,i})]^2}{\sqrt{v_{t-1,i}+\epsilon}} + \left(\frac{p\eta_t G^2(1-\beta_2)}{2\epsilon} - \frac{pL\eta_t^2}{2\sqrt{\epsilon\beta_2}}\right) \frac{\sigma_i^2}{b\sqrt{v_{t-1,i}+\epsilon}},
\end{aligned} \tag{15}
$$

where the first inequality is attained by the fact that $|\nabla f(x_{t,i})| \leq G$*, and the second inequality follows from Lemma 4.*

Going further, from the conditions in Theorem 1, we have the following bounds:

$$\frac{G\sqrt{1-\beta_2}}{\epsilon} \leq \frac{1}{4}, \qquad \frac{L\eta_t}{2\sqrt{\epsilon\beta_2}} \leq \frac{1}{4}. \tag{16}$$

Applying inequality (16) into inequality (15), we obtain a new bound in the following manner:

$$\begin{aligned}
&\mathbb{E}\left[f(x_{t+1,i})|\mathcal{F}^{(t)}\right] \\
&\leq f(x_{t,i}) - \left(p\eta_t - \frac{p\eta_t}{4} - \frac{p\eta_t}{4}\right)\frac{[\nabla f(x_{t,i})]^2}{\sqrt{v_{t-1,i}+\epsilon}} + \left(\frac{p\eta_t G\sqrt{1-\beta_2}}{8} - \frac{p\eta_t}{4}\right)\frac{\sigma_i^2}{b\sqrt{v_{t-1,i}+\epsilon}} \\
&\leq f(x_{t,i}) - \frac{p\eta_t}{2}\frac{[\nabla f(x_{t,i})]^2}{\sqrt{v_{t-1,i}+\epsilon}} + \left(\frac{p\eta_t G\sqrt{1-\beta_2}}{8} - \frac{p\eta_t}{4}\right)\frac{\sigma_i^2}{b\sqrt{v_{t-1,i}+\epsilon}} \\
&\leq f(x_{t,i}) - \frac{p\eta\cdot[\nabla f(x_{t,i})]^2}{2\left(\sqrt{2}G+\epsilon\right)} + \frac{p\eta G\sqrt{1-\beta_2} - 2p\eta}{8\sqrt{\epsilon}}\cdot\frac{\sigma^2}{b},
\end{aligned} \tag{17}$$

where the last inequality is from the fact that $0 \leq v_{t-1,i} \leq 2G^2$*.*

Then applying telescoping sum for inequality (17), we obtain the following bound:

$$\frac{p\eta}{2\left(\sqrt{2}G+\epsilon\right)}\sum_{t=1}^{T}\mathbb{E}\left\|\nabla f(x_{t,i})\right\|^2 \leq f(x_{1,i}) - \mathbb{E}[f(x_{T+1,i})] + \frac{p\eta G\sqrt{1-\beta_2} - 2p\eta}{8\sqrt{\epsilon}}\cdot\frac{\sigma^2 T}{b}. \tag{18}$$

Furthermore, from inequality (18), we further attain that

$$\begin{aligned}
&\sum_{t=1}^{T}\sum_{i=1}^{d}\mathbb{E}\left\|\nabla f(x_{t,i})\right\|^2 \\
&\leq \frac{2\left(\sqrt{2}G+\epsilon\right)}{p\eta}\sum_{i=1}^{d}\left(f(x_{1,i}) - \mathbb{E}[f(x_{T+1,i})]\right) + \frac{2\left(\sqrt{2}G+\epsilon\right)}{p\eta}\frac{p\eta G\sqrt{1-\beta_2} - 2p\eta}{8\sqrt{\epsilon}}\cdot\frac{d\sigma^2 T}{b}.
\end{aligned} \tag{19}$$

Moreover, by the fact that $f(x_i^*) \leq f(x_{t+1,i})$*, we have the following bound:*

$$\sum_{t=1}^{T}\mathbb{E}\left\|\nabla f(\mathbf{x}_t)\right\|^2 \leq \frac{2d\left(\sqrt{2}G+\epsilon\right)}{p\eta}\left(f(\mathbf{x}_1) - f(\mathbf{x}^*)\right) + \frac{\left(\sqrt{2}G+\epsilon\right)\left(G\sqrt{1-\beta_2} - 2p\eta\right)}{4\sqrt{\epsilon}}\cdot\frac{d\sigma^2 T}{b}. \tag{20}$$

Therefore, from inequality (20), we obtain the desired result as follows:

$$\sum_{t=1}^{T}\mathbb{E}\left\|\nabla f(\mathbf{x}_t)\right\|^2 \leq O\left(\frac{f(\mathbf{x}_1) - f(\mathbf{x}^*)}{p\eta T} + \sigma^2\right). \tag{21}$$

The proof of Theorem 1 is completed. ■

As we can see, the proposed algorithm obtains a guaranteed bound that verifies its convergence on nonconvex conditions. Moreover, reviewing inequality (20), the proposed algorithm generates $\mathbf{x}_t$ with constant η, and also has the following bound:

$$\sum_{t=1}^{T}\mathbb{E}\left\|\nabla f(\mathbf{x}_t)\right\|^2 \leq O\left(\frac{1}{T} + \frac{1}{b}\right).$$

Going further, from Definition 1, our proposed algorithm has the SFO complexity of $O(1/\delta^2)$ for achieving a δ-accurate solution under nonconvex cases.

Moreover, from the result in inequality (20), the sampling probability p of block-coordinate has a negative effect on the proposed algorithm's convergence. Indeed, RAda sacrifices some algorithm complexity for a very lightweight computation cost per round. The result in Theorem 1 provides a guaranteed convergence bound. In the following section, we will verify the great advantages of RAda in computing costs through the experiments conducted on public datasets, and report the result curves on running time.

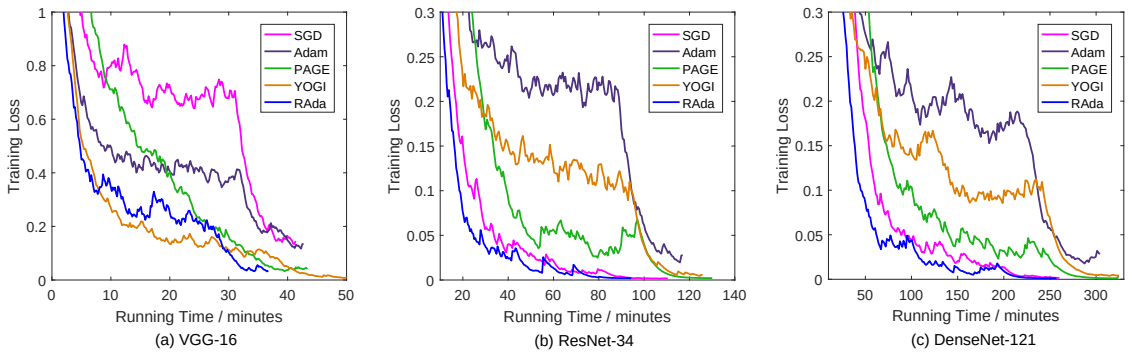

Figure 2: Curve results of training loss *vs.* running time, on CIFAR-10 dataset. RAda has the least running time.

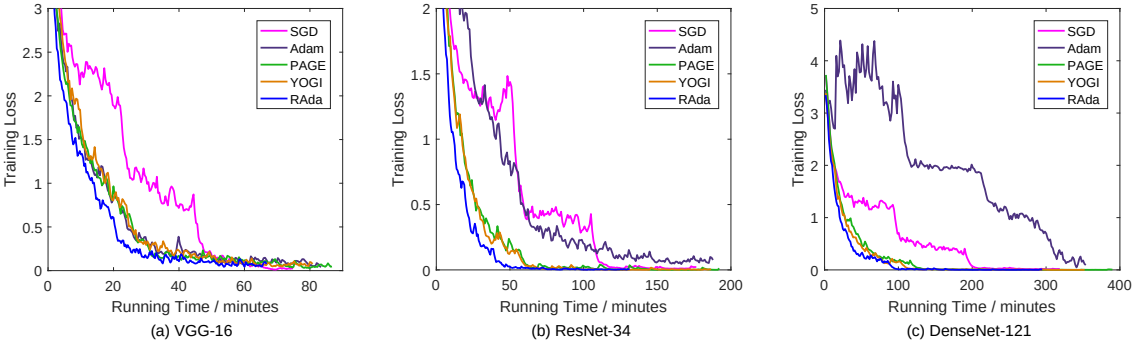

Figure 3: Curve results of training loss *vs.* running time, on CIFAR-100 dataset. RAda has the least running time.

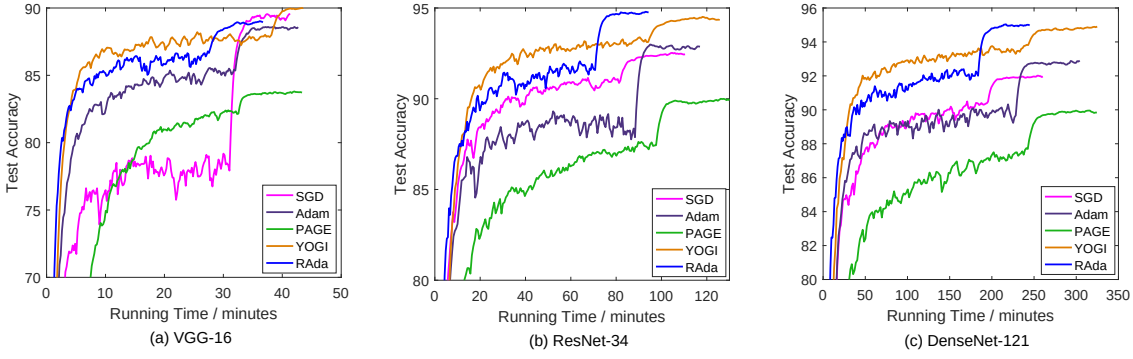

Figure 4: Curve results of test accuracy *vs.* running time, on CIFAR-10 dataset. RAda is the fastest to achieve the desired performance.

6. Experiments

From our theoretical analysis, we gain the insight that the proposed algorithm uses very less computation cost than the other adaptive algorithms per iteration. To verify this insight, we conduct two experiments on image classification and language processing tasks. For the image classification task, we present empirical results of running time *vs.* training loss on public datasets. In addition, we also present experimental results on test accuracy to examine the generalization ability of RAda. For the language processing task, we present empirical results on perplexity *vs.* running time. In this experiment, three LSTM models with different layers are used as the RNN framework on Penn TreeBank dataset. All the experiments are conducted on a machine with one NVIDIA GeForce RTX 3080 GPU and Inter Core i9-10900x CPU.

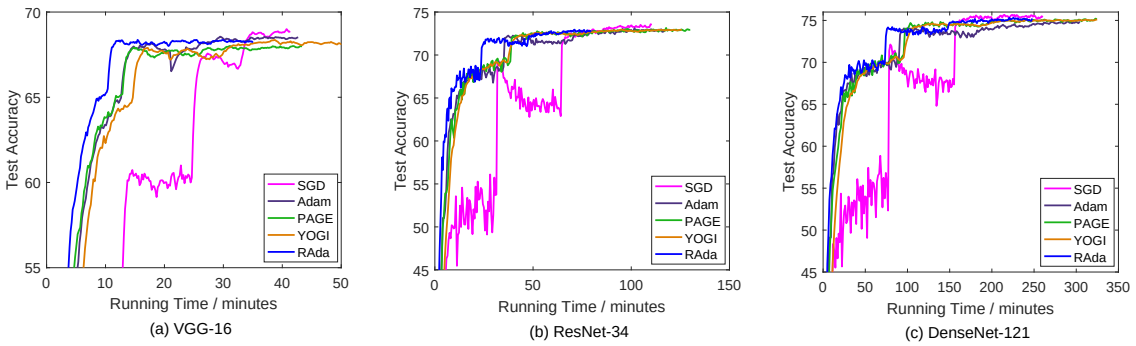

Figure 5: Curve results of test accuracy *vs.* running time, on CIFAR-100 dataset. RAda is the fastest to achieve the desired performance.

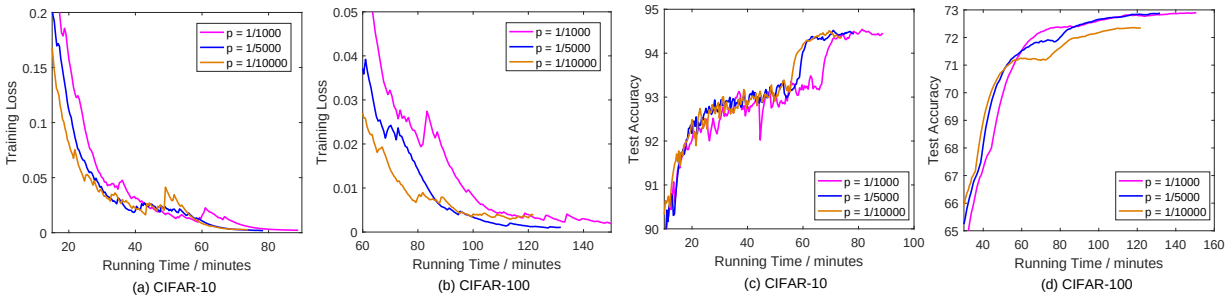

Figure 6: Curve results with different block-coordinate selection probability of RAda in ResNet-34. (a) shows the result of training loss vs. running time on CIFAR-10; (b) shows the result of training loss vs. running time on CIFAR-100; (c) shows the result of test accuracy vs. running time on CIFAR-10; (d) shows the result of test accuracy vs. running time on CIFAR-100.

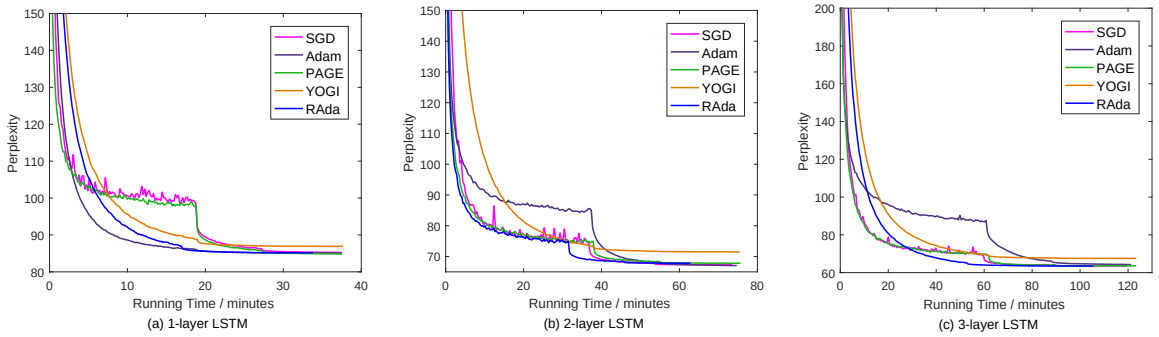

Figure 7: Curve results of RAda on language modeling. (a) shows the result of perplexity vs. running time on 1-layer LSTM model; (b) shows the result of perplexity vs. running time on 2-layer LSTM model; (c) shows the result of perplexity vs. running time on 3-layer LSTM model.

6.1. Experimental Settings

6.1.1. Datasets and Models.

Image Classification Task. Two benchmark datasets are used as baselines in our experiments. One is CIFAR-10, which consists of 60,000 color images of 32×32 resolution, with a total of 10 categories. Moreover, each category contains 6,000 images. The dataset is divided into two subsets, a training set of 50,000 images and a test set of 10,000 images. The other is CIFAR-100 which has 100 categories, which consists of 60,000 color images of 32×32 resolution. We run our experiments in three classical deep models: VGG-16 [36], ResNet-34 [37], and DenseNet-121 [38], which are widely used deep convolutional neural networks.

Language Processing Task. In this part, the famous dataset, Penn TreeBank, is used as baselines. This dataset is formed by annotating esach word with a POS tag. In addition, its training set contains 38,219 sentences and 912,344 tokens, validation set has 5,527 sentences and 131,768 tokens, and test set consists of 5,462 sentences and 129,654 tokens. Moreover, we adopt 1-layer LSTM network, 2-layer LSTM network, and 3-layer LSTM network as the RNN model in this experiment, respectively.

6.1.2. Algorithms

We use the following popular optimization algorithms as our comparison methods: SGD, Adam [19], PAGE [26], YOGI [25]. For SGD, the version with the momentum is used, in which momentum is 0.9 and weight decay coefficient is $5e-4$. The other executed algorithms including the proposed algorithm are all equipped with the adaptive learning rate. The learning rate of YOGI is set to $1e-2$, and the learning rate of Adam is set to $1e-3$. The coefficients of momentums are $\beta_1 = 0.9, \beta_2 = 0.999$. Moreover, Yogi and Rada both have the parameter ϵ of $1e-3$. The parameter ϵ in Adam and PAGE is $1e-8$. The mini-batch size is 128 in our experiments. In addition, $\mathbf{m}_t$ and $\mathbf{v}_t$ are initialed to $\mathbf{0}$.

6.2. Experimental Results and Analysis

6.2.1. Image Classification Task

First, we focus on investigating whether RAda outperforms the compared algorithms on computation cost, which is the main concern of this paper. To visually observe such comparison of each algorithm, we plot the curves of the training loss on running time. We run the algorithms on the image classification task of CIFAR-10, and report the results in Figure 2. We also implement the algorithms for image classification on CIFAR-100, and plot the curves in Figure 3. The results shown in these two figures verify that RAda takes the least running time to reach its desired training loss on VGG-16, ResNet-34, and DenseNet-121. Moreover, we obverse that RAda converges even quicker than the first-order algorithm SGD in running time. Though SGD is a simple first-order algorithm, it still computes the full coordinate gradient each iteration, which consumes a large computation time. These experimental results show that the proposed algorithm RAda may have a huge advantage in computation cost, compared to the other adaptive algorithms and even SGD in some cases.

Second, we conduct experiments to verify whether RAda can perform satisfactory in terms of test accuracy. As seen from the design of RAda, although the gradient calculation cost of each iteration is greatly reduced, some gradient information may also be lost. Therefore, a consequent question arises, i.e. whether RAda will come at the cost of a significant loss of accuracy. For this reason, we report the test accuracy results of all executed algorithms. The results of image classification task on CIFAR-10 and CIFAR-100 are respectively shown in Figure 4, and Figure 5. From the two figures, we observe that the test accuracy of the proposed algorithm can reach the best level in most cases, and even surpass the other comparison algorithms in some cases. Although the gradient information of RAda at each iteration is more sparse, since the computational cost of each iteration is very low, the accuracy loss can be compensated by the additional number of iterations without taking up too much running time.

Finally, we examine how the selection probability p of block-coordinates would affect the performance. According to the original design of the deep models, the hyper-parameter number in VGG-16 is 1,383,575,544, the hyper-parameter number of ResNet-34 is 25,636,712, and that of DenseNet-121 is 8,062,504. In all the above experiments, we set $p = 1/5000$ for RAda. In this group of experiments, we choose the value for p from $\{1/1000, 1/5000, 1/10000\}$ for RAda, and run the algorithm to train ResNet-34 on CIFAR-10. The experiment result is shown in (a) and (b) of Figure 6. Overall, we believe that a probability of $p = 1/5000$ is what we expect. Namely, it not only takes less running time, but also brings the loss to a smaller value. Moreover, the results of test accuracy are shown in (c) and (d) of Figure 6 where the curve of $p = 1/5000$ also works well for both running time and test accuracy. Nevertheless, it would be both interesting and important to analyze how the probability p would affect the performance of the algorithm theoretically. We will leave this exploration as our future work.

6.2.2. Language Processing Task

Language processing is another important task in the field of deep learning. For this reason, we present experiments to further verify the effectiveness of the proposed algorithm on this task. Moreover, we use LSTM network, one of the most commonly used RNN models, to complete the language modeling. In our experiments, 1-layer with 5,293,200 parameters, 2-layer with 13,632,400 parameters, and 3-layer with 24,221,600 parameters LSTM are applied, respectively. Finally, we report the results of perplexity *vs.* running time in Figure 7. Note that the lower of the perplexity, the better the performance of the algorithm. Straightly, the curves in this figure show that the

proposed algorithm reaches a stable low-perplexity state fastest, which means that the proposed algorithm has lower computational cost and shorter running time per iteration than the other compared algorithms, and thus can complete the learning task in a short total running time.

7. Conclusion and Future Work

In this paper, we explore the rationale for the randomized block-coordinate optimization of adaptive training algorithms in nonconvex cases. Specifically, we propose a RBC based adaptive algorithm and prove that it converges when its loss function is nonconvex. Moreover, we provide the empirical evidence to support our theoretical analysis. The proposed algorithm takes very less computation cost than the other algorithms per iteration, and consumes additional iterations to obtain a good performance on test accuracy. Experimental results on image classification task (using CIFAR-10 and CIFAR-100 datasets) and language processing task (using Penn TreeBank dataset) demonstrate the performance of our algorithm.

Although we only validate the low computational cost advantage of our RAda on image classification and NLP tasks, our algorithm could also work similarly on other deep learning tasks, such as GAN and transfer learning tasks, as justified by the theoretical analysis of our algorithm. Additionally, we would prefer to leave the exploration of such tasks as future work. Moreover, another open problem is the optimal sample size for our proposed algorithm, which could inspired from [10, 39]. We also leave this work for the future.

Acknowledgment

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 the Lemma 1

Lemma 1. If function $f(\cdot)$ is Lipschitz continuous gradient with constant $L > 0$ for all $\mathbf{x}_{t+1}, \mathbf{x}_t \in \mathbb{R}^d$ and $t \in [T]$, we have

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) + \left\langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle + \frac{L}{2} \left\| \mathbf{x}_{t+1} - \mathbf{x}_t \right\|^2. \tag{22}$$

Proof. *For all $\mathbf{x}_{t+1}, \mathbf{x}_t \in \mathbb{R}^d$, we have*

$$\begin{aligned} f(\mathbf{x}_{t+1}) &= f(\mathbf{x}_t) + \int_0^1 \left\langle f'\left(\mathbf{x}_t + \omega(\mathbf{x}_{t+1} - \mathbf{x}_t)\right), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle d\omega \\ &= f(\mathbf{x}_t) + \left\langle f'(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle + \int_0^1 \left\langle f'\left(\mathbf{x}_t + \omega(\mathbf{x}_{t+1} - \mathbf{x}_t)\right) - f'(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle d\omega. \end{aligned} \tag{23}$$

Rearranging equation (23), we obtain

$$\begin{aligned} &\left| f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) - \left\langle f'(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle \right| \\ &= \left| \int_0^1 \left\langle f'\left(\mathbf{x}_t + \omega(\mathbf{x}_{t+1} - \mathbf{x}_t)\right) - f'(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle d\omega \right| \\ &\leq \int_0^1 \left| \left\langle f'\left(\mathbf{x}_t + \omega(\mathbf{x}_{t+1} - \mathbf{x}_t)\right) - f'(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \right\rangle \right| d\omega \\ &\leq \int_0^1 \left\| f'\left(\mathbf{x}_t + \omega(\mathbf{x}_{t+1} - \mathbf{x}_t)\right) - f'(\mathbf{x}_t) \right\| \cdot \left\| \mathbf{x}_{t+1} - \mathbf{x}_t \right\| d\omega \end{aligned}$$

$$\leq \int_0^1 \omega L \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 d\omega$$
$$= \frac{L}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2, \tag{24}$$

where the first inequality follows from Triangle inequality, the second inequality is attained from Cauchy-Schwarz inequality, i.e., $\langle x, y\rangle \leq \|x\| \cdot \|y\|$ *for all* $x, y \in \mathbb{R}$*, and the last inequation is yielded from equation* (2)*. Therefore, the proof of Lemma 1 is completed.* ■

B. Proof of the Lemma 2

Lemma 2. If $g_{t,i}, v_{t,i}, \epsilon, \beta_2$ are generated by the proposed algorithm RAda, then we have the following inequality:

$$\frac{|m_{t,i}|}{\sqrt{v_{t,i}+\epsilon} + \sqrt{v_{t-1,i}+\epsilon}} \leq \frac{1}{\sqrt{1-\beta_2}}. \tag{25}$$

Proof. *To prove the desired result, we consider this issue in the following two aspects. On one hand, if* $v_{t-1,i} \geq m_{t,i}^2$*, thus* $sign\big(v_{t-1,i} - m_{t,i}^2\big) = 1$*. Moreover, by the 5-th step of RAda, we attain that*

$$\begin{aligned}
&\frac{|m_{t,i}|}{\sqrt{v_{t,i}+\epsilon} + \sqrt{v_{t-1,i}+\epsilon}} \\
&\leq \frac{|m_{t,i}|}{\sqrt{v_{t-1,i} - (1-\beta_2)m_{t,i}^2 + \epsilon} + \sqrt{v_{t-1,i}+\epsilon}} \\
&\leq \frac{|m_{t,i}|}{\sqrt{m_{t,i}^2 - (1-\beta_2)m_{t,i}^2} + \sqrt{v_{t-1,i}}} \\
&\leq \frac{1}{\sqrt{1-\beta_2}}.
\end{aligned} \tag{26}$$

On the other hand, if $v_{t-1,i} < m_{t,i}^2$*, thus* $sign\big(v_{t-1,i} - m_{t,i}^2\big) = -1$*, and we attain that*

$$\begin{aligned}
&\frac{|m_{t,i}|}{\sqrt{v_{t,i}+\epsilon} + \sqrt{v_{t-1,i}+\epsilon}} \\
&\leq \frac{|m_{t,i}|}{\sqrt{v_{t-1,i} + (1-\beta_2)m_{t,i}^2 + \epsilon}} \\
&\leq \frac{|m_{t,i}|}{\sqrt{(1-\beta_2)m_{t,i}^2}} \\
&\leq \frac{1}{\sqrt{1-\beta_2}}.
\end{aligned} \tag{27}$$

Therefore, the proof of Lemma 2 is completed. ■

C. Proof of the Lemma 3

Lemma 3. If $v_{t,i}, v_{t-1,i}, \beta_2$ are generated by RAda, the we have that $v_{t,i} \geq \beta_2 v_{t-1,i}$.

Proof. *Firstly, if* $v_{t-1,i} \leq m_{t,i}^2$*, and from the step 5 of Algorithm 1, we have*

$$v_{t,i} = v_{t-1,i} + (1-\beta_2)m_{t,i}^2$$

$$= \beta_2 v_{t-1,i} + (1-\beta_2)(v_{t-1,i} + m_{t,i}^2)$$
$$\geq \beta_2 v_{t-1,i}. \tag{28}$$

Secondly, if $v_{t-1,i} > m_{t,i}^2$, then we have the following bound by the step 5 in RAda:

$$v_{t,i} = v_{t-1,i} - (1-\beta_2)m_{t,i}^2$$
$$\geq v_{t-1,i} - (1-\beta_2)v_{t-1,i}$$
$$\geq \beta_2 v_{t-1,i}. \tag{29}$$

Combining inequations (28) and (29), we obtain the desired result. Therefore, the proof of Lemma 3 is completed. ■

D. Proof of the Lemma 4

Lemma 4. For the iterates $\{\mathbf{x}_t\}$ for all $t \in [T]$ in RAda that chooses a mini-batch with size b, the following inequality holds:

$$\mathbb{E}\big[\|m_{t,i}\|^2\big] \leq \big[\nabla f(x_{t,i})\big]^2 + \frac{\sigma_i^2}{b}, \tag{30}$$

for all $i \in [d]$.

Proof. *We first define the following notation:*

$$\zeta_t = \frac{1}{|S_t|}\sum_{s\in S_t}\big(\nabla \ell(x_{t,i}, s) - \nabla f(x_{t,i})\big), \tag{31}$$

where S_t is the total number of samples, and s is the sample of time t.

From equality (31), we attain that

$$\zeta_t + \nabla f(x_{t,i}) = \frac{1}{|S_t|}\sum_{s\in S_t}\Big[\nabla \ell(x_{t,i}, s) - \nabla f(x_{t,i})\Big] + \nabla f(x_{t,i})$$
$$= g_{t,i}. \tag{32}$$

We take the expectation for equality (32) and obtain

$$\mathbb{E}\big[g_{t,i}^2\big] = \mathbb{E}\big[\|\zeta_t + \nabla f(x_{t,i})\|^2\big] = \mathbb{E}\big[\zeta_t^2\big] + \big[\nabla f(x_{t,i})\big]^2$$
$$= \frac{1}{b^2}\mathbb{E}\Big[\Big(\sum_{s\in S_t}\big(\nabla \ell(x_{t,i}, s) - \nabla f(x_{t,i})\big)\Big)^2\Big] + \big[\nabla f(x_{t,i})\big]^2$$
$$= \frac{1}{b^2}\mathbb{E}\Big[\sum_{s\in S_t}\big(\nabla \ell(x_{t,i}, s) - \nabla f(x_{t,i})\big)^2\Big] + \big[\nabla f(x_{t,i})\big]^2$$
$$\leq \frac{\sigma_i^2}{b} + \big[\nabla f(x_{t,i})\big]^2, \tag{33}$$

where the second equality follows from the fact that ζ_t is a mean zero random variable, the third equality uses the fact that $\mathbb{E}[\|\zeta_1 + \ldots + \zeta_k\|^2] = \mathbb{E}[\|\zeta_1\|^2 + \ldots + \|\zeta_k\|^2]$, the inequality follows from Assumption 3.

Moreover, from the 4-th step in RAda, we obtain the following inequality:

$$m_{t,i} = \beta_1^t m_{0,i} + (1-\beta_1)(g_{1,i} + g_{2,i} + \ldots + g_{t,i}). \tag{34}$$

From inequality (34), we further attain that

$$|m_{t,i}| \leq (1-\beta_1)|g_{1,i} + g_{2,i} + \ldots + g_{t,i}| \leq |g_{t,i}|, \tag{35}$$

where the second inequality is due to the fact that RAda selects only one block-coordinate of the gradient per round.

Combining inequalities (33) and (35), we have the following bound:

$$\mathbb{E}\big[m_{t,i}^2\big] \leq \frac{\sigma_i^2}{b} + \big[\nabla f(x_{t,i})\big]^2. \tag{36}$$

Therefore, the proof of Lemma 4 is completed. ■

References

[1] Dong,Y., Liu, Q., Du, B., and Zhang, L. "Weighted Feature Fusion of Convolutional Neural Network and Graph Attention Network for Hyperspectral Image Classification," *IEEE Transactions on Image Processing*, 31: 1559-1572, 2022.

[2] Huang, Y. and Chen, H. H. "Deep face recognition for dim images," *Pattern Recognition*, 126: 108580, 2022.

[3] Perikos, I. and Hatzilygeroudis, I. "Recognizing emotions in text using ensemble of classifiers," *Engineering Applications of Artificial Intelligence*, 51: 191-201, 2016.

[4] Aberdam, A., Litman, R., Tsiper, S., et al. "Sequence-to-Sequence Contrastive Learning for Text Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 15302-15312, 2022.

[5] Luiz F. O. C., Santiago P., Miguel C., Alejandro R., "Constrained Learning with Non-Convex Losses," https://arxiv.org/abs/2103.05134, 2021.

[6] Roy, A., Banerjee, B., et al. "Discriminative Dictionary Design for Action Classification in Still Images and Videos," *Cognitive Computation*, 13: 698-708, 2021.

[7] Li, B., Xu, Z., et al. "A Bibliometric Study and Science Mapping Research of Intelligent Decision," *Cognitive Computation*, 14(3): 989-1008, 2022.

[8] Jang, K., Kang, J., et al. "Ultra-High Dimensional Sparse Representations with Binarization for Efficient Text Retrieval," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1016-1029, 2021.

[9] Zhao, Y., Kang, J., and Long, Q. "Bayesian Multiresolution Variable Selection for Ultra-High Dimensional Neuroimaging Data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15(2): 537-550, 2018.

[10] Arous, G., B., Gheissari, R., and Jagannath, A. "Online stochastic gradient descent on non-convex losses from high-dimensional inference," *Journal of Machine Learning Research*, 22(106:1-106:51), 2021.

[11] Iiduka, H. "Minimization of Stochastic First-order Oracle Complexity of Adaptive Methods for Nonconvex Optimization," https://arxiv.org/abs/2112.07163, 2021.

[12] Zhou, Y., Huang, K., et al. "FastAdaBelief: Improving Convergence Rate for Belief-based Adaptive Optimizer by Strong Convexity," *IEEE Transactions on Neural Networks and Learning Systems*, https://ieeexplore.ieee.org/document/9732534, pp. 1-15, March 2022.

[13] Chen, H., Gao, J., Zhang, W., and Yang, P. "Seismic Acoustic Impedance Inversion via Optimization-Inspired Semisupervised Deep Learning," *IEEE Transactions on Geoscience and Remote Sensing*, 60: 1-11, 2022.

[14] Diakonikolas, J. and Orecchia, L. "Alternating Randomized Block Coordinate Descent," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 1232-1240, 2018.

[15] Zhao, T., Yu, M., et al. "Accelerated Mini-batch Randomized Block Coordinate Descent Method," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pp. 3329-3337, 2014.

[16] Du, K., Ruan, C., and Sun, X. "On the convergence of a randomized block coordinate descent algorithm for a matrix least squares problem," *Applied Mathematics Letters*, 124: 107689, 2022.

[17] Kaushik, H. and Yousefian, F. "A Randomized Block Coordinate Iterative Regularized Subgradient Method for High-dimensional Ill-posed Convex Optimization," in *2019 American Control Conference*, pp. 3420-3425, 2019.

[18] Wang, Q., Cui, Y., et al. "Optimization-based Block Coordinate Gradient Coding," in *IEEE Global Communications Conference*, pp. 1-6, December 2021.

[19] Kingma, D. P., Ba, J.L. Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations, https://arxiv.org/abs/1412.6980, May, 2015.

[20] Zhou, Y., Huang, K., Cheng, C. et al. "LightAdam: Towards a Fast and Accurate Adaptive Momentum Online Algorithm," *Cognitive Computation*, 14(2): 764-779, January 2022.

[21] Zhuang, J., Tang, T., Ding, Y., Takiconda, S., Dvornek, N., Papademetris, X., and Duncan, J. "AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients," in *Neural Information Processing Systems*, 2020.

[22] Mukkamala, M. C. and Hein, M. "Variants of RMSProp and Adagrad with Logarithmic Regret Bounds," in *Proceedings of the 34th International Conference on Machine Learning*, pp. 2545-2553, 2017.

[23] Balles, L. and Hennig, P. "Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 413-422, 2018.

[24] Liu, L., Jiang, H., He, P., et al. "On the Variance of the Adaptive Learning Rate and Beyond," in *8th International Conference on Learning Representations*, https://openreview.net/pdf?id=rkgz2aEKDr, 2020.

[25] Zaheer, M., Reddi, S. J., et al. "Adaptive methods for nonconvex optimization," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 9815-9825, December 2018.

[26] Li, Z., Bao, H., Zhang, X., and Richtárik, P. "PAGE: A Simple and Optimal Probabilistic Gradient Estimator for Nonconvex Optimization," in *Proceedings of the 38th International Conference on Machine Learning*, 139: 6286-6295, 2021.

[27] Zhou, Y., Zhang, M., Zhu, J., et al. "A Randomized Block-Coordinate Adam online learning optimization algorithm," *Neural Computing and Applications*, 32(16): 12671-12684, 2020.

[28] Ghadimi, S. and Lan, "Stochastic first- and Zeroth-order Methods for nonconvex Stochastic programming," *SIAM Journal on Optimization*, 23(4): 2341-2368, 2013.

[29] Duchi, J., Hazan, E., Singer, Y. "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, 12: 2121-2159, 2011.

[30] Chen, X., Liu, S., Sun, R., and Hong, M. "On the Convergence of A Class of Adam-Type Algorithms for Non-Convex Optimization," in *International Conference on Learning Representations*, https://openreview.net/forum?id=H1x-x309tm, 2019.

[31] Simon, L., Jaggi, M., Schmidt, M., and Pletscher, P. "Block-Coordinate Frank-Wolfe Optimization for Structural SVMs," in *Proceedings of the 30th International Conference on Machine Learning*, 28: 53-61, 2013.

[32] Nesterov, Y. E., "Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems," *SIAM J. Optim.*, 22(2): 341-362, 2012.

[33] Xie, T., Liu, B., Xu, Y. *et al.* "A Block Coordinate Ascent Algorithm for Mean-Variance Optimization," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pp. 1073-1083, 2018.

[34] Reddi, S. J., Kale, K., Kumar, S. "On the convergence of adam and beyond," in *Proceedings of the Sixth International Conference on Learning Representations*, https://openreview.net/forum?id=ryQu7f-RZ, February 2018.

[35] Luo, L., Xiong, Y., Liu, Y., Sun, X. "Adaptive gradient methods with dynamic bound of learning rate," in *Proceedings of the Seventh International Conference on Learning Representations*, https://openreview.net/forum?id=Bkg3g2R9FX, September 2019.

[36] Simonyan, K. and Zisserman, A. "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations*, http://arxiv.org/abs/1409.1556, 2015.

[37] He, K., Zhang, X., Ren, S., and Sun, J. "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[38] Huang, G., Liu, Z., Maaten, L., and Weinberger, Q. K. "Densely Connected Convolutional Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2261-2269, 2017.

[39] Xu, P., "Sample-Efficient Nonconvex Optimization Algorithms in Machine Learning and Reinforcement Learning," University of California, Los Angeles, ProQuest Dissertations Publishing, 2021. 28546872.