Linear and Nonlinear Analysis
Volume 7, Number 3, 2021, 317–336

ADAPTIVE METHODS USING ELEMENT-WISE P-TH POWER OF STOCHASTIC GRADIENT FOR NONCONVEX OPTIMIZATION IN DEEP NEURAL NETWORKS

KANAKO SHIMOYAMA AND HIDEAKI IIDUKA

Abstract. A number of useful adaptive methods, such as Adaptive Moment Estimation (Adam) and Adaptive Mean Square Gradient (AMSGrad), have been developed for nonconvex optimization in deep neural networks. These methods use element-wise squared values of the stochastic gradient. This article proposes an adaptive method using the element-wise pth power of the stochastic gradient, wherein the proposed method with $p = 2$ reduces to the existing adaptive methods. The advantages of the proposed method are discussed from the viewpoints of theory and practice. Theoretically, the proposed method can be applied to nonconvex stochastic optimization in deep neural networks. The practical advantage is that the proposed method with $p = 3, 4$, i.e., using the element-wise third or fourth power values of the stochastic gradient, performs better than the existing adaptive methods in image classification.

1. Introduction

In this article, we consider using the following loss minimization for training deep neural networks [8, 4, 14]:

$$\text{minimize } f(\boldsymbol{x}) := \sum_{i\in[n]} f_i(\boldsymbol{x}) \text{ such that } \boldsymbol{x} \in \mathbb{R}^d, \tag{1.1}$$

where $f_i \colon \mathbb{R}^d \to \mathbb{R}$ ($i \in [n] := \{1, 2, \dots, n\}$) denotes a differentiable loss function for sample i and n is the number of samples. The simplest deep learning optimizer for Problem (1.1) is stochastic gradient descent (SGD) [10], which is defined as follows: for all $k = 1, 2, \dots$,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \mathsf{G}(\boldsymbol{x}_k, \xi_k), \tag{1.2}$$

where $\boldsymbol{x}_0 \in \mathbb{R}^d$, $\alpha_k > 0$ is the learning rate, and $\mathsf{G}(\boldsymbol{x}_k, \xi_k)$ denotes the stochastic gradient of the observed loss function f_{ξ_k} at $\boldsymbol{x}_k$. *Adaptive methods* can accelerate SGD and take advantage of an efficient learning rate derived from the element-wise squared values of the stochastic gradient $(\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^2)_{i=1}^d$ (see Tables 1 and 3 for details). Here, Adaptive Gradient (AdaGrad) [5], Root Mean Square Propagation (RMSProp) [11], Adaptive Moment Estimation (Adam) [7], and Adaptive Mean Square Gradient (AMSGrad) [9] are examples of adaptive methods. The performance measure of a deep learning optimizer for the case where each of f_i is *convex* is called the *regret*, denoted by $R(n)$, and it is defined as follows: let $(\boldsymbol{x}_i)_{i=1}^n \subset \mathbb{R}^d$

2020 *Mathematics Subject Classification.* 49M37, 65K05, 90C26.
Key words and phrases. Adaptive method, Deep neural network, Nonconvex optimization, element-wise pth power.

be the sequence generated by a deep learning optimizer. Then, the regret is the difference between the sum of the values of f_i at $\boldsymbol{x}_i$ and the optimal value $f^\star$ of Problem (1.1), i.e.,

$$R(n) := \sum_{i \in [n]} f_i(\boldsymbol{x}_i) - f^\star. \tag{1.3}$$

Kingma and Ba [7] showed that Adam satisfies $R(n)/n \leq \mathcal{O}(1/\sqrt{n})$. However, Reddi, Kale, and Kumar [9] showed a counterexample such that Adam does not always satisfy $R(n)/n \leq \mathcal{O}(1/\sqrt{n})$ (While Adam as proposed in [7] uses $\mathsf{H}_k = \mathsf{diag}(\bar{v}_{k,i}^{1/2})$, Adam in Table 1 uses $\mathsf{H}_k = \mathsf{diag}(\hat{v}_{k,i}^{1/2})$ to guarantee its convergence [6]). They presented AMSGrad [9], which guarantees the convergence of Adam (see Table 3 for the definition of AMSGrad) and showed that it satisfies

$$\frac{R(n)}{n} \leq \mathcal{O}\left(\sqrt{\frac{1+\log n}{n}}\right). \tag{1.4}$$

Table 1. Adam algorithm using element-wise squared values of the stochastic gradient [7, 6] ($\boldsymbol{v}_{-1} = \hat{\boldsymbol{v}}_{-1} = \boldsymbol{0}$, $\gamma, \delta \in [0,1)$)

Table 2. Adam-type algorithm using the element-wise pth power of the stochastic gradient ($\boldsymbol{v}_{-1} = \hat{\boldsymbol{v}}_{-1} = \boldsymbol{0}$, $\gamma, \delta \in [0,1)$)

Table 1	Table 2
$\boldsymbol{m}_k := \beta_k \boldsymbol{m}_{k-1} + (1-\beta_k)\mathsf{G}(\boldsymbol{x}_k, \xi_k)$	$\boldsymbol{m}_k := \beta_k \boldsymbol{m}_{k-1} + (1-\beta_k)\mathsf{G}(\boldsymbol{x}_k, \xi_k)$
$\hat{\boldsymbol{m}}_k := \frac{\boldsymbol{m}_k}{(1-\gamma)^{k+1}}$	$\hat{\boldsymbol{m}}_k := \frac{\boldsymbol{m}_k}{(1-\gamma)^{k+1}}$
$\boxed{\boldsymbol{v}_k := \delta \boldsymbol{v}_{k-1} + (1-\delta)(\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^2)_{i=1}^d}$	$\boxed{\boldsymbol{v}_k := \delta \boldsymbol{v}_{k-1} + (1-\delta)(\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^p)_{i=1}^d}$
$\bar{\boldsymbol{v}}_k := \frac{\boldsymbol{v}_k}{1-\delta^{k+1}}$	$\bar{\boldsymbol{v}}_k := \frac{\boldsymbol{v}_k}{1-\delta^{k+1}}$
$\hat{\boldsymbol{v}}_k = (\hat{v}_{k,i}) := (\max\{\hat{v}_{k-1,i}, \bar{v}_{k,i}\})$	$\hat{\boldsymbol{v}}_k = (\hat{v}_{k,i}) := (\max\{\hat{v}_{k-1,i}, \bar{v}_{k,i}\})$
$\boxed{\mathsf{H}_k = \mathsf{diag}\left(\hat{v}_{k,i}^{1/2}\right)}$	$\boxed{\mathsf{H}_k = \mathsf{diag}\left(\hat{v}_{k,i}^{1/p}\right)}$
$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \alpha_k \mathsf{H}_k^{-1} \hat{\boldsymbol{m}}_k$	$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \alpha_k \mathsf{H}_k^{-1} \hat{\boldsymbol{m}}_k$

Problem (1.1) when each of f_i is *nonconvex* is an important consideration in deep neural networks because the loss functions are not always convex [13, 12, 1]. In this case, we aim to find a stationary point of Problem (1.1), as follows:

$$\text{find a point } \boldsymbol{x}^\star \in \mathbb{R}^d \text{ such that } \nabla f(\boldsymbol{x}^\star) = \boldsymbol{0}, \tag{1.5}$$

where $\nabla f \colon \mathbb{R}^d \to \mathbb{R}^d$ denotes the gradient of f. The performance measure [6, 3, 15] of the deep-learning optimizer for Problem (1.5) is such that, for all $K = 1, 2, \ldots$,

$$\min_{k \in [K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right], \tag{1.6}$$

where $\mathbb{E}[X]$ denotes the expectation of a random variable X. Chen, Liu, Sun, and Hong [3] showed that AMSGrad (see Table 3) can be applied to Problem (1.5) and that it satisfies

$$\min_{k\in[K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] = \mathcal{O}\left(\frac{\log K}{\sqrt{K}}\right). \tag{1.7}$$

Zhuang, Tang, Ding, Tatikonda, Dvornek, Papademetris, and Duncan [15] presented AdaBelief, which stands for adapting stepsizes by the belief in observed gradients, which can be obtained by replacing $\mathsf{G}(\boldsymbol{x}_k, \xi_k)$ in Adam (see Table 1) with $\mathsf{G}(\boldsymbol{x}_k, \xi_k) - \boldsymbol{m}_k$ and showed that AdaBelief satisfies (1.7). Recently, Iiduka [6] proposed an algorithm that unifies these useful adaptive methods, i.e., Adam (Table 1), AMSGrad (Table 3), and AdaBelief, and showed that the algorithm satisfies

$$\min_{k\in[K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right), \tag{1.8}$$

which improves on the previous results [3, 15].

Table 3. AMSGrad algorithm using element-wise squared values of the stochastic gradient [9] ($\boldsymbol{v}_{-1} = \hat{\boldsymbol{v}}_{-1} = \mathbf{0}$, $\delta \in [0,1)$)

$$\boldsymbol{m}_k := \beta_k \boldsymbol{m}_{k-1} + (1-\beta_k)\mathsf{G}(\boldsymbol{x}_k, \xi_k)$$
$$\boxed{\boldsymbol{v}_k := \delta \boldsymbol{v}_{k-1} + (1-\delta)(\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^2)_{i=1}^d}$$
$$\hat{\boldsymbol{v}}_k = (\hat{v}_{k,i}) := (\max\{\hat{v}_{k-1,i}, v_{k,i}\})$$
$$\boxed{\mathsf{H}_k = \mathsf{diag}\left(\hat{v}_{k,i}^{1/2}\right)}$$
$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \alpha_k \mathsf{H}_k^{-1} \boldsymbol{m}_k$$

Table 4. AMSGrad algorithm using the element-wise pth power of the stochastic gradient ($\boldsymbol{v}_{-1} = \hat{\boldsymbol{v}}_{-1} = \mathbf{0}$, $\delta \in [0,1)$)

$$\boldsymbol{m}_k := \beta_k \boldsymbol{m}_{k-1} + (1-\beta_k)\mathsf{G}(\boldsymbol{x}_k, \xi_k)$$
$$\boxed{\boldsymbol{v}_k := \delta \boldsymbol{v}_{k-1} + (1-\delta)(\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^p)_{i=1}^d}$$
$$\hat{\boldsymbol{v}}_k = (\hat{v}_{k,i}) := (\max\{\hat{v}_{k-1,i}, v_{k,i}\})$$
$$\boxed{\mathsf{H}_k = \mathsf{diag}\left(\hat{v}_{k,i}^{1/p}\right)}$$
$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \alpha_k \mathsf{H}_k^{-1} \boldsymbol{m}_k$$

1.1. **Motivation.**

The above discussion shows that adaptive methods are useful for both convex and nonconvex optimization in deep neural networks. Here, we have two motivations related to the previous studies [7, 9, 6, 3] on adaptive methods. The existing adaptive methods use element-wise *squared* values $(\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^2)_{i=1}^d$ of the stochastic gradient $\mathsf{G}(\boldsymbol{x}_k, \xi_k)$ (see the boxed parts in Tables 1 and 3). Here, we are interested in the performance of adaptive methods using the element-wise pth power $(\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^p)_{i=1}^d$ of the stochastic gradient, e.g., Adam and AMSGrad using the element-wise pth power as shown in Tables 2 and 4. See [2] for adaptive methods using the element-wise squared values $(\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^2)_{i=1}^d$ of the stochastic gradient and $\mathsf{H}_k = \mathsf{diag}(\hat{v}_{k,i}^q)$, where $q \in (0, 1/2]$. The proposed algorithms with $p = 2$ in Tables 2 and 4 coincide with Adam and AMSGrad in Tables 1 and 3. Therefore, the first motivation is to clarify that the proposed algorithm can in theory be applied to nonconvex optimization in the same manner as the previous methods [6, 3]. The second motivation is to clarify whether the proposed algorithm outperforms the existing adaptive methods in practice.

1.2. **Contribution.** This article proposes an adaptive method using the element-wise pth power of the stochastic gradient (Algorithm 1 with H_k in Tables 2 and 4). The theoretical contribution is to prove that Algorithm 1 can be applied to nonconvex stochastic optimization in deep neural networks (Theorems 2.3 and 2.4). The practical contribution is to show numerical results for image classification such that Algorithm 1 with H_k and $p = 3, 4$ in Tables 2 and 4, i.e., the adaptive method using element-wise third or fourth power values of the stochastic gradient, performs better than the existing adaptive methods (Section 3).

The remainder of the article is as follows. Section 2 presents the proposed algorithm (Algorithm 1) and its convergence analyses. Section 3 provides numerical comparisons of the proposed algorithm with the existing algorithms. Section 4 concludes the paper with a brief summary.

2. Proposed Algorithm and Its Convergence Analysis

Algorithm 1 is a listing of the proposed method that simplifies to Adam and AMSGrad with $p = 2$.

Algorithm 1 Adaptive method [6] for Problem (1.5)

```
Require: $(\alpha_k)_{k\in\mathbb{N}} \subset (0,1), (\beta_k)_{k\in\mathbb{N}} \subset [0,1), \gamma \in [0,1)$
  $k \leftarrow 0, \boldsymbol{x}_0, \boldsymbol{m}_{-1} \in \mathbb{R}^d, \mathsf{H}_0 \in \mathbb{S}^d_{++} \cap \mathbb{D}^d$
  loop
      $\boldsymbol{m}_k := \beta_k \boldsymbol{m}_{k-1} + (1-\beta_k)\mathsf{G}(\boldsymbol{x}_k, \xi_k)$
      $\hat{\boldsymbol{m}}_k := \dfrac{\boldsymbol{m}_k}{1-\gamma^{k+1}}$
      $\mathsf{H}_k \in \mathbb{S}^d_{++} \cap \mathbb{D}^d$ (see Tables 1, 2, 3, and 4 for examples of $\mathsf{H}_k$)
      Find $\mathsf{d}_k \in \mathbb{R}^d$ that solves $\mathsf{H}_k \mathsf{d} = -\hat{\boldsymbol{m}}_k$
      $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \mathsf{d}_k$
      $k \leftarrow k+1$
  end loop
```

To analyze Algorithm 1, we assume the following:

Assumption 2.1. *The sequence* $(\mathsf{H}_k)_{k\in\mathbb{N}} \subset \mathbb{S}^d_{++} \cap \mathbb{D}^d$, *where* $\mathsf{H}_k := \mathsf{diag}(h_{k,i})$, *satisfies the following conditions:*

(A1) $h_{k+1,i} \geq h_{k,i}$ *for all* $k \in \mathbb{N}$ *and all* $i \in [d]$;

(A2) For all $i \in [d]$, *a positive number* B_i *exists such that* $\sup\{\mathbb{E}[h_{k,i}] \colon k \in \mathbb{N}\} \leq B_i$.

The generated sequence $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ *in Algorithm 1, where* $\boldsymbol{x}_k = (x_{k,i})_{i=1}^d$, *satisfies the following conditions:*

(A3) $D := \max_{i\in[d]} \sup\{(x_{k,i} - x_i)^2 \colon k \in \mathbb{N}\} < +\infty$ *for* $\boldsymbol{x} = (x_i) \in \mathbb{R}^d$;

(A4) A positive number M *exists such that, for all* $k \in \mathbb{N}$, $\mathbb{E}[\|\mathsf{G}(\boldsymbol{x}_k, \xi_k)\|^2] \leq M^2$.

Assumptions (A3) and (A4) are needed to analyze adaptive methods (see, e.g., [7, 9, 6]). Next, we prove a proposition.

Proposition 2.2. *Under (A3), H_k defined as in Tables 1, 2, 3, and 4 satisfies (A1) and (A2).*

Proof: From $\mathsf{H}_k = \mathsf{diag}(\hat{v}_{k,i}^{1/p})$ $(p > 0)$ and $\hat{v}_{k,i} \geq \hat{v}_{k-1,i}$, we have that, for all $k \in \mathbb{N}$ and all $i \in [d]$,

$$h_{k+1,i} = \hat{v}_{k+1,i}^{\frac{1}{p}} \geq \hat{v}_{k,i}^{\frac{1}{p}} =: h_{k,i},$$

which implies that (A1) holds. Let us consider the case where H_k is in Table 2. The continuity of $\mathsf{G}(\cdot, \xi_k)$ and (A3) ensure that $(\mathsf{G}(\boldsymbol{x}_k, \xi_k))_{k\in\mathbb{N}}$ is almost surely bounded, i.e.,

$$M_1 := \sup \left\{ \left\| (\mathsf{G}(\boldsymbol{x}_k, \xi_k)_i^p)_{i=1}^d \right\| : k \in \mathbb{N} \right\} < +\infty.$$

Moreover, the definition of $\boldsymbol{v}_k$ and the triangle inequality guarantee that, for all $k \in \mathbb{N}$,

$$\|\boldsymbol{v}_k\| \leq \delta \|\boldsymbol{v}_{k-1}\| + (1-\delta) M_1.$$

Induction, together with $\boldsymbol{v}_{-1} = \mathbf{0}$, thus shows that, for all $k \in \mathbb{N}$,

$$\|\boldsymbol{v}_k\| = \left(\sum_{i=1}^d |v_{k,i}|^2 \right)^{1/2} \leq M_1$$

almost surely, which, together with the definition of $\bar{\boldsymbol{v}}_k$, implies that

$$\|\bar{\boldsymbol{v}}_k\| = \left(\sum_{i=1}^d |\bar{v}_{k,i}|^2 \right)^{1/2} \leq \frac{M_1}{1-\delta}.$$

Hence, for all $k \in \mathbb{N}$ and all $i \in [d]$, we have

$$|v_{k,i}|^2, |\bar{v}_{k,i}|^2 \leq \frac{M_1^2}{(1-\delta)^2}.$$

The definitions of $\hat{\boldsymbol{v}}_k$ and $\hat{\boldsymbol{v}}_{-1} = \mathbf{0}$ ensure that, for all $k \in \mathbb{N}$ and all $i \in [d]$,

$$\mathbb{E}[h_{k,i}] := \mathbb{E}\left[\hat{v}_{k,i}^{\frac{1}{p}} \right] \leq \left(\frac{M_1}{1-\delta} \right)^{\frac{1}{p}},$$

which implies that (A2) holds. Accordingly, H_k as defined in Table 1 $(p = 2)$ also satisfies (A2).

Let us consider the case where H_k is as in Table 4. A discussion similar to the one showing that H_k defined as in Table 2 satisfies (A2) ensures that H_k defined as in Table 4 also satisfies (A2); i.e., for all $k \in \mathbb{N}$ and all $i \in [d]$,

$$\mathbb{E}[h_{k,i}] := \mathbb{E}\left[\hat{v}_{k,i}^{\frac{1}{p}} \right] \leq M_1^{\frac{1}{p}}.$$

Accordingly, H_k defined as in Table 3 $(p = 2)$ also satisfies (A2). □

The following is a convergence analysis of Algorithm 1 with constant learning rates $\alpha_k = \alpha$ and $\beta_k = \beta$ for Problem (1.5). The proof of Theorem 2.3 is given in Theorem 1 of [6].

Theorem 2.3. *Suppose that Algorithm 1 satisfies Assumption 2.1 and let $\alpha_k := \alpha$ and $\beta_k := \beta$ ($k \in \mathbb{N}$). Then, the following holds for all $\boldsymbol{x} \in \mathbb{R}^d$:*

$$\liminf_{k\to+\infty} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \le \frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\gamma}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}}\beta, \tag{2.1}$$

where $\tilde{B} := \sup\{\max_{i\in[d]} h_{k,i}^{-1/2} : k \in \mathbb{N}\} < +\infty$, $\tilde{M}^2 := \max\{\|\boldsymbol{m}_{-1}\|^2, M^2\}$ and $\tilde{\gamma} := 1-\gamma$, $\tilde{b} := 1-\beta$. Furthermore, the following holds for all $K \ge 1$:

$$\min_{k\in[K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \le \frac{D\sum_{i=1}^d B_i}{2\tilde{b}\alpha K} + \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}}\beta. \tag{2.2}$$

Recall that the performance measure of Algorithm 1 for Problem (1.5) is defined by (1.6), i.e.,

$$\min_{k\in[K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right].$$

Inequality (2.1) in Theorem 2.3 ensures that a subsequence $(\boldsymbol{x}_{k_i})_{i\in\mathbb{N}}$ of $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ exists such that $(\boldsymbol{x}_{k_i})_{i\in\mathbb{N}}$ converges almost surely to a point $\boldsymbol{x}^* \in \mathbb{R}^d$ and

$$\mathbb{E}\left[\langle \boldsymbol{x}^* - \boldsymbol{x}, \nabla f(\boldsymbol{x}^*)\rangle\right] \le \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}}\beta$$

for all $\boldsymbol{x} \in \mathbb{R}^d$. Accordingly, positive constants C_1 and C_2 exist such that

$$\mathbb{E}\left[\|\nabla f(\boldsymbol{x}^*)\|^2\right] \le C_1\alpha + C_2\beta,$$

which implies that Algorithm 1 with constant learning rates α and β will find an approximate stationary point of Problem (1.5). Inequality (2.2) in Theorem 2.3 indicates that Algorithm 1 with constant learning rates α and β ensures that positive constants C_1 and C_2 exist such that, for all $K \ge 1$,

$$\min_{k\in[K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \le \mathcal{O}\left(\frac{1}{K}\right) + C_1\alpha + C_2\beta,$$

which implies that Algorithm 1 with constant learning rates α and β has approximately an $\mathcal{O}(1/K)$ convergence rate. For Problem (1.1) when f_i is convex, Theorem 2.3 leads to the finding that

$$\frac{R(n)}{n} \le \mathcal{O}\left(\frac{1}{n}\right) + C_1\alpha + C_2\beta$$

(see Proposition 1 in [6] for details).

The following is a convergence analysis of Algorithm 1 with diminishing learning rates α_k and β_k for Problem (1.5). The proof of Theorem 2.4 is given in Theorem 2 in [6].

Theorem 2.4. *Suppose that Algorithm 1 satisfies Assumption 2.1 and let α_k and β_k ($k \in \mathbb{N}$) be such that $\sum_{k=0}^{+\infty} \alpha_k = +\infty$, $\sum_{k=0}^{+\infty} \alpha_k^2 < +\infty$, and $\sum_{k=0}^{+\infty} \alpha_k \beta_k < +\infty$. Then, the following holds for all $\boldsymbol{x} \in \mathbb{R}^d$:*

$$\liminf_{k \to +\infty} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq 0. \tag{2.3}$$

Moreover, Algorithm 1 with $\alpha_k := 1/k^\eta$ ($\eta \in [1/2, 1)$) and $\beta_k := \lambda^k$ ($\lambda \in (0,1)$) has the following convergence rate for all $K \geq 1$:

$$\min_{k \in [K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \frac{D \sum_{i=1}^d B_i}{2\tilde{b} K^{1-\eta}} + \frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\gamma}^2 (1-\eta) K^{1-\eta}} + \frac{\tilde{M}\lambda\sqrt{Dd}}{\tilde{b}(1-\lambda)K}, \tag{2.4}$$

where D, $\tilde{b}$, $\tilde{M}$, $\tilde{B}$, and $\tilde{\gamma}$ are the same as in Theorem 2.3.

Inequality (2.3) in Theorem 2.4 ensures that there exists a subsequence $(\boldsymbol{x}_{k_j})_{j \in \mathbb{N}}$ of $(\boldsymbol{x}_k)_{k \in \mathbb{N}}$ such that $(\boldsymbol{x}_{k_j})_{j \in \mathbb{N}}$ converges almost surely to a point $\boldsymbol{x}_* \in \mathbb{R}^d$ satisfying

$$\mathbb{E}\left[\|\nabla f(\boldsymbol{x}_*)\|^2\right] = 0,$$

which implies that Algorithm 1 with diminishing learning rates α_k and β_k can solve Problem (1.5). Inequality (2.4) in Theorem 2.4 indicates that Algorithm 1 with diminishing learning rates $\alpha_k = 1/\sqrt{k}$ and $\beta_k = \lambda^k$ satisfies that, for all $K \geq 1$,

$$\min_{k \in [K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right).$$

For Problem (1.1) when f_i is convex, Theorem 2.4 leads to the finding that

$$\frac{R(n)}{n} \leq \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$$

(see Proposition 2 in [6] for details).

3. Numerical Experiments

This section presents the results of experiments comparing our algorithm with the previous work. We examined the behavior of the algorithms with different learning rates in image classification tasks. The algorithms with $\delta = 0.999$ used in the experiments reported in [7, 9] are as follows, where the ADAM-type algorithms use $\gamma = 0.9$, and the AMSG-type algorithms use $\gamma = 0$.

Algorithm 1 with constant learning rates:

- ADAM-C1: Algorithm 1 with the parameters in Table 2, $p = 2$, $\alpha_k = 10^{-3}$, and $\beta_k = 0.9$
- ADAM-C2: Algorithm 1 with the parameters in Table 2, $p = 2$, $\alpha_k = 10^{-3}$, and $\beta_k = 10^{-3}$
- AMSG-C1: Algorithm 1 with the parameters in Table 4, $p = 2$, $\alpha_k = 10^{-3}$, and $\beta_k = 0.9$
- AMSG-C2: Algorithm 1 with the parameters in Table 4, $p = 2$, $\alpha_k = 10^{-3}$, and $\beta_k = 10^{-3}$
- PADAM1-C1: Algorithm 1 with the parameters in Table 2, $p = 3$, $\alpha_k = 10^{-3}$, and $\beta_k = 0.9$

- PADAM1-C2: Algorithm 1 with the parameters in Table 2, $p = 3$, $\alpha_k = 10^{-3}$, and $\beta_k = 10^{-3}$
- PAMSG1-C1: Algorithm 1 with the parameters in Table 4, $p = 3$, $\alpha_k = 10^{-3}$, and $\beta_k = 0.9$
- PAMSG1-C2: Algorithm 1 with the parameters in Table 4, $p = 3$, $\alpha_k = 10^{-3}$, and $\beta_k = 10^{-3}$
- PADAM2-C1: Algorithm 1 with the parameters in Table 2, $p = 4$, $\alpha_k = 10^{-3}$, and $\beta_k = 0.9$
- PADAM2-C2: Algorithm 1 with the parameters in Table 2, $p = 4$, $\alpha_k = 10^{-3}$, and $\beta_k = 10^{-3}$
- PAMSG2-C1: Algorithm 1 with the parameters in Table 4, $p = 4$, $\alpha_k = 10^{-3}$, and $\beta_k = 0.9$
- PAMSG2-C2: Algorithm 1 with the parameters in Table 4, $p = 4$, $\alpha_k = 10^{-3}$, and $\beta_k = 10^{-3}$

Algorithm 1 with diminishing learning rates:

- ADAM-D1: Algorithm 1 with the parameters in Table 2, $p = 2$, $\alpha_k = 1/\sqrt{k}$, and $\beta_k = 1/2^k$
- ADAM-D2: Algorithm 1 with the parameters in Table 2, $p = 2$, $\alpha_k = 1/k$, and $\beta_k = 1/2^k$
- AMSG-D1: Algorithm 1 with the parameters in Table 4, $p = 2$, $\alpha_k = 1/\sqrt{k}$, and $\beta_k = 1/2^k$
- AMSG-D2: Algorithm 1 with the parameters in Table 4, $p = 2$, $\alpha_k = 1/k$, and $\beta_k = 1/2^k$
- PADAM1-D1: Algorithm 1 with the parameters in Table 2, $p = 3$, $\alpha_k = 1/\sqrt{k}$, and $\beta_k = 1/2^k$
- PADAM1-D2: Algorithm 1 with the parameters in Table 2, $p = 3$, $\alpha_k = 1/k$, and $\beta_k = 1/2^k$
- PAMSG1-D1: Algorithm 1 with the parameters in Table 4, $p = 3$, $\alpha_k = 1/\sqrt{k}$, and $\beta_k = 1/2^k$
- PAMSG1-D2: Algorithm 1 with the parameters in Table 4, $p = 3$, $\alpha_k = 1/k$, and $\beta_k = 1/2^k$
- PADAM2-D1: Algorithm 1 with the parameters in Table 2, $p = 4$, $\alpha_k = 1/\sqrt{k}$, and $\beta_k = 1/2^k$
- PADAM2-D2: Algorithm 1 with the parameters in Table 2, $p = 4$, $\alpha_k = 1/k$, and $\beta_k = 1/2^k$
- PAMSG2-D1: Algorithm 1 with the parameters in Table 4, $p = 4$, $\alpha_k = 1/\sqrt{k}$, and $\beta_k = 1/2^k$
- PAMSG2-D2: Algorithm 1 with the parameters in Table 4, $p = 4$, $\alpha_k = 1/k$, and $\beta_k = 1/2^k$

The compared adaptive methods are variants of Algorithm 1 with $p = 2$. In particular, ADAM-C1 and AMSG-C1 are the same as Adam and AMSGrad [7, 9]. The proposed method was Algorithm 1 with $p = 3, 4$.

Our experiments were conducted on a fast scalar computation server at Meiji University. The environment has two Intel (R) Xeon (R) Gold 6148 (2.4 GHz, 20 cores) CPUs, an NVIDIA Tesla V100 (16GB, 900Gbps) GPU and a Red Hat Enterprise Linux 7.6 operating system. The experimental code was written in Python 3.8.2, and we used the NumPy 1.18.1 package and PyTorch 1.3.0 package.

3.1. **MNIST.** For the image classification, we chose to use a residual network (ResNet) (`https://deepage.net/deep_learning/2016/11/30/resnet.html`), which is a representative model of a convolutional neural network (CNN) (`https://jp.mathworks.com/discovery/convolutional-neural-network.html`). We used a 34-layer ResNet and cross entropy as a loss function. We used the MNIST dataset (`http://yann.lecun.com/exdb/mnist/`), which consists of handwritten digits from 0 to 9. The dataset contains 60,000 training data and 10,000 test data.

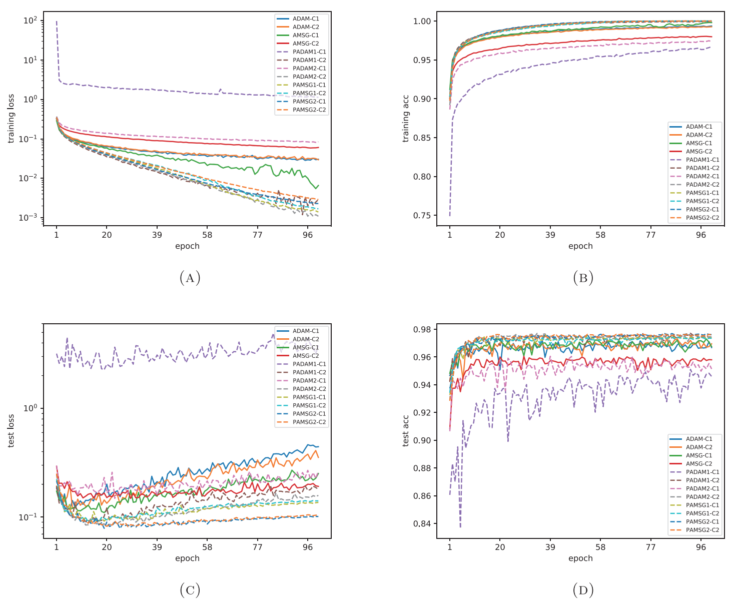


FIGURE 1. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant learning rates versus number of epochs on the MNIST dataset

Figure 1 (resp. Figure 2) shows the results for (a) the training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant (resp. diminishing) learning rates versus the number of epochs on the MNIST dataset. Figure 3 (resp. Figure 4) shows box-plot comparisons of the results in Figure 1 (resp. Figure 2), where the boxes represent the upper and lower quartiles and the interior horizontal lines are the medians.

Figures 1(a), 2(a), 3(a), and 4(a) show that the proposed method, except for PADAM1-C1 and PADAM2-C1, minimized the training loss function faster than the

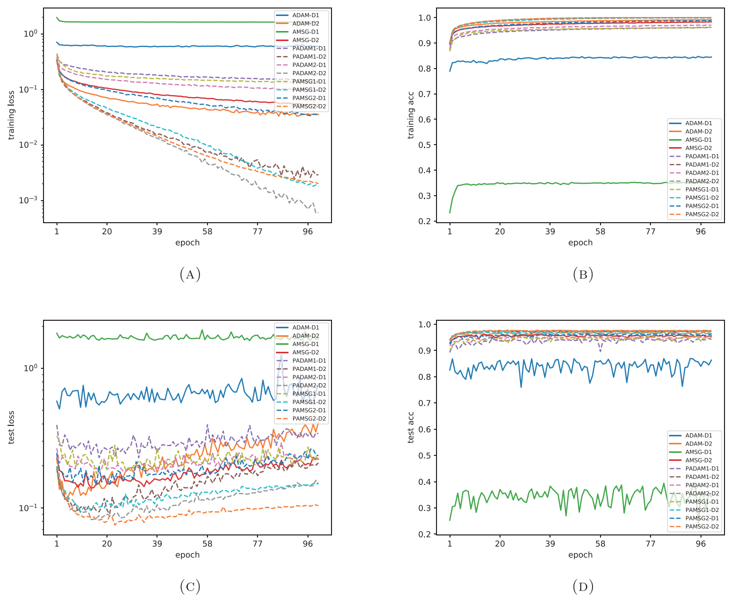
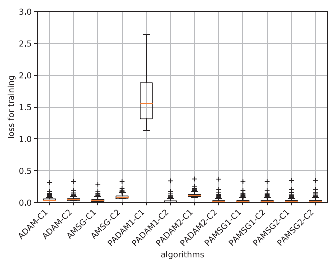


Figure 2. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with diminishing learning rates versus number of epochs on the MNIST dataset

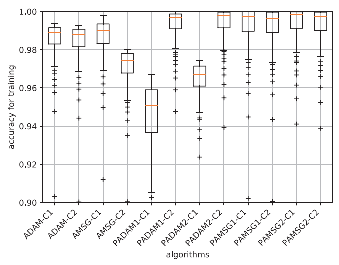
previous algorithms did. Figures 2(a) and 4(a) also show that the previous adaptive methods using $p = 2$ (e.g., ADAM-D1 and AMSG-D1) with diminishing learning rates did not work so well, as also shown in [6]. Meanwhile, the adaptive methods using $p = 3, 4$ (e.g., PADAM1-D1, PADAM2-D1, PAMSG1-D1, and PAMSG2-D1) with diminishing learning rates outperformed the existing adaptive methods using $p = 2$ (e.g., ADAM-D1 and AMSG-D1). Figures 1(c), 2(c), 3(c), and 4(c) show that the proposed method, except for PADAM1-C1, minimized the test loss function faster than the previous ones. Figures 1(b), 1(d), 3(b), and 3(d) show that the proposed method, except for PADAM1-C1, had high accuracy. Figures 2(b), 2(d), 4(b), and 4(d) show that the proposed algorithm was more accurate than the previous ones.

3.2. **CIFAR-10.** We conducted an experiment on the CIFAR-10 (`https://www.cs.toronto.edu/~kriz/cifar.html`). The dataset is a benchmark for image classification and has ten classes. It has 50,000 training data and 10,000 test data, which are labeled images (32×32). Moreover, it has 64 training and test batches.

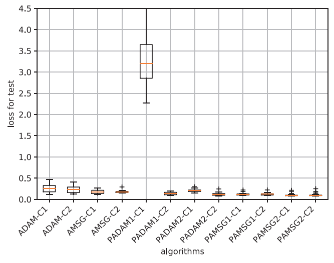
Figures 5–10 show numerical results on CIFAR-10. Figure 5 compares ADAM-C and AMSG-C ($p = 2$) with PADAM1-C and PAMSG1-C ($p = 3$), while Figure 6 compares ADAM-D and AMSG-D ($p = 2$) with PADAM1-D and PAMSG1-D ($p = 3$). Figures 5(a) and 6(a) show that the proposed method minimized the training loss function more than the existing algorithms did. Figures 5(b) and 6(b) show that the proposed algorithm had almost the same training accuracy as



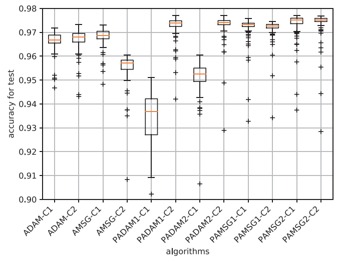
(A)



(B)

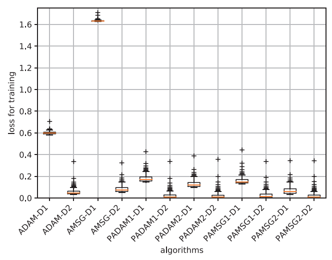


(C)

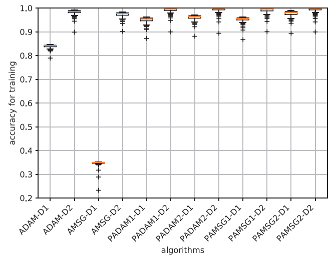


(D)

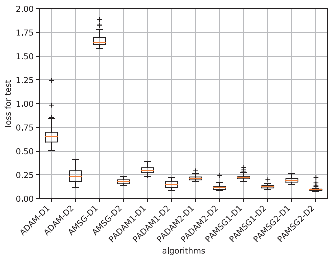
FIGURE 3. Box-plot comparisons of (a) training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant learning rates on the MNIST dataset



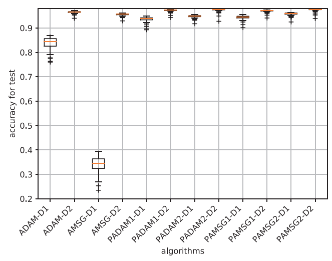
(A)



(B)

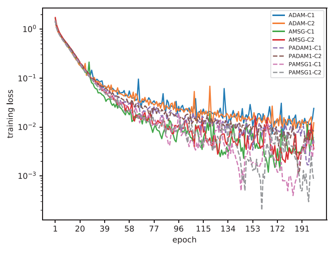


(C)

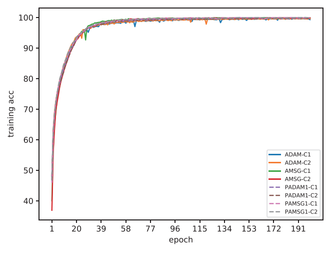


(D)

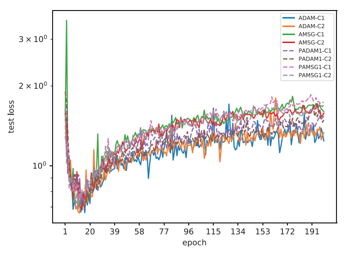
FIGURE 4. Box-plot comparisons of (a) training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with diminishing learning rates on the MNIST dataset



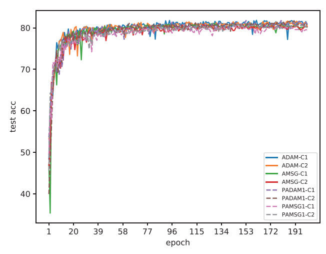
(A)



(B)

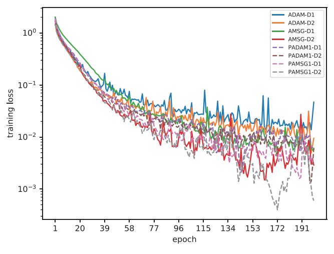


(C)

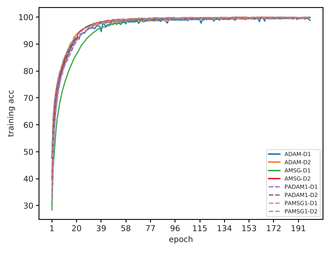


(D)

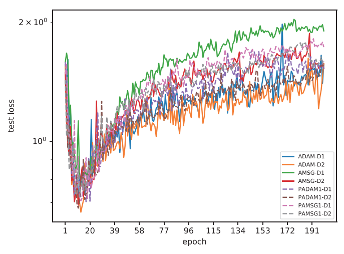
FIGURE 5. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant learning rates ($p = 2, 3$) versus number of epochs on the CIFAR-10 dataset



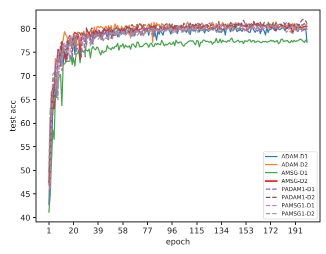
(A)



(B)

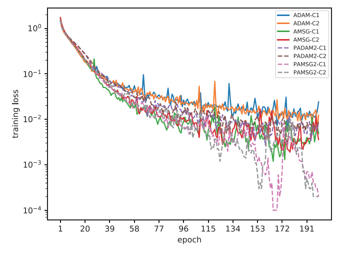


(C)

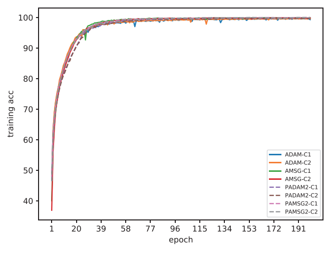


(D)

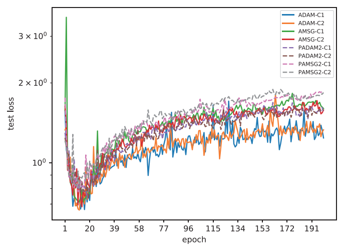
FIGURE 6. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with diminishing learning rates ($p = 2, 3$) versus number of epochs on the CIFAR-10 dataset



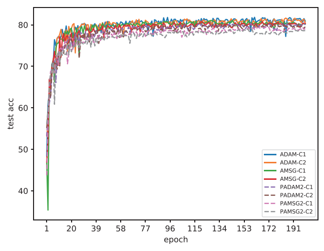
(a)



(b)

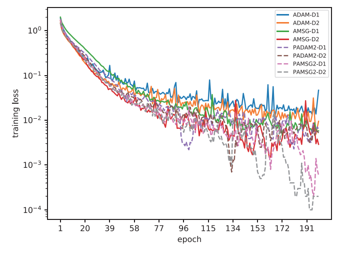


(c)

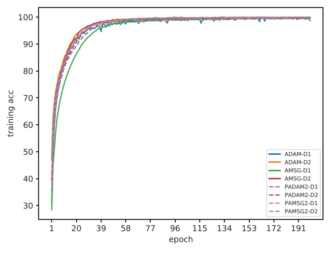


(d)

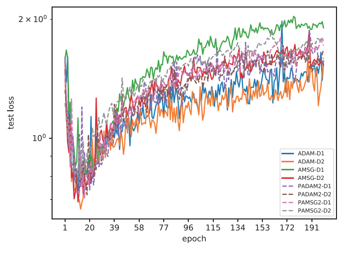
FIGURE 7. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant learning rates ($p = 2, 4$) versus number of epochs on the CIFAR-10 dataset



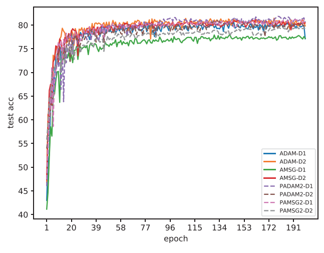
(a)



(b)

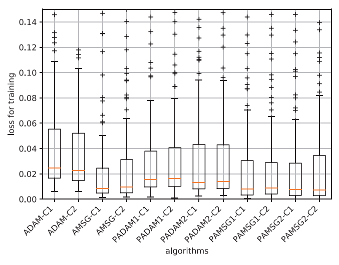


(c)

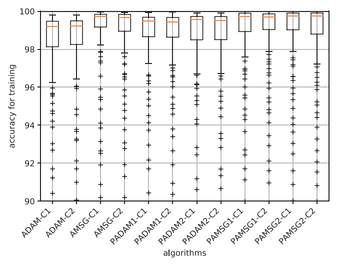


(d)

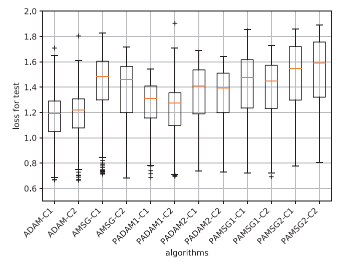
FIGURE 8. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with diminishing learning rates ($p = 2, 4$) versus number of epochs on the CIFAR-10 dataset the existing algorithms. Figures 5(c), 6(c), 5(d), and 6(d) show that the proposed algorithm performed almost as well as the existing algorithms. Figure 7 compares ADAM-C and AMSG-C ($p = 2$) with PADAM2-C and PAMSG2-C ($p = 4$), while Figure 8 compares ADAM-D and AMSG-D ($p = 2$) with PADAM2-D and PAMSG2-D ($p = 4$). The results are similar to those in Figures 5 and 6; i.e., the proposed algorithm performed well.



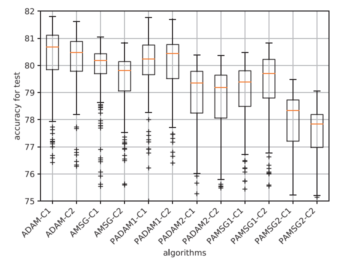
(A)



(B)



(C)



(D)

FIGURE 9. Box-plot comparisons of (a) training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant learning rates on the CIFAR-10 dataset

3.3. **CIFAR-100.** Next, we conducted an experiment that used CIFAR-100 (`https://www.cs.toronto.edu/~kriz/cifar.html`). CIFAR-100 has the same structure as CIFAR-10, but its datasets have super classes, which divide 100 labels into five groups with 20 labels. In our experiment, we used the fine classes as the correct labels instead of the superclasses.

Figures 11–16 show the numerical results on CIFAR-100 (the comparisons in the figures are the same as those in Section 3.2). Figures 11(a) and 12(a) show that PAMSG1 outperformed the existing algorithms. Figures 11(b), 11(c), 11(d), 12(b), 12(c), and 12(d) show that the proposed method performed as well as the existing ones. Figure 13(a) shows that it minimized the training loss function faster than the existing ones did. Figure 13(b) shows that PAMSG2 was more accurate than AMSG. Figures 13(c) and 13(d) show that PADAM2 and PAMSG2 were unstable. Figures 14(a) and 14(b) show that the proposed algorithm outperformed the existing algorithms. However, Figures 14(c) and 14(d) indicate that PADAM2 and PAMSG2

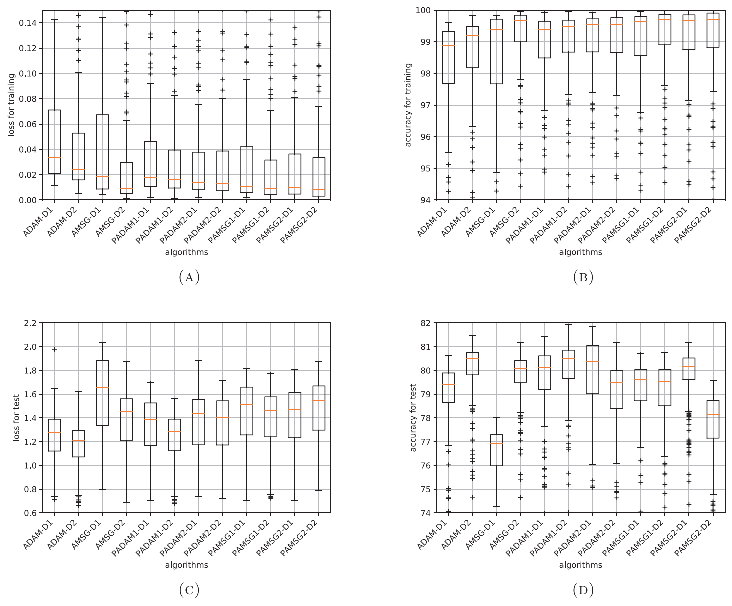


FIGURE 10. Box-plot comparisons of (a) training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with diminishing learning rates on the CIFAR-10 dataset

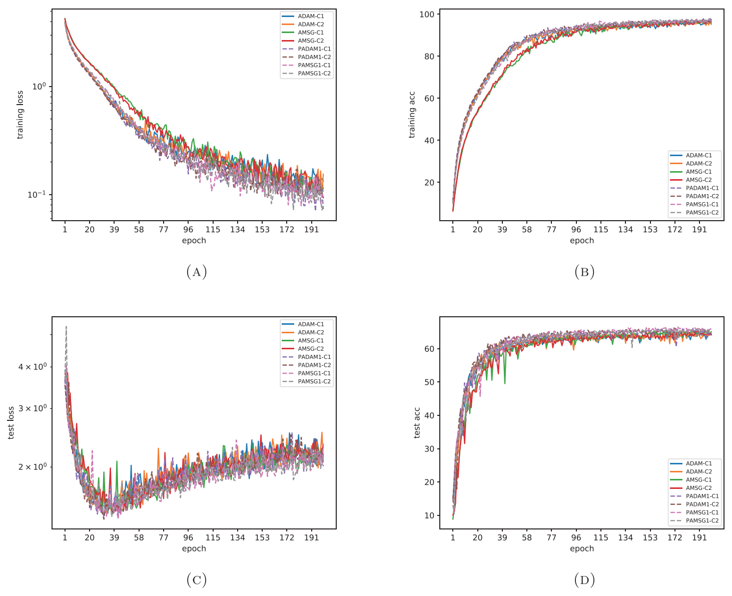
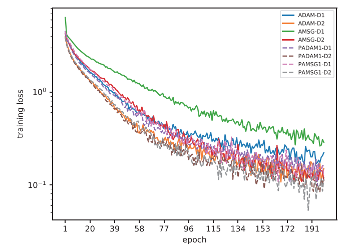
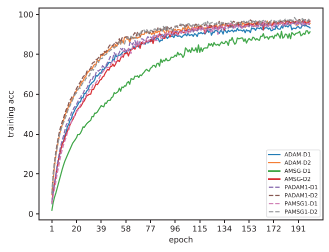


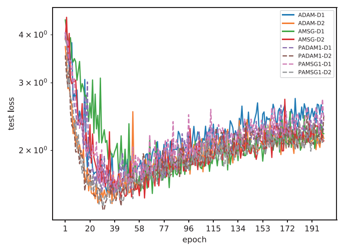
FIGURE 11. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant learning rates $(p = 2, 3)$ versus number of epochs on the CIFAR-100 dataset



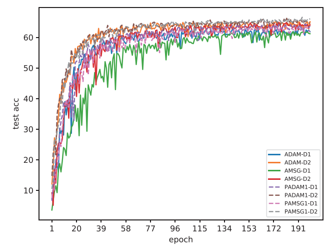
(A)



(B)

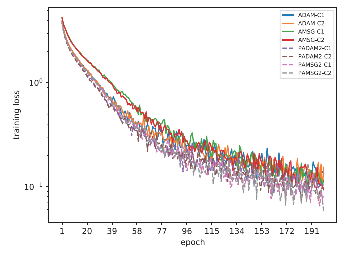


(C)

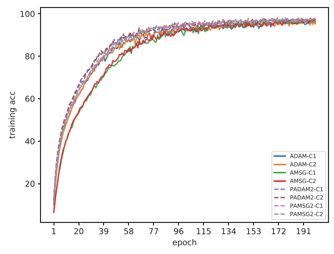


(D)

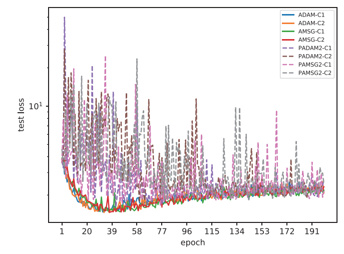
FIGURE 12. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with diminishing learning rates ($p = 2, 3$) versus number of epochs on the CIFAR-100 dataset



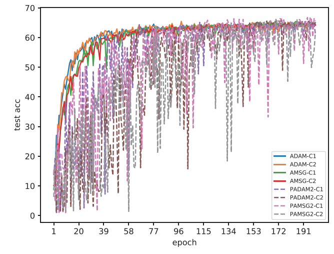
(A)



(B)

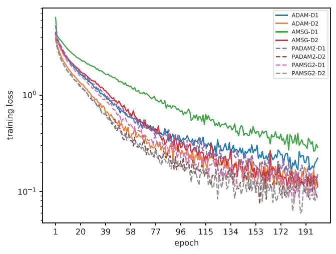


(C)

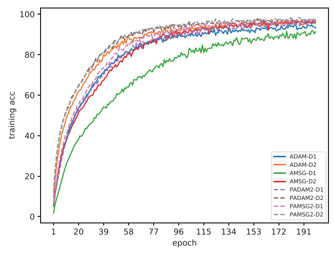


(D)

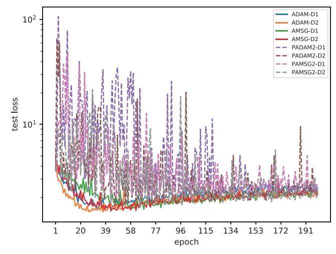
FIGURE 13. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant learning rates ($p = 2, 4$) versus number of epochs on the CIFAR-100 dataset



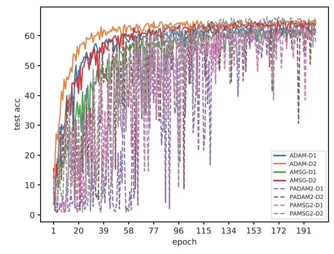
(A)



(B)



(C)



(D)

FIGURE 14. (a) Training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with diminishing learning rates ($p = 2, 4$) versus number of epochs on the CIFAR-100 dataset

were unstable. As shown in Figures 15 and 16, the algorithms with $p = 3$ were better than the algorithms with $p = 4$.

On the complicated datasets, the algorithm with $p = 3$ (PADAM1 and PAMSG1) outperformed the algorithms with $p = 4$ (PADAM2 and PAMSG2) in the terms of the training loss and accuracy. Moreover, the proposed algorithm was better than the existing algorithms in the sense of minimization of the loss functions (see panel (a) in the figures). Therefore, the proposed algorithm with $p = 3, 4$ is superior for loss minimization in deep neural networks.

4. Conclusion and Future Work

This article proposed an adaptive method using the element-wise pth power of the stochastic gradient for solving nonconvex optimization problems in deep neural networks; this method is different from the previous adaptive methods using element-wise squared values of the stochastic gradient. We presented two convergence analyses of the proposed method. The first convergence analysis indicated that the proposed method with constant learning rates will find an approximate stationary point of the nonconvex optimization problem. The second convergence analysis indicated that the proposed method with diminishing learning rates has an $\mathcal{O}(1/\sqrt{K})$ convergence rate for the stationary point problem. We also provide numerical comparisons using the benchmark datasets for image classification of the proposed adaptive methods with the previous adaptive method. The numerical

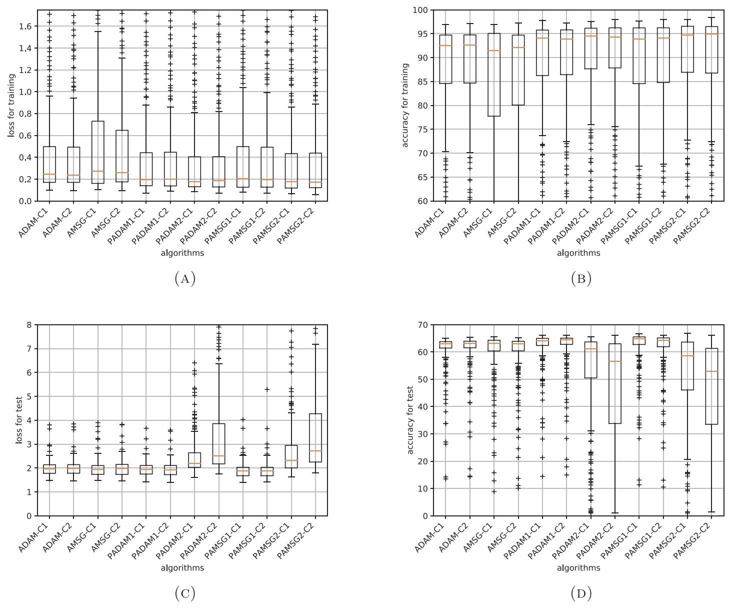


FIGURE 15. Box-plot comparisons of (a) training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with constant learning rates on the CIFAR-100 dataset

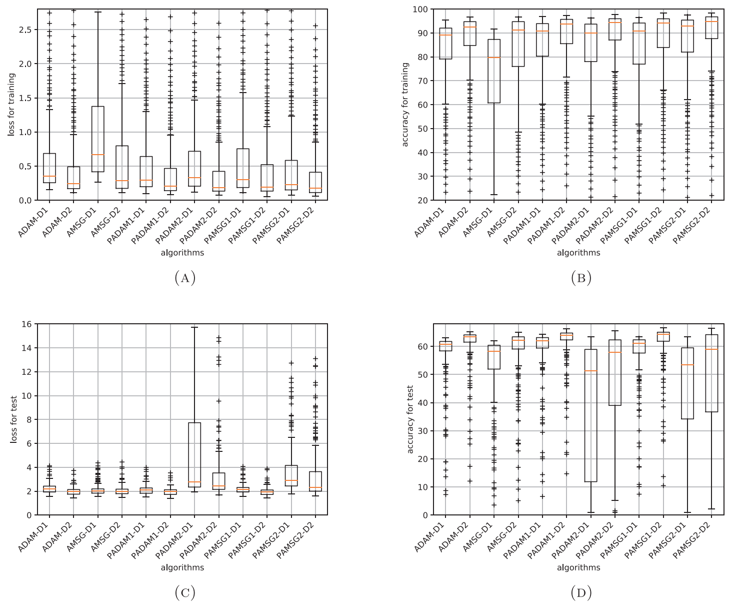


FIGURE 16. Box-plot comparisons of (a) training loss function value, (b) training classification accuracy score, (c) test loss function value, and (d) test classification accuracy score for algorithms with diminishing learning rates on the CIFAR-100 dataset

comparisons showed that the proposed adaptive method using the element-wise third or fourth power of the stochastic gradient performed better than the previous adaptive methods. In particular, it minimized the training loss functions faster and had higher training accuracies compared with the previous methods.

In this study, we applied the proposed and existing adaptive methods to image classification. In the future, we will compare their performances in other classification tasks.

References

[1] M. Arjovsky, S. Chintala and L. Bottou, *Wasserstein GAN,* `https://arxiv.org/pdf/1701.07875.pdf`*, 2017.*

[2] J. Chen, D. Zhou, Y. Tang, Z. Yang, Y. Cao and Q. Gu, Quanquan, *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, International Joint Conferences on Artificial Intelligence Organization, 2020, pp. 3267–3275.

[3] X. Chen, S. Liu, R. Sun and M. Hong, *On the convergence of a class of Adam-type algorithms for non-convex optimization*, in: Proceedings of The International Conference on Learning Representations, 2019.

[4] A.-K. Dombrowski, C. J. Anders, K.-R. Müller and P. Kessel, *Towards robust explanations for deep neural networks*, Pattern Recognition **121** (2022): 108194.

[5] J. Duchi, E. Hazan and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, Journal of Machine Learning Research **12** (2011), 2121–2159.

[6] H. Iiduka, *Appropriate learning rates of adaptive learning rate optimization algorithms for training deep neural networks*, IEEE Transactions on Cybernetics, (to appear), 2021.

[7] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in: Proceedings of The International Conference on Learning Representations, 2015, pp. 1–15.

[8] G. Montavon, S. Lapuschkin, A. Binde, W. Samek and K.-R. Müller, *Explaining nonlinear classification decisions with deep Taylor decomposition*, Pattern Recognition **65** (2017), 211–222.

[9] S. Reddi, J. Kale and and S. Kumar, *On the convergence of Adam and beyond*, in: Proceedings of The International Conference on Learning Representations, 2018, pp. 1–23.

[10] H. Robbins and H. Monro, *A stochastic approximation method*, The Annals of Mathematical Statistics **22** (1951), 400–407.

[11] T. Tieleman and G. Hinton, *RMSProp: Divide the gradient by a running average of its recent magnitude*, COURSERA: Neural Networks for Machine Learning **4** (2012), 26–31.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Advances in Neural Information Processing Systems*, in: Attention is All you Need, vol. 30, 2017, pp. 5998–6008.

[13] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel and Y. Bengio, *Show, attend and tell: Neural image caption generation with visual attention*, in: Proceedings of the 32nd International Conference on Machine Learning, vol. 37, 2015, pp. 2048–2057.

[14] S. Zamboni, Z. T. Kefato, S. Girdzijauskas, C. Norén and L. Dal Col, *Pedestrian trajectory prediction with convolutional neural networks*, Pattern Recognition **121** (2022): 108252.

[15] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris and J. S. Duncan, *AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients*, in: Advances in Neural Information Processing Systems, 2020.

Manuscript received 19 January 2022
revised 25 January 2022

Kanako Shimoyama
Computer Science Course, Graduate School of Science and Technology, Meiji University, Kanagawa 214-8571, Japan
E-mail address: kanako@cs.meiji.ac.jp

Hideaki Iiduka
Department of Computer Science, Graduate School of Science and Technology, Meiji University, Kanagawa 214-8571, Japan
E-mail address: iiduka@cs.meiji.ac.jp