IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 7, 2019, accepted May 24, 2019, date of publication May 30, 2019, date of current version June 13, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2919987*

# Effective Neural Network Training With a New Weighting Mechanism-Based Optimization Algorithm

**YUNLONG YU AND FUXIAN LIU**

Department of Air Defense and Antimissile, Air Force Engineering University, Xi'an 710051, China

Corresponding author: Yunlong Yu (yuyunlong426@126.com)

This work was supported by the National Natural Science Foundation of China under Grant 71771216 and Grant 71701209.

**ABSTRACT** First-order gradient-based optimization algorithms have been of core practical importance in the field of deep learning. In this paper, we propose a new weighting mechanism-based first-order gradient descent optimization algorithm, namely NWM-Adam, to resolve the undesirable convergence behavior of some optimization algorithms which employ fixed sized window of past gradients to scale the gradient updates and improve the performance of Adam and AMSGrad. The NWM-Adam is developed on the basis of the idea, i.e., placing more memory of the past gradients than the recent gradients. Furthermore, it can easily adjust the degree to which how much the past gradients weigh in the estimation. In order to empirically test the performance of our proposed NWM-Adam optimization algorithm, we compare it with other popular optimization algorithms in three well-known machine learning models, i.e., logistic regression, multi-layer fully connected neural networks, and deep convolutional neural networks. The experimental results show that the NWM-Adam can outperform other optimization algorithms.

**INDEX TERMS** Deep learning, optimization algorithm, learning rate, neural network training.

## I. INTRODUCTION

Recently, deep learning has become a significant part of information science research [1]–[4]. And deep learning has achieved outstanding performance in many fields, such as image classification [5]–[7], action recognition [8], [9], image captioning [10], and target localization [11]–[13]. The performance of a deep neural network is mainly determined by the structure of its model and its corresponding optimization algorithm. Therefore, a good optimization algorithm can improve the performance of the deep neural network under circumstance of fixed network architecture.

At present, first-order based optimization algorithms play an important role in deep learning on account of their efficiency and effectiveness in dealing with large-scale optimization problems [14]. In these algorithms, stochastic gradient descent (SGD) [15]–[19] is the representative method to train deep neural networks, in which it iteratively moves in the direction of the negative gradient to update parameters until convergence. Successively, some variants of SGD were proposed, which can automatically adjust the learning rate by using the square root operation of some form of averaging of the squared elements in the past gradients. Adagrad [20], as the first variant, works well for dealing with sparse data. However, this method uses all the past gradients, which can result in fast shrinkage of the learning rate. In order to resolve this problem, some algorithms, e.g., Adadelta [21], RMSprop [22], and Adam [23], were proposed by using the exponential moving average of past squared gradients. Although these optimization methods have achieved good performance in many applications, they can lead to undesirable convergence behavior. AMSGrad [24] was proposed to guarantee convergence by employing the idea, i.e., long-term memory of the past gradients. However, when the new gradients oscillate, AMSGrad may lead to the undesirable estimation. The defects of these existing methods motivate us to find a new way to resolve the problems and further improve the optimization performance.

In this paper, we present a new first-order gradient descent optimization algorithm that includes a more flexible weighting mechanism, which can resolve the undesirable convergence behavior and improve the performance of Adam and AMSGrad. Unlike most of methods, our proposed new weighting mechanism uses a dynamic exponential decay rate

The associate editor coordinating the review of this manuscript and approving it for publication was Qiang Lai.

for second moment estimate instead of a preconfigured and fixed one. In addition, our method can easily adjust the degree to which how much the past gradients weigh in the estimation. The proposed new exponential moving average variant is developed on the basis of the idea, i.e., putting more memory of the past gradients than the recent gradients. The analysis indicates that our method can guarantee convergence, at the same time, the experimental results show that our method can further improve the performance. In the sequel, our method will be referred to as NWM-Adam.

The remainder of this paper is arranged as follows. In Section II, we will present the related works including the most common optimization algorithms. Subsequently, in Section III, we will describe our proposed NWM-Adam algorithm. Next, we provide the convergence analysis of our method in Section IV. Afterwards, in Section V, we will give the description of the experimental design and discusses the results. Finally, in Section VI, we will conclude our work and discuss some future works.

## II. RELATED WORK

Optimization problem is one of the most important research directions in computational mathematics [25]. In the field of deep learning, the selection of optimization algorithms remains top priority in one model. Even if the data sets and model architectures are identical, different optimization algorithms are likely to result in dramatically different training effects. In this section, we will outline some optimization algorithms that are widely used in deep learning.

Gradient descent method minimizes the objective function $J(\theta)$ through updating the model's parameters $\theta \in \mathbb{R}^d$ in the direction of the negative gradient of the objective function $-\nabla_\theta J(\theta)$ with regard to the parameters $\theta$. The updating step in every moment is determined by the learning rate $\eta$.

Batch gradient descent, i.e., vanilla gradient descent, employs the whole training dataset to compute the gradient of the objective function with regard to the parameters $\theta$. However, this method conducts only one update, which cannot update the parameters online. SGD uses each training example to conduct a parameter update, which could be used to learn online. Of course, frequent update of SGD may result in heavy fluctuation of the objective function. This phenomenon is likely to prevent SGD from converging to the exact minimum. Mini-batch gradient descent is the typical method to train neural networks today, in which every mini-batch of training dataset is processed to perform an update.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \qquad \text{(Batch)} \tag{1}$$

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \qquad \text{(Stochastic)} \tag{2}$$

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \qquad \text{(Mini} - \text{batch)} \tag{3}$$

SGD is prone to oscillations when meeting ravines [18]. Therefore, momentum [26] is employed, which can speed up SGD in the correct direction and restrain oscillations.

$$m_t = \gamma m_{t-1} - \eta \cdot \nabla_\theta J(\theta) \tag{4}$$

$$\theta = \theta - m_t \tag{5}$$

On the basis of the original step size, SGD-Momentum adopts the past time step $\gamma m_{t-1}$, in which the coefficient $\gamma$ is usually around 0.9. In this algorithm, the direction of parameters update is not only determined by the current gradient, but also related to the previous descent direction. When using momentum in SGD, the dimensions whose gradient directions are similar will get faster updates, at the same time, the dimensions whose gradient directions have large variations will gain small updates. Therefore, we can speed up convergence and reduce oscillations by using this method.

Deep learning models often involve lots of parameters. Different types of parameters have different update frequency, which means larger update steps for infrequently updated parameters and smaller steps for frequently updated parameters. Adagrad [20] is an algorithm which can achieve this effect. This algorithm introduces the second moment.

$$G_t = diag(\sum_{i=1}^{t} g_{i,1}^2, \sum_{i=1}^{t} g_{i,2}^2, \cdots, \sum_{i=1}^{t} g_{i,d}^2) \tag{6}$$

where $G_t \in \mathbb{R}^{d \times d}$ is the diagonal matrix, in which its element $G_{t,ii}$ is the sum of squared gradients with respect to $\theta_i$ from the initial moment to time $t$. The update for every parameter $\theta_i$ at each time step $t$ is:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\eta}{\sqrt{G_{t-1,ii} + \varepsilon}} \cdot g_{t-1,i} \tag{7}$$

where $\varepsilon$ is usually set to $1e-8$, which can avoid division by zero. $g_{t-1,i}$ is the gradient of the objective function with regard to the parameter $\theta_i$ at time step $t-1$:

$$g_{t-1,i} = \nabla_\theta J(\theta_i) \tag{8}$$

Adagrad can help us adapt the learning rate to the parameters, which has no need of manually tuning the learning rate. However, the continual accumulation of the squared gradients in the denominator will result in the learning rate shrinking. In the end, the learning rate will become especially small, which means that we cannot get any knowledge.

Adadelta [21] modifies the calculation method of the second moment, which resolves the phenomenon that the learning rate of Adagrad changes aggressively. With Adadelta, we implement the accumulation as an exponentially decaying average of past squared gradients. The update rule of Adadelta is:

$$\Delta\theta_{t-1} = -\frac{\sqrt{E\left[\Delta\theta^2\right]_{t-2} + \varepsilon}}{\sqrt{E\left[g^2\right]_{t-1} + \varepsilon}} g_{t-1} \tag{9}$$

$$\theta_t = \theta_{t-1} + \Delta\theta_{t-1} \tag{10}$$

$$E\left[g^2\right]_{t-1} = \gamma E\left[g^2\right]_{t-2} + (1-\gamma) g_{t-1}^2 \tag{11}$$

$$E\left[\Delta\theta^2\right]_{t-2} = \gamma E\left[\Delta\theta^2\right]_{t-3} + (1-\gamma)\Delta\theta_{t-2}^2 \tag{12}$$

RMSprop [22] also confines the window of accumulated past squared gradients to some fixed size instead of accumulating all past squared gradients, which is similar to Adadelta.

This algorithm divides the learning rate by an exponentially decaying average of past squared gradients.

$$E\left[g^2\right]_{t-1} = 0.9E\left[g^2\right]_{t-2} + 0.1g_{t-1}^2 \tag{13}$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{E\left[g^2\right]_{t-1} + \varepsilon}} g_{t-1} \tag{14}$$

Adam [23] can be considered as the combination of RMSprop and momentum. In addition to using an exponentially decaying average of past squared gradients like RMSprop, Adam as well applies an exponentially decaying average of past gradients like momentum.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{15}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{16}$$

where $\beta_1$ and $\beta_2$ are usually set to 0.9 and 0.999, respectively. During the initial stage, $m_t$ and $v_t$ are easily biased towards starting value. Therefore, bias-correction should be computed for the first and second moment.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{17}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{18}$$

Finally, the update rule of Adam is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \tag{19}$$

Zhang *et al.* [27] designed an adaptive exponential decay rate $\beta_t$ for Adam, i.e., AEDR-Adam, in which this exponential decay rate for the first moment estimates and the second raw moment estimates to increase when a large step is taken and to decrease when a small step is taken.

$$\beta_t = 1 - \frac{1}{\iota_t} \tag{20}$$

$$\iota_t = (1 - \frac{m_{t-1}}{\sqrt{v_{t-1}}})\iota_{t-1} + 1 \tag{21}$$

Reddi *et al.* [24] pointed out the problem which exists in the proof of convergence of the Adam algorithm. At the same time, the authors showed an example of convex optimization problem, in which the Adam algorithm cannot converge to an optimal solution. In order to resolve this issue, the authors proposed to modify the Adam algorithm where the idea of ''long-term memory'' of past gradients is employed. The modified Adam is termed as AMSGrad. The main difference of AMSGrad with Adam is that AMSGrad uses the ''max'' operation to all $v_t$ until the present time step and employs this maximum value to the running average of the gradient instead of using original $v_t$ in Adam.

## III. NWM-ADAM

In this section, we describe our proposed NWM-Adam algorithm. The proposed NWM-Adam algorithm uses a dynamic exponential decay rate for second moment estimate instead of a preconfigured and fixed one. Firstly, we provide an universal update rule which can cover many gradient descent optimization algorithms.

$$\theta_t = \theta_{t-1} - \frac{\eta_{t-1}}{\sqrt{V_{t-1}}} m_{t-1} \tag{22}$$

$$m_{t-1} = \phi_{t-1}(\nabla J_1(\theta_1), \ldots, \nabla J_{t-1}(\theta_{t-1})) \tag{23}$$

$$V_{t-1} = \psi_{t-1}(\nabla J_1(\theta_1), \ldots, \nabla J_{t-1}(\theta_{t-1})) \tag{24}$$

where $\phi_{t-1}$ and $\psi_{t-1}$ denote the averaging functions, $\eta_{t-1}$ is the step size, and $\frac{\eta_{t-1}}{\sqrt{V_{t-1}}}$ is the learning rate.

Reddi *et al.* [24] proved that the Adam algorithm cannot converge to an optimal solution even in the simple one-dimensional convex settings, which refutes the convergence of Adam given in [23]. In fact, this fundamental flaw also exists in some most-widely used exponential moving average methods, i.e., RMSprop, Adadelta. The main problem is related to the following quantity.

$$\Gamma_t = \frac{\sqrt{V_t}}{\eta_t} - \frac{\sqrt{V_{t-1}}}{\eta_{t-1}} \tag{25}$$

The above quantity denotes the change of the inverse of the learning rate with regard to each iteration. The update rules of SGD and Adagrad usually result in non-increasing learning rates. On the basis of the update rules of SGD and Adagrad in the previous section, we can observe that $\Gamma_t \succeq 0$ with respect to all $t \in [T]$. However, the positive semi-definiteness of the $\Gamma_t$ cannot be guaranteed with regard to the algorithms which employ exponential moving average to estimate the square of gradient, i.e., RMSprop, Adadelta and Adam. This problem will result in non-convergence.

In order to resolve the aforementioned convergence problem, Reddi *et al.* [24] proposed to employ an idea, i.e., long-term memory of past gradients, to the original Adam algorithm. In this paper, NWM-Adam, i.e., the proposed new exponential moving average variant, is also developed on the basis of this idea, which puts more memory of the past gradients than the recent gradients. In other words, our method employs larger weights on the past gradients than the recent gradients. Our proposed NWM-Adam can make the quantity $\Gamma_t$ positive semi-definite and guarantee convergence, at the same time, our algorithm can further improve the performance of Adam and AMSGrad.

The following presents the details of our proposed NWM-Adam. Eqs. 22, 23 and 24 are the main component of our proposed algorithm. Suppose $m_t$ is the first moment estimate of gradient and $v_t$ is the second moment estimate of gradient. These two estimates are computed as follows.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{26}$$

$$v_t = \beta_{2,t} v_{t-1} + (1-\beta_{2,t}) g_t^2 \tag{27}$$

where $g_t$ denotes the gradient of the objective function $J$ in which cross entropy is used as the objective function. In this algorithm, $\beta_{2,t}$ changes over time step $t$, which is designed as follows.

$$\beta_{2,t} = \frac{(t^\lambda - 1)}{t^\lambda} \tag{28}$$

Therefore, NWM-Adam can easily adjust the degree to which how much the past gradients weigh in the estimation. The key difference of NWM-Adam from Adam is that it employs a changing $\beta_{2,t}$ for controlling the exponential decay rate of moving average of the squared gradient instead of constant $\beta_2$ in Adam. By doing this, our proposed NWM-Adam algorithm can realize that $\Gamma_t$ is positive semi-definite, which can resolve the convergence issue of Adam. The pseudo-code of our proposed algorithm NWM-Adam is presented in Algorithm 1.

**Algorithm 1** *NWM-Adam*, Our Proposed Algorithm for Stochastic Optimization. $g_t$ Denotes the Gradient and $g_t^2$ Represents the Element Wise Square $g_t \odot g_t$. $\varepsilon$ is to $10^{-8}$, Which Can Avoid the Denominator is Zero.

$\eta$: Step size

$\beta_1$: A hyper-parameter for controlling the exponential decay rate of moving average of the gradient $m_t$

$\beta_{2,t}$: A hyper-parameter for controlling the exponential decay rate of moving average of the squared gradient $v_t$

$m_0 \leftarrow 0$: Initialize $1^{st}$ moment vector

$v_0 \leftarrow 0$: Initialize $2^{nd}$ moment vector

**For** $t \in [1, 2, \ldots, T]$

**do**

Get gradients with regard to objective function at time step $t$

$g_t = \nabla_\theta J_t(\theta_t)$

Get the hyper-parameter $\beta_{2,t}$

$\beta_{2,t} = (t^\lambda - 1)/t^\lambda$

Update first moment estimate

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

Update second raw moment estimate

$v_t \leftarrow \beta_{2,t} v_{t-1} + (1 - \beta_{2,t}) g_t^2$

Update parameters

$\theta_{t+1} \leftarrow \theta_t - \eta m_t / (\sqrt{v_t} + \varepsilon)$

**End**

**Until** Meet the stopping criterion

Next, we give an explanation of the weighing scheme of the gradients in NWM-Adam. Let us revisit the Eq. 22, this equation can also be rewritten as follows.

$$\theta_t = \theta_{t-1} - \frac{\eta_{t-1}}{\sqrt{E\left[g^2\right]}} E\left[g\right] \tag{29}$$

With regard to the estimation of $E\left[g^2\right]$, methods using exponential moving average may make this estimation unstable when the gradient magnitude varies a lot from batch to batch. The weighing strategy in NWM-Adam can provide more stable estimation than AMSGrad and Adam. Namely, the weighing scheme of our proposed NWM-Adam algorithm can stabilize the estimation when the new gradients oscillate.

## IV. CONVERGENCE ANALYSIS

In this section, we give the convergence analysis of our proposed method. Suppose the unknown sequence of convex objective functions $J_1(\theta)$, $J_2(\theta)$, $\ldots$, $J_T(\theta)$. At each time step $t$, predicting the parameter $\theta_t$ and evaluating the objective function $J_t$ is our purpose. On account of the unknown sequence in advance, we employ the regret to evaluate our method, which is the summation of all previous difference value between the prediction $J_t(\theta_t)$ and the best value $J_t(\theta^*)$. Then, the defined regret is given as follows.

$$R(T) = \sum_{t=1}^{T} \left[J_t(\theta_t) - J_t(\theta^*)\right] \tag{30}$$

where $\theta^* = \arg\min \sum_{t=1}^{T} J_t(\theta)$. Assume that $\|\nabla J_t(\theta)\|_\infty \leq G_\infty$, and $\|\theta_n - \theta_m\|_\infty \leq D_\infty$ for all $t \in [T]$ and $\forall m, n \in 1, 2, \ldots, T$. At the same time, let $\eta_t = \eta/\sqrt{t}$ and $\beta_{1,t} \leq \beta_1$ for all $t \in [T]$. We employ $g_{1:t,i} = \left[g_{1,i}, g_{2,i}, \ldots, g_{t,i}\right]$ to represent a vector which is the $i^{th}$ dimension of the gradient sequence over all steps till time step $t$.

The changing schedule of $\beta_{2,t}$ satisfies the following two conditions:

(1) $\eta_{t-1}^{-1}\sqrt{v_{t-1,i}} \leq \eta_t^{-1}\sqrt{v_{t,i}}$, $t \in 2, 3, \ldots, T$;

(2) $\delta^{-1}(\sum_{p=1}^{t} g_{p,i}^2)^{0.5} \leq \eta_T^{-1}(\sum_{p=1}^{t}\prod_{q=1}^{t-p}\beta_{2,(t-q+1)}(1-\beta_{2,p})g_{p,i}^2)^{0.5}$, $t \in [T]$.

In order to show that our optimization algorithm has a regret bound, we start with the following:

$$\begin{aligned}\theta_{t+1} &= \prod_{F,\sqrt{V_t}} (\theta_t - \eta_t {V_t}^{-0.5} m_t) \\ &= \min_{x \in F} \left\| V_t^{0.25}(\theta - (\theta_t - \eta_t {V_t}^{-0.5} m_t)) \right\|\end{aligned} \tag{31}$$

Based on the theory in [28], we can obtain the following:

$$\begin{aligned}&\left\| V_t^{0.25}(\theta_{t+1} - \theta^*) \right\|^2 \\ &\leq \left\| V_t^{0.25}(\theta_t - \theta^*) \right\|^2 + \eta_t^2 \left\| V_t^{-0.25} m_t \right\|^2 \\ &\quad - 2\eta_t \left\langle \beta_{1,t} m_{t-1} + (1 - \beta_{1,t}) g_t, \theta_t - \theta^* \right\rangle\end{aligned} \tag{32}$$

Then, we can get the following:

$$\begin{aligned}&\left\langle g_t, \theta_t - \theta^* \right\rangle \\ &\leq 0.5\eta_t (1 - \beta_{1,t})^{-1} \left\| V_t^{-0.25} m_t \right\|^2 \\ &\quad + 0.5\eta_t^{-1}(1 - \beta_{1,t})^{-1} [\left\| V_t^{0.25}(\theta_t - \theta^*) \right\|^2 \\ &\quad - \left\| V_t^{0.25}(\theta_{t+1} - \theta^*) \right\|^2] \\ &\quad + 0.5\beta_{1,t}(1 - \beta_{1,t})^{-1}\eta_t^{-1} \left\| V_t^{0.25}(\theta_t - \theta^*) \right\|^2 \\ &\quad + 0.5\beta_{1,t}(1 - \beta_{1,t})^{-1}\eta_t \left\| V_t^{-0.25} m_{t-1} \right\|^2\end{aligned} \tag{33}$$

$$\begin{aligned}&\sum_{t=1}^{T} J_t(\theta_t) - J_t(\theta^*) \\ &\leq \sum_{t=1}^{T} \left\langle g_t, \theta_t - \theta^* \right\rangle\end{aligned}$$

$$\leq \sum_{t=1}^{T}[0.5\eta_t(1-\beta_{1,t})^{-1}\left\|V_t^{-0.25}m_t\right\|^2 + 0.5\eta_t^{-1}(1-\beta_{1,t})^{-1}[\left\|V_t^{0.25}(\theta_t-\theta^*)\right\|^2 - \left\|V_t^{0.25}(\theta_{t+1}-\theta^*)\right\|^2] + 0.5\beta_{1,t}(1-\beta_{1,t})^{-1}\eta_t^{-1}\left\|V_t^{0.25}(\theta_t-\theta^*)\right\|^2 + 0.5\beta_{1,t}(1-\beta_{1,t})^{-1}\eta_t\left\|V_t^{-0.25}m_{t-1}\right\|^2] \quad (34)$$

We need some intermediate results for further bounding this inequality.

$$\sum_{t=1}^{T}\eta_t\left\|V_t^{-0.25}m_t\right\|^2 = \sum_{t=1}^{T-1}\eta_t\left\|V_t^{-0.25}m_t\right\|^2 + \eta_T\sum_{i=1}^{d}m_{T,i}^2(v_{T,i})^{-0.5} \quad (35)$$

Then, we can get the following inequality.

$$\sum_{t=1}^{T}\eta_t\left\|V_t^{-0.25}m_t\right\|^2 \leq \sum_{t=1}^{T-1}\eta_t\left\|V_t^{-0.25}m_t\right\|^2 + \delta(1-\beta_1)^{-1}\sum_{i=1}^{d}\sum_{j=1}^{T}\beta_1^{T-j}g_{j,i}^2(\sum_{j=1}^{T}g_{j,i}^2)^{-0.5} \leq 2\delta(1-\beta_1)^{-2}\sum_{i=1}^{d}\left\|g_{1:T,i}\right\|_2 \quad (36)$$

Next, we can get the following:

$$\sum_{t=1}^{T}J_t(\theta_t)-J_t(\theta^*) \leq 2\delta(1-\beta_1)^{-3}\sum_{i=1}^{d}\left\|g_{1:T,i}\right\|_2 + 0.5\eta_1^{-1}(1-\beta_1)^{-1}\sum_{i=1}^{d}\sqrt{v_{1,i}}(\theta_{1,i}-\theta_i^*)^2 + 0.5(1-\beta_1)^{-1}\sum_{t=2}^{T}\sum_{i=1}^{d}(\theta_{t,i}-\theta_i^*)^2[\eta_t^{-1}\sqrt{v_{t,i}} - \eta_{t-1}^{-1}\sqrt{v_{t-1,i}}] + 0.5(1-\beta_1)^{-1}\sum_{t=1}^{T}\sum_{i=1}^{d}\beta_{1,t}(\theta_{t,i}-\theta_i^*)^2\eta_t^{-1}\sqrt{v_{t,i}} \quad (37)$$

When $\eta_t^{-1}\sqrt{v_{t,i}} - \eta_{t-1}^{-1}\sqrt{v_{t-1,i}}$ is positive semi-definite, we can get the following regret bound for the sequence $\theta_t$ generated using our proposed NWM-Adam algorithm.

$$R(T) \leq 2(1-\beta_1)^{-3}\delta\sum_{i=1}^{d}\left\|g_{1:T,i}\right\|_2 + \frac{1}{2}D_\infty^2\sqrt{T}\eta^{-1}(1-\beta_1)^{-1}\sum_{i=1}^{d}\sqrt{v_{T,i}} + \frac{1}{2}D_\infty^2(1-\beta_1)^{-1}\sum_{t=1}^{T}\sum_{i=1}^{d}\eta_t^{-1}\beta_{1,t}\sqrt{v_{t,i}} \quad (38)$$

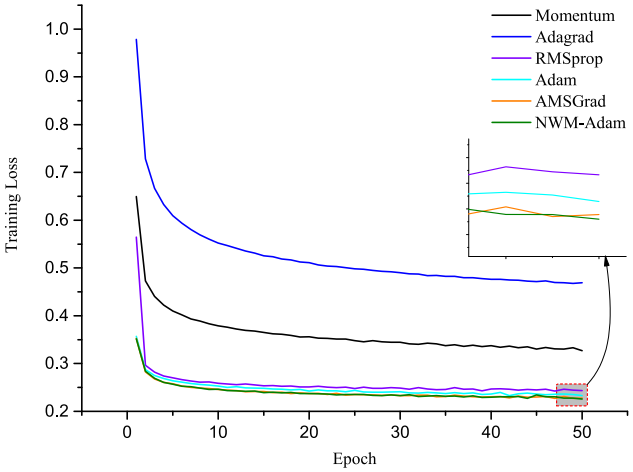


**FIGURE 1.** The training loss of logistic regression using different optimization algorithms.

## V. EXPERIMENTS

In this section, in order to empirically evaluate the effectiveness of the proposed NWM-Adam algorithm, we evaluate the proposed algorithm in three different popular machine learning models, i.e., logistic regression, multi-layer fully connected neural networks, and deep convolutional neural networks. These different models are tested on the MNIST dataset [29] and the CIFAR-10 dataset [30]. By employing these models and datasets, we demonstrate that NWM-Adam can effectively work out the practical deep learning problems.

The experiments in this part compare our proposed NWM-Adam algorithm with other popular gradient descent optimization algorithms, i.e., Momentum, Adagrad, RMSprop, Adam, and AMSGrad. Same parameter initialization is utilized when comparing with different optimization algorithms and the results are showed using the best hyper-parameters. In this section, we first introduce the details of the utilized datasets. Then, we give the experimental results and analyze the performance of our proposed method.

### A. DESCRIPTION OF THE UTILIZED DATASETS

#### 1) MNIST

MNIST is a large database of handwritten digits that is widely used for training and testing in the field of machine learning. This dataset was taken from American Census Bureau employees and American high school students. Its training set contains 60000 grayscale images, and testing set has 10000 grayscale images. These handwritten digits have been preprocessed, resized and centered, and the size of the images is fixed at $28 \times 28$. These images are divided into 10 classes, from 0 to 9.

#### 2) CIFAR-10

CIFAR-10 is a color image dataset which is closer to universal objects. This dataset was established by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It contains 10 classes, including airplane, automobile, bird, car, deer, dog, frog, horse, ship, and truck. This dataset has 50000 $32 \times 32$ training

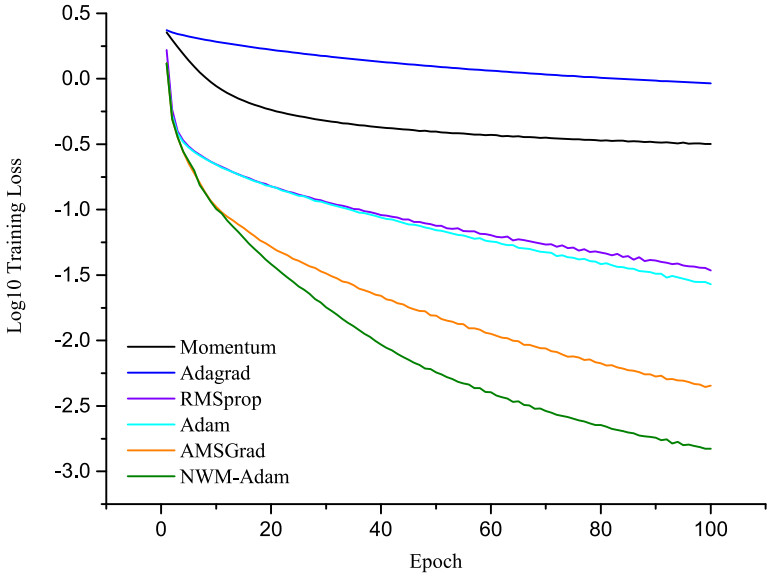


**FIGURE 2.** The training loss of fully connected neural networks (100 hidden units) using different optimization algorithms.

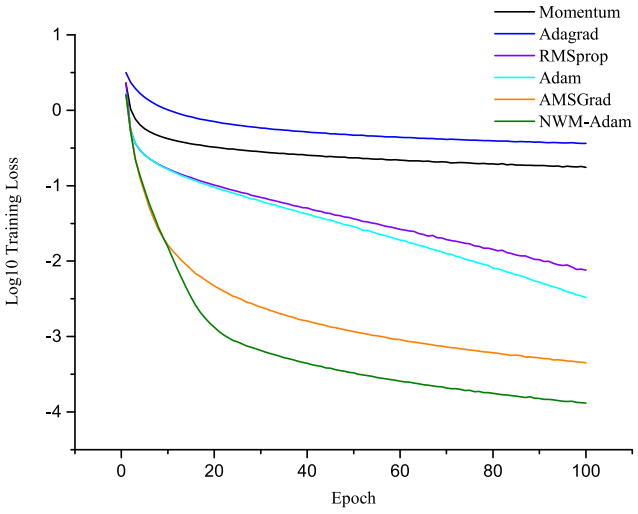


**FIGURE 3.** The training loss of fully connected neural networks (1024-512 hidden units) using different optimization algorithms.

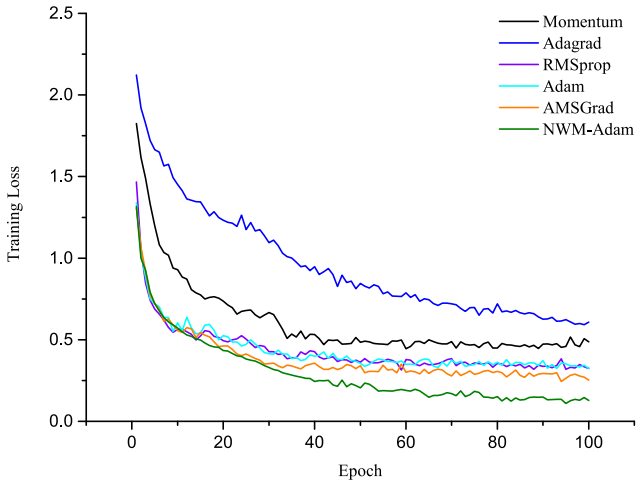


**FIGURE 4.** The training loss of convolutional neural networks (20-layer ResNet) using different optimization algorithms.

images and 10000 test images. Compared with handwritten digits, CIFAR-10 contains real objects in the real world, which is not only noisy, but also has different proportions and features, which makes it very difficult to recognize.

### B. LOGISTIC REGRESSION

For the first experiment, we evaluate our proposed algorithm on logistic regression problem, which can investigate the performance of the algorithm on convex problems. We perform this experiment using the MNIST dataset and compare the result with some popular gradient descent optimization algorithms, where the logistic regression problem gets one of the 10 class labels directly on the 784 dimension image vectors. The step size in this experiment is adjusted by $1/\sqrt{t}$ decay. The minibatch size is set to 128. We report the training loss with respect to epochs in Fig. 1. From the Fig. 1 we can observe that our proposed NWM-Adam algorithm yields similar convergence as AMSGrad and both perform better than Adam, RMSprop, Adagrad and Momentum. We also report the classification accuracy over ten rounds in Table 1. From the Table 1, we can see that our NWM-Adam is slightly better than AMSGrad.

### C. MULTI-LAYER FULLY CONNECTED NEURAL NETWORKS

Multi-layer fully connected neural network is the powerful model with non-convex objective function. In this experiment, we use two neural network models to test the effectiveness of our proposed method in the multiclass classification problem on MNIST dataset.

The first neural network model has one fully connected hidden layer with 100 hidden units. The second neural network model has two fully connected hidden layers with

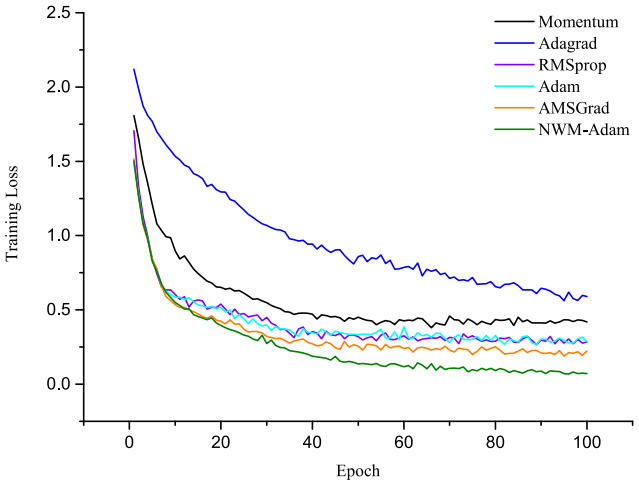


**FIGURE 5.** The training loss of convolutional neural networks (110-layer ResNet) using different optimization algorithms.

**TABLE 1.** Classification accuracies (Mean ± Std) of logistic regression using different optimization algorithms.

| Optimization method | Test accuracy (%) |
|---|---|
| Momentum [26] | $91.44 \pm 0.02$ |
| Adagrad [20] | $89.07 \pm 0.02$ |
| RMSprop [22] | $92.80 \pm 0.04$ |
| Adam [23] | $92.88 \pm 0.05$ |
| AMSGrad [24] | $92.92 \pm 0.03$ |
| NWM-Adam | $92.95 \pm 0.02$ |

1024 hidden units in the first hidden layer and 512 hidden units in the second hidden layer. ReLU is used as the activation function. Softmax is used as a classifier acting on the outputs of the last hidden layer. Furthermore, constant step size is utilized throughout this experiment.

The training loss curves of these two fully connected neural network models by using different optimization algorithms are shown in Fig. 2 and Fig. 3. In order to compare clearly, we use the base-10 logarithm to plot the training loss. Fig. 2 and Fig. 3 all show that NWM-Adam algorithm outperforms other optimization algorithms, i.e., Momentum, Adagrad, RMSprop, Adam, and AMSGrad.

The classification accuracy over ten rounds is presented in Table 2. From the table, we observe that our method outperforms all the other methods in terms of classification accuracy. Furthermore, the good performance of NWM-Adam is relatively stable.

The good performance of our algorithm benefits from proper utilization of the past gradients. Traditional gradient descent optimization algorithms all use a fixed exponential decay rate for second moment estimate, while our algorithm employs a dynamic exponential decay rate for this estimate. Furthermore, our algorithm can easily adjust the degree to which how much the past gradients weigh in the estimation. This is the reason why our algorithm is superior to others.

**TABLE 2.** Classification accuracies (Mean ± Std) of multi-layer fully connected neural networks using different optimization algorithms.

| Optimization method | Test accuracy (%) | |
|---|---|---|
| | 100 hidden units | 1024-512 hidden units |
| Momentum [26] | $91.41 \pm 0.12$ | $94.52 \pm 0.10$ |
| Adagrad [20] | $81.67 \pm 0.62$ | $89.57 \pm 0.17$ |
| RMSprop [22] | $97.61 \pm 0.05$ | $97.85 \pm 0.08$ |
| Adam [23] | $97.71 \pm 0.09$ | $97.96 \pm 0.07$ |
| AMSGrad [24] | $97.83 \pm 0.08$ | $98.24 \pm 0.04$ |
| NWM-Adam | $98.02 \pm 0.06$ | $98.81 \pm 0.02$ |

**TABLE 3.** Classification accuracies (Mean ± Std) of deep convolutional neural networks using different optimization algorithms.

| Optimization method | Test accuracy (%) | |
|---|---|---|
| | 20-layer ResNet | 110-layer ResNet |
| Momentum [26] | $80.25 \pm 0.13$ | $81.69 \pm 0.15$ |
| Adagrad [20] | $74.94 \pm 0.16$ | $75.45 \pm 0.20$ |
| RMSprop [22] | $85.00 \pm 0.22$ | $86.94 \pm 0.21$ |
| Adam [23] | $85.80 \pm 0.18$ | $87.23 \pm 0.19$ |
| AMSGrad [24] | $87.26 \pm 0.16$ | $88.89 \pm 0.15$ |
| NWM-Adam | $88.60 \pm 0.15$ | $90.77 \pm 0.14$ |

### D. DEEP CONVOLUTIONAL NEURAL NETWORKS

In the last experiment, we use deep convolutional neural network (DCNN) to evaluate our method on the standard CIFAR-10 dataset. The residual architecture is used as the DCNN model in this part. In this architecture, the first layer is $3 \times 3$ convolutions. Then, it includes a stack of $3m$ residual blocks with an equal number of blocks for each feature map size. Finally, this architecture ends with a global average pooling, a 10-way fully connected layer, and softmax.

In this experiment, we use the 20-layer ResNet ($m = 6$) and the 110-layer ResNet ($m = 36$). The minibatch size is also set to 128, which is similar to previous experiments. Furthermore, we employ constant step size throughout this experiment.

The training loss curves for this problem are shown in Fig. 4 and Fig. 5. We can see that NWM-Adam performs considerably better than other optimization methods, which is in line with the conclusion of the experiment of multi-layer fully connected neural network. At the same time, we also provide the classification accuracy over ten rounds using different optimization methods, which is shown in Table 3. From the table, we can see that our optimization method can help the neural network further improve its classification accuracy, which demonstrates once again the superiority of our proposed optimization method.

## VI. CONCLUSION

In this paper, our method NWM-Adam has been designed as a more flexible machine learning first-order gradient descent optimization algorithm to resolve the undesirable convergence behavior of some optimization algorithms which

employ fixed sized window of past gradients to scale the gradient updates and improve the performance of Adam and AMSGrad. The idea of our NWM-Adam algorithm is that placing more memory of the past gradients than the recent gradients. In addition, our optimization algorithm can easily control the degree to which how much the past gradients weigh in the estimation. The experimental results have demonstrated that our NWM-Adam optimization algorithm performs better than other popular gradient descent optimization algorithms on some convex and non-convex problems in the field of machine learning. The optimization idea in this paper can also be adopted to other kind of neural networks, like the memristor-based neural networks [31] and discontinuous neural networks [32]. In future work, we plan to develop an improved method which can implement updates for each weight.

## REFERENCES

[1] J. Kim, J. Kim, G.-J. Jang, and M. Lee, "Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection," *Neural Netw.*, vol. 87, pp. 109–121, Mar. 2017.
[2] J. Qiao, G. Wang, W. Li, and M. Chen, "An adaptive deep Q-learning strategy for handwritten digit recognition," *Neural Netw.*, vol. 107, pp. 61–71, Nov. 2018.
[3] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, "Deep learning based inference of private information using embedded sensors in smart devices," *IEEE Netw.*, vol. 32, no. 4, pp. 8–14, Jul./Aug. 2018.
[4] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018.
[5] B. Pan, Z. Shi, and X. Xu, "MugNet: Deep learning for hyperspectral image classification using limited samples," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 108–119, Nov. 2018.
[6] Z. Zhong, J. Li, Z. Luo, and M. Chapman, "Spectral—Spatial residual network for hyperspectral image classification: A 3-D deep learning framework," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 847–858, Feb. 2018.
[7] Y. Yu and F. Liu, "Dense connectivity based two-stream deep feature fusion framework for aerial scene classification," *Remote Sens.*, vol. 10, no. 7, p. 1158, Jul. 2018.
[8] H. Rahmani, A. Mian, and M. Shah, "Learning a deep model for human action recognition from novel viewpoints," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 667–681, Mar. 2018.
[9] D. C. Luvizon, D. Picard, and H. Tabia, "2D/3D pose estimation and action recognition using multitask deep learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2018, pp. 5137–5146.
[10] C. Wang, H. Yang, and C. Meinel, "Image captioning with deep bidirectional lstms and multi-task learning," *ACM Trans. Multimedia Comput., Commun., Appl. (TOMM)*, vol. 14, no. 2, p. 40, 2018.
[11] J. Gao, T. Zhang, X. Yang, and C. Xu, "P2T: Part-to-target tracking via deep regression learning," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 3074–3086, Jun. 2018.
[12] W. Huang, J. Zeng, P. Zhang, G. Chen, and H. Ding, "Single-target localization in video sequences using offline deep-ranked metric learning and Online learned models updating," *Multimedia Tools Appl.*, vol. 77, no. 21, pp. 28539–28565, 2018.
[13] G. Campanella, V. W. K. Silva, and T. J. Fuchs, "Terabyte-scale deep multiple instance learning for classification and localization in pathology," May 2018, *arXiv:1805.06983*. [Online]. Available: https://arxiv.org/abs/1805.06983
[14] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016, *arXiv:1609.04747*. [Online]. Available: https://arxiv.org/abs/1609.04747
[15] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2014, pp. 2933–2941.
[16] H. Robbins and S. Monro, "A stochastic approximation method," in *Herbert Robbins and Sutton Monro*. Berlin, Germany: Springer, 1985, pp. 102–109.
[17] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Proc. IEEE Neural Netw. Signal Process. II. Workshop*, Aug./Sep. 1992, pp. 3–12.
[18] R. Sutton, "Two problems with back propagation and other steepest descent learning procedures for networks," in *Proc. 8th Annu. Conf. Cogn. Sci. Soc.*, 1986, pp. 823–832.
[19] L. Bottou, "Stochastic gradient learning in neural networks," *Proc. Neuro-Nîmes*, vol. 91, no. 8, p. 12, 1991.
[20] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.
[21] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," Dec. 2012, *arXiv:1212.5701*. [Online]. Available: https://arxiv.org/abs/1212.5701
[22] T. Tieleman and G. Hinton, "Lecture 6.5-RMSPROP: Divide the gradient by a running average of its recent magnitude," *COURSERA, Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.
[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[24] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–23. [Online]. Available: https://openreview.net/forum?id=ryQu7f-RZ
[25] A. Antoniou and W.-S. Lu, *The Optimization Problem*. Berlin, Germany: Springer, 2007.
[26] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999.
[27] J. Zhang, F. Hu, L. Li, X. Xu, Z. Yang, and Y. Chen, "An adaptive mechanism to achieve learning rate dynamically," in *Neural Computing and Applications*. New York, NY, USA: Springer, 2018, pp. 1–14.
[28] H. B. McMahan and M. Streeter, "Adaptive bound optimization for online convex optimization," Feb. 2010, *arXiv:1002.4908*. [Online]. Available: https://arxiv.org/abs/1002.4908
[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[30] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.
[31] L. Wang, Z. Zeng, M.-F. Ge, and J. Hu, "Global stabilization analysis of inertial memristive recurrent neural networks with discrete and distributed delays," *Neural Netw.*, vol. 105, pp. 65–74, Sep. 2018.
[32] L. Wang, Z. Zeng, J. Hu, and X. Wang, "Controller design for global fixed-time synchronization of delayed neural networks with discontinuous activations," *Neural Netw.*, vol. 87, pp. 122–131, Mar. 2017.

**YUNLONG YU** received the bachelor's and master's degrees from Air Force Engineering University, Xi'an, China, in 2014 and 2017, respectively, where he is currently pursuing the Ph.D. degree. His research interests include deep learning, pattern recognition, and computer vision.

**FUXIAN LIU** received the bachelor's degree from Lanzhou University, Lanzhou, China, in 1994, and the master's and Ph.D. degrees from Air Force Engineering University, Xi'an, in 1998 and 2001, respectively, where he is currently a Professor. His research interests include deep learning and pattern recognition.