

Article

An Improved Reacceleration Optimization Algorithm Based on the Momentum Method for Image Recognition

Haijing Sun ¹, Ying Cai ², Ran Tao ³, Yichuan Shao ^{1,*}, Lei Xing ⁴, Can Zhang ² and Qian Zhao ⁵¹ School of Intelligent Science and Engineering, Shenyang University, Shenyang 110044, China² School of Information Engineering, Shenyang University, Shenyang 110044, China; caiying990331@163.com (Y.C.)³ Shanghai Maruka Computer Information Technology Co., Ltd., Shanghai 200052, China⁴ School of Chemistry and Chemical Engineering, University of Surrey, Surrey GU2 7XH, UK⁵ School of Science, Shenyang University of Technology, Shenyang 110044, China

* Correspondence: shaoyichuan@syu.edu.cn; Tel.: +86-130-0245-7275

Abstract: The optimization algorithm plays a crucial role in image recognition by neural networks. However, it is challenging to accelerate the model's convergence and maintain high precision. As a commonly used stochastic gradient descent optimization algorithm, the momentum method requires many epochs to find the optimal parameters during model training. The velocity of its gradient descent depends solely on the historical gradients and is not subject to random fluctuations. To address this issue, an optimization algorithm to enhance the gradient descent velocity, i.e., the momentum reacceleration gradient descent (MRGD), is proposed. The algorithm utilizes the point division of the current momentum and the gradient relationship, multiplying it with the gradient. It can adjust the update rate and step size of the parameters based on the gradient descent state, so as to achieve faster convergence and higher precision in training the deep learning model. The effectiveness of this method is further proven by applying the reacceleration mechanism to the Adam optimizer, resulting in the MRGDAdam algorithm. We verify both algorithms using multiple image classification datasets, and the experimental results show that the proposed optimization algorithm enables the model to achieve higher recognition accuracy over a small number of training epochs, as well as speeding up model implementation. This study provides new ideas and expansions for future optimizer research.

Keywords: momentum acceleration; optimization algorithm; deep learning; image recognition; gradient descent algorithm

MSC: 68T07

Citation: Sun, H.; Cai, Y.; Tao, R.; Shao, Y.; Xing, L.; Zhang, C.; Zhao, Q. An Improved Reacceleration Optimization Algorithm Based on the Momentum Method for Image Recognition. *Mathematics* **2024**, *12*, 1759. <https://doi.org/10.3390/math12111759>

Academic Editor: Konstantin Kozlov

Received: 10 April 2024

Revised: 11 May 2024

Accepted: 30 May 2024

Published: 5 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Neural networks have achieved amazing feats in computer vision and other domains thanks to the rapid development of deep learning technologies. In the process of picture categorization, increasingly intricate neural network models are used one after the other in an attempt to increase recognition accuracy. Correspondingly, training speed tends to decrease with increasing model parameter complexity. Therefore, it is essential to select an optimization algorithm that can expedite the convergence speed of the model and ensure high classification accuracy.

The optimizer is at the heart of neural network training [1], minimizing the loss function by adjusting the weight and bias, thereby optimizing the model parameters and helping the network to converge quickly to the optimal solution. Many optimizers have been proposed, and a class of optimization algorithms represented by stochastic gradient descent (SGD) is widely used. These algorithms minimize the objective function by taking the inverse direction of the first step degree of the parameters as the update direction,

but the convergence speed is slow. SGD with momentum [2] includes an acceleration mechanism based on SGD and uses the exponential moving average of the gradient direction at each moment as the updating direction to reduce the oscillation amplitude of the descent path and accelerate the updating in the smooth direction; the Nesterov accelerated gradient is used to replace the current gradient with the gradient at the next moment based on the momentum method, which is more conducive to jumping out of the local optimum. The step size of both is determined by the constant learning rate multiplied by the momentum, and the gradually accumulated momentum has a rigid speed. Following the introduction of this method, people began to study the use of different learning rates for each parameter [3] to improve the convergence speed of the algorithm; that is, adaptive learning rate methods began to propagate. For instance, Adagrad uses the sum of squares of all historical gradients, while AdaDelta and RMSprop introduce the attenuation rates of the two exponential weighted moving averages, β and γ . These methods realize the self-regulation of the learning rate through second-order momentum. Adam combines first-order momentum with second-order momentum, which raises the research in respect of the adaptive learning rate algorithm to a new level, not only adjusting the direction of decline but also realizing self-adaptation of the learning rate. Therefore, many variants of Adam were proposed for “improving” its performance [4], e.g., (Nadam [5], AdamW, Adafactor [6], diffGrad [7], AMSGrad [8], WuC-Adam [9], LCMAdam [10]). These variants aim to address issues such as the need for additional memory consumption and model non-convergence [5,11]. Since they use second-order moments to regulate the learning rate, there are undesired large variances [12] in the early stages, and they may converge to bad/suspicious local optima.

Thus, despite the breadth of modern optimization research, momentum gradient descent and its variants remain the preferred tools for machine learning [13]. In order to accelerate the convergence speed and improve the prediction accuracy of network training, it is imperative to delve into the direction and step size of gradient descent. Rather than allowing the learning rate to self-adjust for modifying the step size, as adaptive algorithms do, our focus lies on the velocity of the gradient. For instance, SGDP [14] projects radial components in order to decelerate the attenuation of effective step size. However, its accuracy leaves much to be desired. QHM introduces additional hyperparameters to weigh the usage proportion of the momentum and gradient with respect to step size calculation, but different hyperparameters should be considered in different models and datasets. AggMo [13] combines multiple velocity vector update parameters with different damping coefficients, which requires a huge amount of computation. We have observed that the proposed optimization algorithm places significant emphasis on the role of accumulated historical momentum but overlooks the impact of the changing relationship between gradient and momentum on the descending gradient during model training. If this relationship can be quantified, it could be utilized to adjust momentum speed, thereby enhancing the effectiveness of the gradient descent step.

In this study, we present a novel optimization algorithm, which is a momentum-based reacceleration enhanced optimization algorithm named momentum reacceleration gradient descent (MRGD), in order to overcome the limitations of the traditional momentum method. Based on the traditional momentum method, the algorithm introduces a reacceleration mechanism, which takes the ratio of the current momentum to the current gradient as the value for monitoring the forward trend. After normalization, the value is multiplied by the original step speed, which can scale the update step size of the parameter at the current time so that parameter updating can be accelerated and smoothed twice only by using the necessary first-order momentum and gradient in situ. Thus, parameter updating is accelerated more accurately and quickly, the parameter update strategy is further optimized, and the global convergence and training efficiency of the algorithm are improved. Through reasonable parameter adjustment, momentum reacceleration gradient descent can adapt to different learning rates better, so as to achieve enhanced performance in the optimization process. In the practical implementation of image recognition, the optimization algorithm

plays a crucial role in minimizing the training epochs of neural networks, conserving computing resources, and ensuring high recognition accuracy.

The innovation points and main work of this study are as follows:

- We propose an improved momentum reacceleration gradient descent algorithm (MRGD) based on the momentum method and verify its performance using multiple image classification datasets through experiments. It is proven that the MRGD algorithm proposed in this paper achieves higher accuracy than the traditional momentum method and Adam algorithm.
- In terms of the sparsity problem, we combine the MRGD algorithm with the Adam algorithm to propose the MRGDAdam algorithm. The experimental results demonstrate that its convergence speed is faster than that of Adam, and its precision performance is also higher than that of Adam. Furthermore, the experimental results also indicate the universality of the proposed method.
- We analyze the fact that the descent rate of the stochastic gradient descent algorithm can be influenced by the relationship between gradient and momentum. The algorithm is verified in the actual task of image classification. The experimental results prove the potential application value of MRGD in the field of optimization algorithms and practical tasks and provide a more effective optimization choice for deep learning model training.
- The algorithm proposed in this paper provides a new idea for the future study of optimization algorithms, and its outstanding training efficiency provides assistance for those studying the practical applications of deep learning.
- The paper is organized as follows. Section 2 introduces the development of the current stochastic gradient descent algorithm, including the constant learning rate stochastic gradient descent algorithms and adaptive learning rate optimization algorithms. Section 3 provides an introduction to the basic principles of the SGD algorithm, and discusses the analysis of gradient descent speed issues, proposing the MRGD algorithm and MRGDAdam algorithm. Section 4 describes our experiments and results analysis demonstrating the effectiveness of the proposed methods. Finally, a conclusion is drawn to summarize the article.

2. Related Work

Depending on whether the learning rate can adapt to the parameters, the optimizer separates the constant learning rate gradient descent algorithm, represented by SGD, and the adaptive learning rate optimization algorithm, represented by Adam.

2.1. Constant Learning Rate Gradient Descent Algorithm

The constant learning rate gradient descent algorithm refers to the use of a consistent and constant learning rate in the parameter updating process, typical of which is the stochastic gradient descent (SGD) algorithm, where steps are taken in the direction of evaluating the negative gradient of the loss function in a small batch. On this basis, SGD with momentum uses the exponential weighted moving average (also known as first-order momentum) to add descent inertia in the process of gradient descent, speeding up the descent speed while suppressing SGD shocks. The Nesterov accelerated gradient uses the next gradient to update the current momentum and then update the parameters, which is a better correction to the direction of parameter updating, but it can still easily fall into a local optimum and the convergence speed is slow, meaning it requires more computing resources and time. QHM [15] uses momentum steps to average an ordinary SGD step size and adds an immediate discount factor, which relies too much on hyperparameters and requires a huge amount of calculation. AggMo [13] combines multiple velocity vectors with different damping coefficients and uses different beta values to smooth and accelerate, which is computationally heavy and unsuitable for large datasets. SWA averages multiple checkpoints during training. Compared with SGD, SWA is helpful for avoiding local minima and improving generalization, but it fails to achieve the expected results due to late

training. AccSGD [16] uses the cumulative gradient to reduce the noise of the stochastic gradient descent algorithm, but it requires a lot of memory and time.

2.2. Adaptive Learning Rate Algorithm

For the unbalanced features of the data, we require the parameters to have different update rates. This means the learning rate can adjust itself automatically by learning data features, so various gradient-based adaptive methods have been proposed. One of these is a method using the gradient divided by the square root of the components of a vector, obtained using the historical sum of the gradient squared (such as Adagrad) or the exponential average (such as RMSprop [17], Adam, and Adadelta). In convex problems, when the gradient is sparse, these methods have theoretical advantages over SGD. The superior performance of these methods comes at a cost; namely, the computational power required to train neural networks with a large number of parameters and the memory capacity required to store these parameters during training need to be greatly enhanced. Nadam with integrated Nesterov, AdamW with decoupling weight attenuation, Adafactor with decomposition parsecs, Adabound [18] with a limited learning rate to increase stability, AdaMax with a momentum of infinite order, AMSGrad with prevention of the non-convergence of the learning rate, and other adaptive learning rate algorithms have been proposed. Radam [12] uses gradient clipping to slow down early oscillation, and Lookahead [19] maintains two sets of weights and interpolates between them to make the descent path more stable. Ranger, which combines the two, has the advantages of a faster descent speed and a more robust process, but it has high computational complexity. Adaptive optimization algorithms require additional memory to store additional accumulators; for example, Adam preserves two added values for each parameter, tripling the memory requirements. Even though Adafactor retains only the sum of each row and column of the moving average of the past gradient square, which greatly reduces the parameter storage, users are more inclined to choose the traditional stochastic gradient descent method after balancing the improved accuracy and memory consumption compared with an optimization algorithm without a second moment.

3. Methods

3.1. SGD Method

In this paper, we focus on the most basic stochastic gradient descent (SGD) algorithm for the function $f(x)$, as shown in Equation (1):

$$f_t(x) = f_{t-1}(x) - \alpha \cdot \nabla f_{t-1}(x). \quad (1)$$

The objective function $f_t(x) - f_{t-1}(x)$ is minimized using the inverse direction of the first step degree of the parameter as the update direction, and the optimal solution is calculated, where $\alpha \in R$ represents the learning rate, and exists as a hyperparameter.

3.1.1. Momentum Method

On the basis of SGD, the classical momentum method adopts the exponential weighted moving average method and uses the accumulated momentum to smooth the oscillation and accelerate the iteration in the optimal direction. $\alpha \in R$ is the learning rate and $\beta \in R$ is the momentum coefficient, which is usually 0.9, as shown in Equations (2) and (3):

$$m_t \leftarrow \beta \cdot m_{t-1} + (1 - \beta) \cdot \nabla f_{t-1}(x), \quad (2)$$

$$f_t(x) \leftarrow f_{t-1}(x) - \alpha \cdot m_t, \quad (3)$$

where m_t represents the momentum buffer, and when $\beta = 0$, the above rule describes the traditional SGD.

3.1.2. Adam Algorithm

Adam is a classical adaptive algorithm. We reproduce the pseudo-code of the Adam optimizer in the following algorithm. Suppose we try to minimize the expected value of a noisy objective function $f(x)$. At each step, a stochastic implementation of f_t is obtained. For example, the loss is calculated on a random small batch of data and the gradient g_t of this function are calculated with respect to the previous parameter. Then, the exponential moving averages of the first and second moments of gradients m_t and v_t are updated, the deviation corrections $\hat{m}_t$ and $\hat{v}_t$ are calculated to account for the zero initialization, and finally the parameters are updated to obtain a new iteration x_t . Step t is repeated in this way, and the last iteration x_t is returned as our approximate solution, as shown in Algorithm 1.

Algorithm 1 Adam

```

1 : Inputs : initial point  $x_0$ , step sizes  $\{\alpha_t\}_{t=1}^T$ , first moment decay  $\beta_1$ , second moment decay  $\beta_2$ ,
   regularization constant  $\epsilon$ 
2 : Initialize  $m_0 = 0$  and  $v_0 = 0$ 
3 : for  $t = 1$  to  $T$  do
4 :    $g_t = \nabla f_t(x_{t-1})$ 
5 :    $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
6 :    $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
7 :    $\hat{m}_t = m_t / (1 - \beta_1^t)$ 
8 :    $\hat{v}_t = v_t / (1 - \beta_2^t)$ 
9 :    $x_t = x_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ 
10: end for

```

3.2. The Proposed Momentum Reacceleration Gradient Descent Algorithm

In the momentum method, the momentum is calculated by the exponential weighted moving average of the historical gradient, replacing the gradient in the process of gradient descent. This approach can help reduce oscillations in the descent path and accelerate convergence rates. However, it is important to note that introducing momentum will lead to a faster reduction in effective step size [13]. Therefore, we must consider both the improved benefits of momentum velocity in terms of direction and magnitude, as well as pay attention to the gradient calculated at each iteration.

As the model training converges towards its local best, it is expected that the idealized gradient will gradually approach 0. At this point, momentum will continue to optimize according to “inertia” in order to jump out of local optima. Similarly, when the model converges towards its global optimum, we can expect that the idealized gradient will gradually approach 0 and the momentum will converge with the magnitude of the gradient. Based on this intuition, we aim to utilize the relationship between them to either accelerate or suppress momentum velocity. This can ultimately assist in guiding our gradient descent path so that it takes fewer detours and allows for quicker convergence of our model.

We propose the momentum reacceleration gradient descent (MRGD) algorithm, an enhanced method based on the momentum method. The ratio of the current momentum to the current gradient is used as the value for monitoring the forward trend. After normalization, the value is multiplied by the original step speed, which can expand or shrink the update step size of the current parameter. Only the necessary first-order momentum and gradient can be used in situ to accelerate and smooth the updating of parameters twice, so as to accelerate the updating of parameters more accurately and quickly. The basic idea is shown in Figure 1.

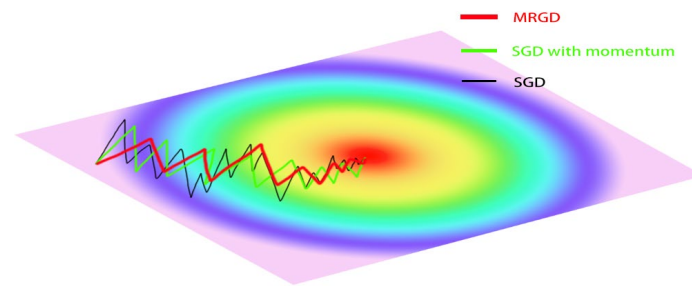


Figure 1. Schematic of the SGD and MRGD optimizer.

Suppose we try to minimize the expected value of a noisy objective function $f(x)$. At each step, a stochastic implementation of f_t is obtained. For example, for the loss calculated on a random small batch of data, and the gradient g_t of this function is calculated with respect to the previous parameter. Then, the exponential moving average of the first moment of gradient m_t is updated, and the ratio of the current time m_t to the current g_t is calculated to obtain r_t . Then, r_t is replaced with negative values and normalized. Finally, the parameters are updated to obtain a new iteration x_t . Step t is repeated in this way, and the last iteration x_t is returned as our approximate solution, as shown in Algorithm 2.

Algorithm 2 MRGD

```

1 : Inputs : initial point  $x_0$ , step sizes  $\{\alpha_t\}_{t=1}^T$ , first moment decay  $\beta$ , regularization constant  $\varepsilon$ ,
   zoom ratio  $\mu$ 
2 : Initialize  $m_0 = 0$  and  $\alpha = 0.001$ 
3 : for  $t = 1$  to  $T$  do
4 :    $g_t = \nabla f_t(x_{t-1})$ 
5 :    $m_t = \beta m_{t-1} + (1 - \beta)g_t$ 
6 :    $r_t = m_t / g_t + \varepsilon$ 
7 :    $r_t(r_t < 0) = -r_t(r_t < 0)$ 
8 :    $\hat{r}_t = (r_t - r_{tmin}) / (r_{tmax} - r_{tmin} + \varepsilon) \cdot \mu$ 
9 :    $x_t = x_{t-1} - \alpha \cdot m_t \cdot \hat{r}_t$ 
10: end for

```

The first-order momentum, i.e., the exponentially weighted moving average m_t , will guide the loss function to move in the direction of the minimization trend. In this paper, one component of each tensor is used to describe the principle of the algorithm. The component in one direction of the momentum is called momentum, and the component in the direction of the gradient is called gradient, which will not be emphasized below.

The ratio of the momentum at time t to the current gradient symbolizes whether the current momentum direction and gradient direction are consistent. If the trend is consistent, which means that the momentum and gradient direction are the same, then the momentum value at time t will be greater than (less than) the gradient value, and the ratio will be more than (less than) one (the ratio depends on the degree of trend consistency between the two), which will be multiplied by the original step size of the update. The step size can be expanded (shrunk) on the original base step size, and the degree of amplification (reduction) depends on the increase (reduction) degree of momentum relative to the gradient, i.e., the ratio. According to the characteristics of the exponential moving average m_t , the change in the ratio of momentum to gradient will not be particularly prominent, so the step size will be adjusted smoothly and moderately. If the trend is inconsistent, the ratio will be negative, which means that the momentum has a tendency to oscillate and decrease, and the update step size should be reduced; that is, the negative value of the ratio is reversed and then multiplied with the step size, so that it will reduce the step size on the basis of an unchanging direction. The result of the ratio is normalized in Formula (8).

Overshoot phenomenon: Similar to the momentum method, our approach may exhibit an overshoot phenomenon. However, due to the ability of the ratio $\hat{r}_t$ to adjust the

descent speed in real time based on both momentum and gradient state, it can effectively regulate the step size. As a result, we are able to set a smaller learning rate, which helps prevent overshooting.

Computational complexity: The computational workload is higher compared to the momentum method because the MRGD algorithm involves element-wise division, inversion, and normalization operations. Its complexity is directly proportional to the number of parameters. Nonetheless, it still maintains a linear complexity similar to that of the momentum method, i.e., being linearly related to both sample feature dimension d and the number of parameters n . This indicates that the algorithm remains highly computationally efficient even in large-scale data scenarios.

For the adaptive learning rate algorithm, we can still follow this optimization idea and improve the first-order momentum. The following is the improved algorithm MRGDAdam based on the Adam optimizer. Suppose we try to minimize the expected value of a noisy objective function $f(x)$. At each step, a stochastic implementation of f_t is obtained. For example, the loss is calculated on a random small batch of data, and the gradient g_t of this function is calculated with respect to the previous parameter. Then, the exponential moving average of the first and second moments of gradients, denoted as m_t and v_t , is updated. And the ratio of the current time m_t to the current g_t is calculated to obtain r_t . Subsequently, r_t is replaced with negative values and normalized. To account for zero initialization, the deviation correction $\hat{m}_t$ and $\hat{v}_t$ are calculated. Finally, the parameters are updated to obtain a new iteration x_t . Step t is repeated in this way, and the last iteration x_t is returned as our approximate solution. The specific steps are shown in Algorithm 3.

Algorithm 3 MRGDAdam

```

1 : Inputs : initial point  $x_0$ , step sizes  $\{\alpha_t\}_{t=1}^T$ , first moment decay  $\beta_1$ , second moment decay  $\beta_2$ ,
   regularization constant  $\varepsilon$ , zoom ratio  $\mu$ 
2 : Initialize  $m_0 = 0$  and  $\alpha = 0.001$ 
3 : for  $t = 1$  to  $T$  do
4 :    $g_t = \nabla f_t(x_{t-1})$ 
5 :    $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
6 :    $r_t = m_t / g_t + \varepsilon$ 
7 :    $r_t(r_t < 0) = -r_t(r_t < 0)$ 
8 :    $\hat{r}_t = (r_t - r_{tmin}) / (r_{tmax} - r_{tmin} + \varepsilon) \cdot \mu$ 
9 :    $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
10 :   $\hat{m}_t = m_t / (1 - \beta_1^t)$ 
11 :   $\hat{v}_t = v_t / (1 - \beta_2^t)$ 
12 :   $x_t = x_{t-1} - \alpha_t \cdot \hat{m}_t \cdot \hat{r}_t / (\sqrt{\hat{v}_t} + \varepsilon)$ 
13 : end for

```

According to the consistency analysis of the momentum and the gradient, the speed of gradient descent is adjusted reasonably. A larger step length is used in the correct direction, while a smaller step length is used in the wrong direction. Additionally, a larger step size is employed to traverse local optima and facilitate quicker convergence to the global optimum. This algorithm offers the advantage of achieving higher recognition accuracy with a small number of training epochs for the model.

3.3. Experimental Methods

3.3.1. Datasets and Tasks

To comprehensively assess the performance of MRGD, we chose several benchmark datasets in computer vision and one for real-world applications, including Mnist, CIFAR-10, CIFAR-100, and Aluminum Profile. Mnist is a handwritten numeric dataset consisting of 60,000 training images and 10,000 test images. The CIFAR-10 dataset contains 60,000 image samples with dimensions of 32×32 pixels, 50,000 for training and 10,000 for testing, and contains 10 categories, each of which is relatively generic, such as cats, dogs, cars, etc. CIFAR-100 is a natural image dataset consisting of 50,000 training images and 10,000 test

images. This dataset was chosen because it is widely available, there is a large amount of previous work to compare it with, and it is challenging and requires highly accurate classifiers for processing.

Aluminum Profile comprises the monitoring image data of aluminum profiles with defects in actual production. Each image contains one or more defects. The dataset consists of 12 defect classifications, of which 1700 have been used for training and 536 for testing. The purpose of choosing this dataset was to investigate the applicability of the proposed optimizer in practice and test its generalization performance. Figure 2 shows a partial sample of the dataset.

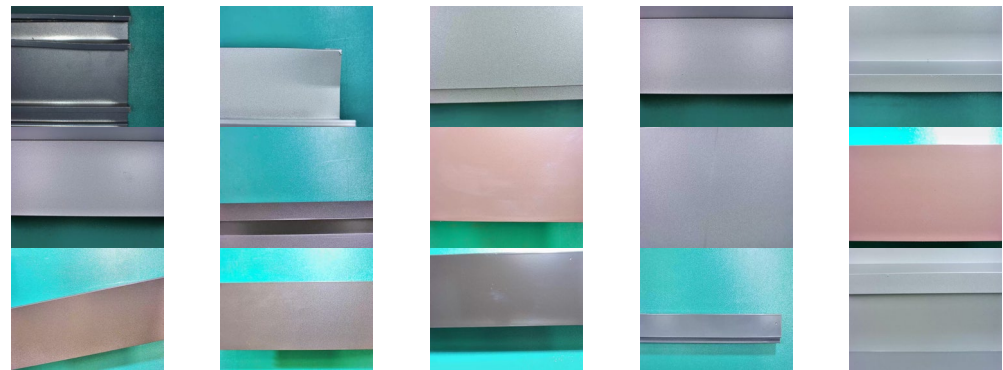


Figure 2. Sample of Aluminum Profile dataset.

3.3.2. Model Architecture

We used image classification model architectures such as ResNet-18, VGG-16, and MobileNetV2 to evaluate the performance of MRGD. ResNet-18 is a small variant in the ResNet series with 18 layers of depth. The introduction of residual connections makes the network easier to train and optimize. ResNet-18 uses basic residual blocks, each containing two convolutional layers, and adds inputs and outputs using skip joins. ResNet-18 [20] has achieved good performance in computer vision tasks such as image classification and target detection, and is widely used in practical projects. The main feature of VGG-16 (Visual Geometry Group-16) is the use of a continuous 3×3 convolution kernel and pooling layer, which give the network a smaller receptive field and a deeper level. VGG-16 has a depth of 16 layers, including 13 convolutional layers and 3 fully connected layers. Its convolutional layer part is simple and structured [21], making the network easy to understand and implement. VGG-16 performs well for image classification tasks; however, due to its deep network structure, it has high computational complexity and a large number of parameters. MobileNetV2 is a lightweight convolutional neural network model proposed by Google to achieve efficient image recognition on mobile devices with limited computing resources. Techniques such as depthwise separable convolution and linear bottlenecking are used to reduce the computational load and parameter number of the models. MobileNetV2 [22] has achieved good performance in lightweight models, enabling fast and accurate image classification on devices with limited computing resources.

3.3.3. Experimental Environment and Parameter Settings

All experiments were conducted on NVIDIA GeForce RTX 3080 servers (NVIDIA technology service co., LTD, Beijing, China). We implemented MRGD and MRGDAdam using the PyTorch (torch2.0.1; python3.11, Facebook Artificial Intelligence Research, New York, NY, America) framework and compared them with SGD with momentum (namely SGD), Adam, and RMSprop.

For each optimizer and dataset combination, we performed a thorough hyperparameter search to find the optimal hyperparameters. Hyperparameters included learning rate, momentum, weight decay, and batch size. We use the Hyperband [23] method to search for optimal hyperparameters, each of which has a range of values. We selected the optimal

hyperparameters according to the performance of the verification set. We used the same training settings and hyperparameters for each optimizer. Each algorithm was trained for 20 epochs and was stopped early based on the performance of the verification set. We use the same data enhancement and preprocessing strategies for each task, with cross-entropy loss as the objective function and several performance indexes (including accuracy and loss) as evaluation indexes. We reported the average performance of each optimizer over multiple runs, and recorded the data using wandb (wandb CLI Version 0.16.2, Weights and Biases, San Francisco, CA, USA).

4. Experimental Results and Analysis

4.1. Experimental Results

In the image classification tasks on the Mnist, CIFAR-10, CIFAR-100, and Aluminum Profile datasets, both MRGD and MRGDAdam showed comparable or even superior performance to other optimizers in terms of the index accuracy and loss values for evaluating the classification tasks. Especially on the multi-classification task CIFAR-100 dataset, MRGD accuracy was significantly higher than for SGD and Adam, and the MRGDAdam accuracy was significantly higher than that of Adam. The accuracy calculation formula is shown in Equation (4):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad (4)$$

where TP = true positive, TN = true negative, FP = false positive, and FN = false negative.

The calculation formula for the loss value is shown in Equation (5):

$$H(P, Q) = -\sum_{i=1}^n P(x_i) \log Q(x_i), \quad (5)$$

where $H(P, Q)$ is the cross-entropy loss, $P(x_i)$ is the expected output of the sample, $Q(x_i)$ is the actual output of the sample, and n is the number of classes.

The comparisons of test (T) accuracy and loss values across various datasets when utilizing different optimizers with the ResNet-18, VGG-16 and MobileNetV2 models are shown in Tables 1–3, respectively.

Table 1. Comparison of test accuracy and loss values on the Resnet-18 model.

Optimizer	Mnist		CIFAR-10		CIFAR-100		Aluminum Profile	
	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)
MRGD	0.0215	0.9946	0.7336	0.858	2.009	0.5872	0.8023	0.7964
SGD	0.0393	0.9876	1.036	0.6302	3.042	0.2572	1.414	0.5734
MRGDAdam	0.0223	0.9943	0.7683	0.8495	2.01	0.5756	0.8011	0.8021
Adam	0.0294	0.9921	0.6549	0.8571	2.54	0.5614	0.8322	0.7569
RMSprop	0.0286	0.993	0.7676	0.8549	2.925	0.5311	0.8566	0.7294

Bold font represents the best-performing data in the same settings.

Table 2. Comparison of test accuracy and loss values on the VGG-16 model.

Optimizer	Mnist		CIFAR-10		CIFAR-100		Aluminum Profile	
	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)
MRGD	0.0255	0.9956	0.7102	0.849	2.163	0.4754	0.932	0.7115
SGD	0.0968	0.9798	1.1901	0.642	3.664	0.1545	1.8173	0.4936
MRGDAdam	0.0272	0.9901	0.6779	0.8618	2.412	0.5034	0.8217	0.7323
Adam	0.0303	0.9850	0.6483	0.8360	2.351	0.3605	0.9674	0.6798
RMSprop	0.0384	0.9856	0.6460	0.8261	2.733	0.3227	0.9827	0.6619

Bold font represents the best-performing data in the same settings..

Table 3. Comparison of test accuracy and loss values on the MobileNetV2 model.

Optimizer	Mnist		CIFAR-10		CIFAR-100		Aluminum Profile	
	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)	T./loss(%)	T./acc.(%)
MRGD	0.0314	0.992	0.7616	0.7868	2.365	0.4398	1.103	0.6950
SGD	0.1260	0.9613	1.5325	0.4235	4.077	0.0704	2.7967	0.4613
MRGDAdam	0.0372	0.9913	0.8863	0.7604	2.571	0.4465	0.9435	0.7091
Adam	0.0351	0.9904	1.0278	0.7544	3.068	0.3908	1.8641	0.5546
RMSprop	0.0323	0.9944	0.9765	0.7683	3.024	0.365	1.986	0.5454

Bold font represents the best-performing data in the same settings.

4.2. Experimental Analysis

In this section, we discuss the suggested experimental setup for the optimizers MRGD and MRGDAdam and compare them to various regularly used optimizers on a variety of benchmark datasets and deep learning models. We use identical training settings and hyperparameters for each optimizer and dataset combination, evaluate each optimizer's performance in the test set using a variety of metrics, and provide the average performance over several runs.

Below is a comparison of the accuracy of the test set monitored using the various models. It can be seen that MRGD and MRGDAdam show higher accuracy and lower loss values, respectively, for the same task.

In addition to the comparisons shown in Tables 1–3, the test accuracy and loss values of each dataset for the ResNet-18, VGG-16, and MobileNetV2 models with respect to different optimizers are shown in Figure 3, Figure 4, and Figure 5, respectively.

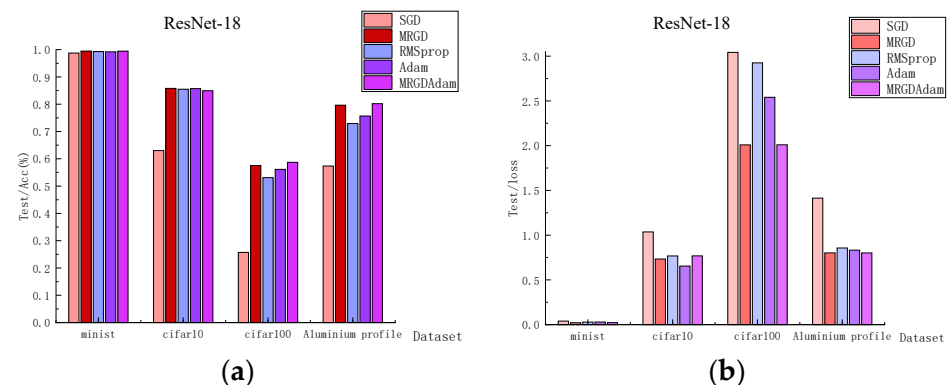


Figure 3. Comparison of optimizer performance across datasets for the ResNet-18 model: (a) comparison of accuracy and (b) comparison of loss values.

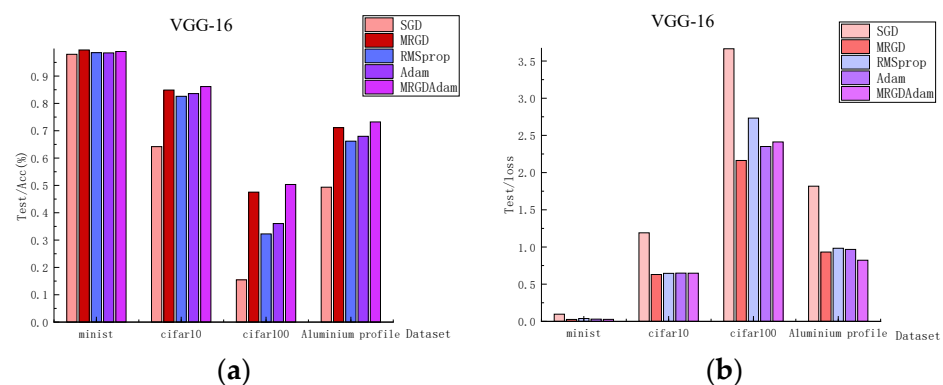


Figure 4. Comparison of optimizer performance across datasets for the VGG-16 model: (a) comparison of accuracy and (b) comparison of loss values.

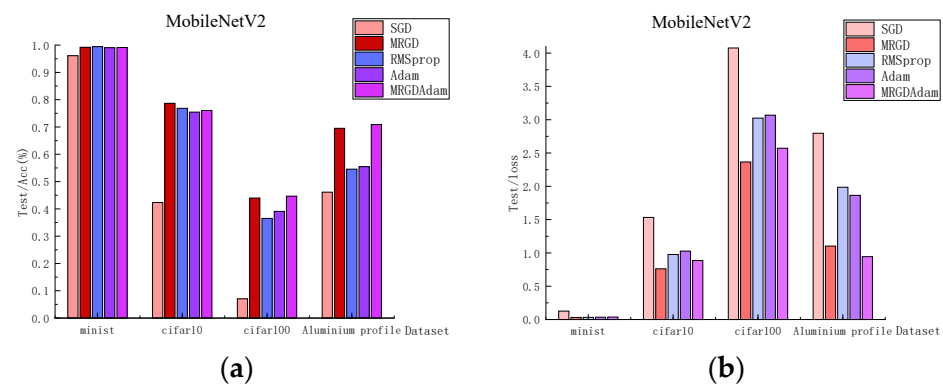


Figure 5. Optimizer performance comparison for each dataset for the MobileNetV2 model: (a) comparison of accuracy, and (b) comparison of loss values.

According to the performance of the validation set in the training process, it can be concluded that MRGD and MRGDAdam can achieve the best effect faster in shorter epochs. The training process of each optimizer's validation set on the CIFAR-100 dataset is shown in Figure 6 below, and the training process of each optimizer's validation set on the CIFAR-10 dataset is shown in Figure 7.

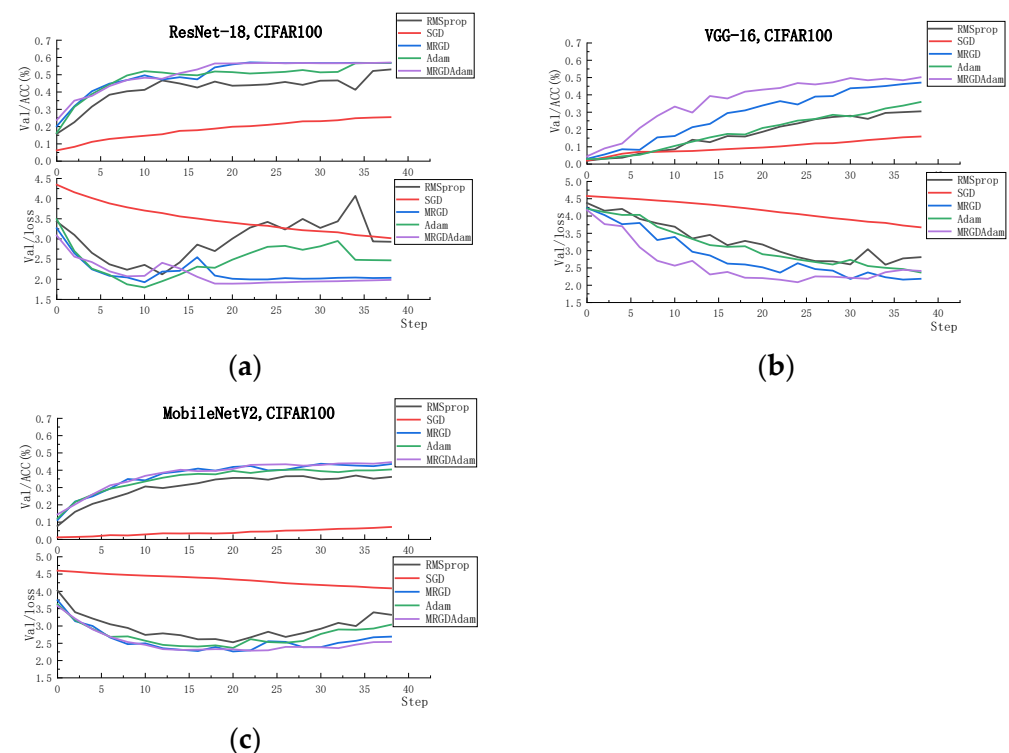


Figure 6. Accuracy (top) and loss trend (bottom) of validation on CIFAR-100 dataset. (a) shows the comparison trend of the accuracy and loss value of each optimizer of CIFAR-100 on the ResNet-18 model, (b) shows the comparison trend of the accuracy and loss value of each optimizer of CIFAR-100 on the VGG-16 model, and (c) shows the comparison trend of the accuracy and loss value of each optimizer of CIFAR-100 on the MobileNetV2 model.

Even though the effectiveness of the proposed optimizer in the test set of CIFAR-10 is not as good as that of CIFAR-100, the increase trend of the accuracy curve and the decrease trend of the loss are observed faster than those of Adam and other optimizers. We determined that the ratio of momentum to gradient can adjust the speed of gradient descent adaptively, and adjust the large variance occurring for various reasons (such as the adaptive learning rate) at the appropriate time so as to help the model converge faster.

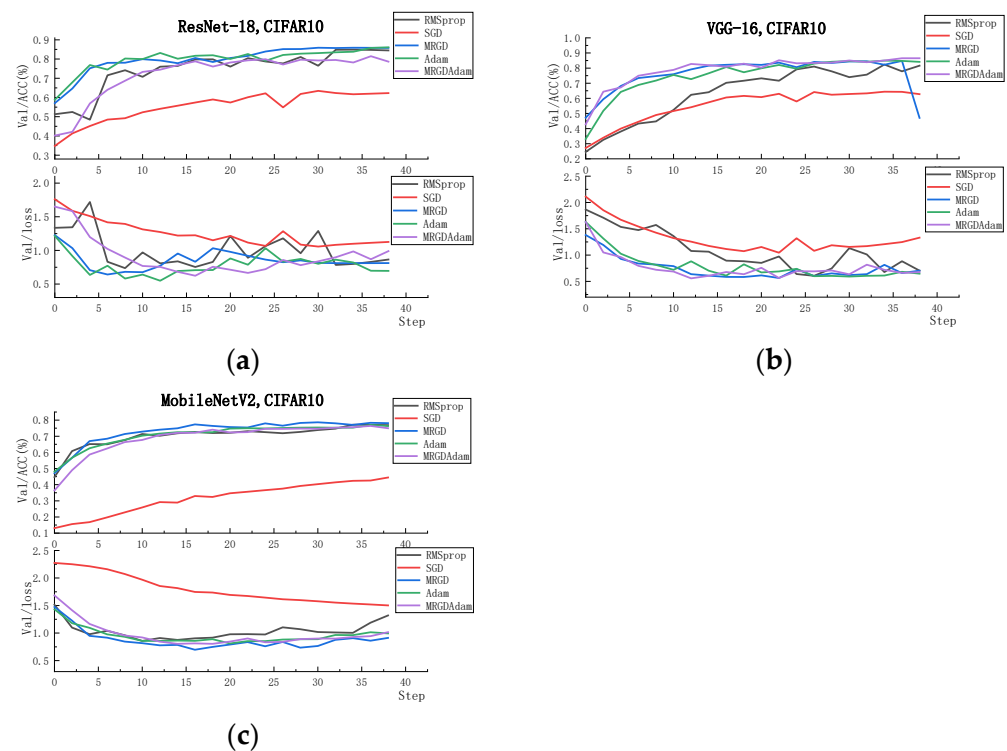


Figure 7. Accuracy (top) and loss trend (bottom) of validation on the CIFAR-10 dataset. (a) shows the comparison trend of the accuracy and loss value of each optimizer of CIFAR-10 on the ResNet-18 model, (b) shows the comparison trend of the accuracy and loss value of each optimizer of CIFAR-10 on the VGG-16 model, and (c) shows the comparison trend of the accuracy and loss value of each optimizer of CIFAR-10 on the MobileNetV2 model.

The results show that the performance of the proposed optimizer, MRGD, on the CIFAR-100 dataset is better than that of the other three optimizers. Specifically, the classification accuracy of MRGD on the ResNet-18 model is 58.72%, while the classification accuracies of Adam, SGD, and RMSprop are 56.14%, 25.72%, and 57.56%, respectively. The classification accuracy of MRGD on VGG-16 is 47.54%, while the classification accuracies of Adam, SGD, and RMSprop are 36.05%, 15.45%, and 32.27%, respectively. The classification accuracy of MRGD on the MobileNetV2 model is 43.98%, while the classification accuracies of Adam, SGD, and RMSprop are 39.08%, 7.04%, and 36.5%, respectively. This shows that the optimizer has significant advantages for image classification tasks.

The performance of MRGD on the Aluminum Profile dataset is also superior. Specifically, the classification accuracy of MRGD on the ResNet-18 model is 79.64%, while the classification accuracies of Adam, SGD, and RMSprop are 75.69%, 57.34%, and 72.94%, respectively. The classification accuracy of MRGD on VGG-16 is 71.15%, while the classification accuracies of Adam, SGD, and RMSprop are 67.98%, 49.36%, and 66.19%, respectively. The classification accuracy of MRGD on the MobileNetV2 model is 69.50%, while the classification accuracies of Adam, SGD, and RMSprop are 55.46%, 46.13%, and 54.54%, respectively.

We also compared the accuracy and loss curves of the validation set for various optimizers during training. In comparison to the other three optimizers, the MRGD optimizer's validation set has smoother and faster convergent accuracy and loss curves. These results suggest that MRGD has the potential to become a competitive alternative to existing optimizers, especially for more complex datasets for multi-classification tasks. For practical applications, projects with a small budget will greatly benefit from using this optimization algorithm.

5. Conclusions

Momentum reacceleration gradient descent is a revolutionary methodology for optimizing and improving the momentum method variants proposed in this research. The forward trend is monitored using the ratio of current momentum to the current gradient. After normalization, this value is multiplied by the original step speed, which can expand or shrink the update step size of the current parameter. Using only the necessary first-order momentum and gradient in situ, the updating of parameters can be accelerated and smoothed twice, thus accelerating the updating of parameters more accurately and quickly. In addition, in this study, the Adam algorithm was improved based on the same idea, and MRGDAdam was proposed. We evaluated this optimizer on image classification tasks through multiple datasets and compared it to currently popular optimizers such as SGD, Adam, and RMSprop.

The experimental results show that in a minimal number of model training epochs, MRGD outperforms alternative optimizers in all indexes, confirming its efficiency across models. The optimizer can increase convergence speed and accuracy across a variety of network configurations. It demonstrates good generalization in practical applications.

In practical applications, especially those requiring real-time decision-making or response, the MRGD optimizer can be used to train models faster, which not only reduces the need for computing resources (such as GPUs) and saves money but also provides faster and more accurate prediction results, thereby improving the user experience. Finally, the method maximizes the value of artificial intelligence.

We also discussed the principles and advantages of the optimizer model through the analysis of gradient updating in the training process. Specifically, the optimizer can also perform well in the adaptive learning rate algorithm. For example, MRGDAdam solves the problem of fast convergence on non-convex optimization problems and the updating of different parameters on sparse data, which provides a diverse solution for complex machine vision in multiple scenarios. Due to the introduction of hyperparameter μ in the normalization operation of r_t , the MRGD algorithm still has some limitations in terms of stability. Next, we will study the rule of μ when the MRGD algorithm is used to achieve the optimal effect for different datasets and models.

In the future, in the field of deep learning, the algorithm proposed in this paper may become a common optimizer model and provide support for further development and application. In future work, we will explore more applications of this optimizer in other computer vision tasks such as image segmentation, and further optimize the strategy to adapt to different training scenarios.

Author Contributions: Conceptualization, methodology, and writing—original draft preparation, Y.C.; software, project administration, and resources, Y.S.; data curation, R.T.; writing—review and editing, supervision, and formal analysis, H.S.; funding acquisition, Q.Z., L.X. and C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: 1. Liaoning Provincial Department of Education Basic Research Project for Higher Education Institutions (General Project), Shenyang University of Technology, Research on Optimization Design of Wind Turbine Cone Angle Based on Fluid Physics Method (LJKZ0159). 2. Basic Research Project of Liaoning Provincial Department of Education “Training and Application of Multimodal Deep Neural Network Models for Vertical Fields” Project Number: JYTMS20231160. 3. Research on the Construction of a New Artificial Intelligence Technology and High-Quality Education Service Supply System in the 14th Five-Year Plan for Education Science in Liaoning Province, 2023–2025, Project Number: JG22DB488. 4. “Chunhui Plan” of the Ministry of Education, Research on Optimization Model and Algorithm for Microgrid Energy Scheduling Based on Biological Behavior, Project No. 202200209. 5. Shenyang Science and Technology Plan “Special Mission for Leech Breeding and Traditional Chinese Medicine Planting in Dengshibao Town, Faku County”, Project No. 22-319-2-26.

Data Availability Statement: The Aluminum Profile dataset that supports the findings of this study is openly available in the “Science Data Bank” at <https://www.scidb.cn/en/s/eIN7Vr>, accessed on 1 February 2024. The Mnist dataset that supports the findings of this study is avail-

able from the National Institute of Standards and Technology (NIST) at <https://blog.csdn.net/bwqiang/article/details/110203835>, accessed on 26 December 2023. These data were derived from the following resources available in the public domain: <http://yann.lecun.com/exdb/mnist/>, accessed on 26 December 2023. The CIFAR-10 and CIFAR-100 datasets are openly available in a public repository. The data that support the findings of this study are openly available from <https://pan.baidu.com/s/1Ef49WYOsest7Ga3Scqqi7w>, accessed on 26 December 2023 at https://blog.csdn.net/qq_43280818/article/details/104405544, accessed on 26 December 2023.

Conflicts of Interest: Author Taotao Ran Tao was employed by the Shanghai Maruka Computer Information Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. Shanghai Maruka Computer Information Technology Co., Ltd. had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Kovachki, N.B.; Stuart, A.M. Continuous time analysis of momentum methods. *J. Mach. Learn. Res.* **2021**, *22*, 1–40.
2. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 17–19 June 2013; pp. 1139–1147.
3. Zhuang, J.; Tang, T.; Ding, Y.; Tatikonda, S.C.; Dvornek, N.; Papademetris, X.; Duncan, J. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18795–18806.
4. Guo, Z.; Xu, Y.; Yin, W.; Jin, R.; Yang, T. A Novel Convergence Analysis for Algorithms of the Adam Family and Beyond. *arXiv* **2021**, arXiv:2104.14840.
5. Dozat, T. Incorporating Nesterov momentum into Adam. In Proceedings of the ICLR 2016 Workshop, San Juan, Puerto Rico, 2–4 May 2016.
6. Shazeer, N.; Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; PMLR; pp. 4596–4604.
7. Dubey, S.R.; Chakraborty, S.; Roy, S.K.; Mukherjee, S.; Singh, S.K.; Chaudhuri, B.B. diffGrad: An optimization method for convolutional neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 4500–4511. [[CrossRef](#)] [[PubMed](#)]
8. Reddi, S.J.; Kale, S.; Kumar, S. On the convergence of Adam and beyond. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
9. Zhang, C.; Shao, Y.; Sun, H.; Xing, L.; Zhao, Q.; Zhang, L. The WuC-Adam algorithm based on joint improvement of Warmup and cosine annealing algorithms. *Math. Biosci. Eng.* **2023**, *21*, 1270–1285. [[CrossRef](#)] [[PubMed](#)]
10. Sun, H.; Zhou, W.; Shao, Y.; Cui, J.; Xing, L.; Zhao, Q.; Zhang, L. A Linear Interpolation and Curvature-Controlled Gradient Optimization Strategy Based on Adam. *Algorithms* **2024**, *17*, 185. [[CrossRef](#)]
11. Li, M.; Luo, F.; Gu, C.; Luo, Y.; Ding, W. Adams algorithm based on adaptive momentum update strategy. *J. Univ. Shanghai Sci. Technol.* **2023**, *45*, 112–119. [[CrossRef](#)]
12. Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; Han, J. On the variance of the adaptive learning rate and beyond. In Proceedings of the 8th International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26–30 April 2020.
13. Lucas, J.; Sun, S.; Zemel, R.; Grosse, R. Aggregated Momentum: Stability Through Passive Damping. In Proceedings of the 7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
14. Heo, B.; Chun, S.; Oh, S.J.; Han, D.; Yun, S.; Kim, G.; Uh, Y.; Ha, J.W. AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Virtual Event, Austria, 3–7 May 2021.
15. Ma, J.; Yarats, D. Quasi-hyperbolic momentum and Adam for deep learning. In Proceedings of the 7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
16. Jain, P.; Kakade, S.M.; Kidambi, R.; Netrapalli, P.; Sidford, A. Accelerating stochastic gradient descent for least squares regression. In Proceedings of the 31st Conference On Learning Theory, Stockholm, Sweden, 5–9 July 2018; pp. 545–604.
17. Shi, N.; Li, D.; Hong, M.; Sun, R. RMSprop converges with proper hyper-parameter. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Virtual Event, Austria, 3–7 May 2021.
18. Luo, L.; Xiong, Y.; Liu, Y.; Sun, X. Adaptive gradient methods with dynamic bound of learning rate. In Proceedings of the 7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
19. Zhang, M.; Lucas, J.; Ba, J.; Hinton, G.E. Lookahead optimizer: K steps forward, 1 step back. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
20. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, Las Vegas, NV, USA, 27–30 June; 2016; pp. 770–778.
21. Shao, Y.; Fan, S.; Sun, H.; Tan, Z.; Cai, Y.; Zhang, C.; Zhang, L. Multi-Scale Lightweight Neural Network for Steel Surface Defect Detection. *Coatings* **2023**, *13*, 1202. [[CrossRef](#)]

22. Shao, Y.; Zhang, C.; Xing, L.; Sun, H.; Zhao, Q.; Zhang, L. A new dust detection method for photovoltaic panel surface based on Pytorch and its economic benefit analysis. *Energy AI* **2024**, *16*, 100349. [[CrossRef](#)]
23. Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.* **2018**, *18*, 1–52.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.