Mathematical Biosciences and Engineering

MBE, 21(1): 1270–1285.
DOI: 10.3934/mbe.2024054
Received: 13 October 2023
Revised: 04 December 2023
Accepted: 13 December 2023
Published: 26 December 2023

http://www.aimspress.com/journal/MBE

Research article

The WuC-Adam algorithm based on joint improvement of Warmup and cosine annealing algorithms

Can Zhang[1], Yichuan Shao[2,*], Haijing Sun[2], Lei Xing[3], Qian Zhao[4] and Le Zhang[2]

[1] School of Information Engineering, Shenyang University, Shenyang 110044, China
[2] School of Intelligent Science & Engineering, Shenyang University, Shenyang 110044, China
[3] School of Chemistry and Chemical Engineering, University of Surrey, GU2 7XH, United Kingdom
[4] School of Science, Shenyang University of Technology, Shenyang 110044, China

* **Correspondence:** Email: 602841309@qq.com; Tel: +13002457275; Fax: +13002457275.

Abstract: The Adam algorithm is a common choice for optimizing neural network models. However, its application often brings challenges, such as susceptibility to local optima, overfitting and convergence problems caused by unstable learning rate behavior. In this article, we introduce an enhanced Adam optimization algorithm that integrates Warmup and cosine annealing techniques to alleviate these challenges. By integrating preheating technology into traditional Adam algorithms, we systematically improved the learning rate during the initial training phase, effectively avoiding instability issues. In addition, we adopt a dynamic cosine annealing strategy to adaptively adjust the learning rate, improve local optimization problems and enhance the model's generalization ability. To validate the effectiveness of our proposed method, extensive experiments were conducted on various standard datasets and compared with traditional Adam and other optimization methods. Multiple comparative experiments were conducted using multiple optimization algorithms and the improved algorithm proposed in this paper on multiple datasets. On the MNIST, CIFAR10 and CIFAR100 datasets, the improved algorithm proposed in this paper achieved accuracies of 98.87%, 87.67% and 58.88%, respectively, with significant improvements compared to other algorithms. The experimental results clearly indicate that our joint enhancement of the Adam algorithm has resulted in significant improvements in model convergence speed and generalization performance. These promising results emphasize the potential of our enhanced Adam algorithm in a wide range of deep learning tasks.

Keywords: deep learning; Adam algorithm; Warmup; cosine annealing strategy; local optimum

1. Introduction

Deep learning has yielded noteworthy advancements in fields such as computer vision and natural language processing. It employs backpropagation techniques to instruct machines on adjusting their internal parameters. By systematically applying multi-layer nonlinear transformations, deep learning automatically acquires data representations and discerns features, thereby extracting abstract information from the data [1]. Deep learning relies on the selection of optimization algorithms and parameter adjustment during the training process. Adam has become a commonly used choice for training deep neural networks due to its self-adaptability and efficiency. Owing to the instability exhibited by the conventional Adam optimizer, particularly during the initial phases of training, there can be fluctuations induced by elevated learning rates or inadequate exploration of the parameter space. Wilson et al. pointed out that the adaptive gradient descent method in the Adam algorithm may not be as good as random gradient descent in some cases [2]. In order to overcome some of the shortcomings of the Adam algorithm, researchers have proposed various improved versions. Reddi et al. [3] pointed out the shortcomings of Adam in proving convergence and proposed a variant of the Adam algorithm, AMSGrad algorithm. This algorithm avoids learning rate oscillations by improving the iterative method of second-order momentum, thereby solving the problem of model non-convergence. Loshchilov and Hutter pointed out that weight attenuation in the Adam algorithm is achieved by directly adding L2 regularization terms in parameter updates. However, this method can result in unstable training procedures [4]. They also proposed the Adam algorithm [4], applying the weight attenuation term to the weight p.data of parameter updates, rather than directly adding it to the gradient p.grad, to some extent solves the problem of unstable Adam training. LAMB introduces a new gradient algorithm based on Adam, making it more suitable for training in handling large batches of data [5]. Sashank J. Reddi et al. pointed out that Adam may experience problems such as slow convergence speed and overfitting in some cases [6]. The Radam [7] proposed by Liu and Jiang et al. introduces a learning rate correction mechanism to solve the problem of unstable training caused by the large fluctuations in the variance of Adam's learning rate due to some reasons. A belief Optimizer introduces a mechanism called "Belief Correction" based on Adam to adjust the update of learning rate [8]. Xu et al. proposed that Yogi [9] uses first-order momentum to replace second-order momentum estimation in Adam, aiming to improve Adam's performance under different learning rate conditions and achieve good training results in large-scale batch data. The core idea of Nadam [10] proposed by Dozat is to combine the advantages of NAG with Adam's adaptive learning rate, and introduce the Nesterov momentum term to improve the gradient update process in Adam. In Adam, the learning rate of adaptive adjustment may increase too quickly under certain circumstances. AdaBound introduces boundaries to limit the range of learning rate, making it controllable and improving model stability [11]. Lookahead Optimizer [12] introduces a "lookahead" based on Adam to accelerate the optimization process. Reyad et al. proposed HN_ Adam [13], modifies the standardized value of the parameter change formula by automatically adjusting the step size of parameter updates during the training period. Chen et al. proposed Lion [14], which uses symbolic functions to handle update volume, to some extent reduces computational complexity, but its performance is not as good as Adam in small batch situations. The Amos [15] proposed by Tian et al. can be regarded as an Adam with theoretical support for adaptive learning rate decay and weight decay, which can use model specific initial information to determine the initial learning rate and decay schedule. In response to the problem of slow convergence in Adam, Liu et al. proposed an improved Adam algorithm based on a combination of adaptive coefficients and composite gradients with random block coordinate descent [16]. Adan adopts a new Nesterov Momentum Estimation (NME) method to estimate the first and second moments in adaptive gradient algorithms to accelerate convergence [17]. However, these improvements may face issues such as unreasonable learning rate adjustment, increased computational complexity and local optima.

Warmup is a common learning rate adjustment strategy in deep learning, used to gradually increase the learning rate during the initial stage of training. Usually, a sudden increase in learning rate may lead to instability during model training, especially in the early stages of training. To address this issue, the algorithm proposed in this article introduces Warmup technology, which gradually increases the learning rate within the first few epochs of training, allowing the model to train more smoothly. The basic idea of Warmup is to use a small learning rate in the early stages of training, gradually increasing to the set initial learning rate within a certain number of iterations. This helps the model to better explore the parameter space in the early stages of training, avoiding oscillation or divergence problems caused by excessive learning rate.

Therefore, we present a collaborative optimization approach that integrates Warmup and cosine annealing strategies into the Adam algorithm in this paper. First, utilizing Warmup technology, we progressively raise the learning rate, enhancing the ability of the model to explore the parameter space more effectively during the initial training phases. Subsequently, we employ the cosine annealing strategy to dynamically fine-tune the learning rate, preventing premature decreases and promoting stable convergence in the later stages of training. To validate the joint enhancement of the Adam algorithm, extensive experiments were conducted on multiple well-established datasets. The experimental results unequivocally demonstrate that our algorithm offers substantial benefits in terms of convergence speed and generalization performance when compared to both the traditional Adam algorithm and other modified variants. The principal contributions of this paper encompass the proposal of an enhanced Adam algorithm grounded in Warmup and cosine annealing, effectively mitigating certain drawbacks associated with conventional Adam algorithms. Furthermore, through comprehensive experimentation on classic datasets, we establish the superior performance of our jointly improved Adam algorithm.

2. Design of WuC-Adam algorithm

2.1. The Adam optimization algorithm

Adam is an adaptive learning rate optimization algorithm based on gradient descent, which combines the advantages of gradient descent algorithm and momentum method. Adam's main idea is to adjust the learning rate by calculating the first-order moment estimation and second-order moment estimation of gradients. The first order moment estimation is the average value of the gradient, while the second order moment estimation is the average value of the square of the gradient. Adam, as one of the commonly used optimization algorithms in neural network models, has the advantage of being able to adaptively adjust the learning rate and adopt different learning rate sizes for different parameters, thus better adapting to the characteristics of different parameters. Adam can also adaptively adjust the size of the learning rate, avoiding the hassle of manually adjusting the learning rate. Moreover, Adam can handle sparse gradients, online settings and non-stationary settings well [18]. In Adam, the parameters used for both are defined as first-order moment m_t and second-order moment v_t, respectively. The moment estimation vectors for the gradient in step t are:

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

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

In the formula, β_1 and β_2 are the exponential decay rates of the first and second moment, respectively, g_t is the gradient of step t, and the calculation formula is:

$$g_t = \nabla_\theta f_t(\theta_t) \tag{3}$$

Among them, θ_t is the parameter vector updated in step t, and $f_t(\theta)$ is the loss function of the t-th iteration in the neural network, $\nabla_\theta f_t(\theta_t)$ is the gradient of the loss function of the t-th iteration of the neural network with respect to parameters.

When the decay rate is very small in the initial stage, the first-order and second-order moment estimates may deviate from 0, resulting in some deviation. To eliminate this deviation, Adam added bias correction to the first-order and second-order moments during the decay process. The formula is as follows:

$$\hat{m} = m/(1-\beta_1^t) \tag{4}$$

$$\hat{v} = v/(1-\beta_2^t) \tag{5}$$

After each iteration, parameter θ will be updated with the following formula:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}+\varepsilon}} \hat{m}_t \tag{6}$$

In the formula: α represents the learning rate; ε is a sufficiently small value greater than 0, mainly to avoid having a denominator of 0.

Although Adam has good performance, it is sensitive to the selection of learning rates and requires careful adjustment of hyperparameters, which cannot converge to the global optimal solution in some non-convex optimization problems.

2.2. Warmup

Warmup was first proposed by He et al. [19] in 2016 as a solution for very deep neural networks to prevent overfitting due to high learning rates. The specific Warmup strategy can be set based on specific problems and models, and common methods include linear Warmup and exponential Warmup. Linear Warmup linearly increases the learning rate within a set number of iterations, while exponential Warmup gradually adjusts the learning rate through exponential growth.

The commonly used Warmup formula is as follows:

$$lr = lr_{base} \frac{step_{current}}{epoch_{Warm} step_{one\_epoch}} \tag{7}$$

Among them, lr_{base} represents the initial learning rate, and the following fractions are used to control the Warmup operation. The numerator represents the current number of steps in the iteration, the denominator $epoch_{Warmup}$ represents the number of epochs set for Warmup before training, and $step_{one\_epoch}$ represents the number of steps required to iterate one epoch in the training set.

We made modifications based on the above formula, and the specific formula is as follows:

$$lr_{Warmup} = lr\frac{step_{current}}{step_{Warmup}} \tag{8}$$

The denominator of the fraction represents the number of Warmup steps initialized, which reduces the time required for the entire Warmup process.

The main purpose of using Warmup is to improve the stability and convergence speed of the model in the early stages of training. In deep learning, learning rate is a very important hyperparameter in optimization algorithms, which determines the magnitude of model parameter updates in each iteration. A suitable learning rate can enable the model to converge to the optimal solution faster during the training process, thereby saving training time and improving model performance. However, in the early stages of training, the parameters of the model may be in a random initialization state or in a poor region, and using a higher learning rate may cause the model to oscillate or fail to converge. This is because an excessive learning rate may lead to excessive parameter updates, making the model unable to steadily move towards the optimal solution. To address this issue, Warmup technology was introduced, which allows us to use a smaller learning rate in the early stages of training and gradually increase the learning rate to the set initial value. The advantage of doing so is that the model undergoes a stable exploration in the early stages of training, avoiding instability caused by excessive learning rate. As the training progresses, the learning rate gradually increases, and the model gradually enters a more suitable state. Then, the set maximum learning rate is used to continue training.

Warmup is usually used in conjunction with other learning rate adjustment strategies to achieve better training results. By properly setting the steps and initial learning rate of Warmup, the convergence speed, stability and generalization ability of the model can be improved when training deep neural networks.

2.3. Cosine annealing strategy

The cosine annealing strategy is a learning rate adjustment strategy used to dynamically adjust the learning rate in optimization algorithms. Its main idea is to simulate the annealing process of the cosine function, periodically changing the learning rate during the training process, so that the model can better make fine adjustments in the later stage of training and improve convergence performance.

The original formula for cosine annealing strategy is as follows:

$$lr = lr + \frac{1}{2}(lr_{\max} - lr)[1 + \cos(\frac{T}{T_I}\pi)] \tag{9}$$

The T in the numerator represents the total number of training cycles to the current stage, and the T_I in the denominator represents the set cosine annealing cycle. Based on the ratio between T and T_I, the learning rate exhibits a cosine like variation.

This article has made modifications to the original cosine annealing strategy, using the relationship between the current number of steps and the set warmup value to calculate the learning rate. The modified formula is as follows:

$$lr \leftarrow lr + \frac{1}{2}(lr_{\max} - lr)[1+\cos(\text{step-}(\text{warmup}\lambda / T_0)\pi)] \tag{10}$$

The values of warmup and T_0 are given during initialization, while λ represents $lr_{\max} / lr$, which is the ratio of the maximum learning rate to the current learning rate.

The characteristic of cosine annealing strategy is that the learning rate will undergo periodic changes within the range of maximum and minimum learning rates according to the annealing curve of the cosine function. This helps to make the model more stable in the later stages of training, avoiding oscillations or jumping out of local optima caused by excessive learning rate, as well as situations where the learning rate is too small and the convergence speed is too slow.

Compared to traditional fixed learning rates, cosine annealing algorithm can converge faster because it can use a smaller learning rate to fine tune model parameters in the later stages of training. The cosine annealing algorithm can avoid oscillations caused by rapid gradient descent during the training process, thereby improving the training stability and generalization ability of the model. The cosine annealing algorithm needs only set a small number of hyperparameters such as maximum learning rate, maximum number of iterations and minimum learning rate, making it easier to adjust and optimize the model.

2.4. WuC-Adam algorithm

When using optimization algorithms, it is necessary to consider the various parameters of the algorithm, and learning rate is a key link. Learning rate needs to consider multiple factors. Generally, different learning rate values need to be set during different training periods. The technology used by Adam is to adjust the adaptive gradient while ensuring that the learning rate does not decay too fast. To achieve this goal, Adam uses exponential weighted averaging for gradients and the square of gradients, which is actually a cumulative calculation of historical gradients. This can effectively reduce the impact of historical gradients on the current gradient. In the initial stage of iteration, due to the acceleration of the first-order momentum of the gradient, Adam converges quickly, resulting in significant oscillations near the optimal value, exhibiting great instability; and when Adam falls into a local optimum, it is also difficult to jump out of the local optimum situation.

In this article, we propose a new adaptive learning rate algorithm for the above problems, which is an improved version of the Adam optimization algorithm that combines Warmup technology and the cosine annealing strategy. The specific steps of this algorithm are as follows:

Initialization: Set the initial learning rate lr and maximum learning rate $lr_{\max}$, with both first-order momentum m and second-order momentum v initialized to 0, and time step t = 0.

Warmup stage: In the first warmup iteration steps of the early training stage, the learning rate linearly increases from a small value to $lr_{\max}$ according to Warmup technology. This helps the model to explore stably in the early stages of training.

Cosine annealing stage: After the preheating stage, starting from the warmup+1 iteration step, the learning rate is periodically adjusted according to the cosine annealing strategy. Specifically, the learning rate increases or decreases according to the curve of the cosine function within each cycle. The cosine annealing strategy makes the model more stable in the later stages of training, which helps to improve the convergence performance of the model.

Parameter update: In each iteration step, calculate the gradient g_t and each parameter θ_t, and update the parameters according to Adam's update formula.

During the training process, the relationship between the current number of steps and the set number of Warmup steps is determined to determine whether to use Warmup to gradually increase the learning rate or to perform periodic transformations on the learning rate through cosine annealing. Through this improvement, the learning rate can be increased in the early stages of training to accelerate convergence, and the search space can be refined in the subsequent stages of training to improve accuracy. This can effectively improve the model's generalization ability and avoid falling into local optima.

Changing from large to small: As the weights of the model are randomly initialized at the beginning of training, choosing a larger learning rate may lead to instability (oscillation) of the model. Choosing to preheat the learning rate can reduce the learning rate within the first few epochs or steps of training. Under the preheat learning rate, the model can gradually stabilize, After the model is relatively stable, choose a pre-set college enrollment rate for training, so that the convergence speed of the model becomes faster and the effect of the model is better.

Algorithm 1

```
1: Input:  initial point $x_0$, first moment decay $\beta_1$, second moment decay $\beta_2$, regularization
 constant $\varepsilon$
2: Initialize $m_0 = 0$ and $v_0 = 0$, $lr$, $lr_{\max}$, warm$up$, $T_0$
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:     $\lambda \leftarrow lr_{\max} / lr$
10:  If  $step \le warmup \times \lambda$
11:        $lr \leftarrow step / warmup \times lr$
12:        If  $lr > lr_{\max}$
13:        break
14:  else do
15:        $lr \leftarrow lr + \frac{1}{2}(lr_{\max} - lr)[1+\cos(\text{step}-(\text{warmup}\lambda / T_0)\pi)]$
16: $x_t = x_{t-1} - lr\,\hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$
end for
Return  $x_t$
```

Changing from small to large: When we train normally, a decrease in learning rate helps with better convergence. When the model learns to a certain extent, the distribution of the model becomes more stable. If a higher learning rate is used, it will disrupt this stability and cause significant fluctuations in the network. It is now very close to the optimal point, and in order to approach this optimal point, a very small learning rate is necessary. The specific algorithm is shown in Algorithm 1.

3. Experimental design and results analysis

3.1. Experimental environment configuration

The experiment implemented the WuC-Adam algorithm based on the PyTorch deep learning framework, which was jointly improved by Warm up and cosine annealing algorithms. The specific versions of software and hardware mainly used in the experiment are shown in Table 1.

Table 1. Experimental software and hardware versions.

Software and Hardware	Version
Python	3.11
torch	2.0.1
torchvision	0.15.2
lightning	2.0.4
GPU	NVIDIA GeForce RTX3080

All experiments were conducted in the Ubuntu server environment with NVIDIA GeForce RTX3080 graphics card configurations, and the entire experimental code was written using the Python lighting framework. The Python language version was 3.11, the torch version was 2.0.1, the torch vision version was 0.15.2 and the lighting version was 2.0.4.

The experiment conducted image classification experiments on three common datasets, MNIST, CIFAR10 and CIFAR100, to test the performance of the WuC-Adam algorithm. Three datasets, MNIST is a grayscale image dataset of handwritten digits; CIFAR10 and CIFAR100 are color image datasets containing different types of items. The data volume, training set, testing set and validation set partitioning and data characteristics of the three datasets are shown in Table 2.

Table 2. Experimental dataset.

Data set	Number of samples	Training set	Test set	Validation set	Category	Data characteristics
MNIST	70,000	55,000	5000	10,000	10	few categories, grayscale images, complex features
CIFAR10	60,000	45,000	5000	10,000	10	few categories, color RGB images, complex features
CIFAR100	60,000	45,000	5000	10,000	100	multiple categories, color RGB images, complex features

3.2. Experimental results and analysis

The WuC-Adam algorithm integrates Warmup and cosine annealing strategies on the basis of Adam, solving the problems of non-convergence and easy falling into local optima caused by the fluctuation of Adam's learning rate, thereby improving the model's generalization ability. In order to comprehensively evaluate the performance advantages of the WuC-Adam algorithm, we selected SGD algorithm, Adagrad, RMSprop, Adam, as well as Nadam and Lion, two improved Adam based algorithms, for comparative experiments. Although the generalization ability of adaptive learning rate algorithms may not be as good as momentum SGD, we will comprehensively evaluate them by conducting multiple experiments on different datasets and learning rate conditions.

Table 3. Comparative experimental results of different optimization algorithms.

Data set	Optimization Algorithm	Accuracy	Loss
MNIST	SGD	94.28%	0.227
	Adagrad	95.91%	0.163
	RMSprop	98.24%	0.078
	Adam	98.18%	0.071
	Nadam	98.32%	0.072
	Lion	98.06%	0.075
	WuC-Adam	**98.87%**	0.081
CIFAR10	SGD	59.46%	1.133
	Adagrad	59.61%	1.283
	RMSprop	81.04%	1.035
	Adam	79.32%	0.951
	Nadam	79.52%	0.957
	Lion	86.52%	1.318
	WuC-Adam	**87.67%**	**0.916**
CIFAR100	SGD	18.56%	3.481
	Adagrad	34.72%	2.629
	RMSprop	49.40%	2.657
	Adam	48.66%	2.401
	Nadam	48.34%	**2.363**
	Lion	55.35%	3.681
	WuC-Adam	**58.88%**	2.852

Experimental setup:

1) Algorithm: The experiment used SGD, Adagrad, RMSporp, Adam, Nadam, Lion and WuC-Adam algorithms as experimental subjects for comparison.

2) Epochs: Set the epoch in the experiment to 100 to ensure that the algorithm is compared under the same number of training rounds.

3) Batch Size: Set the batch size to 128 to maintain consistency in the experiment.

4) Learning rate: The initial learning rate of all algorithms in the experiment is set to 0.0001, and the maximum learning rate of the WuC Adam algorithm is set to 0.01.

5) Multiple experiments: Due to the need to adjust the maximum and minimum learning rates of the WuC-Adam algorithm, we will conduct multiple experiments on each dataset and select the best result as the experimental result.

6) Experimental results: On the MNIST, CIFAR10 and CIFAR100 datasets, 15 comparative experiments were conducted on seven algorithms, and the average of the 15 results was taken as the final result to reflect the advantages of the WuC-Adam algorithm in solving the Adam problem and

improving generalization ability.

The WuC-Adam algorithm aims to improve the model's generalization ability and avoid falling into local optima. In order to test the performance of WuC-Adam algorithm on different neural networks and highlight the contrast of comparative experiments, we trained different types of neural networks on different datasets: Using a simple fully connected neural network for training on MNIST, a dataset composed of grayscale images; ResNet18 residual convolutional neural network is used on two datasets composed of color RGB images, CIFAR10 and CIFAR100. In this experiment, we use a cross dataset as an example to show the convergence process of three algorithms, and the learning rate settings of each algorithm are the same as Table 3. In order to comprehensively present the performance of WuC-Adam algorithm under different learning rate combinations, we designed three different sets of maximum and minimum learning rate combinations and conducted comparative experiments. In Figure 1, we demonstrate the performance of the WuC-Adam algorithm on the CIFAR10 dataset when combined with different learning rates.

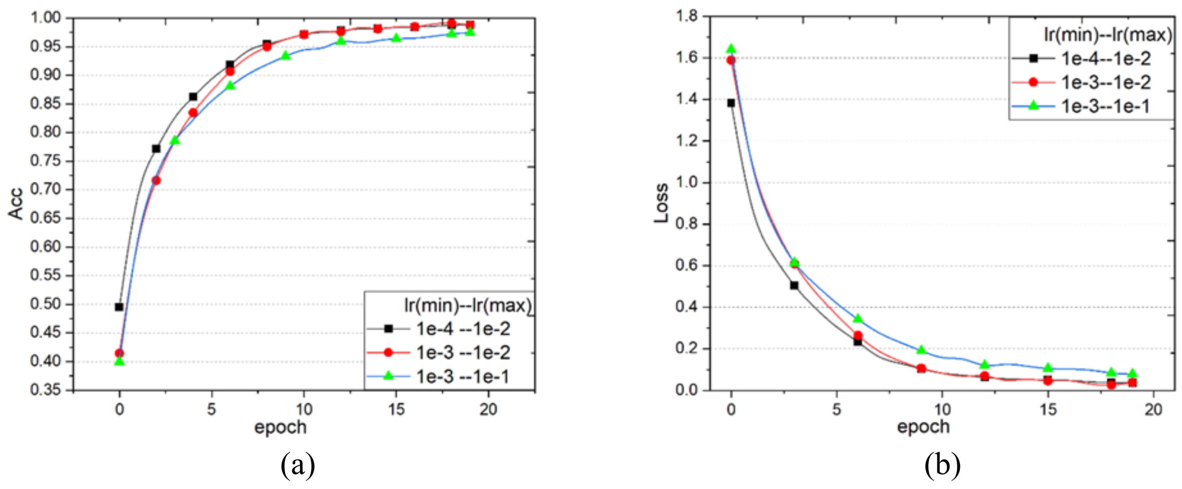


Figure 1. Performance comparison of different learning rate variation intervals on the MNIST dataset. The left figure (a) is a comparison of accuracy, and the right figure (b) is a comparison of loss values. The legend represents the range of variation from the given minimum learning rate to the maximum learning rate.

It can be seen that the optimal effect was achieved using a combination of parameters with a minimum learning rate of 1e-4 and a maximum learning rate of 1e-2. Two parameter combinations with a minimum learning rate set to 1e-3 perform less well in the early stages of training than the combination with a minimum learning rate of 1e-4. This phenomenon may be due to the fact that higher learning rates may lead to model instability in the early stages of training, and cause oscillations near extreme points. However, parameter combinations with a maximum learning rate of 1e-1 perform poorly, compared to two parameter combinations with a maximum learning rate of 1e-2. This may be due to the fact that an excessively high learning rate reduces the model's generalization ability when entering the cosine annealing stage in the later stages of training.

Taking the MNIST dataset as an example, Figure 2 shows the changes in learning rate during the training process using the WUC-Adam algorithm. It can be seen that learning first gradually increases to the maximum learning rate through Warmup, and then uses the cosine annealing strategy to make the learning rate exhibit periodic changes in the cosine function image.

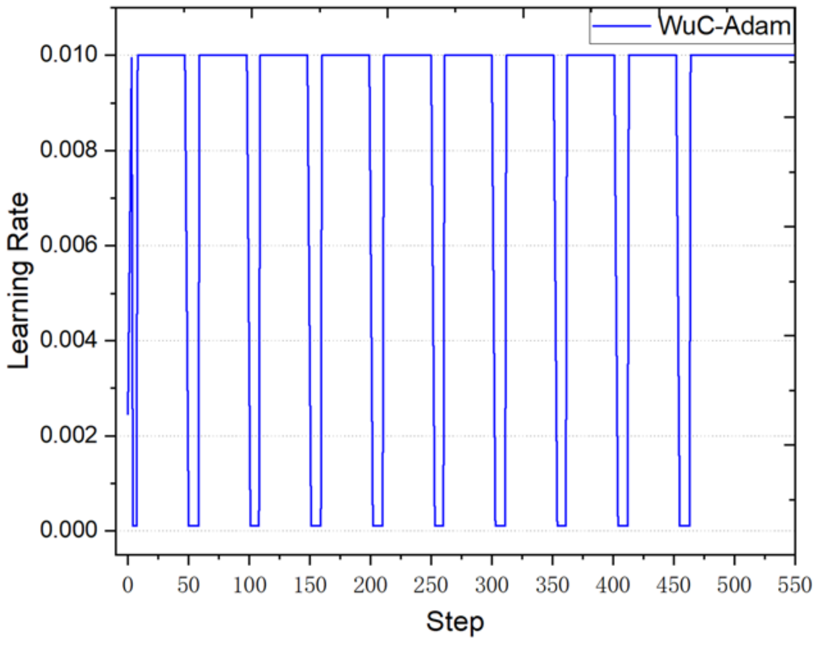


Figure 2. The variation curve of learning rate during WuC-Adam training process.

The reason why the learning rate change curve in Figure 2 presents a cosine like change rather than a complete cosine form change is because the cosine annealing step in the WuC-Adam algorithm uses a modified cosine annealing formula, which varies the learning rate based on the number of steps run, rather than using the proportion of cycles in the original cosine annealing formula. Due to the large number of steps run in each cycle, the learning rate will remain at its maximum for a period of time before continuing with cosine transformation.

Figures 3 and 4 show the training results of seven algorithms on the MNIST dataset and CIFAR10 dataset, respectively.

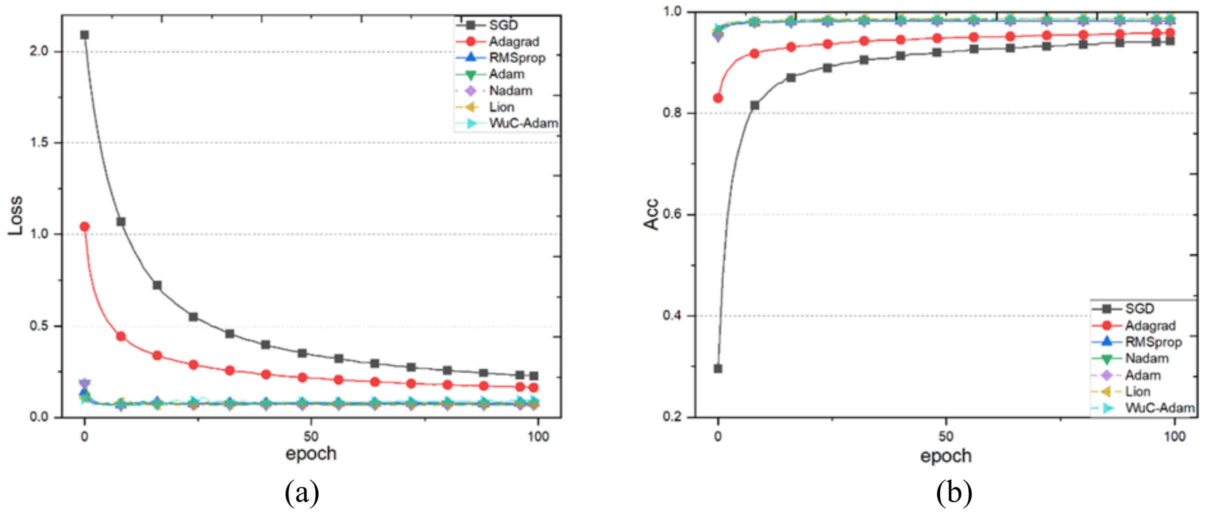
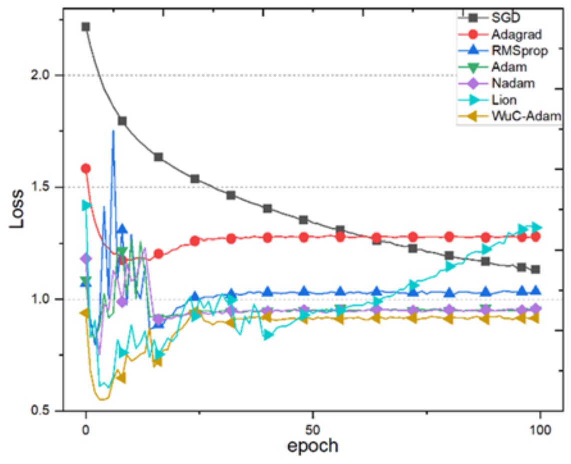
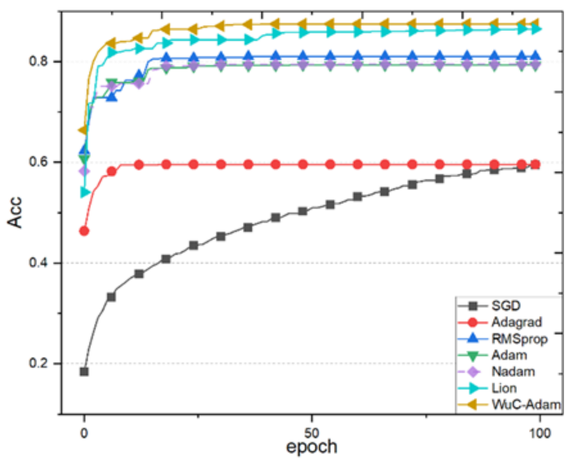


Figure 3. Performance comparison of various algorithms on the MNIST dataset. The left figure (a) shows a comparison of loss values, while the right figure (b) shows a comparison of accuracy.



(a)



(b)

Figure 4. Performance comparison of various algorithms on the CIFAR10 dataset. The left figure (a) shows a comparison of loss values, while the right figure (b) shows a comparison of accuracy.

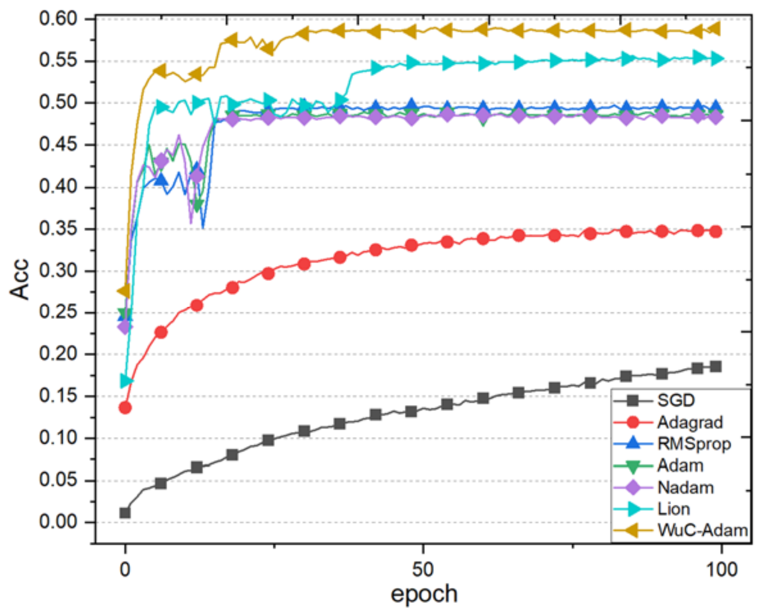


Figure 5. Accuracy of different algorithms on the CIFAR100 dataset.

Based on the results presented in Figure 5, we compared the accuracy of various algorithms with increasing training rounds on the CIFAR-100 dataset. It is evident that the accuracy of SGD is significantly lower than the other six algorithms. Furthermore, the WuC-Adam algorithm exhibits higher accuracy in the early stages of training, far surpassing other algorithms, especially compared to the two improved Nadam and Lion algorithms based on the Adam algorithm, the WuC-Adam algorithm performs better. This advantage can be attributed to the preheating strategy adopted by WuC-Adam, which gradually improves the learning rate in the early stages of training, enhances the stability of the model and significantly improves accuracy in a short period of time. It is worth noting that in the later stage of training, the WuC-Adam algorithm maintained stability without any fluctuations, and at the end, its accuracy curve showed an upward trend.

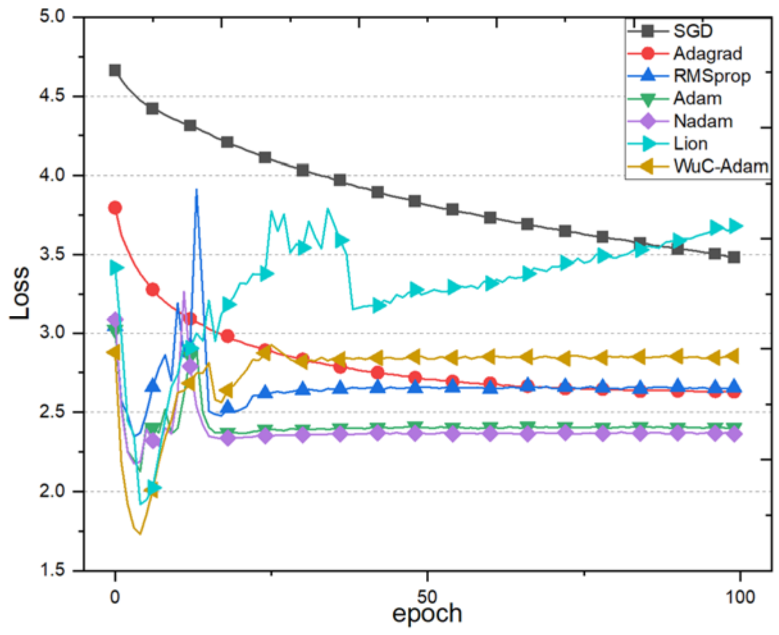


Figure 6. Loss values of different algorithms on the CIFAR100 dataset.

Based on the results presented in Figure 6, we compared the loss values of various algorithms on the CIFAR-100 dataset as the number of training rounds increased. The overall loss value of SGD algorithm is significantly higher than the other six algorithms. The WuC-Adam algorithm exhibits faster convergence speed than other algorithms in the early stages of training, but its loss value is slightly higher in the later stages of training than other algorithms except SGD and Lion. The reason for this result may be due to the fact that in the later stages of training, the WuC-Adam algorithm performs cosine annealing operations to avoid getting stuck in local optima by periodically changing the learning rate. In addition, compared to the Lion algorithm which uses a sign function to handle updates, the WuC-Adam algorithm has a superior final loss performance.

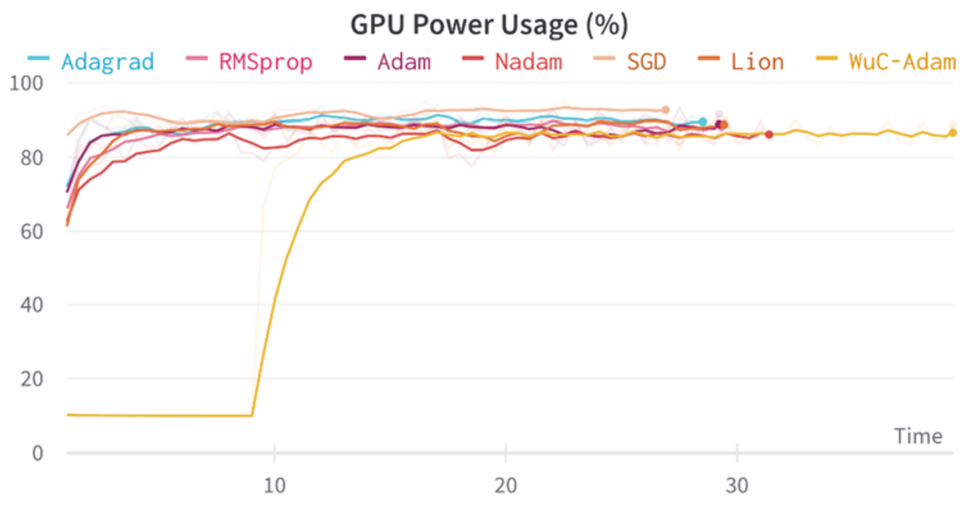


Figure 7. GPU occupancy rates of seven algorithms.

Figure 7 shows the curve graph of the time and GPU usage required for training each algorithm on the CIFAR-100 dataset after smoothing.

The GPU occupancy rate of the WuC-Adam algorithm is significantly lower than the other six algorithms throughout the entire training process, especially in the early Warmup stage of training, where the GPU occupancy rate is less than 20%. In the subsequent cosine annealing stage, the GPU occupancy rate of the WuC-Adam algorithm is lower than other algorithms. However, considering that the WuC-Adam algorithm adopts a constantly changing learning rate during the training process, its training time slightly increases compared to other algorithms.

In summary, compared to the other six algorithms, especially the Adam algorithm, WuC-Adam exhibits better accuracy and more stable convergence on the CIFAR100 dataset, which further validates the advantages of WuC-Adam, especially in solving learning rate related problems and improving generalization ability.

According to the experimental results in Table 3, in the comparative experiments of the three datasets, WuC-Adam showed the highest accuracy, highlighting its overall superiority on the MNIST, CIFAR-10 and CIFAR-100 datasets, fully confirming its excellent generalization ability. This also indicates that the WuC-Adam algorithm can achieve significant results on both simple black and white image datasets and colored RGB image datasets. In addition, after 100 rounds of training, the overall convergence speed of WuC-Adam surpasses other algorithms, especially compared to the standard Adam algorithm, achieving significant optimization. This indicates that WuC-Adam has significantly improved convergence efficiency and the ability to jump out of local optima, not only emphasizing its excellent performance in generalization, but also strengthening its universal applicability on different datasets and tasks. It is worth noting that WuC-Adam exhibits better performance on both fully connected networks (FCNN) and ResNet, further proving that the superiority of this algorithm is not limited by specific neural network types, but has achieved significant results on various network structures.

The experimental results fully validate the advantages of WuC-Adam, especially in terms of generalization ability and convergence speed. It achieved the highest accuracy on different datasets and neural networks, and achieved the lowest loss value during the training process, further proving the effectiveness and applicability of WuC-Adam, making it a potential optimization algorithm that can be used to improve the performance and generalization ability of deep learning models.

4. Conclusions

Compared to traditional Adam algorithms and improved Adam-based algorithms Nadam and Lion, the proposed WuC-Adam algorithm in this paper effectively mitigates the issue of traditional Adam algorithms easily converging to local optima, thereby enhancing the model's generalization ability. However, it is noteworthy that WuC-Adam has also led to a certain increase in its memory requirements. To comprehensively assess the performance of WuC-Adam, we conducted comparative experiments using three widely recognized datasets: MNIST, CIFAR-10 and CIFAR-100. Our evaluation involves comparisons among WuC-Adam, SGD, the Adam algorithm, Adam's predecessors Adagrad and RMSprop algorithms, as well as the Nadam and Lion algorithms improved based on the Adam algorithm. The experiments consistently demonstrate that WuC-Adam achieved higher accuracy and superior convergence on these datasets, exhibiting significant advantages over other algorithms. These results underscore the superiority of WuC-Adam in addressing the issues of traditional Adam and emphasize its pivotal role in achieving substantial performance improvements across multiple datasets.

The improved algorithm WuC-Adam, proposed in this article, effectively addresses a range of issues associated with Adam, such as sensitivity to local optima, overfitting and learning rate fluctuations. This is achieved through the clever combination of preheating and cosine annealing strategies. In our experiments, we empirically validated the superiority of WuC-Adam over traditional Adam on multiple datasets, showcasing notable improvements in accuracy and convergence performance. These experimental results underscore the potential significance and practical value of WuC-Adam, presenting a viable option for enhancing the performance and generalization ability of deep learning models.

Use of AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The work was supported by Basic Research Project for Higher Education Institutions of Liaoning Provincial Department of Education (General Project); Shenyang University of Technology; Research on optimization design of fan cone angle based on fluid physics methods; Project number: LJKZ0159; Liaoning Provincial Education Science 14th Five Year Plan, Research on the Construction of New Artificial Intelligence Technology and High Quality Education Service Supply System, 2023–2025, Project Number: JG22DB488; Ministry of Education's "Chunhui Plan", Research on Optimization Model and Algorithm for Microgrid Energy Scheduling Based on Biological Behavior, Project Number: 202200209; Basic Research Project of Liaoning Provincial Department of Education "Training and Application of Multimodal Deep Neural Network Models for Vertical Fields" Project Number: JYTMS20231160.

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature*, **521** (2015), 436–444. https://doi.org/10.1038/nature14539
2. A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, B. Recht, The marginal value of adaptive gradient methods in machine learning, in *Advances in Neural Information Processing Systems*, **30** (2017).
3. S. J. Reddi, S. Kale, S. Kumar, On the convergence of Adam and beyond, preprint, arXiv:1904.09237.
4. I. Loshchilov, F. Hutter, Fixing weight decay regularization in Adam, 2018. Available from: https://openreview.net/forum?id=rk6qdGgCZ.
5. Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, et al., Large batch optimization for deep learning: Training BERT in 76 minutes, preprint, arXiv:1904.00962.
6. S. J. Reddi, S. Kale, S. Kumar, On the convergence of Adam and beyond, preprint arXiv:1904.09237.

7. L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, et al., On the variance of the adaptive learning rate and beyond, preprint, arXiv:1908.03265.
8. J. Zhuang, T. Tang, Y. Ding, S. C. Tatikonda, N. Dvornek, X. Papademetris, et al., AdaBelief Optimizer: Adapting stepsizes by the belief in observed gradients, in *Advances in Neural Information Processing System*, **33** (2020), 18795–18806.
9. W. Ilboudo, T. Kobayashi, K. Sugimoto, Robust stochastic gradient descent with student-t distribution based first-order momentum, *IEEE Trans. Neural Networks Learn. Syst.*, **33** (2020), 1324–1337. https://doi.org/10.1109/TNNLS.2020.3041755
10. T. Dozat, *Incorporating Nesterov Momentum into Adam*, 2016. Available from: https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ.
11. L. Luo, Y. Xiong, Y. Liu, X. Sun, Adaptive gradient methods with dynamic bound of learning rate, preprint, arXiv:1902.09843.
12. G. Mordido, P. Malviya, A. Baratin, S. Chandar, Lookbehind optimizer: k steps back, 1 step forward, preprint, arXiv:2307.16704.
13. M. Reyad, A. M. Sarhan, M. Arafa, A modified Adam algorithm for deep neural network optimization, *Neural Comput. Appl.*, **2023** (2023), 1–18. https://doi.org/10.1007/s00521-023-08568-z
14. X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, et al., Symbolic discovery of optimization algorithms, preprint, arXiv:2302.06675.
15. R. Tian, A. P. Parikh, Amos: An Adam-style optimizer with adaptive weight decay towards model-oriented scale, preprint, arXiv:2210.11693.
16. M. Liu, D. Yao, Z. Liu, J. Guo, J. Chen, An improved Adam optimization algorithm combining adaptive coefficients and composite gradients based on randomized block coordinate descent, *Comput. Intell. Neurosci.*, **2023** (2023), 1–13. https://doi.org/10.1155/2023/4765891
17. X. Xie, P. Zhou, H. Li, Z. Lin, S. Yan, Adan: Adaptive Nesterov momentum algorithm for faster optimizing deep models, preprint, arXiv:2208.06677.
18. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint, arXiv:1412.6980.
19. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2015), 770–778.

AIMS Press

©2024 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)