algorithms

MDPI

Article

A Linear Interpolation and Curvature-Controlled Gradient Optimization Strategy Based on Adam

Haijing Sun [1], Wen Zhou [2], Yichuan Shao [1,*], Jiaqi Cui [2], Lei Xing [3], Qian Zhao [4] and Le Zhang [1]

[1] School of Intelligent Science and Engineering, Shenyang University, Shenyang 110044, China; sunhaijing@syu.edu.cn (H.S.); zhangle@syu.edu.cn (L.Z.)
[2] School of Information Engineering, Shenyang University, Shenyang 110044, China; zhouwend@outlook.com (W.Z.); q28497187@outlook.com (J.C.)
[3] School of Chemistry and Chemical Engineering, University of Surrey, Surrey GU2 7XH, UK; l.xing@surrey.ac.uk
[4] School of Science, Shenyang University of Technology, Shenyang 110044, China; qzhao@uow.edu.au
* Correspondence: shaoyichuan@syu.edu.cn

Abstract: The Adam algorithm is a widely used optimizer for neural network training due to efficient convergence speed. The algorithm is prone to unstable learning rate and performance degradation on some models. To solve these problems, in this paper, an improved algorithm named Linear Curvature Momentum Adam (LCMAdam) is proposed, which introduces curvature-controlled gradient and linear interpolation strategies. The curvature-controlled gradient can make the gradient update smoother, and the linear interpolation technique can adaptively adjust the size of the learning rate according to the characteristics of the curve during the training process so that it can find the exact value faster, which improves the efficiency and robustness of training. The experimental results show that the LCMAdam algorithm achieves 98.49% accuracy on the MNIST dataset, 75.20% on the CIFAR10 dataset, and 76.80% on the Stomach dataset, which is more difficult to recognize medical images. The LCMAdam optimizer achieves significant performance gains on a variety of neural network structures and tasks, proving its effectiveness and utility in the field of deep learning.

Keywords: deep learning; Adam algorithm; curvature-controlled gradient; linear interpolation; LCMAdam algorithm

Citation: 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. https://doi.org/10.3390/a17050185

Academic Editor: Patrizia Beraldi

Received: 6 April 2024
Revised: 21 April 2024
Accepted: 28 April 2024
Published: 29 April 2024

1. Introduction

Deep learning employs multi-layer neural networks to extract features from large-scale data and is widely used in fields such as computer vision and natural language processing. Optimization algorithms, such as the Adam algorithm, are crucial in deep learning. Although the algorithm performs well in many cases, there are still limitations and challenges. In 2014, Kingma and Ba et al. [1] proposed the Adam algorithm, a stochastic optimization method that combines first-order and second-order moment estimations of the gradient. The algorithm provides a theoretical basis and detailed step-by-step instructions for implementation. Chakrabarti et al. [2] proposed a control theory-based framework for designing adaptive gradient optimizers to address the Adam algorithm's excessive flexibility. Kuangyu Ding et al. [3] introduced the Adam-family method to solve the problem of insufficient weight decay in the Adam algorithm by studying the decoupled weight decay term. The Adam-plus algorithm [4] addresses the performance issues of the Adam algorithm when dealing with large-scale data and high dimensionality by adjusting the learning rate. This adjustment improves the stability and generalization ability of the optimization algorithm. Kavosh Asadi et al. [5] conducted an empirical study on resetting the optimizer in deep reinforcement learning. They proposed a method to reset the optimizer by improving the Adam algorithm with respect to the maladaptation of the learning rate

in the Adam algorithm. Abel C. H. Chen [6] explored the optimized values of each hyper-parameter in various gradient descent algorithms to provide a deeper understanding and guidance for tuning the parameters of optimization algorithms. The study on the EAdam algorithm [7] addressed the effect of the parameter on performance in the Adam optimizer and improved the Adam algorithm's performance to some extent in some cases. Lu Xia et al. [8] proposed a fast adaptive gradient method called the AdamL algorithm to solve the global problem of learning rate in the Adam algorithm. Ran Tian and Ankur P. Parikh [9] proposed Amos, an optimizer with adaptive weight decay that addresses the deficiency of the Adam optimizer in terms of model size. Byeongho Heo et al. [10] proposed an improved version of the AdamP optimizer, which effectively mitigates the shortcomings of the Adam algorithm in high-dimensional optimization problems. The aim is to address the performance degradation of the Adam algorithm on scale-invariant weights. T. Dozat [11] attempted to address the issue of the Adam algorithm's potential poor performance in non-convex optimization problems by integrating the Nesterov momentum approach into the methodology of the Adam optimization algorithm to compensate for its limitations when dealing with highly non-uniform loss function surfaces. Zhang et al. [12] proposed the WuC-Adam algorithm to address the instability of the learning rate and slow convergence speed in the Adam algorithm. Building on this work, Yiming Jiang et al. [13] proposed UAdam to solve the performance issues of the Adam algorithm in non-convex optimization, providing a more general and flexible optimization method. Yan Pan and Yuanzhi Li [14] investigated why Adam converges faster than SGD in the Transformer model. Yichuan Shao et al. [15] improved the Adam algorithm to address the issue of unstable convergence speed in deep learning. They proposed a multi-scale lightweight neural network for detecting steel surface defects and later developed a new PyTorch-based method for detecting dust on the surface of photovoltaic panels [16]. Liu Hailiang [17] proposed an adaptive gradient method based on energy and momentum, which offers a new approach for optimizing the algorithm. Sedjro S. Hotegni et al. [18] investigated the multi-objective optimization problem in sparse deep multi-task learning and proposed new perspectives and methods for optimization in multi-task learning. The StochGradAdam algorithm [19] improved the Adam algorithm by introducing stochastic gradient sampling to enhance the training speed and generalization ability, particularly when dealing with large-scale data. In recent years, deep learning techniques have been widely used in healthcare. Hussam N Fakhouri et al. [20] introduced a cognitive DL retinal vascular splitting hybrid model after synergistically combining the deep learning capabilities of the U-Net architecture with a range of advanced image processing techniques to address the complexity of retinal images. Zhipeng Liu et al. [21] utilized dendritic neurons to address the challenges of medical imaging segmentation. Bujny et al. [22] proposed a deep learning algorithm for accurately localizing coronary arteries in cardiac non-contrast CT images, addressing a gap in the field of coronary artery segmentation.

This study provides insights into the convergence challenges of the Adam algorithm. The research found that, although the Adam algorithm performs well in most cases, its performance may suffer in certain situations, such as when the dataset is highly correlated or when the learning rate decays inappropriately. To address this challenge, two key techniques, the curvature-controlled gradient technique and linear interpolation technique, have been investigated and proposed. LCMAdam algorithm relates the values of second-order moment estimation and first-order moment estimation to the degree of change in the gradient, which reflects the curvature of the parameter space. This strategy can be considered a measure of the curvature of the gradient, hence its name: curvature-controlled gradient. During gradient updates, the rate of the update is controlled by adjusting the size of the gradient based on the curvature of the current gradient. By adjusting the variation in the curvature size, the optimizer can effectively avoid drastic fluctuations in the gradient, thus stabilizing the model training process. Smoothing the gradient update has the effect of allowing the optimizer to search through the parameter space more efficiently and converge to a locally optimal solution faster. Also, linear interpolation is used to fine-tune the learning

rate based on predefined hyperparameters. This modification enables the model to adjust more flexibly to the changes in data distribution, enhancing the model's robustness and effectiveness during the training process. As a result, the model's performance and stability during training are improved.

2. LCMAdam Algorithm Design

2.1. Adam Optimization Algorithm

The Adam algorithm is an optimization algorithm that adapts the learning rate using first-order and second-order moment estimations of the gradient. The first-order moment estimates the gradient direction, while the second-order moment estimates the gradient magnitude. By combining these two estimators, the Adam algorithm can dynamically adjust the learning rate's size to better accommodate changes in different parameters. The algorithm introduces a bias correction mechanism to solve the problem caused by a too-high learning rate at the initial time. Also, the parameters used for both are defined as first-order moments, m_t, and second-order moments, v_t, respectively. The moment estimation vectors for the gradient in step t are shown in Equations (1) and (2):

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

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

where β_1 and β_2 are the exponential decay rates of the first-order and second-order moments, respectively, and g_t is the gradient of the step, t. The formula is shown in (3):

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

The parameter values updated by the neural network model at the t-th iteration are denoted by θ_t, the usage loss function computed by the neural network model at the t-th iteration is denoted by f_t, and the rate of change in the neural network model to the loss function relative to the parameters at the t-th iteration is denoted by $\nabla_\theta f_t(\theta_t)$.

To address bias in estimating first- and second-order moments when the decay rate is very small in the initial stage, the Adam algorithm corrects these moments to optimize parameter updates more efficiently. The formulas are shown in (4) and (5):

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

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

After each iteration, Equation (6) is used to calculate the update of parameter θ.

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

where θ represents the value of the learning rate and ε represents a sufficiently small but greater-than-zero value that prevents a denominator of zero.

Although the Adam algorithm performs well in many cases, this algorithm is very sensitive to the learning rate setting, which requires fine-tuning its hyperparameters. In some cases, these hyperparameter settings may cause the algorithm to fail to reach the globally optimal solution, especially when dealing with certain non-convex optimization problems.

2.2. Curvature-Controlled Gradient Strategy

Controlling and adjusting the difference between the squares of the second-order moment estimates and the first-order moment estimates indirectly reflects information about the curvature of the gradient in the parameter space. This allows optimization algorithms to better adjust the magnitude and direction of the gradient, updating the parameters more efficiently and avoiding the possibility of converging to a locally optimal solution. The stability of the optimization process can be improved by using the minimum

value to adjust the size of the gradient, which helps to control the oscillations and volatility of the gradient. This results in faster convergence to the optimal solution during the training process.

When the gradient's curvature is large, the gradient changes faster, at which point the curvature is reduced to make the gradient update smoother. Conversely, when the gradient's curvature is smaller, the gradient changes more slowly, so increases the curvature to speed up the gradient update.

First, the difference between the squared second-order moment estimate of the gradient and the first-order moment estimate is computed, and the resulting value is defined as the curvature. The formula is shown in (7):

$$curvature = v_t - m_t^2 \tag{7}$$

This curvature reflects the trend of the gradient change. A larger curvature indicates a sharp change in the gradient, while the opposite indicates a smoother change in the gradient. To limit the curvature magnitude, it was restricted to a range of 1.0. This is performed to balance the parameter update and prevent the gradient from becoming too large or too small. If the curvature exceeds this range, it is adjusted to 1.0; otherwise, the original value is maintained. The formula is presented in (8):

$$GR = \min(|curvature|, 1.0) \tag{8}$$

The gradient is normalized to a unit vector, which maintains the direction of the gradient while scaling its length to 1.0. This stabilizes and facilitates the adjustment of the gradient update magnitude. This process ensures that the proper gradient size is maintained when updating the parameters. The LCMAdam algorithm can balance gradient variations more efficiently and converge to the optimal solution faster by dynamically adjusting the curvature.

2.3. Linear Interpolation Strategy

The adaptive adjustment of the learning rate using the linear interpolation technique can ensure that historical information is fully utilized during the training process and the size of the learning rate is dynamically adjusted according to the current situation. This adaptive adjustment can effectively avoid the performance degradation caused by a too large or too small learning rate, thus improving the stability and efficiency of the optimization algorithm in the training process.

Drawing on the concept of the microelement method, the learning rate is adjusted. This approach replaces the use of a fixed learning rate. A learning rate value between the maximum and minimum learning rates was calculated using a linear interpolation method based on the relative position of the current training step in the cycle, as shown in Equation (9):

$$lr = initial\_lr - (initial\_lr - \min\_lr) \cdot (t / period) \tag{9}$$

The variables used in this formula are: lr for the learning rate after approximation, $initial\_lr$ for the maximum value that can be reached by the learning rate, $\min\_lr$ for the minimum value of the learning rate, t for the relative position in the cycle in which it is currently located, and $period$ for the period of interpolation. The purpose of this interpolation is to dynamically adjust the learning rate during the training process to improve the model's convergence or adaptation to the training data. As training progresses, the learning rate is tuned to enhance the performance and stability of the optimization algorithm.

2.4. LCMAdam Algorithm

The selection of the learning rate is a critical aspect of neural network training. The Adam algorithm adjusts the learning rate by adaptive gradients and uses exponentially weighted averaging to balance historical and current gradients for smoother parameter

updates. However, during the initial stages of training, the Adam algorithm may cause rapid convergence and oscillations, as well as the issue of being trapped in a local optimum point, which can affect the stability and speed of training. To address the above problems, an LCMAdam algorithm is proposed, which is improved on the basis of the Adam algorithm, and the specific steps of the algorithm are as follows.

To initialize, set the initial learning rate to lr, the maximum learning rate to $initial\_lr$, the minimum learning rate to $\min\_lr$, the maximum change in the learning rate to $\max\_lr\_change$, the relative position, t, in the cycle and the cycle length to $period$, and initialize the first-order momentum to m_t and the second-order momentum to v_t as zero.

Curvature adjustment stage: The curvature of the gradient was calculated, a correction factor, GR, was calculated based on the curvature value of the gradient, and it was ensured that the correction factor did not exceed 1.0. Then, the gradient was multiplied by this correction factor, GR, so that the magnitude of the gradient was regulated by the curvature. Finally, the gradient was processed, i.e., the gradient vector was divided by its parameter to ensure stability and convergence during the gradient update process.

Learning rate adjustment phase: After obtaining the value set by initialization, the value of the current learning rate was determined by calculating the relative position, t, of the current step in the cycle, and the learning rate was adjusted according to the global number of steps by linear interpolation.

Parameter update: By controlling the variation range of the learning rate, it ensures that the learning rate does not exceed the specified maximum variation value during the training process, and that the learning rate does not fall below the specified minimum value.

The LCMAdam optimizer introduces a linear interpolation adjustment and a curvature control gradient strategy. This allows the model to adaptively adjust the learning rate based on the actual situation, accelerating convergence speed, and improving training efficiency. This section comprehensively discusses the learning rate and gradient adjustment, which makes the LCMAdam algorithm more flexible in coping with different data situations when training deep learning models. The algorithm for this approach is shown in Algorithm 1.

Algorithm 1: LCMAdam

Input : initial point θ_0, first $-$ moment decay β_1, sec ond $-$ moment decay β_2, regularization constant ε
Initialize $m_0 = 0$ and $v_0 = 0$, lr, $initial\_lr$, $\min\_lr$, t, $period$
For 0 to $period$ perform
$g_t = \nabla f_t(\theta_{t-1})$
$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
$\hat{m}_t = m/(1-\beta_1^t)$
$\hat{v}_t = v/(1-\beta_2^t)$
$curvature = v_t - m_t^2$
$momentum = \min(|curvature|, 1.0)$
$t = step\%period$
$lr = initial\_lr - (initial\_lr - \min\_lr) \cdot (t/period)$
$lr = \max(\min\_lr, lr - \max\_lr\_change)$
$\theta_{t+1} = \theta_t - lr \cdot \hat{m}_t/(\sqrt{\hat{v}} + \varepsilon)$
End for
Return θ_t

3. Experimental Design and Analysis of Results

3.1. Experimental Environment Configuration

The experiments utilized the Python programming language and the PyTorch deep learning framework. The improved LCMAdam algorithm was also employed. Details regarding the main software versions used in the experiments are shown in Table 1.

Table 1. Main software versions in the experiment.

Software	Version
Python	3.10
Torch	2.0.1
Torchvision	0.15.0
Lightning	2.1.2
Cuda	Cu118

The experimental code was written using the Lightning framework. The Python language version used was 3.10, with Torch version 2.0.1 and torchvision version 0.15.0. Lightning version 2.1.2 was also used. Experiments were conducted on three datasets: MNIST, CIFAR10, and Stomach. MNIST is a grayscale image dataset of handwritten digits, with each image being 28 × 28 pixels in size. CIFAR10 is a color image dataset that contains different types of image classifications, and each image is 32 × 32 pixels in size. Stomach is a medical imaging dataset with a total of 1885 images containing eight classifications. Before training, the images were preprocessed into sizes of 224 × 224. Table 2 displays the experimental dataset.

Table 2. Experimental dataset.

Dataset	Number of Samples	Training Set	Test Set	Validation Set	Category	Data Characteristics
MNIST	70,000	55,000	5000	10,000	10	Image size unification, data diversity, moderate data volume
CIFAR10	60,000	45,000	5000	10,000	10	RGB image, relatively small scale, small image size
Stomach	1885	900	485	500	8	RGB image, few categories, recognition difficulty

3.2. Experimental Results and Analysis

In order to comprehensively assess the performance advantage of the LCMAdam algorithm over other optimization algorithms, a series of classical optimization algorithms were selected for comparative experiments (the initial conditions were the same for each experiment, and the comparison was only the difference in the accuracy and loss values between the different algorithms), including SGD, Adagrad, Adalelta, Adam, Nadam, and StochGradAdam. Multiple experiments were conducted with different datasets and learning rate conditions to comprehensively evaluate their performance. The comparison of the experimental results of different optimization algorithms is shown in Table 3.

The experimental setup was as follows:

(1) Algorithm selection: the SGD, Adagrad, Adadelta, Adam, Nadam, StochGradAdam, and LCMAdam algorithms were selected for comparison.
(2) Number of training rounds: to ensure a fair comparison, the number of training rounds (Epoch) in the experiment was set to 100.
(3) Batch size: to maintain consistency in the experiment, we set the batch size to 128.
(4) Learning rate setting: the initial learning rate of all algorithms was set to 0.001. The maximum learning rate of the LCMAdam algorithm was set to 0.005, the minimum learning rate was set to 0.001, and the maximum change of learning rate was set to 0.001.
(5) Other hyperparameters: the cycle length (period) in the LCMAdam optimizer was set to 1.1.
(6) Multiple experiments: since the LCMAdam algorithm needed to adjust the cycle length and the maximum variation in the learning rate, we conducted multiple experiments on each dataset and selected the best results as the experimental results.

(7) Experimental results: 17 comparative experiments were conducted on MNIST, CIFAR10, and Stomach datasets, and the average of the 17 results was taken as the final result to reflect the advantages of the LCMAdam algorithm in solving the Adam algorithm problem and improving the generalization ability.

Table 3. Comparison of experimental results of different optimization algorithms.

Dataset	Optimization Algorithm	Accuracy	Loss
MNIST	SGD	97.18%	0.092
	Adagrad	98.68%	0.067
	Adalelta	96.42%	0.1297
	Adam	98.64%	0.060
	NAdam	98.44%	0.0638
	StochGradAdam	98.11%	0.0695
	LCMAdam	98.49%	**0.060**
CIFAR10	SGD	48.32%	1.466
	Adagrad	29.87%	1.95
	Adalelta	25.97%	1.973
	Adam	68.49%	1.22
	NAdam	69.85%	1.685
	StochGradAdam	69.06%	1.032
	LCMAdam	**75.20%**	1.28
Stomach	SGD	57.60%	1.171
	Adagrad	69.99%	0.9581
	Adalelta	56.00%	1.665
	Adam	74.20%	0.883
	NAdam	74.80%	0.817
	StochGradAdam	70.20%	0.9307
	LCMAdam	**76.80%**	**0.66**

The LCMAdam algorithm aims to improve the generalization ability of the model and solve the oscillation problem caused by the algorithm's non-convergence. To evaluate the performance of the LCMAdam algorithm on various neural networks, different datasets were used for training. The MNIST dataset, consisting of grayscale images with a size of 28 × 28 pixels each, was used to train a simple fully connected neural network. On the other hand, the CIFAR10 and Stomach datasets were used to train a MobileNetV2 lightweight neural network. These two datasets contain color RGB images. To compare the algorithms, we selected cross datasets and observed their convergence on all three datasets. The learning rate settings are the same as in Table 3. In order to comprehensively demonstrate the performance of the LCMAdam algorithm under different combinations of hyperparameter settings, three different combinations of maximum variation values of the learning rate and cycle lengths were designed and comparative experiments were conducted. In Figures 1 and 2, the performance of the LCMAdam algorithm is shown for different combinations of hyperparameter settings on the MNIST and CIFAR10 datasets. The algorithm's performance varies depending on the values of *initial_lr* and min_*lr*, max_*lr_change* and *period* (when validating a dataset, the best performance parameter value is taken for the other dataset).

The minimum learning rate parameter, min_*lr*, is held constant at 0.001. Based on the observations in Figure 1, the LCMAdam algorithm performs best when the learning rate parameter range is increased from 0.001 to 0.005. When the maximum learning rate parameter was set to 0.05 and 0.01, the program lagged behind the maximum learning rate parameter of 0.005 in the later stages of training, although it slightly outperformed the maximum learning rate parameter of 0.005 in the early stages of training. This phenomenon may be due to the learning rate range being set too large, which can result in an inappropriate value of the learning rate during later training phases, ultimately affecting

the search for the local optimal solution. The initial learning rate value remains unchanged at 0.001, and by combining the calculated values of some formulas in this algorithm, the minimum learning rate is set to 0.001 without any other changes.

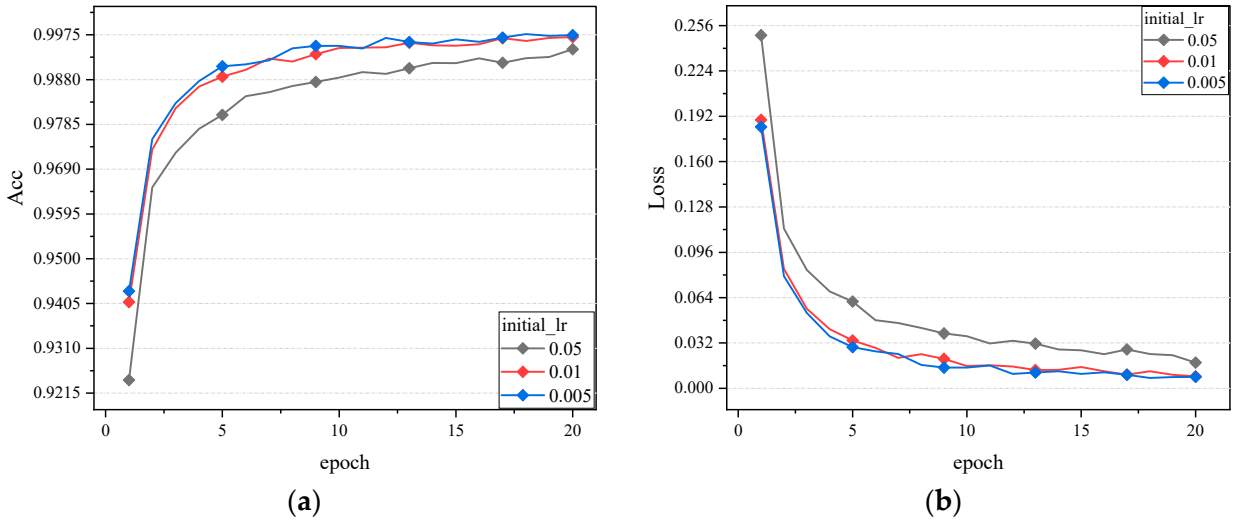


Figure 1. Performance comparison of the MNIST dataset for different learning rate variation intervals: (**a**) comparison of accuracy; (**b**) comparison of loss values.

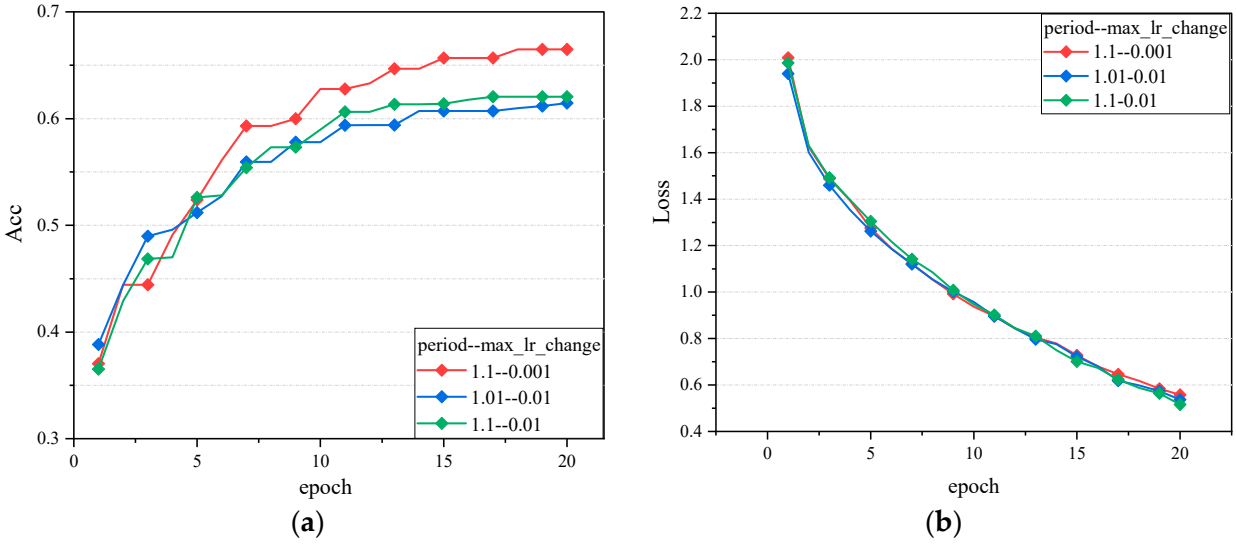
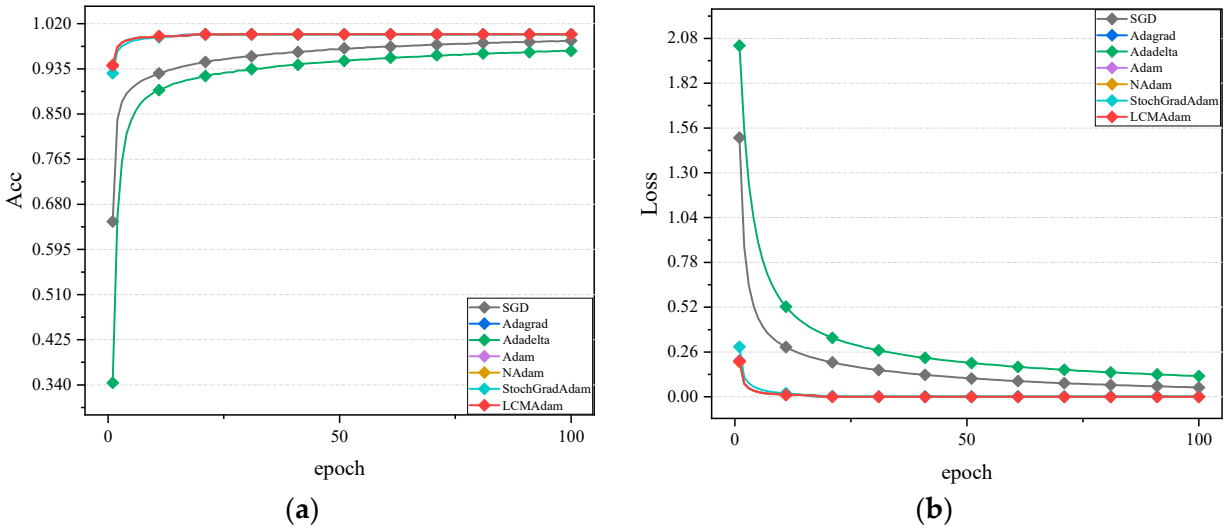


Figure 2. Comparison of the performance of the CIFAR10 dataset for different learning rates in terms of maximum change and cycle length: (**a**) comparison of accuracy; (**b**) comparison of loss values.

The variation in the maximum variation in the learning rate max_*lr_change* and the cycle length *period* in Figure 2 also affects the performance of the LCMAdam algorithm. The experimental results indicate that the LCMAdam algorithm performs best when the learning rate maximum variation parameter and the cycle length parameter are set to 1.1 and 0.001, respectively. However, changing the maximum amount of change in the learning rate parameter to 0.01 results in a good performance in the early stages of training, but a decrease in performance in the later stages. When adjusting these two parameters simultaneously, the performance is still not as good as when they are set to 1.1 and 0.001. This phenomenon may be due to improper parameter settings resulting in a large variation in *lr* in the final calculation.

For training, the hyperparameters are set to the set of data on which the LCMAdam algorithm performs best on the three datasets. Figure 3 shows the performance comparison of various algorithms for the MNIST dataset.



(**a**) (**b**)

Figure 3. Performance comparison of various algorithms on the MNIST dataset: (**a**) comparison of accuracy; (**b**) comparison of loss values.

The results in Figure 4 show that, on the CIFAR10 dataset, although the accuracy of the LCMAdam algorithm is slightly lower than that of the Adam and NAdam algorithms at the beginning of the training period, the accuracy of the LCMAdam algorithm surpasses that of the remaining six algorithms as the training progresses, and the algorithm improves its accuracy by 6.71% compared to the Adam algorithm. The accuracy of the LCMAdam algorithm is improved by 5.35% and 6.14% compared to the improved Adam algorithm-based NAdam algorithm and StochGradAdam algorithm, respectively. This advantage can be attributed to the curvature-controlled gradient strategy employed by the LCMAdam optimizer. During the training process, the LCMAdam algorithm ensures the smoothness of the gradient update by limiting the curvature of the gradient. By adjusting the change in the size of the curvature, the LCMAdam algorithm can effectively avoid drastic fluctuations in the gradient, thus improving the stability of model training.

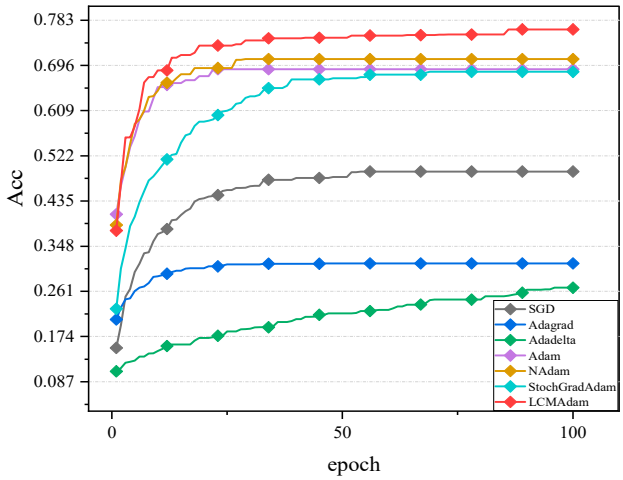


Figure 4. Accuracy of different algorithms for the CIFAR10 dataset.

Figure 5 shows that the LCMAdam algorithm has a slightly lower loss value than the other algorithms at the beginning of training. As the number of training rounds increases, the Adam algorithm is lower than the algorithm by 0.06 in comparison.

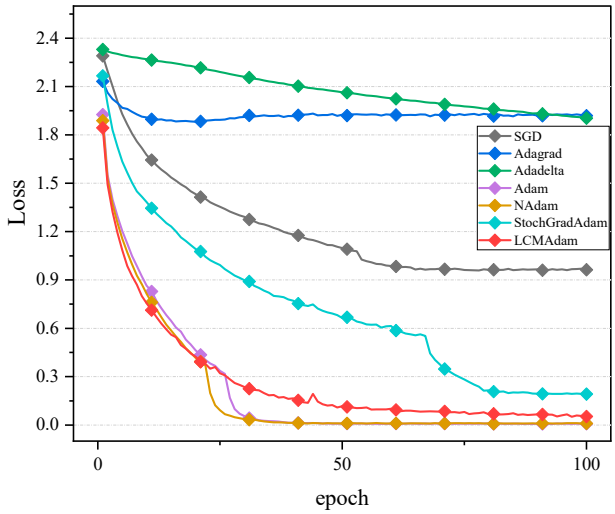


Figure 5. Loss values of different algorithms for the CIFAR10 dataset.

The accuracy of several different algorithms was compared on the Stomach dataset. According to the results in Figure 6, the accuracy of the Adadelta and SGD algorithms is significantly lower than the other five algorithms as the number of training rounds increases. The LCMAdam algorithm improves by 2.6%, 2%, and 6.6% compared to the Adam algorithm as well as the NAdam algorithm and StochGradAdam algorithm, which are improved based on the Adam algorithm. This advantage can be attributed to the linear interpolation control change learning rate strategy, which constantly changes the value of the learning rate during training and utilizes linear interpolation to approximate the learning rate for adjustment. By adjusting the learning rate in a timely and precise manner, the LCMAdam algorithm can converge to the optimal solution more efficiently and be more responsive to changes and challenges during training, which in turn improves performance and stability.

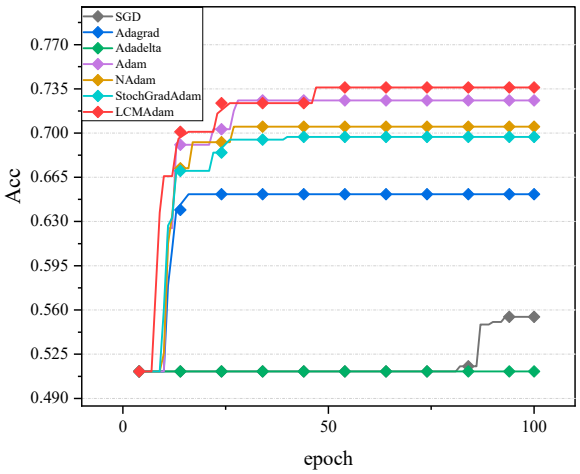


Figure 6. Accuracy of different algorithms on the Stomach dataset.

The loss values for the test set of several different algorithms were compared for the Stomach dataset. The results in Figure 7 show that, as the number of training rounds increases, the Adadelta algorithm has significantly higher loss values than the other optimization algorithms. The LCMAdam algorithm outperforms the other six algorithms

with lower loss values. Specifically, when compared to the Adam algorithm, as well as the NAdam algorithm and the StochGradAdam algorithm, which are based on the improvement of the Adam algorithm; the LCMAdam algorithm reduces the loss values by 0.223, 0.157, and 0.2707.

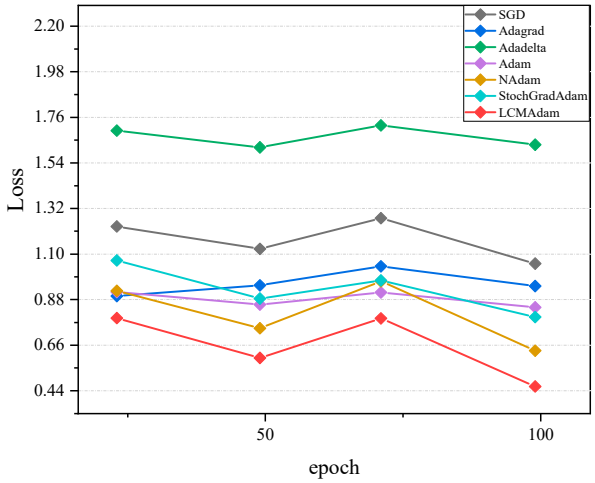


Figure 7. Loss values for different algorithms for the Stomach dataset.

Based on Figure 8, on the CIFAR10 dataset, the LCMAdam algorithm has slightly lower GPU power consumption compared to the StochGradAdam algorithm, but still outperforms the Adam algorithm and other compared algorithms. This is due to the LCMAdam algorithm's dynamic learning rate adjustment strategy, which results in a slightly longer training time compared to some other algorithms. The LCMAdam algorithm's dynamic learning rate adjustment strategy offers more precise guidance for model optimization, thereby enhancing the model's convergence speed and performance to some extent.

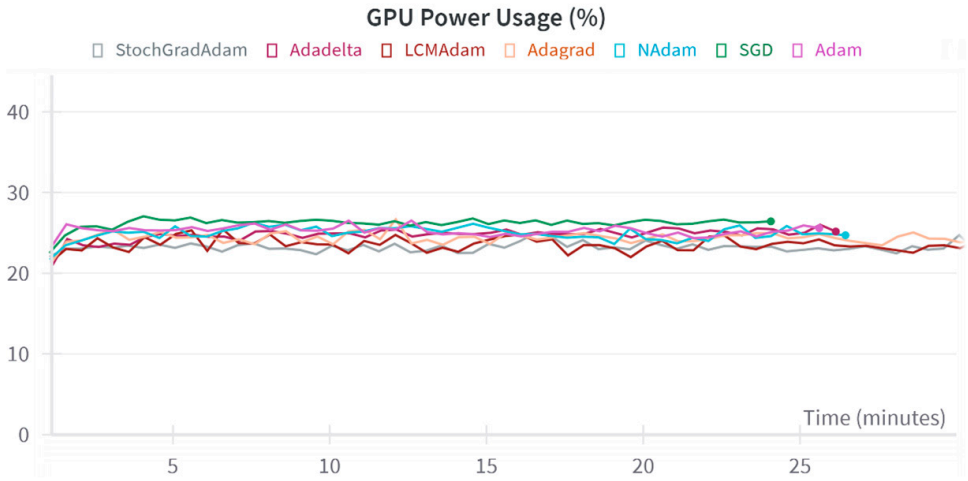


Figure 8. GPU power consumption of different algorithms for the CIFAR10 dataset.

Based on the observations in Figure 9, it can be seen that the GPU power consumption of the seven algorithms is basically equal when facing a more difficult dataset to recognize, such as Stomach, but there is a slight difference in their processing time. The LCMAdam algorithm shows a faster processing speed when dealing with this type of more difficult dataset, an advantage that may bring significant benefits in real-world scenarios.

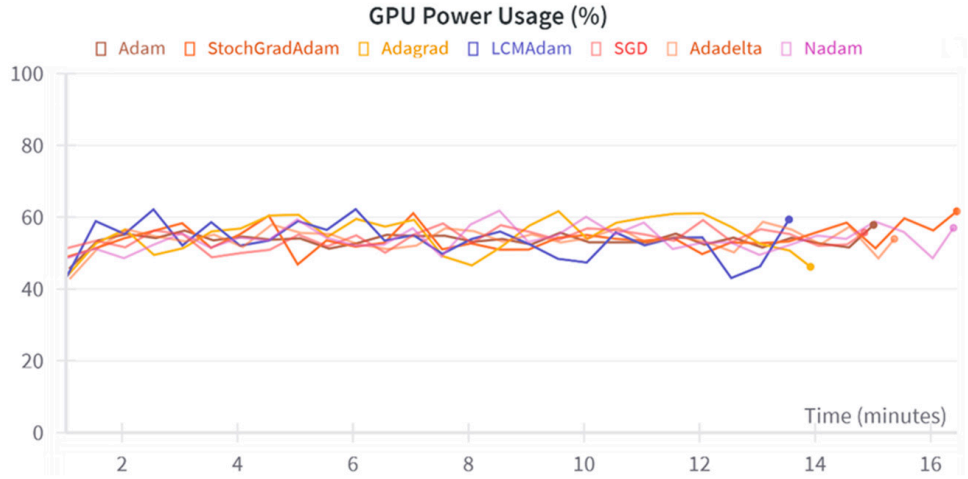


Figure 9. GPU power consumption of different algorithms for the Stomach dataset.

The LCMAdam algorithm demonstrates superior performance not only in processing simple grayscale images, but also in processing color image data. After 100 rounds of training, the LCMAdam algorithm processes images with slightly higher accuracy than the other algorithms; especially compared to the standard Adam algorithm, the LCMAdam algorithm achieves significant optimization. The benefits of the LCMAdam algorithm are emphasized in improving image recognition accuracy and reducing oscillation ability. The experimental results confirm the superiority of the LCMAdam algorithm, particularly its robust performance. The algorithm presents high accuracy under different datasets and achieves low loss values during training. This shows that the LCMAdam algorithm has unique advantages in dealing with complex data structures.

4. Conclusions

In this study, a new optimization algorithm, LCMAdam, is proposed to further improve the problems and deficiencies of the Adam optimization algorithm. Comprehensive experimental validation confirms that the LCMAdam algorithm outperforms traditional optimization methods, such as Adam, on several standard datasets. The LCMAdam algorithm demonstrates significant improvements in stability and convergence speed for complex problems. The LCMAdam algorithm offers a more flexible and efficient training approach by combining curvature-controlled gradient and linear interpolation to adapt the learning rate adjustment. The LCMAdam algorithm has made significant progress in improving the performance and stability of neural network training, but there are still some limitations. One such limitation is the computational efficiency challenge that may be faced when dealing with extremely large-scale datasets. In addition, the performance of the LCMAdam algorithm may be affected by the characteristics of the dataset, especially in sparse data or highly unbalanced classification tasks, where additional tuning may be required to achieve optimal performance. The results indicate that the LCMAdam algorithm enhances the model's performance on standard test sets and real-world applications, including medical image processing. This provides a promising approach for improving performance and generalization of deep learning models.

Author Contributions: Conceptualization, methodology, and writing—original draft preparation, W.Z.; software, project administration, and resources, Y.S.; data curation, J.C.; writing—review and editing, supervision, and formal analysis, H.S.; funding acquisition, Q.Z. and L.X. and L.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 location of the Python code used in this paper is https://github.com/zhou0618/LCMAdam (accessed on 6 April 2024); CIFAR10 dataset: https://www.kaggle.com/datasets/gazu468/cifar10-classification-image (accessed on 6 April 2024); and Stomach datasets: https://doi.org/10.1038/s41597-020-00622-y (accessed on 6 April 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The list of abbreviations and symbols is shown below.

LCMAdam	Linear Curvature Momentum Adam
GR	correction factor
initial_lr	maximum learning rate
min_lr	minimum learning rate
max_lr_change	maximum variation in the learning rate
period	cycle length

References

1. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**. [CrossRef]
2. Chakrabarti, K.; Chopra, N. A Control Theoretic Framework for Adaptive Gradient Optimizers in Machine Learning. *arXiv* **2023**. [CrossRef]
3. Ding, K.; Xiao, N.; Toh, K.-C. Adam-family Methods with Decoupled Weight Decay in Deep Learning. *arXiv* **2023**. [CrossRef]
4. Liu, M.; Zhang, W.; Orabona, F.; Yang, T. Adam^{+}: A Stochastic Method with Adaptive Variance Reduction. *arXiv* **2020**. [CrossRef]
5. Asadi, K.; Fakoor, R.; Sabach, S. Resetting the Optimizer in Deep RL: An Empirical Study. *arXiv* **2023**. [CrossRef]
6. Chen, A.C.H. Exploring the Optimized Value of Each Hyperparameter in Various Gradient Descent Algorithms. *arXiv* **2022**. [CrossRef]
7. Yuan, W.; Gao, K.-X. EAdam Optimizer: How ε Impact Adam. *arXiv* **2020**. [CrossRef]
8. Xia, L.; Massei, S. AdamL: A fast adaptive gradient method incorporating loss function. *arXiv* **2023**. [CrossRef]
9. Tian, R.; Parikh, A.P. Amos: An Adam-style Optimizer with Adaptive Weight Decay towards Model-Oriented Scale. *arXiv* **2022**. [CrossRef]
10. 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. *arXiv* **2021**. [CrossRef]
11. Dozat, T. Incorporating Nesterov Momentum into Adam. 2016. Available online: https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ (accessed on 6 April 2024).
12. 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. MBE* **2024**, *21*, 1270–1285. [CrossRef]
13. Jiang, Y.; Liu, J.; Xu, D.; Mandic, D.P. UAdam: Unified Adam-Type Algorithmic Framework for Non-Convex Stochastic Optimization. *arXiv* **2023**. [CrossRef]
14. Pan, Y.; Li, Y. Toward Understanding Why Adam Converges Faster Than SGD for Transformers. *arXiv* **2023**. [CrossRef]
15. 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]
16. 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]
17. Liu, H.; Tian, X. An Adaptive Gradient Method with Energy and Momentum. *Ann. Appl. Math.* **2022**, *38*, 183–222. [CrossRef]
18. Hotegni, S.S.; Berkemeier, M.; Peitz, S. Multi-Objective Optimization for Sparse Deep Multi-Task Learning. *arXiv* **2024**. [CrossRef]
19. Yun, J. StochGradAdam: Accelerating Neural Networks Training with Stochastic Gradient Sampling. *arXiv* **2024**. [CrossRef]
20. Fakhouri, H.N.; Alawadi, S.; Awaysheh, F.M.; Alkhabbas, F.; Zraqou, J. A cognitive deep learning approach for medical image processing. *Sci. Rep.* **2024**, *14*, 4539. [CrossRef]

21. Liu, Z.; Zhang, Z.; Lei, Z.; Omura, M.; Wang, R.-L.; Gao, S. Dendritic Deep Learning for Medical Segmentation. *IEEECAA J. Autom. Sin.* **2024**, *11*, 803–805. [CrossRef]
22. Liu, C.; Fan, F.; Schwarz, A.; Maier, A. AnatoMix: Anatomy-aware Data Augmentation for Multi-organ Segmentation. *arXiv* **2024**. [CrossRef]