algorithms

MDPI

Article

An Improved Adam's Algorithm for Stomach Image Classification

Haijing Sun [1], Hao Yu [2], Yichuan Shao [1,*], Jiantao Wang [2], Lei Xing [3], Le Zhang [1] and Qian Zhao [4]

[1] School of Intelligent Science and Engineering, Shenyang University, Shenyang 110044, China; sunhaijing@syu.edu.cn (H.S.); snowise@syu.edu.cn (L.Z.)
[2] School of Information Engineering, Shenyang University, Shenyang 110044, China; yuhao9733@outlook.com (H.Y.); wangjt17888@outlook.com (J.W.)
[3] School of Chemistry and Chemical Engineering, University of Surrey, Guildford GU2 7XH, UK; l.xing@surrey.ac.uk
[4] School of Science, Shenyang University of Technology, Shenyang 110044, China
* Correspondence: shaoyichuan@syu.edu.cn

Abstract: Current stomach disease detection and diagnosis is challenged by data complexity and high dimensionality and requires effective deep learning algorithms to improve diagnostic accuracy. To address this challenge, in this paper, an improved strategy based on the Adam algorithm is proposed, which aims to alleviate the influence of local optimal solutions, overfitting, and slow convergence rates by controlling the restart strategy and the gradient norm joint clipping technique. This improved algorithm is abbreviated as the CG-Adam algorithm. The control restart strategy performs a restart operation by periodically checking the number of steps and once the number of steps reaches a preset restart period. After the restart is completed, the algorithm will restart the optimization process. It helps the algorithm avoid falling into the local optimum and maintain convergence stability. Meanwhile, gradient norm joint clipping combines both gradient clipping and norm clipping techniques, which can avoid gradient explosion and gradient vanishing problems and help accelerate the convergence of the optimization process by restricting the gradient and norm to a suitable range. In order to verify the effectiveness of the CG-Adam algorithm, experimental validation is carried out on the MNIST, CIFAR10, and Stomach datasets and compared with the Adam algorithm as well as the current popular optimization algorithms. The experimental results demonstrate that the improved algorithm proposed in this paper achieves an accuracy of 98.59%, 70.7%, and 73.2% on the MNIST, CIFAR10, and Stomach datasets, respectively, surpassing the Adam algorithm. The experimental results not only prove the significant effect of the CG-Adam algorithm in accelerating the model convergence and improving generalization performance but also demonstrate its wide potential and practical application value in the field of medical image recognition.

Keywords: deep learning; Adam algorithm; control restart strategy; gradient norm joint clipping; CG-Adam algorithm

Citation: Sun, H.; Yu, H.; Shao, Y.; Wang, J.; Xing, L.; Zhang, L.; Zhao, Q. An Improved Adam's Algorithm for Stomach Image Classification. *Algorithms* **2024**, *17*, 272. https://doi.org/10.3390/a17070272

Academic Editor: Patrizia Beraldi

Received: 20 May 2024
Revised: 17 June 2024
Accepted: 19 June 2024
Published: 21 June 2024

1. Introduction

With the wide application of deep learning in various fields, the optimization algorithm of the model is crucial for the improvement of the training effect. Among many optimization algorithms, the Adam algorithm has received widespread attention for its excellent convergence speed and stable performance. However, with the increase in model complexity and training data size, the Adam algorithm also faces some challenges, such as gradient vanishing, gradient explosion, and overfitting. To solve these problems, researchers have proposed various optimization strategies. For example, Yun [1] proposed a new method to accelerate neural network training through a stochastic gradient sampling technique in his research, and Xia and Massei [2] explored a fast adaptive gradient method, which integrates the loss function and aims to improve the performance of the

algorithm. Meanwhile, Tang [3] corrected the DP bias in the Adam algorithm, emphasizing the importance of considering privacy protection in optimization algorithms. In addition, Kleinsorge et al. [4] proposed the ELRA optimization method, and Hong and Lin [5] investigated the problem of the high-probability convergence of the Adam algorithm under an unbounded gradient, all of which provided new perspectives and solutions for the development of the optimization algorithm. Zhuang [6] explored adaptive strategies in non-convex optimization problems, focusing on how the learning rate can be automatically adjusted by an adaptive algorithm to achieve a near-optimal convergence rate in the absence of prior knowledge. Zhang et al. [7] proposed a new gradient descent method, the asymmetric momentum method, which improves the performance on non-sparse gradient datasets by improving the momentum allocation during gradient descent. Son and Yang [8] introduced a new learning rate scheduling algorithm designed to speed up the convergence of optimization algorithms by automatically adjusting the learning rate, especially when dealing with noisy gradients in small batch optimization scenarios. Bao Zhang [9] proposed the SADAM algorithm, which employs a stochastic strategy to escape stagnation points and saddle points with the aim of improving accuracy without the need for batch processing. Wang and Klabjan [10] proposed a variance reduction version based on Adam that focuses on convergence improvement in specific scenarios. Li et al. [11] provided a proof of the strict convergence of the Adam algorithm for a wide range of optimization objectives. He et al. [12] explored the convergence of the Adam algorithm under non-convex objectives and provided theoretical guarantees of the effectiveness of the Adam algorithm in a wider range of application scenarios by relaxing the hyper-parameter conditions, thus broadening the scope of its applicability to complex optimization problems. Bu et al. [13] proposed an automatic gradient trimming technique to improve the ability of the Adam algorithm to deal with the gradient explosion problem, which aims to reduce the need for manual parameter tuning and make the optimization process more efficient and stable by automatically adjusting the trimming threshold. Notsawo [14] investigated the possibility of improving the efficiency of the Adam algorithm on large-scale datasets by introducing a stochastic averaging gradient method to reduce the number of gradients that need to be computed in each iteration and speed up the optimization process. Chen et al. [15] proposed a new adaptive learning rate method that can automatically adjust the learning rate during the training process by making the learning rate differentiable, aiming to improve the efficiency of the training process and the final performance of the model. Chen [16] explored the optimal value of each hyper-parameter for various gradient descent methods and provided a method to optimize the selection of hyper-parameters by analyzing the effect of hyper-parameters on the training process in detail. Bieringer et al. [17] proposed a new algorithm, the AdamMCMC algorithm, which provides a new quantitative strategy for uncertainty estimation in deep neural network methods by combining the Metropolis-adjusted Langevin algorithm and momentum-based optimization. Despite the progress made in optimization algorithms in these studies, there are still limitations in dealing with non-convex optimization problems, large-scale datasets, and gradient explosion and overfitting problems. Existing stochastic gradient sampling techniques perform well in accelerating neural network training, but when faced with non-convex optimization problems, they tend to fall into local optimal solutions and are difficult to find the global optimal solution. Although the adaptive gradient method improves the performance of the algorithm by integrating the loss function, it underperforms on large-scale datasets, and the computational overhead and convergence speed during the training process still need to be optimized. New optimization methods and high-probability convergence studies provide new theoretical perspectives and solutions, but their effectiveness in practical applications has not yet been fully verified, especially in real-world complex datasets and tasks. Overall, existing studies still have significant shortcomings in enhancing the stability of Adam's algorithm and adapting it to complex optimization problems, especially when dealing with nonconvex optimization, large-scale datasets, gradient explosion, and overfitting

problems; the stability and convergence speed of the existing methods need to be improved urgently [18,19].

In order to compensate for the limitations of the existing Adam optimizer, an improved strategy is proposed in this study that combines the control restart strategy with the Gradient norm joint clipping technique, aiming to improve the stability and convergence speed of the algorithm. The control restart strategy helps avoid local optimal solutions and maintain the stability of the algorithm, while the Gradient norm joint clipping technique avoids the gradient explosion and vanishing problems by restricting the size of the gradient and the number of paradigms, while at the same time accelerating the convergence process. Combining the advantages of both of the above, the CG-Adam algorithm not only performs excellently in accelerating the convergence and improving the stability but also helps to improve the generalization ability of the model and reduce overfitting. In this paper, the principle, implementation method, and effect of the CG-Adam algorithm in practical applications will be introduced in detail. Through theoretical analysis and experimental validation, the advantages of this joint optimization strategy in improving model training effects, accelerating convergence speed, and preventing overfitting will be demonstrated. Meanwhile, the adaptability of this strategy in different models and scenarios is explored, providing new optimization ideas and methods for researchers and developers in the field of deep learning [20].

2. Design of the CG-Adam Algorithm

2.1. The Adam Optimization Algorithm

The Adam algorithm is an adaptive learning rate optimizer that combines the advantages of the gradient descent and momentum strategies. Its core mechanism is based on the computation of first-order and second-order moment estimates of the gradient, which are used to dynamically adjust the learning rate. The first-order moments represent the mean of the gradient, and the second-order moments represent the mean of the squared gradient. As a frequently used optimization tool in neural network training, the Adam algorithm is notable for its ability to adaptively adjust the learning rate, allowing for personalized learning rate adjustments for each parameter to better fit the unique needs of each parameter. This approach avoids the complexity of manually setting the learning rate and shows better adaptability for handling sparse gradients, real-time data streams, and non-static datasets. Although Adam performs well in most cases, it may fall into local optimal solutions on some complex lossy surfaces, thus limiting the effectiveness of the optimization. In the implementation of the Adam algorithm, the parameters of the first-order moments and second-order moments, which are responsible for balancing the contribution of first-order and second-order moments, respectively, further optimize the learning process. In the Adam algorithm, the parameters used for both are defined as first-order moments and second-order moments, respectively. The moment estimation vectors for the gradient are shown in Equations (1) and (2):

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

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

In Equations (1) and (2), β_1 and β_2 represent the exponential decay rates for the first-order moment and second-order moment, respectively; g_t is the gradient at step t and is calculated as shown in Equation (3):

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

In Equation (3), θ_t is the parameter vector updated at step t; $f_t(\theta_t)$ is the loss function of the t-th iteration in the neural network; and g_t is the gradient of the loss function of the t-th repetition of the neural network with respect to the parameters. When the decay rate is very small in the initial stage, the first-order moment and second-order moment estimates

may deviate from zero, leading to some bias. In order to eliminate this bias, Adam applies a bias correction to the first-order moments and second-order moments during the decay process, as shown in Equations (4) and (5):

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

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

After each iteration, the parameter θ will be updated, as shown in Equation (6):

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

In Equation (6), θ represents the learning rate; ε is a small value greater than 0, primarily used to prevent division by zero in the denominator. Despite the good performance of the Adam optimization algorithm, it is sensitive to the choice of learning rate and requires a careful adjustment of hyperparameters. In some non-convex optimization problems, the hyperparameters may not converge to the global optimum.

2.2. Gradient Norm Joint Clipping

Gradient norm joint clipping trims both the gradient and the norm to ensure the stability and reasonableness of the gradient during the optimization process. By limiting the gradient to a suitable range, the gradient explosion and gradient vanishing problems can be avoided, and it helps to accelerate the convergence of the optimization process. Therefore, the use of the gradient norm joint clipping technique can effectively improve the training effect of the optimization algorithm and increase the performance and reliability of the algorithm. In addition, this technique enhances the stability of training by maintaining the proper scale of the gradient during optimization and reducing unwanted fluctuations. It also allows for a more flexible setting of gradient thresholds and fine control of the optimizer's behavior, enabling the optimization algorithm to adaptively adjust at different training stages, improving overall training efficiency and model quality. With these improvements, gradient norm joint clipping provides a stronger foundation for optimization algorithms to effectively address complex training challenges and improve the performance of the final model. The mathematical expression for gradient norm joint clipping is shown in Equation (7):

$$g_t' = g_t/\max(1, \|g_t\|_2/clip\_value) \tag{7}$$

In Equation (7), g_t is the original gradient of the model parameters at step t of the iteration. g_t' is the clipped gradient. $\|g_t\|_2$ is the L2 norm of the original gradient, which is the length of the gradient vector in Euclidean space. $clip\_value$ is the default clipping threshold that specifies the maximum allowable length of the gradient vector. $\max(1, \|g_t\|_2/clip\_value)$ calculates a clipping factor, which is the greater between 1 and the maximum value of the ratio of the gradient L2 norm to the $clip\_value$ threshold. If $\|g_t\|_2$ is less than or equal to the trimming threshold, the clipping factor is 1; otherwise, it is $\|g_t\|_2/clip\_value$. g_t' stands for cropping the gradient t by dividing g_t by the cropping factor. If the L2 norm of the gradient is less than or equal to the cropping threshold, the cropping factor is 1, and the gradient is unchanged; otherwise, the gradient is scaled.

Gradient norm joint clipping can simultaneously limit the size of the gradient and the number of gradient paradigms, effectively controlling the magnitude of the gradient change, thus improving the stability of the optimization algorithm. By clipping the gradient and the number of gradient paradigms, the effects of gradient explosion, gradient vanishing, and outliers on the optimization process can be prevented, which helps the model converge faster and improves the model's generalization ability.

2.3. Control Restart Strategy

The core idea of the control restart strategy is to periodically check the number of steps during the optimization process and perform a restart operation once the number of steps reaches a predefined restart period. The restart operation consists of resetting the optimizer state and the step counter to their initial values so that the optimization process can restart in the next cycle. The introduction of this restart strategy effectively improves the robustness and efficiency of the optimization algorithm. By dynamically adjusting the restart cycle, the algorithm is able to adaptively optimize according to the characteristics of the problem, further improving its performance. This flexible restart strategy provides an effective solution for the application of optimization algorithms to cope with optimization challenges in different scenarios. In addition to this, the control restart strategy has some additional advantages. It prevents the optimization process from falling into local optimal solutions and thus explores better global solutions. In addition, the strategy is able to better handle non-convex optimization problems by restarting the optimization process periodically, which makes the optimizer perform better in the face of complex and multi-peaked loss functions. Taken together, the control restart strategy not only improves the robustness and efficiency of the algorithm but also enhances the adaptability and stability of the algorithm in dealing with complex optimization problems, making it an indispensable and important technique in solving practical optimization tasks. The control restart strategy is shown in Equation (8):

$$t' = t \bmod T \tag{8}$$

In Equation (8), t is the current number of iterations; mod denotes the modulo operation; and t' denotes the control of the step count within a cycle T when executing a restart cycle. The step counter t' indicates the number of steps in the current cycle. At the beginning, t' is initialized to 0, and the value of t' is increased by 1 for each step performed as the optimization process proceeds. The value of t' is reset to 0 when t' reaches the cycle length T in order to restart the counting, thus completing one cycle. The introduction of this restart cycle effectively controls the step counting of the optimization algorithm so that the number of steps in each cycle can be well controlled and managed, thus improving the stability and efficiency of the optimization algorithm. The deviation correction factor takes into account the restart logic, and the deviation correction formula can be redefined, as shown in Equations (9)–(12):

$$m_{t'} = \beta_1 m_{t'-1} + (1-\beta_1) g_t \tag{9}$$

$$v_{t'} = \beta_2 v_{t'-1} + (1-\beta_2) g_t^2 \tag{10}$$

$$\hat{m}_{t'} = m_t / \left(1-\beta_1^{t'}\right) \tag{11}$$

$$\hat{v}_{t'} = v_t / \left(1-\beta_2^{t'}\right) \tag{12}$$

In Equations (9)–(12), the restart strategy converts the timestamp t into t', which gives the parameter update process the property of periodic adjustment through t' and is able to better adapt to the periodically changing data distribution and model parameters. These adjustments not only take into account the bias correction and dynamic adjustment of the gradient estimation in each cycle but also ensure the periodic impact of the bias correction, thus improving the stability and generalization performance of the algorithm.

The formula for *denom* is shown in Equation (13):

$$denom = \sqrt{\hat{v}_t + \varepsilon} \tag{13}$$

In Equation (13), *denom* denotes the denominator term in the step adjustment; $\hat{v}_t$ denotes the bias-corrected second-order moment estimate, which represents the moving

average of the squared gradient; and ε denotes a small constant that is usually used to prevent the denominator from being zero. By adding a small constant ε to the bias-corrected second-order moment estimate and then taking its square root, we ensure that the denominator term *denom* is not too small. This helps to avoid errors in the division operation during parameter updating due to the second-order moment estimates being too small and also ensures the stability of the gradient update.

The formula for *step_size* is shown in Equation (14):

$$step\_size = lr / \left(1 - \beta_1^{t'}\right) \tag{14}$$

In Equation (14), *step_size* can be adaptively adjusted according to the actual situation. This adaptivity can make the optimization algorithm better adapted to different data distributions, thus improving the robustness and generalization performance of the algorithm.

The formula for θ_{new} is shown in Equation (15):

$$\theta_{new} = \theta - \frac{\hat{m}_{t'}}{denom} step\_size \tag{15}$$

In Equation (14), by comprehensively considering the gradient normalization and step size control, the parameter update rule can effectively optimize the parameters and improve the performance and convergence speed of the optimization algorithm. At the same time, the algorithm's adaptability to different data is also enhanced by adaptively adjusting the step size and normalized gradient.

By introducing a control restart strategy, the CG-Adam optimizer adds significant flexibility and efficiency to the Adam algorithm. This design not only helps avoid local optimum traps during training and accelerates model convergence but also further enhances the generalization ability of the model by facilitating a broader exploration of the parameter space.

2.4. The CG-Adam Algorithm

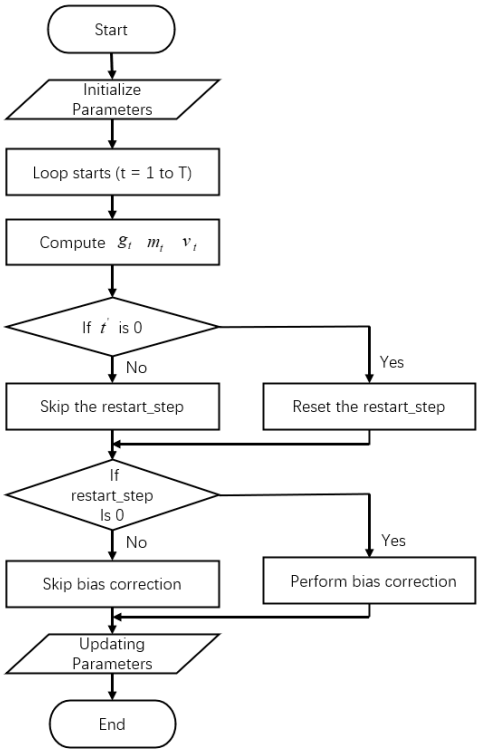
In the use of optimization algorithms, various parameters need to be considered, and the learning rate is crucial. The learning rate needs to be determined by a combination of factors and is usually set at different values during different training phases. The Adam algorithm employs an adaptive adjustment of the learning rate technique to ensure that the learning rate does not decay too quickly. In order to achieve this goal, Adam performs an exponentially weighted average of the gradient and its square, which is essentially an accumulation calculation of the historical gradient and can effectively attenuate the influence of the historical gradient on the current gradient. In the initial stage of iteration, the Adam algorithm is prone to rapid convergence due to the rapid accumulation of the first-order momentum of the gradient, resulting in significant oscillations around the optimum and exhibiting instability. When the algorithm falls into a local optimum, it is also difficult to jump out of the local optimum.

In this paper, a new adaptive learning rate algorithm is proposed for the above problem; CG-Adam is an improved version of the Adam optimization algorithm that combines the control restart strategy and Gradient norm joint clipping. The specific steps of this algorithm are shown in Algorithm 1.

For a clearer description of Algorithm 1, the algorithm flowchart (Scheme 1) is shown below.

Algorithm 1 CG-Adam

1: Input: initial point θ_t, first moment decay β_1, second moment decay β_2, Gradient clipping threshold *clip_value*, Restart period T.
2: Initialize: set the initial value of momentum m_0 and v_0 to 0. Initialize t' to 0 for cycle counting. Initialize restart_step to 0 to control the restart strategy. Set the initial step of the optimizer to 0. i.e., $m_0 = 0, v_0 = 0$, $t' = 0$, restart_step = 0, step = 0.
3: For t = 1 to T do
4: $g_t = \nabla_\theta f_t(\theta_t)$
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: if t' == 0 then
8: restart_step = 0
9: end for
10: if restart_step == 0 then
11: $\hat{m}_{t'} = m_t / \left(1 - \beta_1^{t'}\right)$
12: $\hat{v}_{t'} = v_t / \left(1 - \beta_2^{t'}\right)$
13: for each θ parameter in group[params] do
14: $m_{t'} = \beta_1 m_{t'-1} + (1-\beta_1) g_t$
15: $v_{t'} = \beta_2 v_{t'-1} + (1-\beta_2) g_t^2$
16: $g_t' = g_t / \max(1, \|g_t\|_2 / clip\_value)$
17: $denom = \sqrt{\hat{v}_t} + \varepsilon$
18: $step\_size = lr / \left(1 - \beta_1^{t'}\right)$
19: $\theta_{new} = \theta - \frac{\hat{m}_{t'}}{denow} step\_size$
20: end for
21: update t' to the value of the next cycle, $t' = t \bmod T$
22: restart_step = (restart_step + 1) mod T
end for
Return: Returns the final optimized parameter θ_t.



Scheme 1. Process generalization of the steps of Algorithm 1.

3. Experimental Design and Analysis of Results

3.1. Experimental Environment Configuration

This experiment is based on the deep learning framework of Pytorch. The CG-Adam algorithm is jointly improved by the control restart strategy and Gradient norm joint clipping. The specific versions of the software mainly used in the experiment are shown in Table 1.

Table 1. The specific versions of the software mainly used in the experiment.

Software	Version
Python	3.10
torch	2.0.1
torchvision	0.15.0
lightning	2.1.2
Wandb	0.16.0

In order to test the effectiveness of the CG-Adam algorithm, image classification experiments were conducted on three commonly used datasets: the MNIST dataset, the CIFAR10 dataset, and the Stomach dataset. MNIST is a binary image dataset containing handwritten digits. CIFAR-10 is a dataset containing color images for the image classification task. The images cover a wide range of object classes. The Stomach image dataset encompasses multiple categories. Characterizations of the datasets are shown in Table 2.

Table 2. Characterizations of the datasets.

Dataset	Number of Samples	Training Set	Test Set	Validation Set	Category
MNIST	70,000	55,000	10,000	5000	10
CIFAR10	60,000	45,000	10,000	5000	10
Stomach	1885	900	485	500	8

The Stomach dataset is a medical image dataset focused on gastrointestinal diseases. It contains over 110,000 high-quality images and a large amount of video data, all collected and annotated by specialized medical experts from the same Behram Hospital. This dataset is valuable for the automated detection and categorization of gastrointestinal diseases, as well as for medical education and research. Through the application of machine learning and data analytics, this gastrointestinal dataset helps facilitate the development of medical technology and improve the accuracy of clinical diagnosis. In this paper, experiments were conducted using 1885 labeled images from this dataset.

3.2. Experimental Results and Analysis

The improved algorithm proposed in this paper inherits the advantages of the adaptive learning rate of the Adam algorithm and enhances the robustness and efficiency of the whole optimization process. This improvement strategy makes the CG-Adam algorithm more suitable for a wide range of deep learning tasks, especially when encountering gradient instability and optimization challenges, and ensures more reliable performance. In order to thoroughly evaluate the performance benefits of the CG-Adam algorithm, this study selected the SGD algorithm, Adagrad algorithm, Adadelta algorithm, and Adam algorithm, and the improved algorithms, the Nadam algorithm and the StochGradAdam algorithm, and conducted a series of comparative experiments. Through extensive testing on multiple datasets and under different learning rate settings, the aim is to gain a deeper understanding of the performance of these algorithms. The comparison of the experimental results for different optimization algorithms is shown in Table 3. The reason for the bold numbers in the table is that CG-Adam shows the best results when compared to the other six algorithms.

Table 3. The comparison of the experimental results for different optimization algorithms.

Dataset	Optimization Algorithm	Accuracy	Loss
MNIST	SGD	97.36%	0.091
	Adagrad	97.84%	0.066
	Adadelta	96.42%	0.133
	Adam	98.52%	0.064
	Nadam	98.50%	0.062
	StochGradAdam	97.82%	0.078
	CG-Adam	**98.59%**	**0.059**
CIFAR10	SGD	49.09%	1.467
	Adagrad	33.54%	1.852
	Adadelta	25.58%	1.979
	Adam	69.55%	1.232
	Nadam	68.87%	1.638
	StochGradAdam	68.07%	1.04
	CG-Adam	**70.7%**	1.181
Stomach	SGD	57.6%	1.17
	Adagrad	68.40%	0.992
	Adadelta	55.80%	2.110
	Adam	69.80%	1.046
	Nadam	67.66%	2.10
	StochGradAdam	67.2%	1.166
	CG-Adam	**73.2%**	1.020

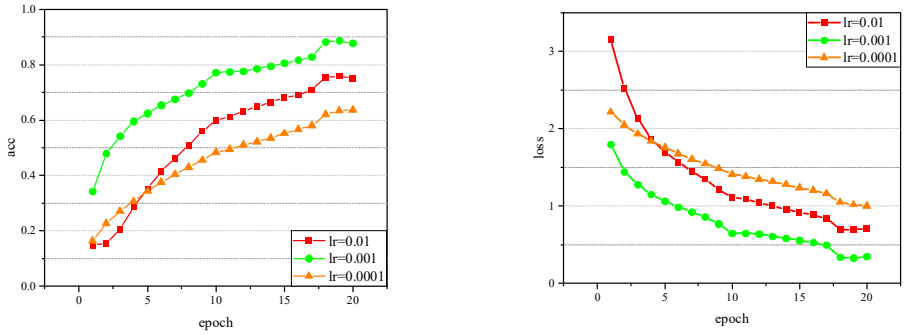
Based on the comparative experimental data in Table 3, it can be concluded that the performance of the improved CG-Adam algorithm is superior to the performance of the unimproved Adam algorithm. The accuracy of the CG-Adam algorithm on the CIFAR10 dataset reaches 70.7%, which is 1.15% higher than that of the unimproved Adam algorithm, and the loss value of the CG-Adam algorithm is 1.181, which is 0.051 lower than that of the unimproved Adam algorithm. The accuracy of the CG-Adam algorithm on the Stomach dataset reaches 73.2%, which is 3.4% higher than that of the unimproved Adam algorithm, and the loss value of the CG-Adam algorithm is 1.020, which is 0.026 lower than that of the unimproved Adam algorithm. The CG-Adam algorithm achieves better accuracy on all three datasets, which demonstrates that the proposed CG-Adam algorithm can be used in the processing of different types of datasets (both basic black-and-white images and more complex color RGB images) and verifies its outstanding generalization capability.

The experimental setup was as follows:

(1) The epoch in this experiment is set to 100 to ensure that the algorithms are compared with the same number of training rounds.
(2) The batch size in this experiment is set to 128 to maintain the consistency of the experiment.
(3) The initial learning rate of all algorithms in this experiment is set to 0.001.
(4) Due to the need to adjust the restart period and cropping value of the CG-Adam algorithm, multiple experiments are performed on each dataset, and the best experimental results are selected.
(5) Comparison experiments are conducted on the MNIST, CIFAR10, and Stomach datasets, respectively.

The CG-Adam algorithm aims to improve the generalization ability of the model and avoid falling into local optima. To test the performance of the CG-Adam algorithm on different neural networks, different types of neural networks were trained on different datasets. A simple fully connected neural network was used for training on the MNIST dataset, and a MobileNetV2 lightweight neural network was used for training on the CIFAR10 and Stomach datasets. In this experiment, in order to present the performance of the CG-Adam algorithm with different combinations of learning rates, three different

combinations were designed. The performance of the CG-Adam algorithm on the CIFAR10 dataset when combined with different learning rates is shown in Figure 1.



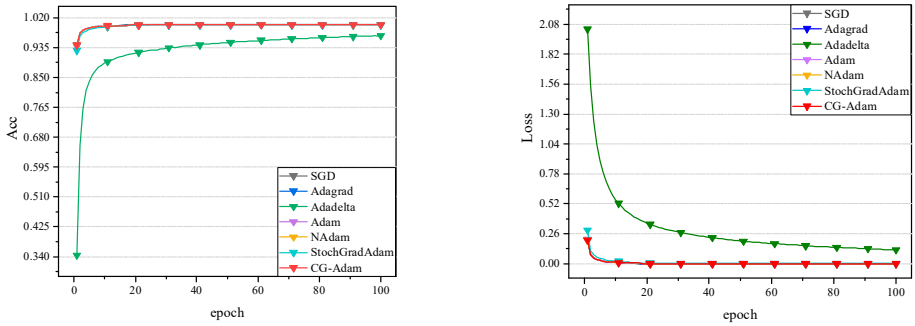
(**a**) (**b**)

Figure 1. The comparison of performance with different learning rates on the CIFAR10 dataset. (**a**) The comparison of accuracy. (**b**) The comparison of loss values.

In Figure 1, it is shown that in the three sets of experiments with different learning rates, the CG-Adam algorithm with a learning rate of 0.001 achieves the optimal values of acc and loss, so the learning rate of 0.001 is set as the initial value of the CG-Adam algorithm.

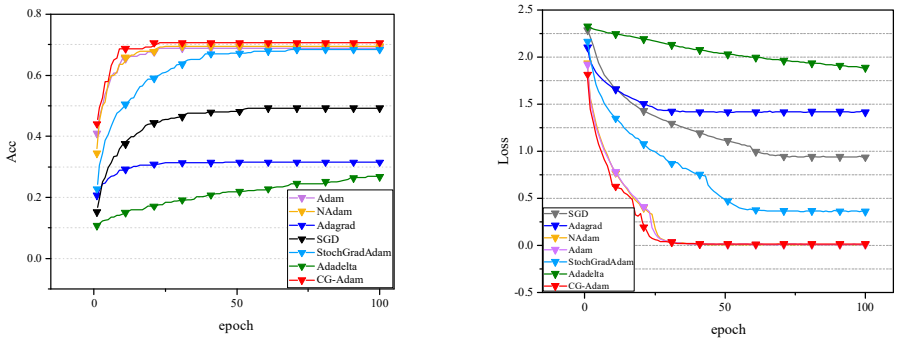
The training results of the seven algorithms on the MNIST dataset are shown in Figure 2.

The training results of the seven algorithms on the CIFAR10 dataset are shown in Figure 3.



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

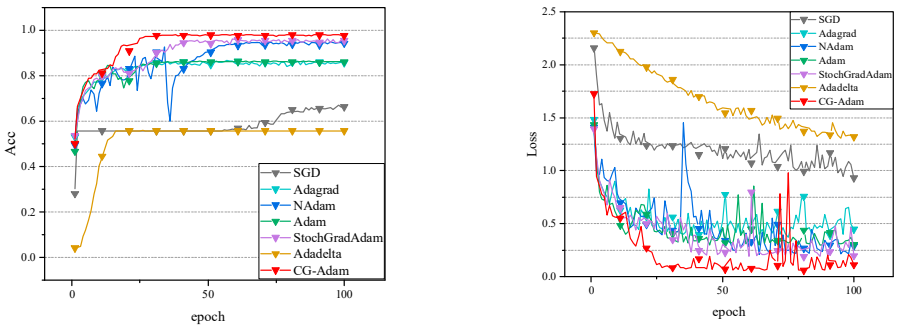
Figure 2. The training results of the seven algorithms on the MNIST dataset. (**a**) The comparison of accuracy. (**b**) The comparison of loss values.



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

Figure 3. The training results of the seven algorithms on the CIFAR10 dataset. (**a**) The comparison of accuracy. (**b**) The comparison of loss values.

The training results of the seven algorithms on the Stomach dataset are shown in Figure 4.



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

Figure 4. The training results of the seven algorithms on the Stomach dataset. (**a**) The comparison of accuracy. (**b**) The comparison of loss values.

As can be seen in Figure 4, the SGD algorithm and Adadelta algorithm significantly lag behind the other algorithms in terms of accuracy and loss values. The CG-Adam algorithm shows high accuracy from the beginning of the training phase, especially when compared to other algorithms based on the improvement of Adam (e.g., the Nadam algorithm and the StochGradAdam algorithm). This achievement is attributed to CG-Adam's controlled restart strategy, which resets the algorithm's state when a specific cycle is reached, thus indirectly enabling efficient tuning of the learning rate. This mechanism not only facilitates the rapid improvement of the model's performance at the early stage of training but also maintains the stability of the model throughout the training process, which ultimately leads to a steady upward trend in the accuracy curve of the CG-Adam algorithm. Throughout the training process, the CG-Adam algorithm outperforms Adam and the other five algorithms.

The GPU occupancy comparison of seven algorithms is shown in Figure 5.

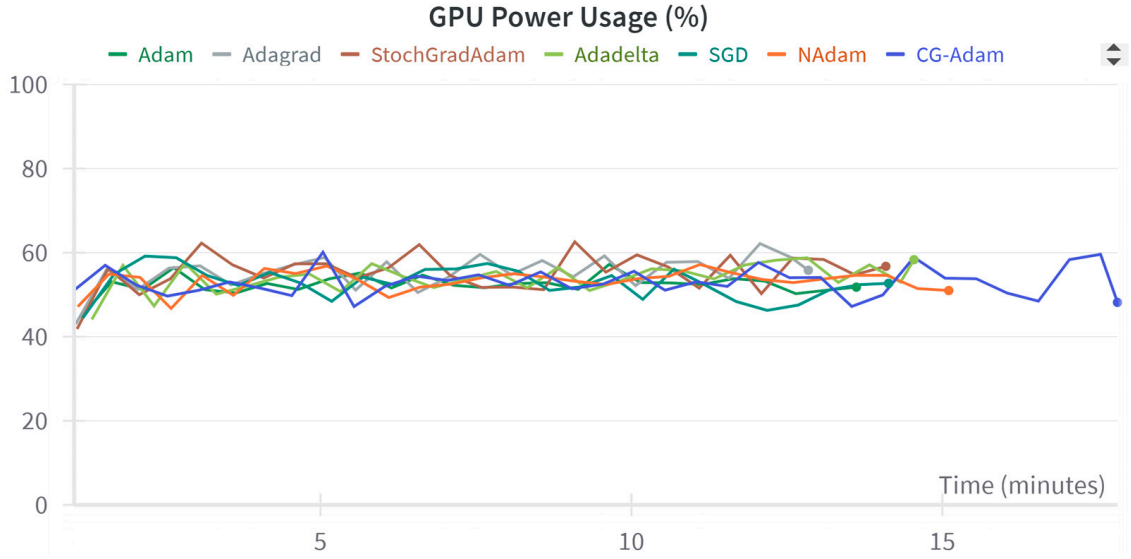


Figure 5. GPU occupancy comparison of seven algorithms.

The time and GPU utilization required to train each algorithm on the smoothed Stomach dataset can be seen in Figure 5. Throughout the training process, the CG-Adam algorithm has lower GPU utilization compared to the other six algorithms. Considering that the CG-Adam algorithm employs a controlled restart strategy to indirectly change the learning rate during training, its training time is slightly increased compared to the

other algorithms. This is due to the fact that the control restart strategy requires adjusting the learning rate in a specific cycle, which may lead to slower model convergence, thus lengthening the training time. For some application scenarios that require fast iterations, this increased training time may be detrimental. In addition, the low GPU occupancy of the CG-Adam algorithm may mean that GPU resources are underutilized at certain moments. This reduction in resource utilization may pose an efficiency problem in some HPC environments, especially if there is a need to maximize hardware resource utilization.

In the comparison study between the CG-Adam algorithm and the other six algorithms, the CG-Adam algorithm shows higher accuracy and more robust convergence on the Stomach dataset, which further proves that the CG-Adam algorithm has a significant advantage in improving the learning rate adjustment problem and model generalization ability. After 100 training iterations, CG-Adam not only converges faster than other algorithms but also improves the optimization efficiency and the ability to get rid of local optimums, especially compared with the Adam algorithm. This not only emphasizes CG-Adam's advantage in generalization performance but also highlights its wide application in different datasets and tasks.

The experimental data fully demonstrate the sophistication of the CG-Adam algorithm, especially in improving the generalization ability and accelerating the convergence speed. The fact that CG-Adam achieves the highest accuracy on different datasets and maintains the highest accuracy values during the training process further confirms the efficiency and wide applicability of CG-Adam.

4. Conclusions

In this study, the CG-Adam algorithm effectively avoids the shortcoming of the Adam algorithm that tends to fall into local optimums during the training process, which greatly improves the generalization of the model over multiple levels. Although the performance of the CG-Adam algorithm is significantly improved, its memory requirement is increased compared with the Adam algorithm. The experimental results on three datasets, MNIST, CIFAR10, and Stomach, show that the CG-Adam algorithm outperforms other algorithms in terms of accuracy and convergence speed, showing its obvious performance advantages. The algorithm proposed in this paper not only highlights the effectiveness of CG-Adam in overcoming the inherent defects of the Adam algorithm but also proves the key role of CG-Adam in improving the performance of multi-dataset processing.

Author Contributions: Conceptualization, methodology, and writing—original draft preparation, H.Y.; software, project administration, and resources, Y.S.; data curation, J.W.; writing—review and editing, supervision, and formal analysis, H.S.; funding acquisition, Q.Z., 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; and 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: CIFAR10: https://www.kaggle.com/datasets/gazu468/cifar10-classification-image (accessed on 1 May 2022); Stomach: https://doi.org/10.1038/s41597-020-00622-y (accessed on 28 August 2020); and MNIST: https://www.cvmart.net/dataSets/detail/236 (accessed on 24 December 2021).

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

Abbreviations

The list of abbreviations and symbols is shown below.

CG-Adam	Control restart strategy gradient norm joint clipping Adam
AdamMCMC	Combining Metropolis-adjusted Langevin with momentum-based optimization
SADAM	Stochastic Adam
lr	Learning rate
ELRA	Exponential learning rate adaption gradient descent

References

1. Yun, J. StochGradAdam: Accelerating Neural Networks Training with Stochastic Gradient Sampling. *arXiv* **2024**, arXiv:2310.17042. [CrossRef]
2. Xia, L.; Massei, S. AdamL: A fast adaptive gradient method incorporating loss function. *arXiv* **2023**, arXiv:2312.15295. [CrossRef]
3. Tang, Q.; Shpilevskiy, F.; Lécuyer, M. DP-AdamBC: Your DP-Adam Is Actually DP-SGD (Unless You Apply Bias Correction). *arXiv* **2023**, arXiv:2312.14334. [CrossRef]
4. Kleinsorge, A.; Kupper, S.; Fauck, A.; Rothe, F. ELRA: Exponential learning rate adaption gradient descent optimization method. *arXiv* **2023**, arXiv:2309.06274. [CrossRef]
5. Hong, Y.; Lin, J. High Probability Convergence of Adam Under Unbounded Gradients and Affine Variance Noise. *arXiv* **2023**, arXiv:2311.02000. [CrossRef]
6. 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]
7. Zhuang, Z. Adaptive Strategies in Non-convex Optimization. *arXiv* **2023**, arXiv:2306.10278. [CrossRef]
8. Zhang, G.; Zhang, D.; Zhao, S.; Liu, D.; Toptan, C.M.; Liu, H. Asymmetric Momentum: A Rethinking of Gradient Descent. *arXiv* **2023**, arXiv:2309.02130. [CrossRef]
9. Song, Z.; Yang, C. An Automatic Learning Rate Schedule Algorithm for Achieving Faster Convergence and Steeper Descent. *arXiv* **2023**, arXiv:2310.11291. [CrossRef]
10. Zhang, W.; Bao, Y. SADAM: Stochastic Adam, A Stochastic Operator for First-Order Gradient-based Optimizer. *arXiv* **2022**, arXiv:2205.10247. [CrossRef]
11. Wang, R.; Klabjan, D. Divergence Results and Convergence of a Variance Reduced Version of ADAM. *arXiv* **2022**, arXiv:2210.05607. [CrossRef]
12. Li, H.; Rakhlin, A.; Jadbabaie, A. Convergence of Adam Under Relaxed Assumptions. *arXiv* **2023**, arXiv:2304.13972. [CrossRef]
13. He, M.; Liang, Y.; Liu, J.; Xu, D. Convergence of Adam for Non-convex Objectives: Relaxed Hyperparameters and Non-ergodic Case. *arXiv* **2023**, arXiv:2307.11782. [CrossRef]
14. Bu, Z.; Wang, Y.-X.; Zha, S.; Karypis, G. Automatic Clipping: Differentially Private Deep Learning Made Easier and Stronger. *arXiv* **2023**, arXiv:2206.07136. [CrossRef]
15. 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]
16. Notsawo, P.J.T. Stochastic Average Gradient: A Simple Empirical Investigation. *arXiv* **2023**, arXiv:2310.12771. [CrossRef]
17. Chen, B.; Wang, H.; Ba, C. Differentiable Self-Adaptive Learning Rate. *arXiv* **2022**, arXiv:2210.10290. [CrossRef]
18. Chen, A.C.H. Exploring the Optimized Value of Each Hyperparameter in Various Gradient Descent Algorithms. *arXiv* **2022**, arXiv:2212.12279. [CrossRef]
19. Bieringer, S.; Kasieczka, G.; Steffen, M.F.; Trabs, M. AdamMCMC: Combining Metropolis Adjusted Langevin with Momentum-based Optimization. *arXiv* **2023**, arXiv:2312.14027. [CrossRef]
20. 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]