symmetry

MDPI

Article

An Improved BGE-Adam Optimization Algorithm Based on Entropy Weighting and Adaptive Gradient Strategy

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

[1] School of Intelligent Science Engineering, Shenyang University, Shenyang 110044, China; shaoyichuan@syu.edu.cn (Y.S.); snowise@syu.edu.cn (L.Z.)
[2] School of Information Engineering, Shenyang University, Shenyang 110044, China; wangjt17888@outlook.com (J.W.); yuhao9733@outlook.com (H.Y.)
[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: sunhaijing@syu.edu.cn; Tel.: +86-1300-245-7275

Abstract: This paper introduces an enhanced variant of the Adam optimizer—the BGE-Adam optimization algorithm—that integrates three innovative technologies to augment the adaptability, convergence, and robustness of the original algorithm under various training conditions. Firstly, the BGE-Adam algorithm incorporates a dynamic β parameter adjustment mechanism that utilizes the rate of gradient variations to dynamically adjust the exponential decay rates of the first and second moment estimates (β_1 and β_2), the adjustment of β_1 and β_2 is symmetrical, which means that the rules that the algorithm considers when adjusting β_1 and β_2 are the same. This design helps to maintain the consistency and balance of the algorithm, allowing the optimization algorithm to adaptively capture the trending movements of gradients. Secondly, it estimates the direction of future gradients by a simple gradient prediction model, combining historic gradient information with the current gradient. Lastly, entropy weighting is integrated into the gradient update step. This strategy enhances the model's exploratory nature by introducing a certain amount of noise, thereby improving its adaptability to complex loss surfaces. Experimental results on classical datasets, MNIST and CIFAR10, and gastrointestinal disease medical datasets demonstrate that the BGE-Adam algorithm has improved convergence and generalization capabilities. In particular, on the specific medical image gastrointestinal disease test dataset, the BGE-Adam optimization algorithm achieved an accuracy of 69.36%, a significant improvement over the 67.66% accuracy attained using the standard Adam algorithm; on the CIFAR10 test dataset, the accuracy of the BGE-Adam algorithm reached 71.4%, which is higher than the 70.65% accuracy of the Adam optimization algorithm; and on the MNIST dataset, the BGE-Adam algorithm's accuracy was 99.34%, surpassing the Adam optimization algorithm's accuracy of 99.23%. The BGE-Adam optimization algorithm exhibits better convergence and robustness. This research not only demonstrates the effectiveness of the combination of these three technologies but also provides new perspectives for the future development of deep learning optimization algorithms.

Keywords: deep learning; BGE-Adam optimization algorithm; dynamic β adjustment; gradient prediction model; entropy weighting

Citation: Shao, Y.; Wang, J.; Yu, H.; Sun, H.; Xing, L.; Zhao, Q.; Zhang, L. An Improved BGE-Adam Optimization Algorithm Based on Entropy Weighting and Adaptive Gradient Strategy. *Symmetry* **2024**, *16*, 623. https://doi.org/10.3390/sym16050623

Academic Editor: Aviv Gibali

Received: 9 April 2024
Revised: 29 April 2024
Accepted: 13 May 2024
Published: 17 May 2024

1. Introduction

In today's rapidly evolving field of artificial intelligence, deep learning is reshaping our understanding of machine learning with its powerful data representation capabilities. Its applications in image recognition, natural language processing, autonomous driving, and other fields have proven its enormous potential for solving complex pattern recognition and high-dimensional data-processing problems [1]. The success of deep neural networks relies on effective training methods, among which optimization algorithms play a crucial

role. Optimization algorithms not only determine the convergence speed of model training, but also directly affect the stability of the training process and the final performance of the model. The Adam optimizer [2], one of the most popular optimization algorithms, has gained widespread recognition in the field of deep learning for its excellent adaptability and efficient computational performance. By combining the momentum method with an adaptive learning rate adjustment mechanism, the Adam optimization algorithm enhances the stability and convergence speed of the training process. Nonetheless, researchers have found that the original Adam algorithm has some issues in practical applications: the Adam optimizer still has limitations in finding the global optimal solution, particularly when dealing with complex loss surfaces, where the standard Adam optimizer may fall into local minima or converge slowly in certain situations [3].

Consequently, researchers have been dedicated to proposing various improvements to enhance the performance and stability of the Adam algorithm, as well as to extend its application scope. Substantial research has been conducted to refine and explore the Adam optimization algorithm. Z. Zhang and others [4] introduced an improved variant called ND-Adam, based on controlling the variance in parameter update directions, aimed at maintaining a more consistent update path. However, although this method has shown enhancements in direction consistency, it may negatively affect convergence speed for certain problems. Reyad, M and colleagues [5] proposed the HN Adam optimization algorithm, an iteration over Adam designed to bolster its precision and generalization capabilities. However, this too comes with its own set of challenges, requiring fine-tuning of newly introduced hyperparameters, thus adding to its complexity. Juntang Zhuang et al. [6] put forward the AdaBelief Optimizer, accounting for the disparity between predicted and observed gradients with the goal of promoting stability and rapid convergence. Nevertheless, this approach might lead to overcorrection at times, impacting training dynamics. John Duchi et al. [7] introduced the AdaGrad algorithm, adapting learning rates to fit each dimension of the parameters, offering better performance with sparse data. However, the approach suffers from a monotonically decreasing learning rate, which could prematurely halt learning. X. Chen, C. Liang, and others [8] devised the AdamW optimization algorithm, refining the weight update rules through the separation of weight decay, an approach better suited to addressing regularization issues, though it might require further adjustments to the decay rate to prevent overfitting. Luo Liangchen and colleagues [9] came up with the AdaBound optimization algorithm that amalgamates the benefits of AdaGrad and SGD, dynamically bounding the learning rate to approximate that of SGD. However, the algorithm's effectiveness depends on the choice of boundary functions, demanding continuous adjustments to these boundaries, thus increasing the complexity of the optimization process. Matthew D. Zeiler and others [10] proposed the AdaDelta optimization algorithm, which mitigates the rapid decay of learning rates to zero by considering historical gradient information. Yet, it also possesses limitations, calling for careful adjustment of its hyperparameters to suit specific optimization problems. Variants like AdaMax and Nadam improved second-moment estimations [11], adjusting learning rates more effectively across various gradient magnitudes [12]. N. Landro, I. Gallo, and R. La Grassa [13] explored a novel optimization method combining Adam and SGD [14]. A potential drawback of merging these two approaches is the additional hyperparameter adjustments needed to decide when and how to integrate both optimizers, possibly making the process more complex. Dokkyun Yi and others [15] put forward an Adam-based improved optimization algorithm, specifically targeting efficient optimization methods for non-convex cost functions. The proposed approach dealt with the local minima issue by introducing a cost function in the parameter update rules of the ADAM method. However, these modified algorithms do not always guarantee superior performance over Adam in practical applications. Such enhancements may confront issues like unreasonable hyperparameter adjustment, increased computational complexity, and local optima [16].

In light of these challenges, this paper introduces a novel variant of the Adam optimizer—BGE-Adam. This optimizer aims to enhance the adaptability and robustness of

the original algorithm under various training conditions by integrating three innovative technologies. First, by introducing a dynamic adjustment mechanism for the β parameters, BGE-Adam is able to capture the trends in gradient changes more flexibly, making more effective use of gradient information. Second, in conjunction with a simple gradient prediction model, the optimizer can anticipate the future direction of gradients, thus allowing for more accurate adjustment of the parameter update strategy. Finally, by incorporating entropy weighting into the gradient update steps, BGE-Adam enhances the model's ability to explore complex loss landscapes, effectively improving the model's adaptability to intricate loss surfaces. Extensive experimental validation was conducted on multiple standard datasets, and the results show that BGE-Adam can accelerate the training process and improve the performance of the final model in most cases. These achievements not only validate the effectiveness of the proposed combination of techniques but also pave new avenues for the future development of deep learning optimizers. By delving deeper into the potential of these innovative technologies, we expect to offer more efficient and robust optimization tools for the deep learning field, further advancing the application and development of deep learning technologies across various domains.

2. Related Work

The Adam optimizer, initially introduced by Kingma and Ba, has been widely used due to its robustness and efficiency across a broad array of tasks[2]. However, recent advancements have suggested that the static hyperparameters in the Adam algorithm, namely β_1 and β_2, which control the exponential decay rates for the moment estimates, could be suboptimal in dynamically changing the landscapes of the loss functions [17,18]. This is why dynamically adjusting the beta parameter may help. Adaptability: If the algorithm can dynamically adjust the β value based on the current change in the gradient, the algorithm can react more flexibly to different types of data encountered during training and different training stages. For example, in flat areas of the loss surface, it may be necessary to rely more on historical gradient information to maintain direction and speed, while in complex surface areas, it may be necessary to rely less on historical information to quickly adapt to new gradients. Robustness: By maintaining the β parameter within a reasonable range, that is, setting an upper and lower limit, extreme situations can be avoided. This is because a value of β very close to 1 or very small may lead to unstable optimization behavior—close to 1 may make the optimizer over-smooth in the gradient information and thus, unresponsive to new gradient changes; too small a value may cause The optimizer is too sensitive to gradient noise, making the steps too violent and unstable. By setting reasonable limits, one can ensure that the optimizer finds a balance between these two extremes and achieves robust performance.

Dynamically adjust the β parameter based on the gradient change rate, and adjust $\beta 1$ and $\beta 2$ in real time based on the historical information of the gradient in the algorithm: These adjustments can make the optimizer more adaptable to the local characteristics of the loss function, thereby improving the efficiency and stability of optimization. This process can be divided into the following steps. In order to calculate the gradient change rate cr_t, we need to compare the gradient of the current step g_t with the gradient of the previous step g_{t-1}. The calculation process is as follows shown in Equation (1) in the next section, in which ϵ is a small positive number that prevents the denominator from becoming zero. With the gradient change rate, the optimizer will adjust the parameter based on this rate. When the change rate is high, the gradient fluctuates greatly. At this time, $\beta 1$ may be reduced to reduce momentum and $\beta 2$ may be increased to improve stability; conversely, $\beta 1$ may be increased to increase momentum and $\beta 2$ may be reduced to track faster changes. The calculation process is shown in Equations (2) and (3), where f(cr_t) is some function that defines the position of the β value between its minimum and maximum values according to the gradient rate of change. Algorithms that dynamically adjust β parameters may perform better when dealing with complex loss functions than traditional algorithms. In particular, when the loss function has a steep region (possibly a local minimum) or a flat region

(possibly a saddle point), the algorithm can adapt to different regions more flexibly and avoid falling into the local optimum or wandering around the saddle point, thus improving convergence. At the same time, reducing oscillations and accelerating convergence can improve the efficiency of the optimization process.

Add entropy weights in the parameter update step: Based on the concepts of simulated annealing and noise injection technology, entropy weight introduces random elements into the optimization process [19]. Introducing the optimization method of entropy weight helps the algorithm perform the necessary random exploration by properly injecting noise into parameter updates. The benefits and operating principles of this mechanism can be understood from the following aspects. In the early stages of optimization, the noise introduced by high entropy weights allows the algorithm to explore a wide area of the solution space, which increases the possibility of jumping out of the local optimum and discovering a better solution space. This "exploration" in a high-entropy state helps the algorithm obtain more information about the characteristics of the objective function. Secondly, introducing an entropy weight dynamic adjustment strategy can help the algorithm avoid falling into a local minimum. When falling into a local optimum during optimization, the standard gradient descent method may have difficulty in escaping because around the local optimum point, the gradient of the objective function is close to or equal to zero. The random perturbation introduced by the entropy weight gives the algorithm a certain probability to cross obstacles near the local optimum and explore other possible better regions. When implemented, the entropy weight adjustment can be dynamically adjusted based on the current state of the algorithm and historical performance data. If slow progress or stalled convergence is detected, the entropy weight can be temporarily increased to add random perturbations to help the algorithm "unhook" from the jam. Conversely, as the algorithm gradually approaches the potential optimal solution, gradually reducing the entropy weight can reduce noise and fine-tune the solution closer to the optimal solution.In summary, by rationally introducing and adjusting entropy weights in the update step, the exploratory nature of the optimization algorithm is enhanced, thereby improving the ability to avoid local optima and discover better global solutions.

3. Design of the BGE-Adam Algorithm

3.1. Dynamically Adjusted β-Parameter Mechanisms

For the standard Adam algorithm, the β parameters (β_1 and β_2) are set as hyperparameters before training begins and remain unchanged throughout the process. These parameters have a critical impact on the decay of the first-order moment estimation (mean) and the second-order moment estimation (uncentered variance). To adjust β_1 and β_2, BGE-Adam introduces a mechanism based on the gradient's rate of change to dynamically adjust these two parameters: During the parameter refinement process, if the direction of gradient change is consistent over several consecutive steps, this indicates that the current direction is reliable, and thus, the value of β_1 can be increased to give more weight to the momentum, thereby accelerating the learning process. Conversely, if the direction of gradient change varies frequently, this may suggest proximity to a local minimum, in which case the value of β_1 can be reduced to lessen the impact of momentum and increase the model's sensitivity to new information. The same strategy can be applied to β_2 to adjust the sensitivity to the rate of gradient change. During initialization, BGE-Adam sets upper and lower limits for changes in the β_1 and β_2 parameters, ensuring dynamic adjustments occur within this range. BGE-Adam adapts to changes in gradients by dynamically adjusting these hyperparameters; the formula for its dynamic adjustment is as follows:

The calculationof the gradient change rate cr_t is shown in Equation (1):

$$cr_t = \frac{\|g_t - g_{t-1}\|}{\|g_{t-1}\| + \epsilon} \tag{1}$$

The dynamicadjustments of β_1 and β_2 are shown in Equations (2) and (3):

$$\beta_{1,t} = \beta_{1,\min} + (\beta_{1,\max} - \beta_{1,\min}) \times (1 - cr_t) \tag{2}$$

$$\beta_{2,t} = \beta_{2,\min} + (\beta_{2,\max} - \beta_{2,\min}) \times (1 - cr_t) \tag{3}$$

The term cr_t represents the gradient change rate at the time step, ϵ is a very small number to avoid division by zero error, g_t and g_{t-1}, respectively, denote the gradient at the current and previous time step, and "min" and "max" represent the minimum and maximum values of hyperparameters, respectively.

3.2. Gradient Prediction Model

BGE-Adam introduces a gradient prediction mechanism to assist in forecasting the direction of future gradients, thereby enhancing optimization performance. This prediction principle is completed by taking the weighted average of the current and past gradients. The parameter α is the weight factor of historical gradients in this weighted average. By adjusting α, one can control the model's sensitivity to changes in gradients as well as the extent to which historical information is considered when predicting the next gradient. Within the gradient prediction model, the α parameter governs the degree to which historical gradients influence the current gradient forecast. When α is close to 1, the predicted gradient will rely more on historical gradients, suggesting that the model considers that gradient changes will not be significant, and thus it filters out some of the noise in the gradient, making the prediction smoother. When α is small and far from 1, the model gives more weight to the current gradient, and the predicted gradient will respond more quickly to new gradient information. The working principle here is similar to that of an exponentially weighted moving average, a common technique used to smooth time series data. The α parameter determines the degree of smoothing or conservatism of the gradients based on historical gradient information. The gradient prediction model can help to regulate the gradient update path, and its calculation is shown in Equation (4):

$$\hat{\mathbf{g}}_t = \alpha \cdot \hat{\mathbf{g}}_{t-1} + (1 - \alpha) \cdot \mathbf{g}_t \tag{4}$$

where $\hat{\mathbf{g}}_t$ is the predicted gradient at time step t, α is the weight factor, and $\hat{\mathbf{g}}_{t-1}$is the predicted gradient at time step $t - 1$.

3.3. Entropy Weights

To enhance the optimization process's exploratory nature and enable the model to escape potential local optima, BGE-Adam incorporates entropy weighting during the parameter update step. The adjustment of entropy weight aims to introduce randomness into the update steps of the optimization process, thereby assisting the model in avoiding local minima and increasing the likelihood of encountering a global minimum. This concept takes inspiration from the thermodynamic notion of "entropy", which fundamentally measures the disorder within a system [20]. Within optimization algorithms, the introduction of certain randomness can emulate a form of "thermodynamic noise" [21], aiding in the escape from local optima and the discovery of additional potential solutions. More specifically, the BGE-Adam optimizer adjusts the update steps by incorporating a stochastic perturbation term associated with a hyperparameter ω. This term is generated by invoking the torch.randn_like(p.data) method, which yields random numbers matching the shape and size of the current parameters and conforms to a Gaussian distribution. The mean of these random numbers is then calculated and multiplied by ω as the weighting of entropy. Subsequently, during parameter updates, the model's predicted next gradient is multiplied by the computed entropy adjustment coefficient e_t introducing entropy-weighted random perturbations into each parameter's update step. By injecting a certain amount of noise into the updates, this increases randomness and helps prevent the optimization process from easily becoming stuck in local minima. The introduced entropy weighting adjustment

adds randomness to the weight updates, which can, to some extent, improve the likelihood of the algorithm avoiding local minima and exploring global minima. This approach can also enhance the model's generalization performance in certain contexts as it encourages the exploration of less thoroughly searched regions of the parameter space. The specific computation is demonstrated by the following Equation (5):

$$e_t = 1 + \omega \cdot \mathcal{N}(0, 1) \tag{5}$$

In which ω is the hyperparameter that controls the level of noise and $\mathcal{N}(0, 1)$ denotes the standard normal distribution.

Integrating the three strategies mentioned, the parameter update rules are demonstrated in the following Equations (6)–(10):

$$m_t = \beta_{1,t} \cdot m_{t-1} + (1 - \beta_{1,t}) \cdot g_t \tag{6}$$

$$v_t = \beta_{2,t} \cdot v_{t-1} + (1 - \beta_{2,t}) \cdot g_t^2 \tag{7}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_{1,t}^t} \tag{8}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_{2,t}^t} \tag{9}$$

$$\theta_{t+1} = \theta_t - \eta_t \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \cdot e_t \tag{10}$$

In this context, θ_t represents the parameter to be optimized, m_t and v_t are the first-order and second-order moment estimates of the gradient, respectively, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first-order and second-order moments, η_t is the learning rate, and e_t is the entropy weight adjustment. The hyperparameter description table in the BGE-Adam optimization algorithm is shown in Table 1.

Table 1. Hyperparameters Description of the Integrated Adam Optimizer.

Hyperparameter	Description	Default Value
`lr` (Learning Rate)	Controls the update step size of the model at each iteration	0.001
`alpha`	Determines the weight ratio of the predicted gradient to the actual gradient	0.5
`betas` (β Parameters)	A pair of values used to compute the moving averages of the gradient and its square	(0.9, 0.999)
`eps` (ε)	A small number to prevent division by zero errors	1×10^{-8}
`weight_decay`	Weight decay, used for regularization and to prevent overfitting	0
`entropy_weight`	Entropy weight, used to introduce randomness into the parameter space during optimization	0.01
`amsgrad`	Boolean value indicating whether to use the AMSGrad variant to prevent sudden changes in gradient updates	False
`beta1_max`	The maximum adjustment value of beta1	0.9
`beta1_min`	The minimum adjustment value of beta1	0.5
`beta2_max`	The maximum adjustment value of beta2	0.999
`beta2_min`	The minimum adjustment value of beta2	0.9

3.4. BGE-Adam Algorithm

During the step function of the algorithm, the change rate between the current gradient and the gradient from the previous time step is first calculated. This change rate, computed by the $compute_gradient_change_rate$ function, is used to dynamically adjust the values of β_1 and β_2. The logic for dynamic adjustment is computed through the $compute_dynamic_\beta$ function, which updates the values of the β parameters based on the gradient change rate and the predetermined minimum and maximum values. This improvement allows the optimizer to adapt to the trend in the gradients, enabling it to react quickly when there is a significant change in gradient direction by adjusting the decay rates of the first-order and second-order moment estimates. This helps the optimizer to modify its behavior more flexibly, showing better performance at different optimization stages. Gradient prediction is implemented within the GradientPredictionModel class, where a simple Exponential Moving Average

(EMA) model is used to predict the following gradients. At each step, the actual gradient is combined with the predicted gradient to estimate the direction of future gradients. By predicting future gradient directions, the optimizer can adjust its update strategy in advance, potentially avoiding overly large parameter updates and instability. This assists in smoothing the optimization path and accelerating convergence. Finally, in the parameter update section of the step function, entropy weight adjustment is incorporated. This is achieved by adding a certain amount of random noise to each weight update. The introduction of entropy weight (multiplied by random normal noise) encourages the optimizer to explore a broader parameter space, enhancing its ability to escape potential local minima. The introduction of entropy weight is akin to adding an exploratory mechanism during the optimization process, allowing the algorithm not only to focus on the current best direction of gradient descent but also to perform a random exploration to some extent. This random exploration can help avoid local optima and has the potential to discover better global solutions. In summary, BGE-Adam merges three strategies, dynamic β adjustments, gradient prediction, and entropy weighting, offering a new adaptive method for gradient descent. With this algorithm, the optimization process can make more refined adjustments to model parameters while maintaining a stable update step size, which has enhanced the algorithm's ability to explore the solution space, thus improving convergence speed and robustness of the model in different application scenarios. The specific computational process of the improved optimization algorithm BGE-Adam is shown in Algorithm 1—The specific computational process of the improved optimization algorithm BGE-Adam.

Algorithm 1: BGE-Adam.

Input: initial point P_0, first moment decay, second moment decay, regularization constant

Output: p_{new}

Input: initial point P_0, first moment decay, second moment decay, regularization con-stant
Initialize m_0 and v_0, θ_0, ω, α, $\beta_{1_{\min}}$, $\beta_{1_{\max}}$, $\beta_{2_{\min}}$, $\beta_{2_{\max}}$
for $t = 1$ **to** T
  $g_t = \nabla_\theta f_t(\theta_{t-1})$
  $m_t = \beta_{1_{t-1}} \cdot m_{t-1} + (1 - \beta_{1_{t-1}}) \cdot g_t$
  $v_t = \beta_{2_{t-1}} \cdot v_{t-1} + (1 - \beta_{2_{t-1}}) \cdot g_t^2$
  $cr_t = \frac{\|g_t - g_{t-1}\|}{\|g_{t-1}\| + \epsilon}$
  $\beta_{1_t} = \beta_{1_{\min}} + (\beta_{1_{\max}} - \beta_{1_{\min}}) \times (1 - cr_t)$
  $\beta_{2_t} = \beta_{2_{\min}} + (\beta_{2_{\max}} - \beta_{2_{\min}}) \times (1 - cr_t)$
  $\hat{m}_t = \frac{m_t}{1-\beta_1^t}; \hat{v}_t = \frac{v_t}{1-\beta_2^t}$
  $bias\_correction1 = 1 - \beta_1^t; bias\_correction2 = 1 - \beta_2^t$
  $step\_size = \frac{lr \cdot \sqrt{1-\beta_2^t}}{1-\beta_1^t}$
  $\hat{g} = \alpha \cdot g_{\text{prev}} + (1 - \alpha) \cdot g$
  if p in $grad\_pred\_model$
    $pred\_grad = grad\_pred\_model[p].pred(grad)$
  else
    $grad\_pred\_model[p].pred(grad) = GradPredModel(group['\alpha'])$
    $pred\_grad = grad\_pred\_model[p].pred(grad)$
  else do
  $e_t = 1 + \omega \cdot N(0, 1)$
  $entropy\_adjustment = 1 + entropy\_weight \cdot E[N(p.data)]$
  $p_{\text{new}} \leftarrow p - step\_size \times \frac{\hat{g}}{\sqrt{\hat{v}} + \epsilon} \times entropy\_adjustment$
end for
Return p_{new}

4. Experimental Results and Analyses

4.1. Experiment Environment and Configuration

This study implements the improved BGE-Adam optimization algorithm based on the PyTorch deep learning framework [22]. This algorithm is an enhancement of the standard Adam optimizer, and dynamically adjusts the values of β_1 and β_2 based on the rate of change of gradients and predicts the next gradient, alongside integrating an entropy weight into the parameter update adjustment strategy. The main software versions used in the experiments are shown in Table 2.

Table 2. Main software versions used in the experiments.

Software Resources	Version
Python	3.11.3
torch	2.0.0+cu11.8
torchvision	0.15.1+cu11.8
torchaudio	2.0.1+cu11.8
lightning	2.0.4
wandb	W&B Local 0.47.2

The table lists the versions of Python and several important libraries used in the experiments. This research was conducted using the lightning-template-hydra framework, developed in Python, to validate the performance of the algorithm. The experiments have strict requirements for the versions of the libraries and Python, with a Python environment version requirement of 3.9 or above and a PyTorch version requirement of 2.0.0 or above. This study tests the performance of the BGE-Adam optimization algorithm on multiple datasets: MNIST, CIFAR10, and a specialized medical image dataset referred to here as "Medical". Each of these three datasets contains a different number of images and image quality. The MNIST dataset includes 70,000 grayscale images with 10 categories representing handwritten digits 0–9. The CIFAR10 dataset [23] contains 60,000 color images in 10 categories. The Medical dataset consists of 1885 color images, divided into 8 categories based on the severity of the conditions [24]. In the experiments, all three datasets are divided into three parts, including a training set, a validation set, and a test set. The experimental datasets are shown in Table 3.

Table 3. Experimental dataset.

Dataset	Number of Datasets	Training Set	Validation Set	Test Set	Classification
CIFAR10	60,000	45,000	5000	10,000	10
MNIST	70,000	55,000	5000	10,000	10
Medical	1885	1400	200	285	8

4.2. Experimental Results and Analysis

The BGE-Adam optimization algorithm improves upon the traditional Adam optimizer by introducing a mechanism that dynamically adjusts the β_1 and β_2 values according to the gradient change rate and predicting the next gradient step. By doing so, the algorithm tailors the influence of past gradients on both the first-order (mean) and second-order (uncentered variance) moment estimates, which can lead to better performance and faster convergence compared to using fixed β values. Secondly, the BGE-Adam algorithm integrates a gradient prediction mechanism that helps the optimizer traverse the loss surface more smoothly, preventing excessive oscillations. This contributes to a steadier and often quicker convergence to potentially better global minima as the model can anticipate and correct its path during optimization. Thirdly, by incorporating entropy weight adjustments into the learning rate, the BGE-Adam optimizer introduces randomness into the optimization process, encouraging the algorithm to explore new regions in the parameter space rather than merely fluctuating around the current minimum. The concept of entropy, borrowed from thermodynamics, symbolizes a measure of disorder; hence, its adjustment

introduces a form of 'controlled chaos', which can be particularly effective in avoiding local minima and increasing the likelihood of discovering superior global minima. This feature is invaluable when dealing with complex and high-dimensional problems, as it expands the robustness of the optimization process against becoming trapped in suboptimal solutions.

To validate the performance of the BGE-Adam optimization algorithm, we conducted comparative experiments with several existing optimizers: SGD, Adam, Adadelta, NAdam, Adagrad, and Adamax. Through testing these optimization algorithms on different datasets, we found that the BGE-Adam optimization algorithm outperforms the others in both accuracy and loss reduction, the experimental results are compared as shown in Table 4.

Table 4. Comparison of experimental results.

Dataset	Optimization Algorithm	Accuracy	Loss
MNIST	Adam	99.23%	0.04474
	Adadelta	83.70%	0.4897
	Adamax	98.84%	0.0629
	Adagrad	89.09%	0.4041
	Nadam	99.30%	0.03188
	SGD	97.11%	0.1095
	BGE-Adam	99.34%	0.0756

Table 4. *Cont.*

Dataset	Optimization Algorithm	Accuracy	Loss
CIFAR10	Adam	70.11%	1.195
	Adadelta	27.37%	1.951
	Adamax	55.45%	1.693
	Adagrad	29.58%	1.974
	Nadam	68.95%	2.439
	SGD	48.66%	1.478
	BGE-Adam	71.4%	1.458
Medical	Adam	67.66%	3.481
	Adadelta	60.85%	2.629
	Adamax	66.81%	2.657
	Adagrad	67.23%	2.401
	Nadam	67.66%	2.363
	SGD	68.09%	3.681
	BGE-Adam	69.36%	2.852

4.3. Experimental Setup

(1) Optimization Algorithms: Comparative experiments were conducted between the BGE-Adam optimization algorithm and six existing optimization algorithms: SGD, Adam, Adadelta, Adamax, Adagrad, and NAdam.

(2) Dataset Selection: Ten sets of comparative experiments were conducted on the seven optimization algorithms on the traditional datasets, MNIST, CIFAR10, and a medical image dataset for gastrointestinal disease diagnostics. The average of the results from the ten comparative experiments was taken as the final experimental outcome.

(3) Network Model Selection: The comparative experiments in this study utilize the PyTorch deep learning architecture, selecting the lightweight neural network MobileNetV2 for comparison experiments. To evaluate the performance of optimizers more fairly, the BGE-Adam optimization algorithm and the six other optimizers in the comparison all employ the same network model architecture.

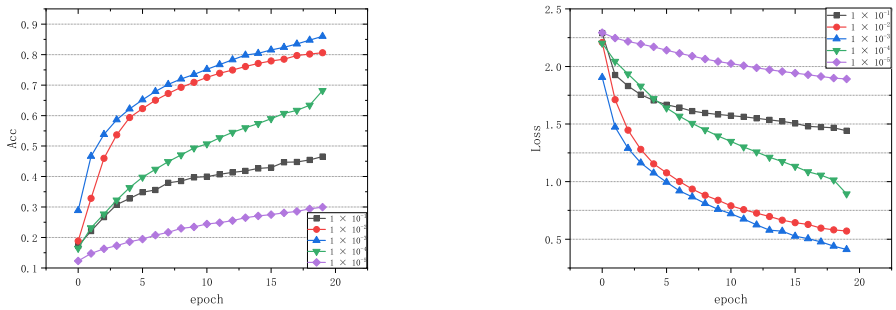
(4) Hyperparameter Initialization: In addition to the inherent parameters within the optimizers, the same initial hyperparameters, including learning rate and weight decay, are set for each optimizer. The batch size used in the experiments is determined to be 128, and the number of training iterations is set to 100.

(5) Model Training: Each optimizer is used to train the model for 50 epochs separately to ensure the repeatability of the experimental results.

4.4. Experimental Implementation

Determination of Hyperparameter Learning Rate lr: Before conducting the comparative experiment of optimization algorithms, the BGE-Adam optimization algorithm is tested using different learning rates. The experimental results are shown in Figure 1.

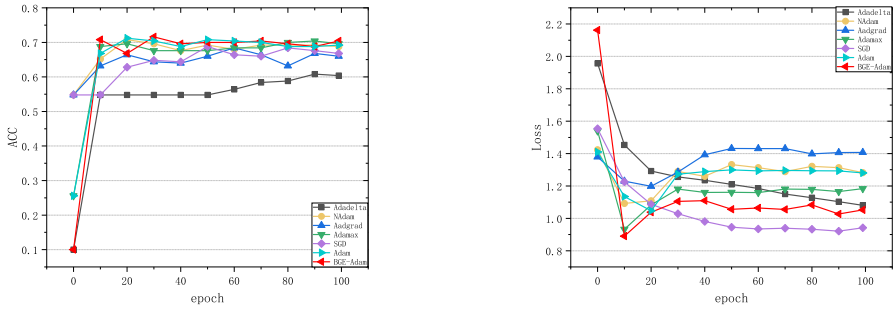
The BGE-Adam optimization algorithm was tested with three different learning rates (1×10^{-1} , 1×10^{-2}, 1×10^{-3}, 1×10^{-4}, and 1×10^{-5}) on the CIFAR10 dataset, with the number of iterations (epochs) set to 20. The left (a) shows the accuracy when tested on the validation set. When the learning rate (lr) is set to 1×10^{-3}, the accuracy is the highest, indicating the best performance. The right (b) shows the loss values, and again, when the lr is set to 1×10^{-3}, the loss values are the smallest, indicating the best performance. When the lr is set to 1×10^{-4}, the accuracy is low and the loss values are high, indicating the worst performance. Based on the experimental results, an initial learning rate of 1×10^{-3} was finally determined to be the best choice.



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

Figure 1. Comparison of training results with different learning rates for BGE-Adam settings. (**a**) Comparison of accuracy values for different learning rates; (**b**) Comparison of loss values for different learning rates.

Figure 2 shows the training comparison results of the seven optimization algorithms in the comparison on the Medical gastrointestinal disease diagnosis medical dataset. Figure 3 compares the training results of the seven optimization algorithms on the MNIST dataset.

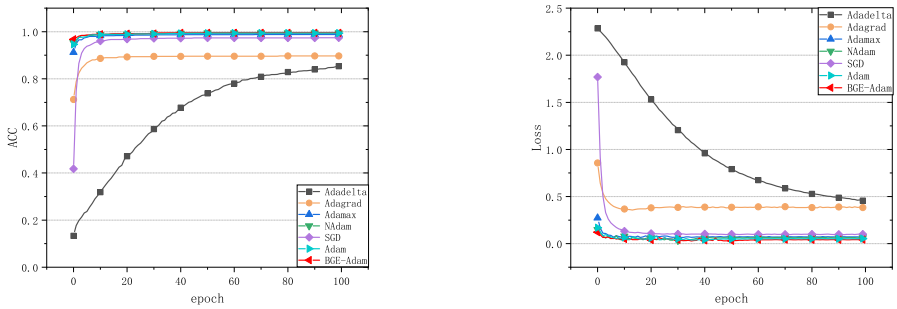


(**a**) (**b**)

Figure 2. Comparison between different optimization algorithms on the Medical Gastrointestinal Condition Diagnosis dataset. (**a**) Accuracy on validation set; (**b**) Loss value on validation set.

Based on the experimental results shown in Figure 4, it can be observed that as the number of training iterations increases, the Adam-based Adadelta algorithm, when applied to the CIFAR10 dataset, converges slower, achieves lower accuracy, and exhibits poorer performance compared to the other optimization algorithms. Additionally, the BGE-Adam optimization algorithm demonstrates higher accuracy right from the beginning of the training, significantly outperforming other optimization algorithms in the early stages of training. With increasing iterations, the accuracy of BGE-Adam continues to

stabilize and improve, eventually converging with an advantage over other improved optimization algorithms, particularly showing better convergence and accuracy than the Adam optimization algorithm. The superior performance of BGE-Adam can be attributed to a key feature of the algorithm: the dynamic adjustment of the first- and second-order momentum factors (β_1 and β_2). This adjustment is dependent on the rate of change of the gradient, allowing the algorithm to flexibly adjust the step size according to the historical trend in gradient changes. This helps in adjusting the intensity of the learning rate at different stages of model training, such as the initial phase and the convergence phase, thereby achieving faster convergence and higher accuracy on the CIFAR-10 dataset. Furthermore, BGE-Adam also employs a gradient prediction model to forecast the next gradient, which may enhance the efficiency of gradient descent. By considering information from past gradients to predict the direction and magnitude of the next gradient, the gradient prediction model may make BGE-Adam's search in the parameter space more precise.



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

Figure 3. Comparison of different optimization algorithms on MNIST dataset. (**a**) Accuracy on validation set; (**b**) Loss value on validation set.

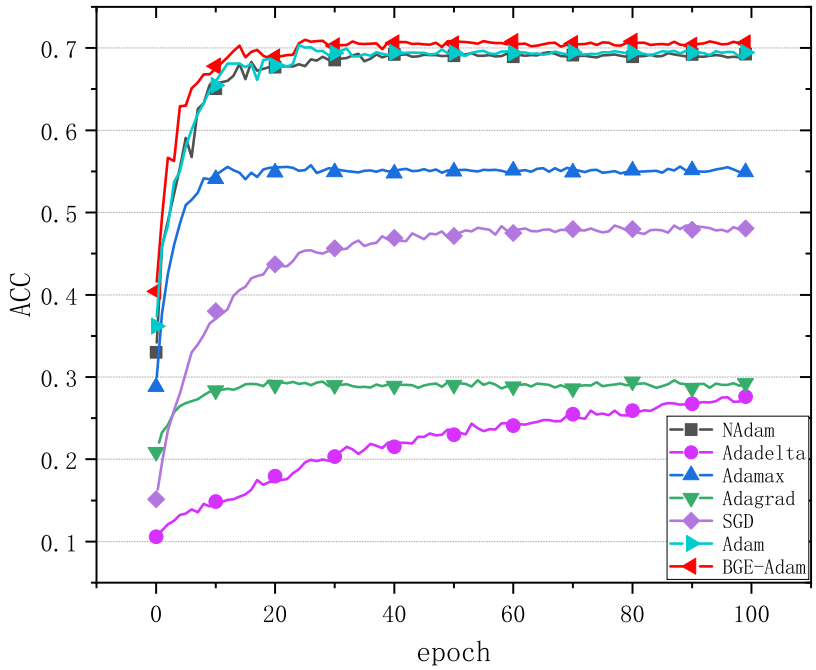


Figure 4. Comparison of accuracy of different optimization algorithms on CIFAR10 dataset.

Figure 5 demonstrates that during the early stages of training, the BGE-Adam optimization algorithm converges significantly faster than the other algorithms. The Adam-based Adadelta optimization algorithm exhibits significantly higher loss values compared to the other optimization algorithms, indicating larger losses. The Adam, Nadam, and BGE-Adam algorithms all converge relatively quickly. By the 40th iteration, the BGE-Adam algorithm

achieves the minimum loss value, which is significantly lower than that of the other six optimization algorithms at the same iteration, resulting in the best convergence performance. The reason for this phenomenon may be that BGE-Adam incorporates entropy weights (entropy weight) into parameter updates, which adds randomness to the update steps. This randomness can help the algorithm bypass local minima and explore more regions of the loss function, ultimately finding a better global optimal solution. As the number of iterations increases and approaches the later stages, the loss value of BGE-Adam tends to stabilize. In terms of the final convergence results, BGE-Adam's loss value is lower than the minimum loss values achieved by SGD (Stochastic Gradient Descent) and Adamax, indicating better convergence compared to other algorithms. This suggests that BGE-Adam is effective in minimizing the loss function and achieving better optimization outcomes.

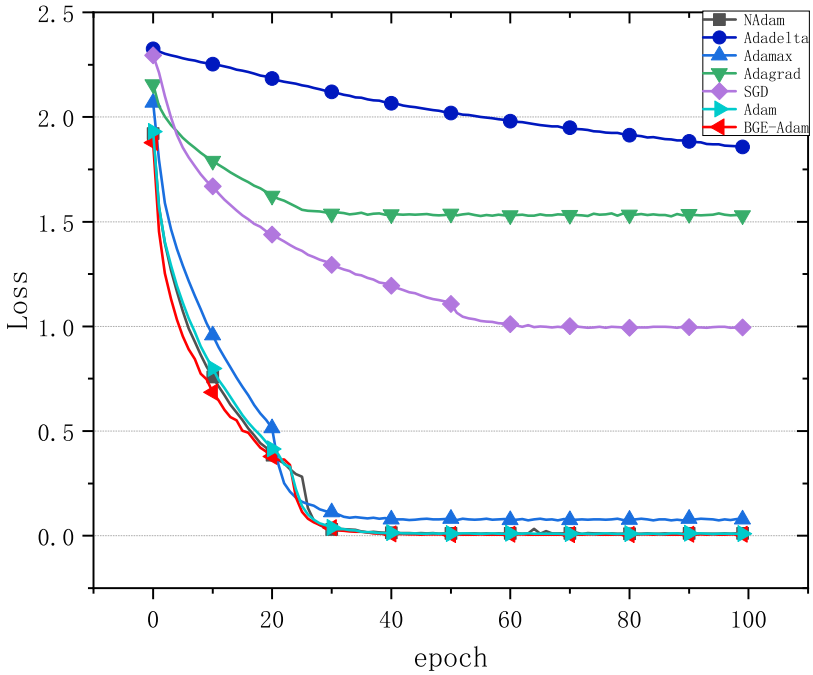


Figure 5. Comparison of loss values of different optimiaation algorithms on CIFAR10 dataset.

The graph in Figure 6 compares the training time and GPU utilization of the seven optimization algorithms in the comparison on the CIFAR10 dataset. The BGE-Adam optimization algorithm stands out for having a significantly lower GPU utilization rate than the other six algorithms. This is achieved by adaptively adjusting the exponential decay rates of the first- and second-order moment estimates based on the rate of change of the gradient. This allows BGE-Adam to reduce the frequency of parameter updates without compromising performance. Furthermore, by incorporating historical gradient information to estimate the future direction of gradients, BGE-Adam may optimize the training process, making the algorithm more precise and efficient during gradient updates. This can potentially reduce unnecessary computations and parameter adjustments, thereby decreasing the computational load on the GPU. However, as the number of iterations increases, BGE-Adam may occasionally update the parameters more conservatively to ensure more stable convergence to the global optimum, while this can enhance the final performance of the model, it may also result in a slight increase in training time. In summary, BGE-Adam offers a balance between performance and computational efficiency by adaptively adjusting the decay rates of moment estimates and incorporating historical gradient information. This can lead to lower GPU utilization and a more efficient training process. The trade-off is that the algorithm may require more cautious parameter updates, which can slightly increase training time but potentially improve the final performance of the model.

Overall, in the experiments on the CIFAR10 dataset, the BGE-Adam algorithm demonstrated superior performance compared to the other six optimization methods in the com-

parison. In particular, when compared to the widely used Adam algorithm, BGE-Adam shows not only a significant improvement in accuracy but also a more stable convergence. This experimental result further confirms the unique advantages of BGE-Adam among many optimization algorithms. It is particularly noteworthy that BGE-Adam exhibits exceptional strength in addressing issues related to learning rate sensitivity and enhancing the generalization capability of the model. These characteristics make BGE-Adam a powerful choice for optimization in scenarios where controlling the learning rate and improving the model's ability to generalize are critical.

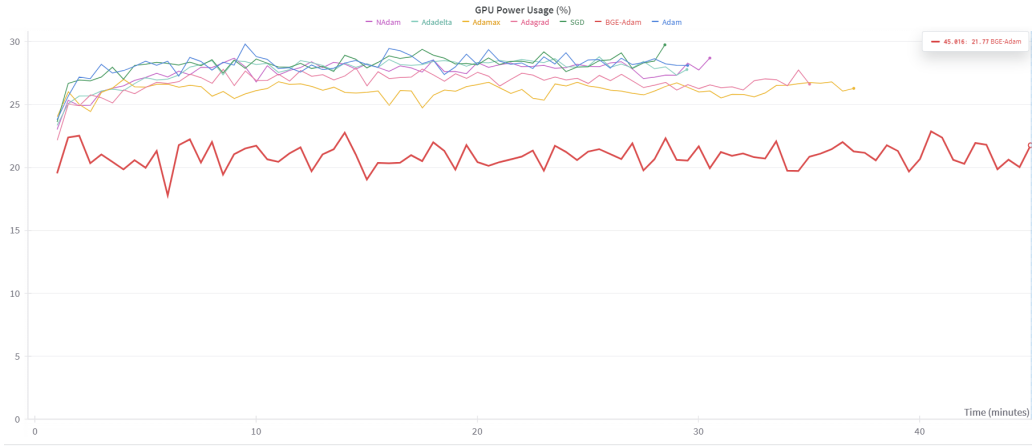


Figure 6. GPU occupancy for the seven algorithms in the comparison.

The research findings from this study showcase the exceptional performance of the BGE-Adam algorithm in addressing critical optimization issues during deep learning training. Compared to the traditional Adam algorithm, BGE-Adam is more efficient in adjusting the learning rate and avoiding the fluctuations and instability that may occur during training. This ensures the robustness of the model throughout the training process. More impressively, BGE-Adam has an enhanced ability to generalize and adapt to new data through its improved optimization mechanism. It exhibits stronger generalization capabilities, which is a significant advantage in the field of machine learning. Extensive testing on the CIFAR10, MNIST datasets, and Medical diagnostic datasets has demonstrated that BGE-Adam can consistently maintain its superiority under different environments and conditions. These results are not only valuable for researchers aiming for high training efficiency and model accuracy but also for practitioners in the field of machine learning, providing insights and inspiration for real-world applications. The robustness, stability, and generalization capabilities of BGE-Adam make it a promising candidate for optimization in deep learning tasks, where consistent and high-quality model performance is crucial.

Based on the data presented in Table 2, it is evident from the comparative studies conducted on three different datasets that BGE-Adam exhibits superior performance in terms of accuracy. Whether on the handwritten digit dataset MNIST, the image recognition dataset CIFAR-10, or the medical gastrointestinal disease diagnosis dataset, BGE-Adam demonstrates outstanding overall performance. This clearly demonstrates the algorithm's broad applicability across different types of datasets, along with its remarkable generalization capability and robustness. The BGE-Adam algorithm performs excellently across various types of image datasets, including grayscale and color images. In particular, after 100 training iterations, the algorithm demonstrates rapid convergence and stability, surpassing other comparative optimization algorithms, especially when compared to the standard Adam algorithm, with significantly higher accuracy. This indicates the superior performance of BGE-Adam in enhancing stability and escaping local optima. This performance of the algorithm not only highlights its outstanding generalization performance but also reinforces its applicability across different datasets and tasks. Further research

indicates that BGE-Adam exhibits superior performance whether applied to fully connected neural networks or convolutional neural networks, further demonstrating that its benefits are not limited to specific network structures. In experiments, the BGE-Adam algorithm demonstrates its multifaceted advantages, especially in terms of its generalization ability and rapid convergence, which are crucial characteristics. The BGE-Adam optimization algorithm achieves an accuracy of 71.4% on the CIFAR-10 test dataset, surpassing the 70.6% accuracy of the Adam algorithm. On the Medical test dataset, the accuracy is 69.36%, higher than Adam's 67.66%. On the MNIST dataset, the accuracy of the BGE-Adam algorithm is 99.34%, surpassing the 99.23% of the Adam optimization algorithm. The BGE-Adam optimization algorithm achieves the best accuracy on three datasets and different network structures and reaches the minimum loss value during training, which experimentally reinforces the potential of BGE-Adam as an algorithm for improving the performance and generalization power of deep learning models [25].

5. Conclusions

The BGE-Adam optimization algorithm, based on the standard Adam algorithm, not only delivers higher accuracy but also demonstrates improved adaptability and robustness, especially in high-noise environments such as medical datasets.As depicted in Figure 1, the algorithm performs optimally with a learning rate set to 1×10^{-3}. It exhibits faster convergence rates, greater stability, and enhanced robustness across multiple datasets. Extensive experimental comparisons of seven optimization algorithms, namely, Adam, SGD, Adadelta, Adamax, NAdam, Adagrad, and BGE-Adam, were conducted on the CIFAR-10 and MNIST and a medical image diagnosis datasets. The results, illustrated in Figures 2–4, indicate that the BGE-Adam optimization algorithm surpasses the other algorithms in the comparison in terms of accuracy. Figure 5 demonstrates that BGE-Adam converges significantly faster in terms of the loss values on the CIFAR-10 dataset. Specifically, BGE-Adam displays expedited convergence and heightened accuracy, which is particularly vital for tasks requiring iterative training of large neural networks. The algorithm's improved stability reduces fluctuations during parameter updates throughout training, aiding in more robust convergence of the models. Its outstanding robustness is manifested in higher accuracy across various datasets, especially in high-noise datasets such as medical images, showcasing its formidable adaptability to outliers and input noise, while pursuing high precision, BGE-Adam has consistently outperformed other optimizers in various test sets, affirming the efficacy of integrated and innovative techniques in the realm of optimization algorithms. This provides new avenues for the future development of deep learning models from a fresh perspective.

The BGE-Adam algorithm addresses the problem of the Adam optimization algorithm becoming trapped in local optima in certain cases; BGE-Adam's design considers enhancing global search capability by avoiding suboptimal solutions through more efficient gradient utilization strategies. Meanwhile, the Adam algorithm may exhibit slow startup phenomena in the initial training phase; BGE-Adam accelerates the model's learning speed in the early stages of training by dynamically adjusting optimization parameters. Sometimes, Adam performs poorly in convergence on specific tasks or data; the improvement mechanism adopted by BGE-Adam aims to provide more consistent convergence behavior and reduce performance fluctuations on specific types of datasets. We believe that BGE-Adam has the potential to be further validated and applied in various deep learning applications such as visual recognition and complex medical image analysis due to its outstanding performance. However, the improved BGE-Adam optimization algorithm also has some limitations. The current BGE-Adam optimization algorithm does not support sparse gradients, which limits its application in processing large-scale sparse datasets commonly used in natural language processing and recommendation systems. Future research can further explore BGE-Adam's performance on a wider range of datasets and different model architectures, as well as its efficacy and feasibility in practical applications. Additionally, how to more effectively utilize the innovative techniques within BGE-Adam to adapt to the

increasing complexity and diversity of deep learning models remains a worthy subject for further investigation.

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

Funding: This research was supported by the Liaoning Provincial Department of Education's Higher Education Foundation Research Project (General Project), Shenyang University of Technology, Project number: LJKZ0159; the Liaoning Provincial Department of Education Science "14th Five-Year Plan", Research on the Construction of New Technologies of Artificial Intelligence and High-Quality Education Service Supply System, 2023–2025. Project number: JG22DB488; the Ministry of Education's "Chunhui Plan", Research on Optimization Model and Algorithm of Microgrid Energy Scheduling Based on Biological Behavior, Project number: 202200209; the Liaoning Provincial Department of Education's Basic Research Project "Training and Application of Vertical Field Multi-Mode Deep Neural Network Model", Project number: JYTMS20231160; and the 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/wangjiantao1/BGE-Adam/tree/master(accessed on 29 March 2024). The website for the standard dataset CIFAR10 dataset is https://www.kaggle.com/datasets/gazu468/cifar10-classification-image (accessed on 1 May 2022). The website for the medical datasets is https://doi.org/10.1038/s41597-020-00622-y (accessed on 28 August 2020).

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

References

1. Anjum, M.; Shahab, S. Improving Autonomous Vehicle Controls and Quality Using Natural Language Processing-Based Input Recognition Model. *Sustainability* **2023**, *15*, 5749. [CrossRef]
2. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
3. Kashyap, R. A survey of deep learning optimizers–First and second order methods. *arXiv* **2023**, arXiv:2211.15596.
4. Zhang, Z.; Ma, L.; Li, Z.; Wu, C. Normalized Direction-preserving Adam. *arXiv* **2018**, arXiv:1709.04546.
5. Reyad, M.; Sarhan, A.; Arafa, M. A modified Adam algorithm for deep neural network optimization. *Neural Comput. Appl.* **2023**, *35*, 17095–17112. [CrossRef]
6. Zhuang, J.; Tang, T.; Ding, Y.; Tatikonda, S.; Dvornek, N.; Papademetris, X.; Duncan, J.S. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In Proceedings of the Advances in Neural Information Processing Systems, NeurIPS, Online, 6–12 December 2020.
7. Duchi, J.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
8. Yao, Z.; Gholami, A.; Shen, S.; Mustafa, M.; Keutzer, K.; Mahoney, M. ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning. *Proc. Aaai Conf. Artif. Intell.* **2021**, *35*, 10665–10673. [CrossRef]
9. Luo, L.; Xiong, Y.; Liu, Y.; Sun, X. Adaptive Gradient Methods with Dynamic Bound of Learning Rate. *arXiv* **2019**, arXiv:1902.09843.
10. Gill, K.; Sharma, A.; Anand, V.; Gupta, R. Brain Tumor Detection using VGG19 model on Adadelta and SGD Optimizer. In Proceedings of the 2022 6th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 1–3 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1407–1412.
11. Wang, J.; Cao, Z. Chinese text sentiment analysis using LSTM network based on L2 and Nadam. In Proceedings of the 2017 IEEE 17th International Conference on Communication Technology (ICCT), Chengdu, China, 27–30 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1891–1895.
12. Zhang, Q.; Zhang, Y.; Shao, Y.; Liu, M.; Li, J.; Yuan, J.; Wang, R. Boosting Adversarial Attacks with Nadam Optimizer. *Electronics* **2023**, *12*, 1464. [CrossRef]
13. Landro, N.; Gallo, I.; La Grassa, R. Mixing ADAM and SGD: A Combined Optimization Method. *arXiv* **2020**, arXiv:2011.08042.
14. Woodworth, B.; Patel, K.; Stich, S.; Dai, Z.; Bullins, B.; Mcmahan, B.; Shamir, O.; Srebro, N. Is Local SGD Better than Minibatch SGD? In Proceedings of the 37th International Conference on Machine Learning, Online, 13–18 July 2020; PMLR; 2020; pp. 10334–10343.
15. Yi, D.; Ahn, J.; Ji, S. An Effective Optimization Method for Machine Learning Based on ADAM. *Appl. Sci.* **2020**, *10*, 1073. [CrossRef]

16. 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.* **2024**, *21*, 1270–1285. [CrossRef] [PubMed]
17. Chen, X.; Liu, S.; Sun, R.; Hong, M. On the Convergence of A Class of Adam-type Algorithms for Non-Convex Optimization. *arXiv* **2018**, arXiv:1808.02941.
18. Reddi, S.J.; Kale, S.; Kumar, S. On the Convergence of Adam and Beyond. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
19. Jimenez Rezende, D.; Mohamed, S. Variational information maximisation for intrinsically motivated reinforcement learning. In Proceedings of the Advances in Neural Information Processing Systems, NeurIPS, Montreal, QC, Canada, 7–12 December 2015.
20. Lhermitte, E.; Hilal, M.; Furlong, R.; O'Brien, V.; Humeau-Heurtier, A. Deep Learning and Entropy-Based Texture Features for Color Image Classification. *Entropy* **2022**, *24*, 1577. [CrossRef] [PubMed]
21. Chinea Manrique de Lara, A. On the theory of deep learning: A theoretical physics perspective (Part I). *Phys. A Stat. Mech. Its Appl.* **2023**, *632*, 129308. [CrossRef]
22. Shao, Y.; Zhang, C.; Xing, L.; Sun, H.; Zhao, Q.; Zhang, L. A new dust detection method for photovoltaic panel surface based on Pytorch and its economic benefit analysis. *Energy AI* **2024**, *16*, 100349. [CrossRef]
23. Khanday, O.M.; Dadvandipour, S.; Lone, M.A. Effect of filter sizes on image classification in CNN: A case study on CFIR10 and Fashion-MNIST datasets. *IAES Int. J. Artif. Intell. (IJ-AI)* **2021**, *10*, 872. [CrossRef]
24. Sutton, R.T.; Zaïane, O.R.; Goebel, R.; Baumgart, D.C. Artificial intelligence enabled automated diagnosis and grading of ulcerative colitis endoscopy images. *Sci. Rep.* **2022**, *12*, 2748. [CrossRef] [PubMed]
25. 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]