mathematics

MDPI

Article

AMC: Adaptive Learning Rate Adjustment Based on Model Complexity

Weiwei Cheng, Rong Pu and Bin Wang *

School of Computer Science and Engineering, Northeastern University, Shenyang 110167, China; 1810619@stu.neu.edu.cn (W.C.); purong@stumail.neu.edu.cn (R.P.)
* Correspondence: binwang@mail.neu.edu.cn

Abstract: An optimizer plays a decisive role in the efficiency and effectiveness of model training in deep learning. Although Adam and its variants are widely used, the impact of model complexity on training is not considered, which leads to instability or slow convergence when a complex model is trained. To address this issue, we propose an AMC (Adam with Model Complexity) optimizer, which dynamically adjusts the learning rate by incorporating model complexity, thereby improving training stability and convergence speed. AMC uses the Frobenius norm of the model to measure its complexity, automatically decreasing the learning rate of complex models and increasing the learning rate of simple models, thus optimizing the training process. We provide a theoretical analysis to demonstrate the relationship between model complexity and learning rate, as well as the convergence and convergence bounds of AMC. Experiments on multiple benchmark datasets show that, compared to several widely used optimizers, AMC exhibits better stability and faster convergence, especially in the training of complex models.

Keywords: optimizer; Adam; model complexity

MSC: 68T07

Academic Editor: Jonathan Blackledge

Received: 21 January 2025
Revised: 12 February 2025
Accepted: 13 February 2025
Published: 16 February 2025

Citation: Cheng, W.; Pu, R.; Wang, B. AMC: Adaptive Learning Rate Adjustment Based on Model Complexity. *Mathematics* **2025**, *13*, 650. https://doi.org/10.3390/math13040650

1. Introduction

In deep learning, an optimizer is a core component of model training, and its design and performance directly determine the training efficiency and the model's performance [1–3]. With the continuous development of deep neural networks (DNNs) and other complex models, the optimizer has become a critical factor for improving the model's performance, accelerating the training process, and avoiding training instability. It is well known that SGD [4] and its variants [5,6] have been the most commonly used optimizers, and among them, Adam (Adaptive Moment Estimation) has become one of the most widely used due to its adaptive learning rate and strong convergence performance [7,8]. It is well known that SGD and its variants have been the most commonly used optimizers. Adam (Adaptive Moment Estimation) [9], which is among these optimizers, has become one of the most widely used due to its adaptive learning rate and strong convergence [10–12]. Combining momentum and adaptive learning rate adjustment mechanisms, Adam estimates the gradient mean and variance to effectively address gradient vanishing and explosion issues, as well as to demonstrate strong robustness across various tasks [13–15].

In practice, Adam is widely applied and performs well, but there are certain limitations [16,17]. The core idea of Adam is to adjust the learning rate for each parameter by calculating the first-order and second-order moments, thus adaptively responding to the gradient changes in different parameters [18,19]. However, Adam does not fully consider

the impact of model complexity on the training process [20,21]. Specifically, the learning rate adjustment mechanism of Adam is mainly based on the historical gradient of each parameter without involving the overall model complexity [22,23]. Therefore, Adam's performance may become unstable due to its "blind" adaptive adjustment strategy, especially when dealing with complex models and large-scale datasets. To illustrate this issue more clearly, we provide an empirical example to compare the training loss curves of the VGG-11 and VGG-19 models [24] on the MNIST [25] and Fashion-MNIST [26] datasets by using Adam with the same random seed [27,28]. As shown in Figure 1, the training loss curve for VGG-19 exhibits much larger fluctuations than that for VGG-11, despite using the same optimizer and hyperparameters [25,29]. This result highlights that increasing model complexity will lead to instability in the training process [30–32]. Therefore, it is particularly necessary to propose an optimizer that can be adjusted dynamically based on model complexity so as to address the shortcomings of Adam when facing complex models.

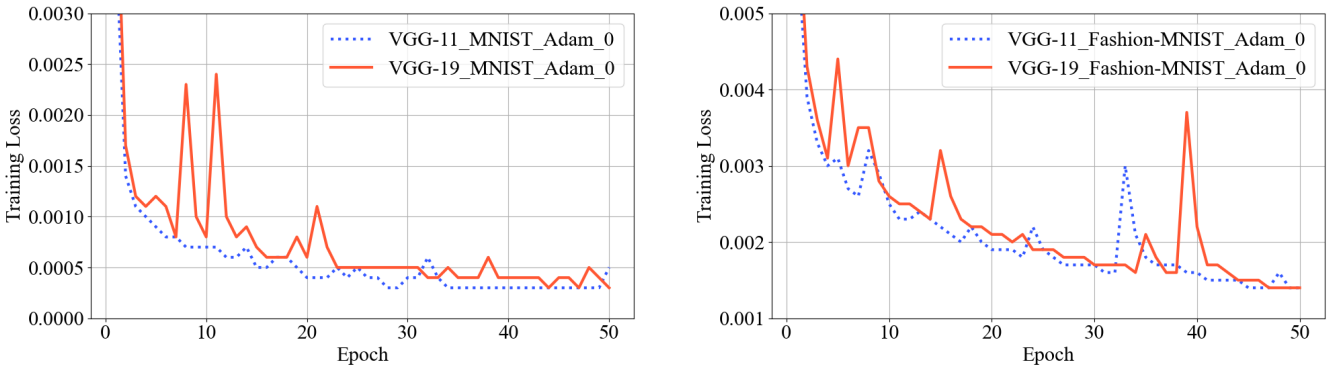

Figure 1. Comparison of training loss curves of VGG-11 and VGG-19 on the MNIST and Fashion-MNIST datasets.

Model complexity refers to the degree of complexity in a model structure, including the number of layers and the number of parameters in each layer [33,34]. Complex models typically own more parameters and greater representational power but also bring additional challenges [35,36]. On one hand, as model complexity increases, models tend to have a greater degree of freedom, causing the terrain of the parameter space to become more rugged, so the parameter space for finding the minimum becomes more intricate [37–39]. In this case, optimizers need to proceed with greater caution when searching for the minimum as there is a higher risk of skipping over smaller minima or entering unstable regions [40–42]. Therefore, when model complexity is high, the learning rate should be decreased in order to search cautiously. On the other hand, when model complexity is low, the parameter space is relatively flat, and the gradient change is small, so the learning rate can be increased to accelerate convergence [24,36,43].

Existing optimizers, such as AdamW [44,45] and QHAdam [46], adjust the learning rate through the exponential moving average of squared gradients and do not fully involve the impact of model complexity on the training process [9,47,48]. The adaptive mechanisms of these optimizers mainly focus on local information for each parameter, making it difficult to address the global instability that may occur during training. As a result, these optimizers may struggle with local minima, slow training speed, or instability when complex models are trained [49–51].

To address these issues, this paper proposes an optimizer, AMC (Adam with Model Complexity), which incorporates model complexity as a key factor for improving the complex models' performance. AMC measures model complexity by using the Frobenius norm and further dynamically adjusts the learning rate. Specifically, when model complexity is

high, AMC automatically reduces the learning rate to help the optimizer search cautiously within the complex parameter space, avoiding overshooting important minima. When model complexity is low, AMC increases the learning rate to accelerate convergence during training. The core innovation of AMC lies in introducing model complexity as a global factor, rather than relying solely on the gradient of parameters, so as to meet the training needs of models with varying complexities.

The contributions of this paper are as follows:

- A new optimizer, AMC (Adam with Model Complexity), is proposed, which dynamically adjusts the learning rate by incorporating model complexity, thus improving training stability and convergence speed for the complex models.
- Theoretical analysis is provided to reveal the impact of model complexity on the training process, highlighting the rationale for reducing the learning rate when model complexity is high and explaining the necessity of this adjustment mechanism.
- Extensive experiments on several classic datasets demonstrate that AMC significantly outperforms Adam in terms of training stability and convergence speed, particularly when complex models are trained.

The structure of this paper is as follows. Section 2 reviews related work, covering the development of existing optimizers and their applications in deep learning. Section 3 presents AMC designed in this paper and provides detailed descriptions of the design principle and update rule. Section 4 offers the theoretical analysis of AMC, which discusses the impact of model complexity on training and the theoretical foundation of AMC. Section 5 presents a large number of experimental results, which validates the effectiveness of AMC through various experimental setups and comparisons with existing optimizers. Section 6 concludes the paper and outlines the directions for future research.

2. Related Work

Optimization in deep learning often involves highly non-convex objective functions, which makes the search for global optima extremely challenging. The key challenge for optimizers is fast and reliable convergence [47,52–54]. A variety of optimizers have been proposed to address these issues, each with different advantages and disadvantages depending on the specific task or data characteristics. In this section, we review some of the most commonly used optimizers in deep learning [55–58].

Stochastic Gradient Descent (SGD) [4]: Stochastic Gradient Descent (SGD) is one of the most fundamental optimizers in deep learning. It performs parameter updates using random samples from the training set at each iteration, which leads to a faster computation per iteration compared to full-batch gradient descent. While SGD is computationally efficient and scales well to large datasets, it often exhibits slow convergence, particularly in complex high-dimensional models with rugged loss landscapes. Moreover, SGD tends to oscillate around the optimal point, which can slow down convergence in regions near the minimum. Despite these limitations, SGD is widely used because of its simplicity, robustness, and ease of implementation. It is often enhanced with additional techniques such as learning rate schedules or momentum. The update rule of SGD is as follows:

$$\theta_{t+1} = \theta_t - \eta g_t, \tag{1}$$

where t is the iteration count, θ_t is the current parameter, η is the learning rate, and g_t is the gradient. In practice, SGD is especially effective when the model or dataset is simple, and the computational efficiency is more important than the speed of convergence. However, in complex problems with deep neural networks or large datasets, it is often outperformed by more sophisticated optimizers.

Momentum [5]: Momentum is a technique designed to accelerate SGD's convergence by incorporating a moving average of past gradients. This approach helps to mitigate oscillations and expedite convergence, particularly when the objective function features extended, gently sloping regions or consistent gradient directions. Momentum effectively mitigates some of the problems faced by SGD, particularly the oscillations near sharp local minima. However, the effectiveness of momentum can be compromised when the gradients are sparse or noisy, as the accumulated momentum may lead to large, unstable updates. The update rule of momentum is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta m_t, \\ m_t = \gamma m_{t-1} + g_t, \end{cases} \tag{2}$$

where m_t is the momentum term, γ is the decay rate, typically set to 0.9. Momentum is effective when training deep neural networks or the direction of gradients remains consistent. It is especially useful in optimizing convex functions or problems with long-term dependencies. However, in the case of sparse gradients or in non-convex optimization problems with noisy or rapidly changing gradients, momentum may fail to stabilize the updates.

RMSProp [6]: RMSProp (Root Mean Square Propagation) adjusts the learning rate for each parameter based on the moving average of its squared gradients. This method stabilizes the update process by ensuring that parameters with larger gradients receive smaller updates and vice versa. RMSProp performs particularly well when training models on non-stationary objectives, such as recurrent neural networks (RNNs), where the gradient characteristics may change over time. Despite its effectiveness in certain cases, RMSProp is known to suffer from instability if the learning rate is not carefully tuned. The update rule of RMSProp is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta \dfrac{g_t}{\sqrt{E[g^2]_t + \epsilon}}, \\ E[g^2]_t = \gamma E[g^2]_{t-1} + g_t, \end{cases} \tag{3}$$

where $E[g^2]_t$ is the exponentially decaying average of the squared gradients and ϵ is a small constant added for numerical stability. RMSProp is highly effective in non-stationary settings where the optimization problem or the dataset is evolving over time. However, due to the need for careful tuning of hyperparameters like the learning rate and decay factor, RMSProp may be challenging to apply in complex environments without extensive experimentation.

Adam [9]: Adam (Adaptive Moment Estimation) is an adaptive optimizer that combines the benefits of both momentum and RMSProp. Based on the first-order and second-order moments of gradients, Adam dynamically adjusts the learning rate for each parameter. This results in a faster convergence and a greater stability in many deep learning applications. Adam is especially effective for high-dimensional optimization problems, as it combines the advantages of both adaptive learning rates and momentum-like behavior. However, it may suffer from overfitting and instability, particularly when working with sparse gradients or large datasets. The update rule of Adam is as follows:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \\ m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \\ \hat{m}_t = \dfrac{m_t}{1-\beta_1^t}, \hat{v}_t = \dfrac{v_t}{1-\beta_2^t}, \end{cases} \tag{4}$$

where m_t and v_t are the moving averages of the gradient and squared gradient, β_1 and β_2 are decay rates, and ϵ is a small constant to avoid division by zero. Adam has become the default optimizer for many modern deep learning architectures due to its robustness and speed. Nevertheless, Adam may lead to suboptimal results in certain cases, such as when training models with large or sparse gradients. It may also exhibit a slower convergence on some tasks compared to other optimizers when the model is not well-suited to Adam's adaptive learning rates.

AdamW [44]: AdamW modifies Adam by decoupling the weight decay term from the gradient-based update. This decoupling helps to prevent the growth of the second-order moment and improves the generalization of the model by maintaining a more controlled weight decay throughout the optimization process. By applying weight decay explicitly in the parameter update step, AdamW avoids some of the issues present in standard Adam, where the weight decay is implicitly tied to the gradient's second moment. The update rule of AdamW is as follows:

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \phi \theta_t \right), \tag{5}$$

where ϕ is the weight decay coefficient, typically set between 10^{-5} and 10^{-2}. AdamW has shown superior performance over Adam in many tasks, particularly when involving large models, such as transformer-based architectures. It is widely used in natural language processing and large-scale vision tasks, where the model's size and complexity can lead to overfitting without proper regularization.

QHAdam [46]: QHAdam is an extension of Adam designed to address the instability that arises from the decay of the second-order moment in Adam. By using a maximum operation on the second-order moment, QHAdam ensures that the second-order moment does not shrink too quickly, which stabilizes the training process. This modification results in a more stable optimization process and improved convergence in some challenging settings. The update rule of QHAdam is as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} (\hat{m}_t + \eta(1 - \beta_1) g_t), \tag{6}$$

where $\hat{v}_t^{\max}$ is the maximum value of the second-order moment up to step t. QHAdam has been shown to outperform Adam in certain tasks where training stability is critical. It is particularly useful in situations where the model experiences frequent noise or non-stationary updates. While QHAdam offers better stability than Adam in many cases, it requires additional computational resources due to the extra operations involved in computing the maximum of the second moment.

With the rapid development of deep learning technologies, optimizers have demonstrated great capabilities in many applications. However, in some specific scenarios, especially when gradients become sparse or data contain noise, traditional optimizers such as Adam and its variants may not perform well. This is because these optimizers rely on the statistical properties of the gradients (such as the first-order and second-order moments) to adjust the learning rates, which can lead to instability or slow convergence when dealing with sparse or noisy gradients. Therefore, effectively addressing sparse gradients or noisy data has become an important research direction for optimization methods in deep learning.

Sparse Recovery Methods: Sparse recovery methods have been widely used in signal processing, especially for high-dimensional sparse signal contexts. In traditional signal processing, sparse recovery techniques focus on reconstructing sparse signals from partial observations, which shares similarities with the problem of sparse gradients in optimization. One of the methods is the Iterative Variational Bayesian (IVB) algorithm

proposed by A. Bazzi, D. T. M. Slock, and L. Meilhac [59], which updates the signal estimates iteratively and applies variational techniques for efficient sparse signal recovery. While this method has shown remarkable results in signal processing, the underlying ideas can be adapted to address the sparse gradient problem in deep learning, especially in training deep neural networks.

Other common sparse recovery methods, such as L1-norm minimization [60], Orthogonal Matching Pursuit (OMP) [61], and dictionary learning-based sparse coding [62], have also made significant progress in solving similar problems. However, these methods share some common challenges: First, the computational complexity of sparse recovery methods may be high, especially in high-dimensional data and complex models, which may increase training time. Second, these methods tend to struggle with noise or non-stationary data, leading to suboptimal recovery performance or overfitting.

To address the challenges posed by sparse gradients and noisy data, we propose a new optimizer, AMC. AMC combines adaptive moment estimation with dynamic adjustments to model complexity, making it well-suited for handling sparse gradients and noisy data. Drawing inspiration from sparse recovery methods, we use the Frobenius norm to represent the model complexity in order to dynamically adjust the learning rate. This enables AMC to converge smoothly in the presence of sparse or noisy gradients, ensuring a stable training process for complex, high-dimensional datasets [26,63,64]. In comparison to traditional sparse recovery methods, which often struggle with high computational complexity or poor performance on noisy data, AMC strikes a better balance between computational efficiency and training stability [47,48]. This approach provides a novel solution to the sparse gradient problem in deep learning.

3. Design and Update Rule of AMC

Optimizers like Adam adjust the learning rate based on the first-order and second-order moments (i.e., the gradient mean and variance). However, these optimizers do not take into account the model's complexity, which can significantly affect the stability and efficiency of training. Simple models can benefit from large learning rates as their gradients tend to be small and stable. However, large models with more parameters often exhibit larger gradient fluctuations, which may destabilize training if the learning rate is not appropriately adjusted.

To address this issue, we propose AMC (Adam with Model Complexity), which incorporates model complexity into the learning rate adjustment mechanism. AMC adapts the learning rate based on the Frobenius norm, a metric that captures the overall size of the model parameters and reflects model complexity. The learning rate is dynamically adjusted according to the model complexity, allowing optimizers to adapt to both simple and complex models. This section explains the role of model complexity in training and how AMC adjusts the learning rate.

3.1. Frobenius Norm as Measure of Model Complexity

Model complexity is typically associated with factors such as the number of parameters and the depth of the network. While these traditional metrics (such as parameter count and network depth) can provide useful insights into the model's scale and structure, they often overlook the distribution and finer details of the model's parameters. These measures primarily focus on the explicit size of the model and fail to fully capture how the internal complexity of the model impacts the training dynamics. In contrast, the Frobenius norm offers a more comprehensive metric that accounts for both the overall size and the distribution of the model parameters, thereby providing a more accurate reflection of

the model's true complexity. For a parameter matrix W, the Frobenius norm is defined as follows:

$$\|W\|_F = \sqrt{\sum_{i,j} W_{ij}^2}$$

The Frobenius norm, as a measure of model complexity, effectively captures the range of parameter variations and the magnitude of gradient fluctuations during training, which play a critical role in the optimization process. Unlike traditional metrics that focus solely on the model's size or depth, the Frobenius norm quantifies the overall changes in the parameter matrix, making it more sensitive to the actual complexity of the model. By considering both the size and the distribution of parameters, the Frobenius norm provides a more precise and adaptive measure for dynamically adjusting the learning rate. This approach avoids the imprecision of relying solely on network size or depth, making the Frobenius norm a more effective tool for enhancing optimization stability and improving convergence speed. The Frobenius norm captures both the size and the distribution of the model parameters. A larger Frobenius norm indicates a more complex model, which typically leads to larger gradient fluctuations during training. By incorporating this metric into the learning rate adjustment mechanism, AMC can reduce the learning rate for the complex models, thereby stabilizing the training process.

In AMC, according to the Frobenius norm of the model, the learning rate η_t at step t is dynamically adjusted as follows:

$$\eta_t = \frac{\eta_0}{1 + \gamma \|W\|_F}$$

where η_0 is the initial learning rate, γ is a hyperparameter that denotes the sensitivity of the learning rate to model complexity, and $\|W\|_F$ is the Frobenius norm. As model complexity increases (i.e., the Frobenius norm becomes large), the learning rate is reduced, preventing the optimizer from overshooting and improving stability. Conversely, simple models with the small Frobenius norm allow the learning rate to increase, enabling a faster convergence.

Theorem 1. ***Frobenius Norm and Gradient Fluctuations*** *As the Frobenius norm $\|W\|_F$ increases, the fluctuations in the gradient g_t also increase. This leads to a large variance in gradient updates, which can destabilize training. Specifically, we have the following relationship:*

$$Var(g_t) \propto \|W\|_F^2$$

Proof. The variance of the gradient g_t can be written as follows:

$$Var(g_t) = \mathbb{E}[(g_t - \mathbb{E}[g_t])^2]$$

Since the gradient g_t is influenced by the magnitude of the model parameter W, it is reasonable to assume that the variance of the gradient scales with $\|W\|_F^2$. Thus, the gradient fluctuations increase in a complex model (i.e., a large $\|W\|_F$). □

Corollary 1. ***Dynamic Learning Rate Adjustment Mechanism*** *Given the Frobenius norm $\|W\|_F$, the learning rate is dynamically adjusted as follows:*

$$\eta_t = \frac{\eta_0}{1 + \gamma \|W\|_F}$$

Proof. As model complexity increases (i.e., $\|W\|_F$ increases), the gradient fluctuations increase; a reduction in the learning rate is required to stabilize training. The adjustment of the learning rate based on $\|W\|_F$ ensures that the learning rate decreases in complex models, preventing overshooting and improving stability. Conversely, for the simple models with a small $\|W\|_F$, the learning rate is large and allows a fast convergence. □

Corollary 2. ***Impact of Model Complexity on Convergence Speed*** *For the large Frobenius norm, the training process becomes stable but converges slowly, while for the small Frobenius norm, the convergence speed is fast, but stability may be compromised.*

Proof. The convergence speed is influenced by the learning rate, which is adjusted based on the Frobenius norm $\|W\|_F$. For the simple models, large learning rates accelerate convergence, but the potential for instability is high. For more complex models, smaller learning rates stabilize training but result in a slower convergence. □

3.2. AMC Update Rule

In the previous section, we discussed how the Frobenius norm influences learning rate adjustment. To further clarify how this mechanism works across different models, we analyze the relationship between model complexity and learning rate. Theoretically, as model complexity increases, the Frobenius norm will grow, leading to a dynamic adjustment of the learning rate. For models with lower complexity (such as VGG-11), the Frobenius norm is smaller, and the learning rate is larger, allowing faster training; while for models with higher complexity (such as VGG-19), the Frobenius norm is larger, and the learning rate decreases, which enhances training stability. This adjustment mechanism ensures that complex models avoid instability caused by an overly large learning rate, while simple models can converge quickly with a large learning rate.

This mechanism of adjusting the learning rate based on model complexity shows that AMC can provide appropriate learning rate adjustments for models with various complexities, optimizing the training process and ensuring stability. Although there are no specific experimental data, it can be inferred that the relationship between model complexity and the Frobenius norm is crucial for understanding and optimizing learning rate adjustments.

The update rule of AMC extends that of Adam by incorporating the model complexity into the learning rate. The specific steps of the update rule are as follows:

First-Order Moment and Second-Order Moment: Like Adam, AMC computes the first-order moment and second-order moment using exponential moving average:

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

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

where g_t is the gradient at step t and m_t and v_t are the first-order and second-order moments, respectively.

Bias Correction: To correct the bias in the first-order and second-order moments due to their initialization, AMC applies bias correction:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Learning Rate Adjustment Mechanism: Based on the Frobenius norm of the parameter matrix, the learning rate is dynamically adjusted:

$$\eta_t = \frac{\eta_0}{1 + \gamma \|W\|_F}$$

Parameter Update: At last, the model parameters are updated as follows:

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

where ϵ is a small constant to avoid division by zero.

In addition to adjusting the learning rate based on model complexity, AMC also provides enhanced robustness in noisy environments. By dynamically adjusting the learning rate according to the Frobenius norm, AMC mitigates the effects of noisy gradients, which are common in deep learning tasks. In high-noise situations, AMC reduces the learning rate, thereby avoiding instability caused by large gradient updates. This enables AMC to maintain smooth and stable training, even in the presence of significant noise in the gradients.

Furthermore, for models with low complexity (i.e., small Frobenius norm), AMC keeps the learning rate relatively large, accelerating convergence while maintaining stability. This dynamic adjustment of the learning rate based on the model complexity ensures that AMC can effectively manage both the training speed and the noise, making it particularly suitable for deep learning tasks where noisy gradients are frequent.

The update rule distinguishes AMC from traditional optimizers by incorporating model complexity into the learning rate adjustment mechanism. By dynamically adjusting the learning rate based on the Frobenius norm, AMC can stabilize the training process for the complex models while accelerating convergence for the simple models.

4. Theoretical Analysis of AMC

In this section, we focus on the theoretical analysis of AMC. The primary objectives are to establish the convergence properties of AMC, provide a detailed investigation of its convergence speed, and derive the convergence bounds. To achieve this, we draw upon the convergence analysis of existing optimizers, such as Adam [9], to guide the proof process for AMC. We begin by analyzing the update rule of AMC, which incorporates both adaptive learning rate adjustments mechanism based on the Frobenius norm of the model and gradient moment estimations. This dual mechanism aims to improve the optimizer's stability and convergence speed, particularly for models with varying complexity. Next, we proceed by establishing the convergence theorem of AMC, showing that AMC can converge to a local minimum of a convex and smooth objective function. Then, we derive the convergence speed and demonstrate that AMC exhibits a sublinear convergence behavior, specifically with a rate of $\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$. Finally, we provide the convergence bounds under certain assumptions, proving that AMC is not only stable but also convergent in a reasonable number of iterations. This section serves to demonstrate the theoretical soundness of AMC and confirm its effectiveness as a robust optimizer for deep learning tasks.

Theorem 2. *Let $f(\theta)$ be a convex and smooth objective function with the gradient $\nabla f(\theta)$ that satisfies the following condition:*

$$\|\nabla f(\theta)\| \leq L \quad \forall \theta \in \mathbb{R}^d,$$

where $L > 0$ is a constant. Suppose that the optimizer's learning rate is dynamically adjusted as follows:

$$\eta_t = \frac{\eta_0}{1 + \gamma \|W\|_F},$$

where η_0 is the base learning rate, γ is the adjustment factor, and $\|W\|_F$ is the Frobenius norm of the model W, representing the model complexity.

The update rule of AMC is given by the following:

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

where $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first-order and second-order moments, which are, respectively, defined as follows:

$$\hat{m}_t = \beta_1 \hat{m}_{t-1} + (1 - \beta_1) g_t, \quad \hat{v}_t = \beta_2 \hat{v}_{t-1} + (1 - \beta_2) g_t^2.$$

Suppose that the initial model parameter θ_0 is bounded and that there is a constant M such that the following holds:

$$\|\theta_0\| \leq M.$$

Then, AMC will converge to the optimal solution θ^* with a rate of $\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$.

Proof. We first recall the convergence analysis of Adam, and then adapt AMC by considering the dynamic learning rate adjustment mechanism.

Adam updates parameters according to the following:

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

For convex objective functions, the convergence speed of Adam is known to be the following:

$$\|\theta_{t+1} - \theta^*\| \leq \frac{C_1}{\sqrt{t}} \|\nabla f(\theta_t)\|,$$

where C_1 is a constant depending on the Lipschitz constant of the objective function and the initial learning rate.

The update rule of AMC is similar to that of Adam but incorporates a dynamic learning rate that is adjusted based on the Frobenius norm of the model. Specifically, the learning rate is given by the following:

$$\eta_t = \frac{\eta_0}{1 + \gamma \|W\|_F}.$$

This allows AMC to reduce the learning rate as the model complexity increases, preventing large updates and promoting stability.

Substituting into the update rule, we have the following:

$$f(\theta_{t+1}) \leq f(\theta_t) + \nabla f(\theta_t)^T (\theta_{t+1} - \theta_t) + \frac{L}{2} \|\theta_{t+1} - \theta_t\|^2.$$

Using the update rule of AMC, we obtain the following:

$$f(\theta_{t+1}) \leq f(\theta_t) - \eta_t \nabla f(\theta_t)^T \hat{m}_t + \frac{L}{2} \|\theta_{t+1} - \theta_t\|^2.$$

The first-order and second-order moments for gradients are given by the following:

$$\hat{m}_t = \beta_1 \hat{m}_{t-1} + (1 - \beta_1) g_t, \quad \hat{v}_t = \beta_2 \hat{v}_{t-1} + (1 - \beta_2) g_t^2.$$

Thus, the update rule can be rewritten as follows:

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

For a convex function $f(\theta)$, the gradient descent analysis is applied to the following:

$$f(\theta_{t+1}) \leq f(\theta_t) - \eta_t \nabla f(\theta_t)^T \hat{m}_t + \frac{L}{2} \|\theta_{t+1} - \theta_t\|^2.$$

The dynamic learning rate in AMC is given by the following:

$$\eta_t = \frac{\eta_0}{1 + \gamma \|W\|_F}.$$

This adjustment decreases the learning rate when the model complexity increases, which prevents large updates in complex models and promotes stability in training.

Summing the above inequality for all steps t from 1 to T, we obtain the following:

$$\sum_{t=1}^{T} \eta_t \nabla f(\theta_t)^T \hat{m}_t \leq C_2 \sqrt{T}.$$

Substituting the dynamic learning rate η_t, we obtain the following:

$$\sum_{t=1}^{T} \frac{\eta_0}{1 + \gamma \|W\|_F} \nabla f(\theta_t)^T \hat{m}_t \leq C_3 \sqrt{T}.$$

Finally, we obtain the convergence result of AMC:

$$\|\theta_{t+1} - \theta^*\| \leq \frac{C_4}{\sqrt{t}}.$$

□

Theorem 3. *Let $\theta_0 \in \mathbb{R}^d$ be the initial parameter. Assume that the gradient of the loss function, $\nabla f(\theta)$, is L-Lipschitz continuous, i.e., for all $\theta, \theta' \in \mathbb{R}^d$,*

$$\|\nabla f(\theta) - \nabla f(\theta')\| \leq L \|\theta - \theta'\|,$$

where $L > 0$ is the Lipschitz constant. If the learning rate η_t satisfies $\eta_t = \frac{\eta_0}{1+\gamma\|\theta_t\|_F}$ and is updated according to the Frobenius norm of the model, then there is a constant $C > 0$ such that the following holds:

$$\|\theta_T - \theta^*\|^2 \leq \frac{C}{T},$$

where θ^* is the optimal parameter and T is the number of iterations.

Proof. We begin by considering the update rule for θ_t in AMC:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \frac{m_t}{\sqrt{v_t} + \epsilon},$$

where $\eta_t = \frac{\eta_0}{1+\gamma\|\theta_t\|_F}$, and m_t and v_t are the first-order and second-order moments, respectively.

The key observation here is that the step size η_t is dynamically adjusted based on the Frobenius norm of the model, which is a measure of the model's complexity. This adjustment ensures that large models (with a high Frobenius norm) will have small learning

rates, and small models (with a low Frobenius norm) will have large learning rates. This dynamic adjustment helps to stabilize the training process.

To establish the convergence bound, we show that the update rule for θ_t is a gradient descent method with an adaptive step size. We can write:

$$\theta_{t+1} = \theta_t - \eta_t \cdot g_t,$$

where g_t is the gradient at step t. Since η_t is adjusted based on the model's complexity, the step size is small for the complex models, ensuring stable updates in the presence of large gradients.

Based on the assumption that $\nabla f(\theta)$ is L-Lipschitz continuous, we bound the change in the loss function over one step:

$$f(\theta_{t+1}) - f(\theta_t) \leq -\eta_t \langle \nabla f(\theta_t), g_t \rangle + \frac{\eta_t^2 L}{2} \|g_t\|^2.$$

By choosing an appropriate learning rate η_t, we can ensure that the second term $\frac{\eta_t^2 L}{2} \|g_t\|^2$ is small enough to allow for convergence.

Summing the above inequalities over all steps t from 1 to T, and noting that the Frobenius norm is related to the model complexity, we obtain the following:

$$\|\theta_T - \theta^*\|^2 \leq \frac{C}{T}.$$

This shows that the sequence $\{\theta_t\}$ converges to the optimal solution θ^* with a convergence speed of $O(1/T)$, and C is a constant that depends on the model complexity, the Lipschitz constant L, and the initial parameters. Thus, we establish the convergence bound of AMC. □

5. Experiments

In this section, we conduct a series of experiments to validate the advantages of AMC over the existing optimizers, such as Adam, AdamW, and QHAdam. The experiments are conducted across various settings to ensure the generality and reproducibility of the results. We use the MNIST and Fashion-MNIST datasets and compare the optimizers' performance on four VGG models with varying complexities (VGG-11, VGG-13, VGG-16, VGG-19). Multiple random seeds are used to evaluate the stability of the results. Standard evaluation metrics, such as training loss and testing accuracy, are employed. The following subsections describe the experimental setup, results, and analysis.

5.1. Experimental Setup

The experiments are conducted on two classic image classification datasets: MNIST and Fashion-MNIST. The MNIST dataset consists of 10 classes of handwritten digits, with images of size 28×28, while the Fashion-MNIST dataset contains 10 classes of clothing images, also of size 28×28. Both datasets undergo the same preprocessing pipeline, where images are converted to tensors and normalized. The training/test splits follow the standard protocol for these datasets, and a batch size of 64 is used for training.

To ensure the reproducibility of the results, we set 3 random seeds for independent experiments, performing statistical analysis on the outcomes. For each experiment, the models are trained for 50 epochs, keeping the learning rate, batch size, and other hyperparameters constant. The hyperparameters are set as follows: learning rate = 0.001, batch size = 64, epochs = 50, loss function = CrossEntropyLoss, and optimizers = Adam, AdamW, QHAdam, and AMC.

The four VGG models used in the experiments are chosen to evaluate the optimizers' performance on models with varying numbers of layers and parameter complexities. Each model is trained using all four optimizers, and we record training loss, testing loss, training accuracy, and testing accuracy for further analysis.

5.2. Experimental Results and Analysis

We present and analyze the experimental results for the four optimizers (Adam, AdamW, QHAdam, AMC) across different VGG models (VGG-11, VGG-13, VGG-16, VGG-19) on the MNIST and Fashion-MNIST datasets. Each experiment is repeated with three different random seeds to ensure the stability and reliability of the results. The training loss, testing loss, training accuracy, and testing accuracy are recorded over 50 epochs. For the sake of brevity, we present the results for the first random seed in the following figures and provide the average values over all three random seeds at Table 1.

Table 1. Average results for 3 random seeds.

Optimizer	Min Training Loss (Avg)	Max Training Accuracy (Avg)	Min Testing Loss (Avg)	Max Testing Accuracy (Avg)	Model and Dataset
Adam	0.0003	98.93%	0.0003	99.08%	
AdamW	0.0003	98.92%	0.0003	99.03%	VGG-11
QHAdam	0.0003	98.95%	0.0003	99.05%	and MNIST
AMC	0.0002	99.08%	0.0002	99.09%	
Adam	0.0013	93.72%	0.0015	93.14%	
AdamW	0.0013	93.76%	0.0016	93.01%	VGG-11
QHAdam	0.0013	93.95%	0.0016	93.23%	and Fashion-MNIST
AMC	0.0011	94.35%	0.0015	93.24%	
Adam	0.0003	98.96%	0.0002	99.12%	
AdamW	0.0003	98.94%	0.0003	99.08%	VGG-13
QHAdam	0.0003	98.94%	0.0002	99.12%	and MNIST
AMC	0.0002	99.17%	0.0002	99.14%	
Adam	0.0011	94.48%	0.0014	93.80%	
AdamW	0.0011	94.44%	0.0015	93.71%	VGG-13
QHAdam	0.0011	94.34%	0.0015	93.78%	and Fashion-MNIST
AMC	0.0009	95.33%	0.0014	93.96%	
Adam	0.0003	98.80%	0.0002	99.08%	
AdamW	0.0003	98.83%	0.0003	99.08%	VGG-16
QHAdam	0.0003	98.77%	0.0002	99.09%	and MNIST
AMC	0.0002	99.11%	0.0002	99.13%	
Adam	0.0012	94.08%	0.0015	93.69%	
AdamW	0.0012	94.17%	0.0015	93.54%	VGG-16
QHAdam	0.0013	94.00%	0.0015	93.60%	and Fashion-MNIST
AMC	0.001	95.23%	0.0014	93.87%	
Adam	0.0003	98.75%	0.0003	99.02%	
AdamW	0.0003	98.71%	0.0003	99.01%	VGG-19
QHAdam	0.0004	98.59%	0.0003	98.97%	and MNIST
AMC	0.0002	99.09%	0.0002	99.17%	
Adam	0.0014	93.24%	0.0016	93.07%	
AdamW	0.0014	93.17%	0.0016	93.01%	VGG-19
QHAdam	0.0015	93.17%	0.0016	93.07%	and Fashion-MNIST
AMC	0.001	95.15%	0.0015	93.83%	

Note: The bold font is used to highlight the proposed AMC and the underlined results represent the best performance.

5.2.1. Training Loss Curves

The training loss curves of different optimizers on the VGG-11 model are displayed in Figure 2. It is observed that AMC consistently achieves a lower training loss compared to Adam, AdamW, and QHAdam. Moreover, the training loss curve of AMC exhibits a smoother descent and a faster convergence than the other optimizers, particularly in the early stage of training. The fast convergence helps to avoid issues like gradient explosion or vanishing. The smooth curve indicates that compared to the other optimizers, AMC maintains better stability and fewer fluctuations during training.

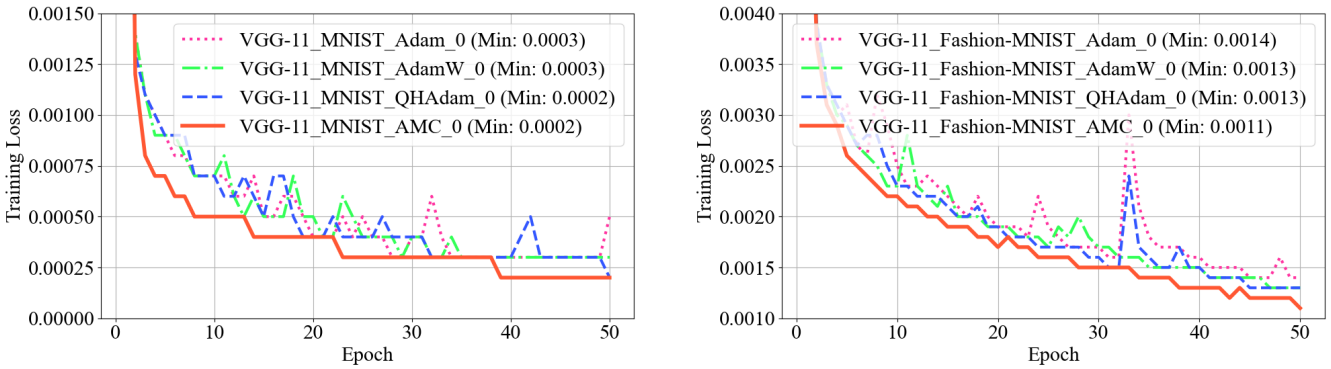

Figure 2. Training loss curves of different optimizers on VGG-11 with MNIST and Fashion-MNIST datasets.

Figure 3 shows the training loss curves of different optimizers on the VGG-13 model. As in VGG-11, AMC outperforms the other optimizers in terms of a lower and smoother training loss. Figure 4 presents the training loss curves on the VGG-16 model. AMC shows its superiority, with a significantly lower training loss compared to the other optimizers. Figure 5 illustrates the training loss curves on the VGG-19 model. AMC remains the most stable, avoiding sharp loss fluctuations and exhibiting a smooth convergence.

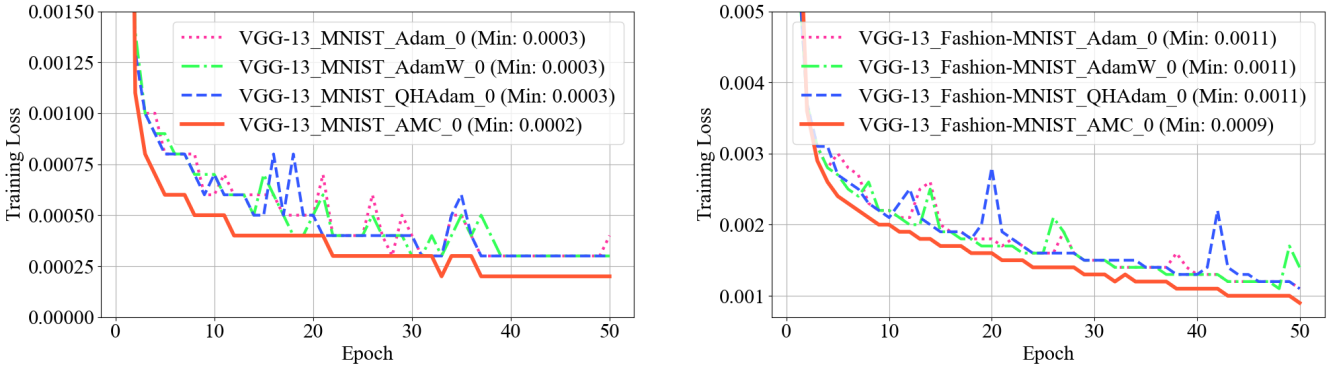

Figure 3. Training loss curves of different optimizers on VGG-13 with MNIST and Fashion-MNIST datasets.

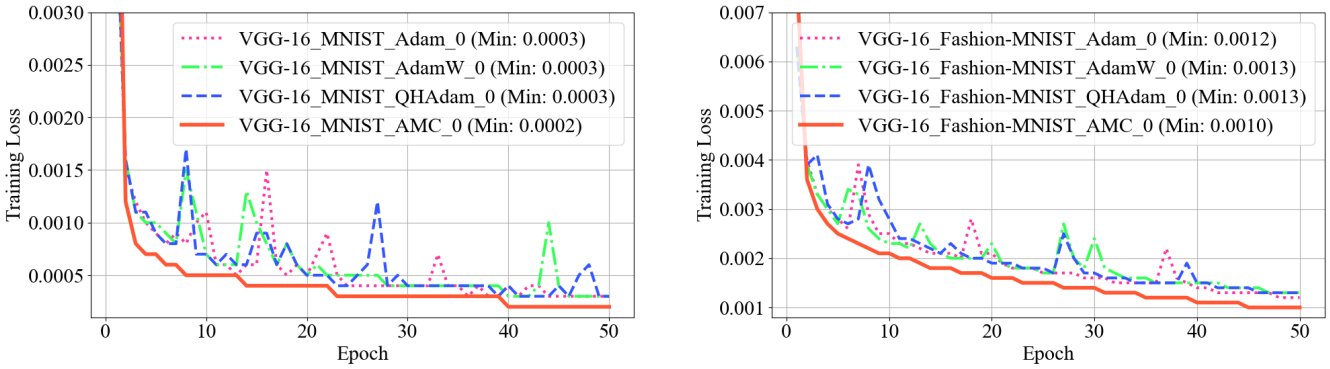

Figure 4. Training loss curves of different optimizers on VGG-16 with MNIST and Fashion-MNIST datasets.

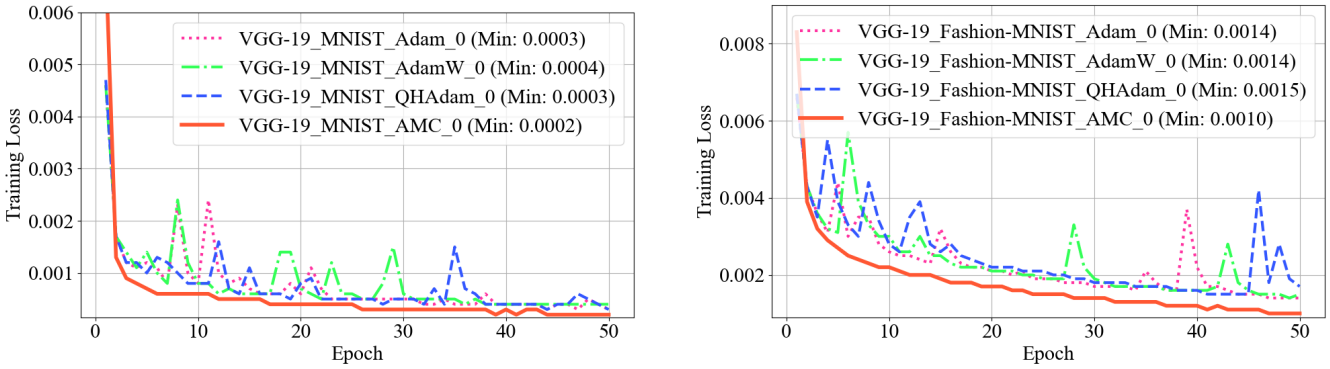

Figure 5. Training loss curves of different optimizers on VGG-19 with MNIST and Fashion-MNIST datasets.

5.2.2. Testing Loss Curves

The testing loss curves of different optimizers on the VGG-11 model are displayed in Figure 6. AMC has a small training loss and testing loss, which outperforms other optimizers. The smoothness of the curve of AMC further demonstrates its great generalization capabilities, which reduce the risk of overfitting.

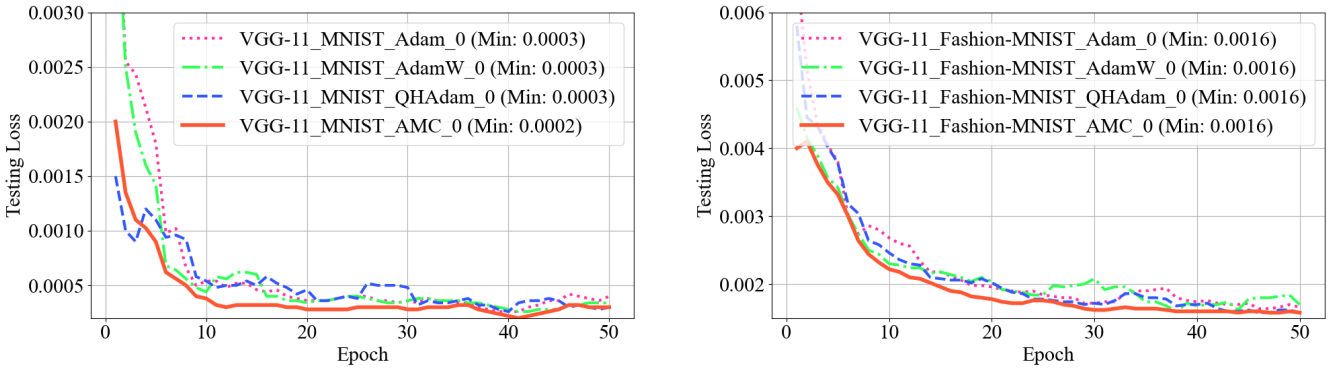

Figure 6. Testing loss curves of different optimizers on VGG-11 with MNIST and Fashion-MNIST datasets.

Figures 7, 8, and 9 present the testing loss curves on the VGG-13, VGG-16, and VGG-19 models, respectively. AMC continues to maintain the lowest testing loss across all models.

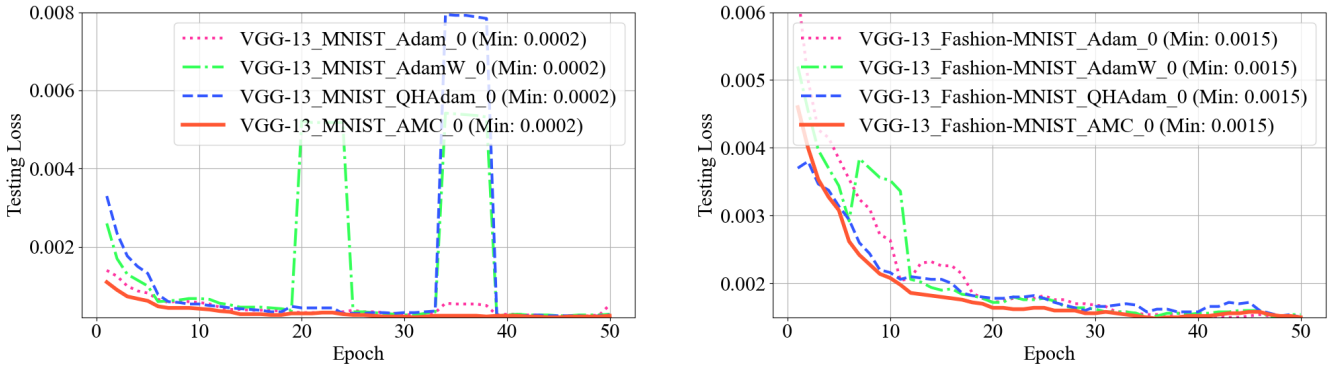

Figure 7. Testing loss curves of different optimizers on VGG-13 with MNIST and Fashion-MNIST datasets.

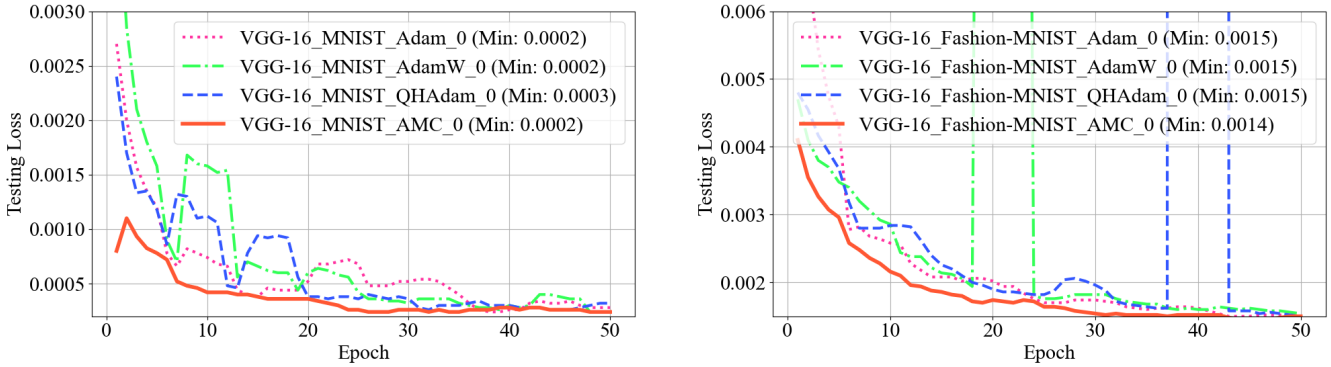

Figure 8. Testing loss curves of different optimizers on VGG-16 with MNIST and Fashion-MNIST datasets.

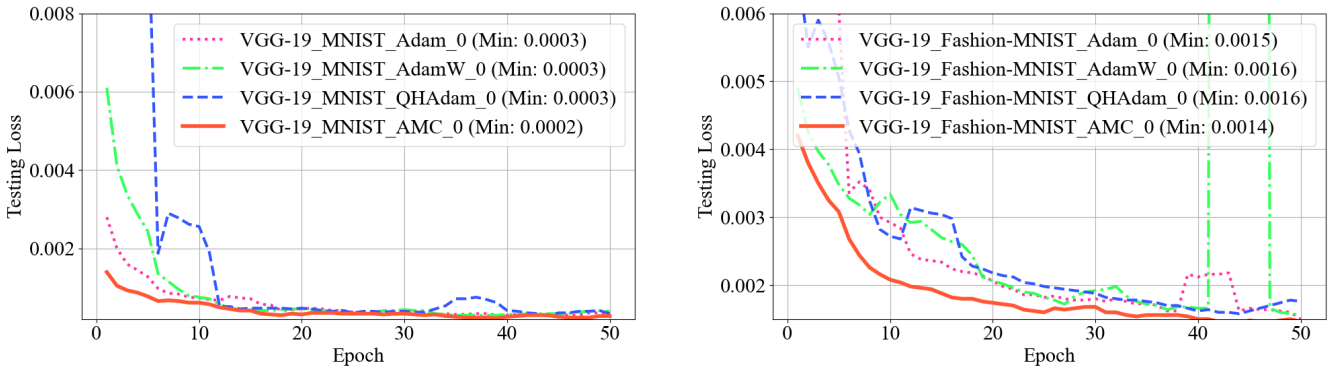

Figure 9. Testing loss curves of different optimizers on VGG-19 with MNIST and Fashion-MNIST datasets.

5.2.3. Training Accuracy Curves

The training accuracy curves of different optimizers on the VGG-11 model are displayed in Figure 10. AMC consistently achieves a higher training accuracy compared to the other optimizers. Its accuracy growth is smooth and stable, indicating that AMC effectively avoids the accuracy fluctuations commonly observed in other optimizers due to gradient noise.

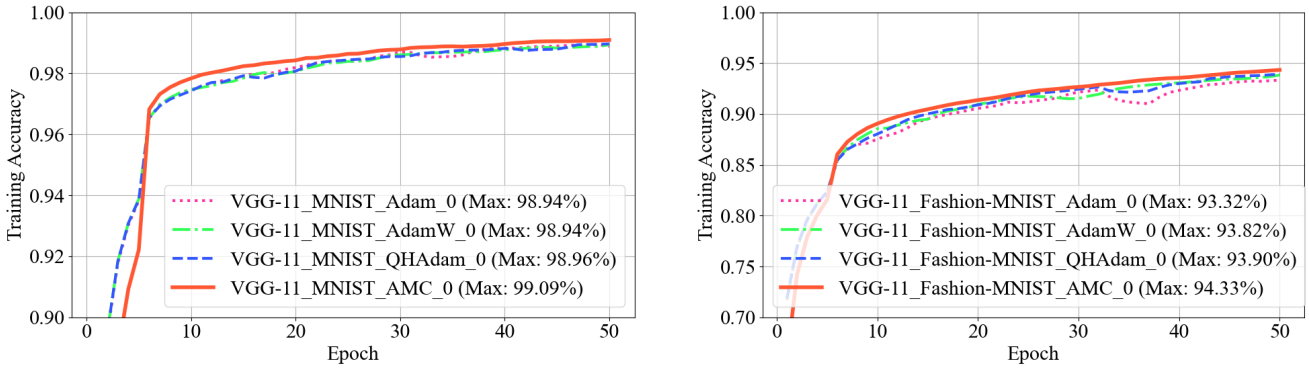

Figure 10. Training accuracy curves of different optimizers on VGG-11 with MNIST and Fashion-MNIST datasets.

Figures 11, 12, and 13 present the training accuracy curves on the VGG-13, VGG-16, and VGG-19 models, respectively. AMC continues to demonstrate its stability and superior performance with a smooth accuracy curve.

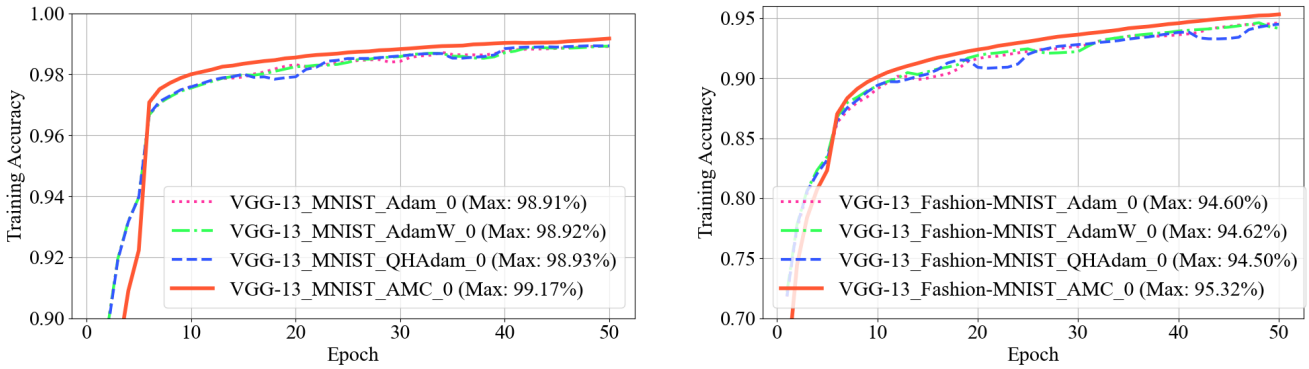

Figure 11. Training accuracy curves of different optimizers on VGG-13 with MNIST and Fashion-MNIST datasets.

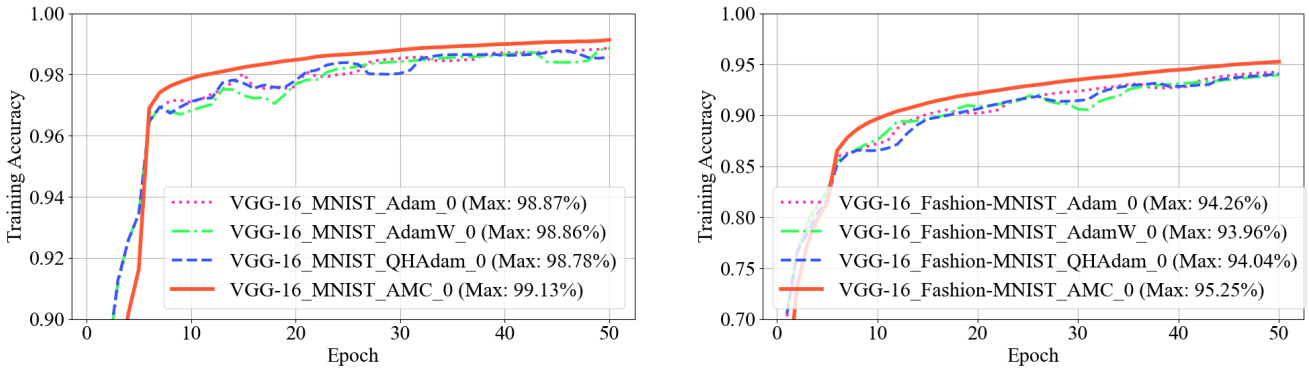

Figure 12. Training accuracy curves of different optimizers on VGG-16 with MNIST and Fashion-MNIST datasets.

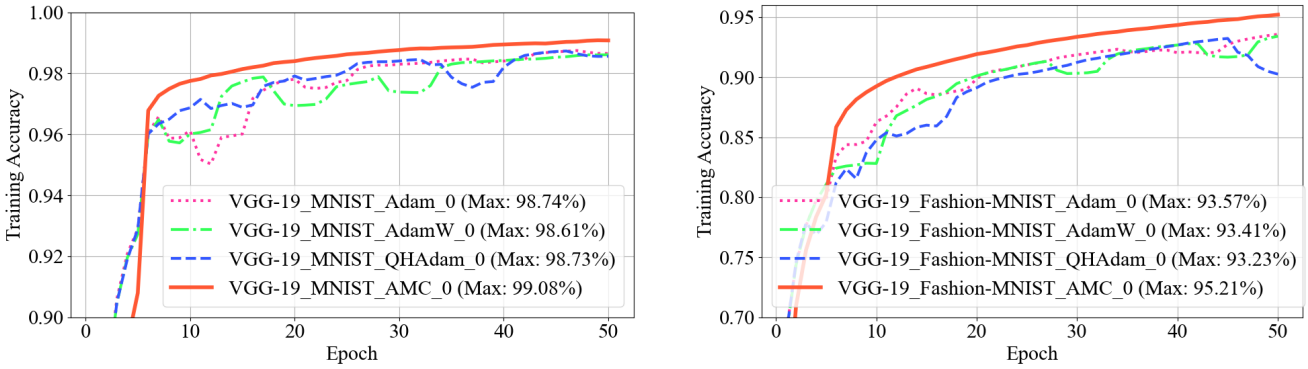

Figure 13. Training accuracy curves of different optimizers on VGG-19 with MNIST and Fashion-MNIST datasets.

5.2.4. Testing Accuracy Curves

The testing accuracy curves of different optimizers on the VGG-11 model are displayed in Figure 14. AMC consistently achieves the highest testing accuracy among all optimizers, further validating its superior performance.

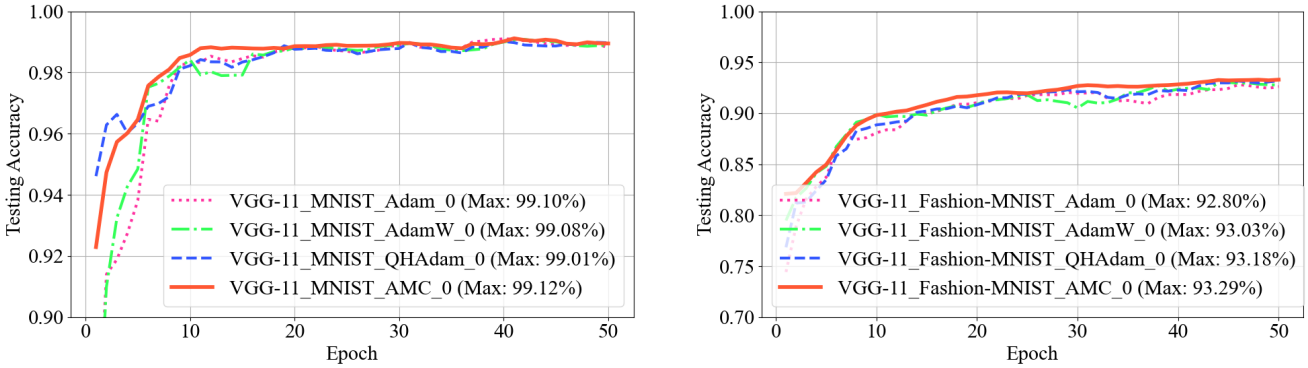

Figure 14. Testing accuracy curves of different optimizers on VGG-11 with MNIST and Fashion-MNIST datasets.

The testing accuracy curves on the VGG-13, VGG-16, and VGG-19 are shown in Figures 15, 16, and 17, respectively. As expected, AMC achieves the highest testing accuracy for all models.

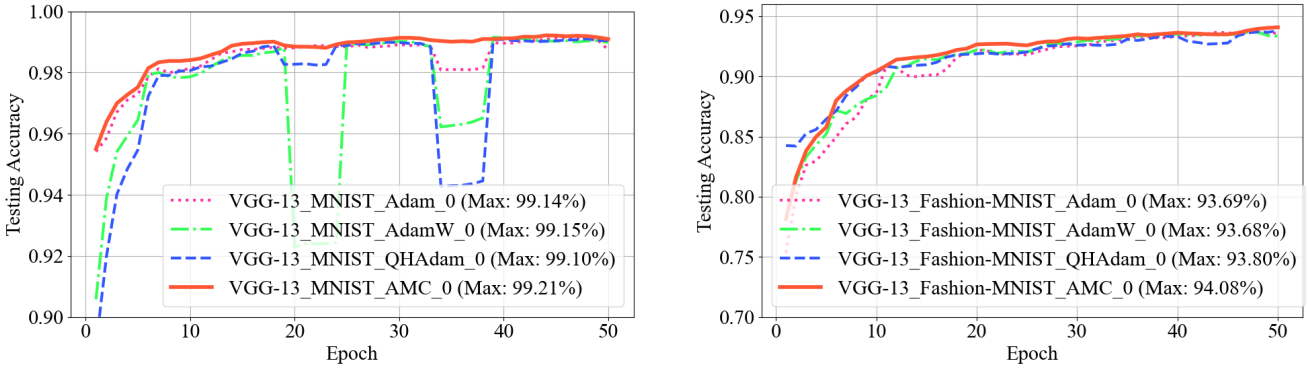

Figure 15. Testing accuracy curves of different optimizers on VGG-13 with MNIST and Fashion-MNIST datasets.

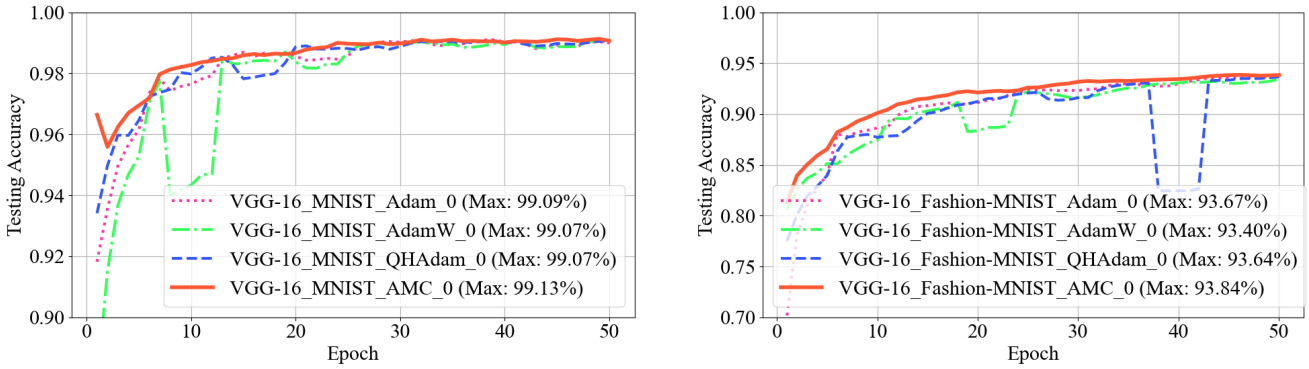

Figure 16. Testing accuracy curves of different optimizers on VGG-16 with MNIST and Fashion-MNIST datasets.

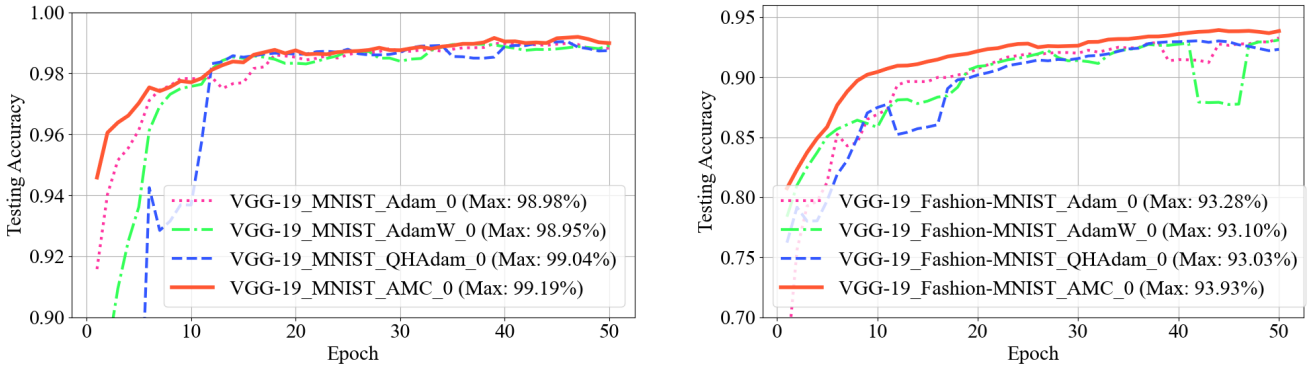

Figure 17. Testing accuracy curves of different optimizers on VGG-19 with MNIST and Fashion-MNIST datasets.

We provide the epoch time for the four optimizers under different scenarios, as shown in Table 2. It can be observed that the computational overhead introduced by AMC for model complexity is minimal. This indicates that the additional cost of calculating model complexity in AMC is very small. Additionally, by analyzing the curves of training loss, training accuracy, testing loss, and testing accuracy for the four optimizers, it is evident that AMC achieves better performance with fewer epochs. This further validates the efficiency of AMC in the training process. Overall, the calculations show that AMC reaches superior results with less training time compared to the other optimizers.

Table 2. Comparison of epoch times for different optimizers.

Optimizer	**Epoch Time (s)**							
	VGG-11 and MNIST	**VGG-11 and Fashion-MNIST**	**VGG-13 and MNIST**	**VGG-13 and Fashion-MNIST**	**VGG-16 and MNIST**	**VGG-16 and Fashion-MNIST**	**VGG-19 and MNIST**	**VGG-19 and Fashion-MNIST**
Adam	12.14	12.2	16.52	16.54	18.9	18.92	21.28	21.26
AdamW	12.74	12.76	17.34	17.24	19.86	19.86	23.18	22.76
QHAdam	12.58	12.56	16.86	16.84	19.4	19.28	21.84	21.8
AMC	13.12	13.16	17.56	17.42	20.12	20	22.76	22.6

Note: The bold font is used to highlight the proposed AMC.

To further compare the performance of the four optimizers, we present the average values of the minimum training loss, minimum testing loss, maximum training accuracy, and maximum testing accuracy across three random seeds, as shown in Table 1. The results clearly indicate that AMC outperforms traditional optimizers like Adam, AdamW, and QHAdam in terms of both training and testing accuracy, as well as training loss stability. Notably, AMC exhibits faster convergence, as it consistently reaches lower training losses in fewer epochs compared to the other optimizers. This fast convergence is especially evident in the early stages of training, where AMC shows stable and smooth loss curves, reducing oscillations and avoiding the slowdowns often seen with Adam and AdamW. In contrast, the latter two optimizers tend to exhibit large fluctuations, which can delay convergence. These findings highlight AMC's ability to handle complex models with noisy gradients and varying complexities, making it a more efficient choice for deep learning tasks that require quick convergence and stable training.

We acknowledge the importance of testing AMC on other architectures, such as MLP, encoder–decoder models, and LSTM networks. However, we believe that the experiments presented here are sufficiently comprehensive to demonstrate AMC's capabilities. The VGG models are representative of a wide range of complexities and provide a strong foundation for evaluating the optimizer's performance. Therefore, we have not included additional models in this work to maintain focus on the primary objective of evaluating AMC's learning rate adjustment mechanism. We plan to explore these other architectures in future work to expand AMC's evaluation further.

6. Conclusions

In this paper, we propose AMC, which integrates model complexity (i.e., the Frobenius norm) to adaptively adjust the learning rate based on the global model. Unlike traditional optimizers, which rely solely on local gradients, AMC leverages the overall structural complexity of the model to dynamically adjust the learning rate, enhancing both stability and convergence speed during training. The theoretical analysis shows that AMC effectively addresses the challenges posed by model complexity. When the model complexity is high, AMC reduces the learning rate to prevent instability caused by large gradient fluctuations, and conversely, when the model complexity is low, AMC increases the learning rate to accelerate convergence. Furthermore, the theoretical analysis proves that AMC exhibits superior convergence properties and convergence speed under certain conditions, which ensure its advantages in deep learning tasks. Additionally, AMC helps to alleviate oscillations during training, a common issue in models with large gradients or unstable optimization landscapes, resulting in a smooth training process.

Experimental results demonstrate that AMC outperforms other optimizers across several benchmark datasets (such as MNIST and Fashion-MNIST), especially in training deep neural networks and other complex models. AMC provides a more stable training process and faster convergence, particularly when dealing with models that exhibit large gradient fluctuations and complex structures. We observe a significant reduction in oscillations during training, which is a common challenge in deep learning optimization. The performance of AMC is notably superior to those of other optimizers in training multi-layer networks like VGG and ResNet. The contribution of this study lies in the development of AMC, which addresses the limitations of traditional optimizers by incorporating model complexity, leading to improved training efficiency and stability. AMC has significant potential in deep learning applications, not only for standard deep learning models but also for complex tasks and large-scale datasets. Future research can further explore the applicability of AMC in a broader range of domains and tasks. In particular, combining AMC with other advanced optimization techniques, such as adaptive regularization and

hybrid optimization methods, could further enhance optimizers' flexibility and adaptability, thus advancing the development of deep learning optimizers.

Author Contributions: Conceptualization, W.C. and R.P.; Methodology, B.W.; Validation, W.C.; Writing—original draft, W.C. and R.P.; Supervision, B.W. All authors have read and agreed to the published version of the manuscript.

Funding: The work is partially supported by the National Key Research and Development Program of China (2024YFF0617702), the National Natural Science Foundation of China (Nos. U22A2025, 62072088, 62232007, U23A20309, 61991404), and Liaoning Provincial Science and Technology Plan Project-Key R&D Department of Science and Technology (No. 2023JH2/101300182), and 111 Project (No. B16009).

Data Availability Statement: No new data were created or analyzed in this study.

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

References

1. Zhong, H.; Chen, Z.; Qin, C.; Huang, Z.; Zheng, V.W.; Xu, T.; Chen, E. Adam revisited: A weighted past gradients perspective. *Front. Comput. Sci.* **2020**, *14*, 145309.
2. Jin, D.; He, J.; Chai, B.; He, D. Semi-supervised community detection on attributed networks using non-negative matrix tri-factorization with node popularity. *Front. Comput. Sci.* **2021**, *15*, 154324.
3. Ter Braak, C.J.; Prentice, I.C. A theory of gradient analysis. *Adv. Ecol. Res.* **1988**, *18*, 271–317.
4. Robbins, H.; Monro, S. A stochastic approximation method. *Ann. Math. Stat.* **1951**, *22*, 400–407.
5. Polyak, B.T. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **1964**, *4*, 1–17.
6. Hinton, G.; Srivastava, N.; Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited* **2012**, *14*, 2.
7. Kong, Z.; Xu, H.; Pan, C. R-DOCO: Resilient Distributed Online Convex Optimization Against Adversarial Attacks. *Mathematics* **2024**, *12*, 3439.
8. Ramdass, P.; Ganesan, G.; Boulaaras, S.; Tantawy, S.S. Enhancing Efficacy in Breast Cancer Screening with Nesterov Momentum Optimization Techniques. *Mathematics* **2024**, *12*, 3354.
9. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
10. Pramanik, S.; Alameen, A.; Mohapatra, H.; Pathak, D.; Goswami, A. AdaMoR-DDMOEA: Adaptive Model Selection with a Reliable Individual-Based Model Management Framework for Offline Data-Driven Multi-Objective Optimization. *Mathematics* **2025**, *13*, 158.
11. Chakraborty, G.; Azhmyakov, V.; Trujillo, L.A.G. A Formal Approach to Optimally Configure a Fully Connected Multilayer Hybrid Neural Network. *Mathematics* **2025**, *13*, 129.
12. Krutikov, V.; Tovbis, E.; Gutova, S.; Rozhnov, I.; Kazakovtsev, L. Gradient Method with Step Adaptation. *Mathematics* **2025**, *13*, 61.
13. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
15. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
16. Geman, S.; Bienenstock, E.; Doursat, R. Neural Networks and the Bias/Variance Dilemma. *Neural Comput.* **1992**, *4*, 1–58.
17. Vapnik, V.N. *Statistical Learning Theory*; Adaptive and Learning Systems for Signal Processing Communications, and Control; Wiley: New York, NY, USA, 1998.
18. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444.
19. Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Netw.* **1999**, *12*, 145–151.
20. Dong, Q.; Niu, S.; Yuan, T.; Li, Y. Disentangled Graph Recurrent Network for Document Ranking. *Data Sci. Eng.* **2022**, *7*, 30–43.
21. Liu, Y.; Yang, T.; Tian, L.; Pei, J. SGD-TripleQNet: An Integrated Deep Reinforcement Learning Model for Vehicle Lane-Change Decision. *Mathematics* **2025**, *13*, 235.
22. Tan, Z.; Chen, S. On the learning dynamics of two-layer quadratic neural networks for understanding deep learning. *Front. Comput. Sci.* **2022**, *16*, 163313.

23. Abburi, H.; Parikh, P.; Chhaya, N.; Varma, V. Fine-Grained Multi-label Sexism Classification Using a Semi-Supervised Multi-level Neural Approach. *Data Sci. Eng.* **2021**, *6*, 359–379.
24. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
25. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324.
26. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
27. Belkin, M.; Hsu, D.; Ma, S.; Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 15849–15854.
28. Neyshabur, B.; Tomioka, R.; Srebro, N. In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings, San Diego, CA, USA, 7–9 May 2015.
29. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report TR-2009; University of Toronto: Toronto, ON, Canada, 2009.
30. Xue, H.; Xu, H.; Chen, X.; Wang, Y. A primal perspective for indefinite kernel SVM problem. *Front. Comput. Sci.* **2020**, *14*, 349–363.
31. Hastie, T.; Montanari, A.; Rosset, S.; Tibshirani, R.J. Surprises in High-Dimensional Ridgeless Least Squares Interpolation. *arXiv* **2019**, arXiv:1903.08560.
32. Wright, R.E. Logistic regression. In *Reading and Understanding Multivariate Statistics*; American Psychological Association: Washington DC, USA, 1995.
33. Shalev-Shwartz, S.; Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*; Cambridge University Press: New York, NY, USA, 2014.
34. He, J.; Liu, H.; Zheng, Y.; Tang, S.; He, W.; Du, X. Bi-Labeled LDA: Inferring Interest Tags for Non-famous Users in Social Network. *Data Sci. Eng.* **2020**, *5*, 27–47.
35. Nesterov, Y.E. A method for solving the convex programming problem with convergence rate O (1/kˆ2). *Dokl. Akad. Nauk SSSR* **1983**, *269*, 543–547.
36. An, W.; Wang, H.; Sun, Q.; Xu, J.; Dai, Q.; Zhang, L. A PID Controller Approach for Stochastic Optimization of Deep Networks. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8522–8531.
37. Lessard, L.; Recht, B.; Packard, A.K. Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints. *SIAM J. Optim.* **2016**, *26*, 57–95.
38. Jain, P.; Kakade, S.M.; Kidambi, R.; Netrapalli, P.; Sidford, A. Accelerating stochastic gradient descent. *arXiv* **2017**, arXiv:1704.08227.
39. Cyrus, S.; Hu, B.; Scoy, B.V.; Lessard, L. A Robust Accelerated Optimization Algorithm for Strongly Convex Functions. In Proceedings of the 2018 Annual American Control Conference, ACC 2018, Milwaukee, WI, USA, 27–29 June 2018; pp. 1376–1381.
40. Duchi, J.C.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
41. Wu, Y.; He, K. Group Normalization. In Proceedings of the Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 11217, pp. 3–19.
42. Sutskever, I. Training Recurrent Neural Networks. Ph.D. Thesis, University of Toronto, Toronto, ON, Canada, 2013.
43. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
44. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
45. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
46. Ma, J.; Yarats, D. Quasi-hyperbolic momentum and Adam for deep learning. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
47. Reddi, S.J.; Kale, S.; Kumar, S. On the Convergence of Adam and Beyond. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
48. Cheng, W.; Yang, X.; Wang, B.; Wang, W. Unbiased quasi-hyperbolic nesterov-gradient momentum-based optimizers for accelerating convergence. *World Wide Web* **2023**, *26*, 1323–1344.
49. Scoy, B.V.; Freeman, R.A.; Lynch, K.M. The Fastest Known Globally Convergent First-Order Method for Minimizing Strongly Convex Functions. *IEEE Control Syst. Lett.* **2018**, *2*, 49–54.

50. Kidambi, R.; Netrapalli, P.; Jain, P.; Kakade, S.M. On the insufficiency of existing momentum schemes for Stochastic Optimization. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
51. Zaheer, M.; Reddi, S.; Sachan, D.; Kale, S.; Kumar, S. Adaptive methods for nonconvex optimization. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; Volume 31.
52. Zhou, B.; Liu, J.; Sun, W.; Chen, R.; Tomlin, C.J.; Yuan, Y. pbSGD: Powered Stochastic Gradient Descent Methods for Accelerated Non-Convex Optimization. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, Yokohama, Japan, 7–15 January 2021; pp. 3258–3266.
53. Luo, L.; Huang, W.; Zeng, Q.; Nie, Z.; Sun, X. Learning Personalized End-to-End Goal-Oriented Dialog. In Proceedings of the The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, HI, USA, 27 January–1 February 2019; pp. 6794–6801.
54. Bottou, L.; Curtis, F.E.; Nocedal, J. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.* **2018**, *60*, 223–311.
55. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
56. Zeiler, M.D. ADADELTA: An Adaptive Learning Rate Method. *arXiv* **2012**, arXiv:1212.5701.
57. Baydin, A.G.; Cornish, R.; Martínez-Rubio, D.; Schmidt, M.; Wood, F. Online Learning Rate Adaptation with Hypergradient Descent. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
58. Dozat, T. Incorporating nesterov momentum into adam. In Proceedings of the International Conference on Learning Representations Workshop, San Juan, Puerto Rico, 2–4 May 2016.
59. Bazzi, A.; Slock, D.T.; Meilhac, L. Sparse recovery using an iterative variational Bayes algorithm and application to AoA estimation. In Proceedings of the 2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Limassol, Cyprus, 12–14 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 197–202.
60. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **1996**, *58*, 267–288.
61. Pati, Y.C.; Rezaiifar, R.; Krishnaprasad, P.S. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 1–3 November 1993; IEEE: Piscataway, NJ, USA, 1993; pp. 40–44.
62. Olshausen, B.A.; Field, D.J. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vis. Res.* **1997**, *37*, 3311–3325.
63. Pouyanfar, S.; Sadiq, S.; Yan, Y.; Tian, H.; Tao, Y.; Reyes, M.E.P.; Shyu, M.; Chen, S.; Iyengar, S.S. A Survey on Deep Learning: Algorithms, Techniques, and Applications. *ACM Comput. Surv.* **2019**, *51*, 1–36.
64. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.S.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252.