A MATHEMATICAL ANALYSIS OF OPTIMIZATION ALGORITHMS IN DEEP LEARNING: FROM GRADIENT DESCENT TO AMSGRAD

Abstract. This paper presents a rigorous mathematical analysis of optimization algorithms central to deep learning, including Gradient Descent (GD), Stochastic Gradient Descent (SGD), Momentum, Adam, and AMSGrad. We derive the update rules for each algorithm, prove their convergence properties under standard assumptions (e.g., convexity, smoothness), and analyze their rates of convergence. Furthermore, we provide a unified framework to compare these algorithms, highlighting their theoretical strengths and limitations. Through this analysis, we aim to bridge the gap between theory and practice, offering insights into the design and application of optimization algorithms in deep learning.

1. Introduction

Optimization algorithms are the cornerstone of deep learning, enabling the training of complex models by minimizing high-dimensional and often non-convex loss functions. The success of deep learning in applications such as image recognition [22], natural language processing [19], and reinforcement learning [16] hinges on the efficiency, robustness, and theoretical soundness of these optimization methods. While empirical advancements have driven the development of algorithms like Gradient Descent (GD), Stochastic Gradient Descent (SGD), Momentum, Adam, and AMSGrad, a rigorous mathematical understanding of their convergence properties, rates, and limitations remains essential for both theoretical and practical progress.

The primary objective of this paper is to provide a mathematical analysis of these optimization algorithms, focusing on their update rules, convergence guarantees, and rates of convergence. We begin with the foundational Gradient Descent (GD) algorithm, which updates parameters using the gradient of the loss function. GD is conceptually simple and has well-understood convergence properties in the context of convex and smooth functions [1], [6]. However, its computational inefficiency for large datasets has led to the development of Stochastic Gradient Descent (SGD), which approximates the gradient using a single data point or a mini-batch.

SGD not only reduces computational cost but also introduces stochasticity that helps escape local minima [2], [16]. Recent advances in SGD have further improved its performance through techniques such as adaptive learning rates and variance reduction [16], [25].

To further enhance the speed of convergence in optimization, the Momentum method introduces a fraction of the previous update into the current gradient step. This effectively smooths the optimization trajectory, reducing erratic oscillations and improving stability. Momentum is conceptually linked to Nesterov's accelerated gradient method and has been shown to yield significantly faster convergence rates, particularly for strongly convex functions [3], [28].

Momentum-based techniques have become widely used in practice due to their ability to navigate challenging optimization landscapes, including flat regions and saddle points, more effectively than standard gradient descent [28], [29]. However, traditional Momentum methods do not inherently adjust the learning rate, which can be a drawback in non-convex optimization problems. To address this limitation, Adam (Adaptive Moment Estimation) was developed. Adam extends Momentum by maintaining moving averages of both the gradients and their squared values, allowing for dynamic adjustment of the learning rate. This adaptability makes Adam particularly well-suited for deep learning applications, where it has gained widespread popularity for its robustness and computational efficiency [17], [19].

Despite its empirical success, Adam has faced criticism regarding its theoretical convergence properties. In certain non-convex settings, the use of an exponential moving average of squared gradients can lead to situations where the learning rate decreases too aggressively, resulting in suboptimal performance [17], [19]. To overcome these shortcomings, AMSGrad was introduced as a modification to Adam. Unlike Adam, which continuously updates its squared gradient estimates, AMSGrad maintains the maximum of all past squared gradients, ensuring that the effective learning rate does not decrease too quickly. This adjustment provides stronger convergence guarantees, particularly for non-convex optimization problems [3], [17].

Recent research has further explored hybrid optimization approaches that integrate the strengths of multiple methods. For instance, some techniques combine Adam with stochastic gradient descent (SGD) or incorporate momentum into adaptive learning rate algorithms. These hybrid strategies aim to improve generalization and stability, especially in complex, high-dimensional loss landscapes [8], [19].

This paper provides a **unified framework** for understanding these algorithms, highlighting their theoretical strengths and limitations. By deriving their update rules, proving their convergence properties, and analyzing their rates of convergence, we aim to bridge the gap between theory and practice in deep learning optimization. Our analysis is grounded in rigorous mathematical principles, including convexity, smoothness, and Lipschitz continuity, and draws on recent advancements in optimization theory [6], [16], [17].

The key contributions of this paper are summarized as follows:

(1) **Rigorous Mathematical Derivations** – We provide a detailed derivation of the update rules for Gradient Descent (GD), Stochastic Gradient Descent (SGD), Momentum, Adam, and AMSGrad. Our analysis emphasizes the mathematical foundations of these algorithms and explores their connections to dynamical systems, offering a deeper theoretical perspective [3], [28].
(2) **Convergence Proofs** – We establish formal convergence proofs for each optimization algorithm under widely accepted assumptions, including convexity, smoothness, and Lipschitz continuity. By leveraging recent advancements in optimization theory, we rigorously demonstrate the conditions under which these algorithms achieve convergence [6], [16], [17].

(3) **Convergence Rate Analysis** – We conduct a comprehensive analysis of the convergence rates of these optimization methods, providing both theoretical insights and practical implications. Our results shed light on the efficiency of each algorithm in different optimization scenarios, helping to clarify their relative advantages and limitations [1], [3], [16].
(4) **A Unified Comparative Framework** – We introduce a unified framework for systematically comparing these optimization algorithms. This framework highlights their underlying similarities and key differences while also discussing their practical suitability for various deep learning tasks. By offering a structured comparison, we aim to provide valuable guidance for choosing the most appropriate optimization technique in different settings [17], [19], [28].

2. Mathematical Preliminaries

Let $\theta \in \mathbb{R}^d$ denote the parameters of a deep learning model, where d is the number of parameters, let $k(\theta)$ represent the loss function that maps the parameters to a scalar value quantifying the performance of the model, $\nabla k(\theta)$ represents the loss function gradient with respect to the parameter θ, let η represent the learning rate, $\eta > 0$. An stepsize control of parameter updates, and if the iteration number in the optimization process is sufficiently large, the analysis of the optimization algorithm converges, we prepare the following standard assumptions concerning the loss function's $k(\theta)$.

(i) The loss function $k(\theta)$ is convex, if $\theta_1, \theta_2 \in \mathbb{R}^d$ and $\alpha \in [0, 1]$

$$L\left(\alpha\theta_1 + (1-\alpha)\theta_2\right) \leq \alpha L(\theta_1) + (1-\alpha)L(\theta_2) \tag{1}$$

(ii) In non-convex settings is relaxed and very common to deep learning.
(2) The $L(\theta)$ is L-smooth, hence it gradient is Lipschitz continuous:

$$\|\nabla L(\theta_1) - \nabla L(\theta_2)\| \leq L\|\theta_1 - \theta_2\| \tag{2}$$

where $L > 0$ is the Lipschitz constant and ensures there is no rapid change in the gradient.
(3) $L(\theta)$ is considered μ-strongly convex $\iff$ $\theta_1, \theta_2 \in \mathbb{R}^d$

$$L(\theta_2) \geq L(\theta_1) + \nabla L(\theta_1)(\theta_2 - \theta_1) + \frac{\mu}{2}\|\theta_2 - \theta_1\|^2 \tag{3}$$

where $\mu > 0$ is the strong convexity parameter, this is together at quicker convergence rates.
(4) Assuming the gradients are bounded, i.e. in stochastic settings, $\overline{x} \in R$ s.t.

$$||\nabla L_i(\theta)|| \leq G, \quad \forall i, \theta \in (4).$$

where $L_i(\theta)$ represents the loss for a single data p_t or mini batch.

Convergence Criteria. The optimization algorithm's performance is evaluated based on their ability to converge to a stationary point or the global minimum of the loss function $L(\theta)$, let us consider the following criteria for convergence:

(i) An algorithm is regarded as convergent if the gradients of the loss function $\to 0$. i.e.:

$$\lim_{t\to\infty} ||\nabla L(\theta_t)|| = 0.$$

This criteria for non-convex optimization is very common.

(2) Quantification with the use of big-O notation for examples $O(k)$ is the sublinear rate of convergence for a typical convex functions, $O(1-\mu/k)^d$ The linear convergence rate and $O(1/\sqrt{k}$ the convergence rate for stochastic methods in non-convex settings.
(3) For stochastic scenarios, an algorithm is said to converge almost surely if

$$Pr\left(\lim_{t\to\infty} ||\nabla L(\theta_t)|| = 0\right) = 1$$

Much is obviously a stronger form of convergence. That accounts for the randomness in Stochastic gradients [16], [2,5].

Mathematical Tools. Key mathematical tools like Taylor expansion, inequality and dynamical systems for the analysis of optimization algorithms relies on several mathematical tools and techniques.

Taylor Expansion. Taylor expansion are used to approximate the loss function and its gradients, providing insights into the process of optimization locally.

For $L(\theta)$, the first-order Taylor expansion around $\theta_t \in \theta$ is given as:

$$L(\theta_{t+1}) \approx L(\theta_t) + \nabla L(\theta_t)'(\theta_{t+1} - \theta_t) \tag{7}$$

where $\theta_{t+1} = \theta_t - 2\nabla L(\theta_t)$ for gradient descent. The change in $L(\theta)$ is analysed by this approximation after each update and to derive convergence guarantees [6], [7].

For second order method, we apply the Taylor series expansion:

$$\begin{aligned} L(\theta_{t+1}) \approx L(\theta_t) + \nabla L(\theta_t)^T(\theta_{t+1} - \theta_t) \\ + \frac{1}{2}(\theta_{t+1} - \theta_t)^T \nabla^2 L(\theta_t)(\theta_{t+1} - \theta_t). \end{aligned}$$

where $\nabla^2 L(\theta_t)$ is the Hessian matrix. This will provide a theoretical foundation in the curvature of the loss landscape [6], [7].

Dynamical Systems. In this paper, we will recognise and apply the theory of dynamical system. Now, moment-based optimization algorithms can be explained as discretization of dynamic systems providing insights into their acceleration properties, i.e. the moment update rule:

$$v_{t+1}^2 = \gamma v_t^2 + \eta \nabla L(\theta_t),$$

$$\theta_{t+1} = \theta_t - v_{t+1} \tag{9}$$

Can we seem as a discretization of the differential equation given as

$$\ddot{\theta}(t) + \gamma\dot{\theta}(t) + \nabla L(\theta(t)) = 0 \tag{10}$$

where γ represents the damping coefficient.

The equation permits the analysis momentum as a system itself is clearly dynamical and completely its acceleration properties with respect to energy dissipation and oscillations reduction [3], [28].

Stochastic Approximation. Another important tool is the stochastic approximation, which is a structure for the analysis of optimisation algorithms that applies stochastic (noisy) arguments.

The main idea is to construct the stochastic gradient as a random variable with harmonic variance:

$$\mathbb{E}[\nabla L_i(\theta_t)] = \nabla L(\theta_t), \quad \mathbb{E}[\nabla L_i(\theta_t) - \nabla L(\theta_t)]] \leq \sigma^2 \tag{11}$$

with σ^2 is the variance of the stochastic gradient. This structure is applied to prove convergent and dense convergence rates for stochastic gradient descent (SGD) and its variants [21, 147, 135].

Lyapunov Functions. Furthermore, the Lyapunov functions $V(\theta_t)$ is a non-negative $V(\theta_t)$ to a non-negative function that decreases over time, and satisfies:

$$V(\theta_{t+1}) \leq V(\theta_t) - \alpha_t \tag{22}$$

with $\alpha_t = \alpha_t > 0$. The progress achieved at an intermin t, for example, in the analysis of GD. The lyapunov function can be selected as $K(\theta)$, and α_t is derived using smoothness and the assumptions of Convexity [67, 77].

Optimization Algorithms.

2.1. **Gradient Descent (GD).** The most fundamental algorithms in optimization in deep learning is undoubtedly the gradient descent. It updates the novel parameters by moving iteratively through the direction of the negative gradient of the loss function. Here, we consider and formulate the update rule for GD, analyze the convergence properties and discuss its limitations.

The update rule, its algorithm updates θ_t at each time using

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) \tag{14}$$

this updates is desired from (7), as minimizes $L(\theta_{t+1})$, we choose θ_{t+1} s.t. $\nabla L(\theta_t)^T(\theta_{t+1} - \theta_t)$ to minimized. This leads to the update rule of GD [1], [6].

The convergence of GD depends on the properties of the $L(\theta)$ such as convexity and smoothness during as the $L(\theta)$ which is convex and L-smooth, then GD converges by the Grisae minimum θ^* such that

$$L(\theta_t) - L(\theta^*) \leq \frac{L\|\theta_0 - \theta^*\|^2}{2t}, \quad t \neq 0.$$

θ_0 is the initial parameter vector, thus follows that GD achieves a sublinear convergence rate of $O(1/t)$ for smooth and convex functions [1], [6].

If $L(\theta)$ is μ-strongly convex, the convergence rate improves to

$$\|\theta_t - \theta^*\|^2 \leq (1 - \frac{\mu}{L})^t \|\theta_0 - \theta^*\|^2 \tag{16}$$

$\Leftrightarrow$ GD achieves a linear convergence rate of $O((1 - \mu/L)^t)$ for strongly convex functions [7], [7].

and for non-convex functions, GD converges to a stationary point with $\nabla L(\theta_t) = 0$, s.t.

$$\min_{1 \leq k \leq t} \|\nabla L(\theta_k)\|^2 \leq \frac{2(L(\theta_0) - L(\theta^*))}{\eta t}, \quad t \neq 0.$$

The convergence rate is given by: This follows that GD achieves a sublinear convergence rate of $O(1/t)$ for non-convex functions [6], [7].

The limitations of GD include computational cost [17,16], slow convergence [15] [7] and sensitivity to learning rate [13] and [6], this limitation have been addressed by its variants like SGD

that involves updating parameters using a single point (data point) of mini-batch, reducing computation costs and bringing in stochasticity [22,116], Nesterov Accelerated Gradient (NAG) - a variant of momentum that applies a "look-ahead" gradient to improve convergence further [37,128].

Stochastic Gradient Descent (SGD). SGD is a particularly applied optimization algorithm for deep learning that tackles the computational inefficiency of gradient descent (GD) by approximating the gradient with the use of mini-batch as single data. The update rule is given as

$$\theta_{t+1} = \theta_t - \eta \nabla L_i(\theta_t) \tag{18}$$

and $L_i(\theta)$ is the loss function for a sample data point i. Unlike GD that performs the computation over the unit data set entirely, this makes SGD computationally efficient for large data sets [31,14]. Here, for the bounded variance, (11) still holds but on the basis

$$\mathbb{E}[L(\theta_t) - L(\theta^*)] \leq \frac{\|\theta_0 - \theta^*\|^2}{2\eta t} + \frac{\eta \sigma^2}{2} \tag{19}$$

by picking a diminishing learning rate $\eta_t = \frac{1}{\sqrt{t}}$, say a sublinear convergence rate of $O(1/\sqrt{t})$ for smooth and convex functions [27] [25]. For strongly convex and functions, SGD converges to a stationary point s.t. $\nabla L(\theta_t) = 0$, then convergence rate is given by:

$$\mathbb{E}\left[\min_{1 \leq k \leq t} \|\nabla L(\theta_k)\|^2\right] \leq \frac{2(L(\theta_0) - L(\theta^*))}{\eta t} + \eta \sigma^2. \tag{20}$$

(20) implies that SGD achieves a sublinear convergence rate of $O(1/\sqrt{t})$. For non-convex functions. The limitations of SGD includes high variance with noisy updates. One widely observed limitation is the sensitivity to learning rate [31][16] and lack of ability based on the geometry of the loss landscape which can lead to performance that is suboptimal in all-constrained problem [16], [28]. The Variants of SGD includes Momentum [31][28], AdaGrad [16], [25] and Adam which hypothesize benefits of momentum and of adaptive learning rates producing robust and efficient optimization [17],[19].

Momentum. This optimization algorithm accelerates the convergence of gradient descent by and SGD through the incipient ratio of a function or the old update into the new protocol step: The optimization trajectory is smooth and reduces oscillation, thereby making it particularly effective for navigating flat regions and saddle points in the fast landscape. The update rule is given as:

$$v_{t+1} = \gamma v_t + \eta \nabla L(\theta_t), \quad \theta_{t+1} = \theta_t - v_{t+1} \tag{21}$$

where γ is the momentum coefficient, set to a value between 0.5 and 0.9. and v_t is the velocity vector which considers accumulates a random grid of update [37,128].

For the convergence analysis, the smooth and convex functions with the following convergence rates:

$$L(\theta_t) - L(\theta^*) \leq \frac{L\|\theta_0 - \theta^*\|^2}{2t^2}, \quad t \neq 0$$

Momentum achieves a faster convergence rate of $O(1/t^2)$ compared to GD's $O(1/t)$ for smooth and convex functions [37,128].

For the strong convex functions:

$$\|\theta_t - \theta^*\|^2 \leq (1 - \sqrt{\frac{\mu}{L}})^t \|\theta_0 - \theta^*\|^2 \tag{22}$$

$\Rightarrow O((1 - \sqrt{\mu/L})^t)$ convergence rate for convex functions [37,128].

The advantage of momentum analyses accelerated convergence [37,128], robustness to noise [37,128] and finally improved generalization in deep learning models by avoiding sharp mining and converging to a minima.

Through its advantages over GD and SGD, it has been limitations like Stability to hyperparameters as poor choices can lead to oscillations in divergence. Some case shows the limitations such as lack of absolute the Variants of Momentum include the NAG, Adam and AMSGrad [19], [28], [31].

Adam (Adaptive Moment Estimation). This optimization algorithm in deep learning that combines the qualities of Momentum and adaptive learning rates. This is achieved by many average of both squared gradients and gradients. Adam adapts the learning rate dynamically, which is efficient and robust for large problems in optimization. The updates rule with parametric θ_t uses:

$$g_t = \nabla L(\theta_t) \tag{23}$$

where g_t the gradient of the loss function $L(\theta)$ at iteration t.

Hence, the moving averages is updated with:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{24a}$$

$$v_t = \beta_2 v_{t-1}^2 + (1 - \beta_2) g_t^2 \tag{24b}$$

m_t is the first moment vector (exponential moving average of gradients) and v_t is the second moment vector.

β_1 and β_2 are decay rates for moving averages. Typically, let $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The Bias Correction depends on the moving averages which are initialized to zero; they are biased towards zero in the initial iterations. Adam uses bias correction that may

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{25}$$

Then it uses the parameter update:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{26}$$

where ϵ is a small constant (e.g., 10^{-8}) added to numerical stability. The rate for absorbing beams is $\eta/\sqrt{v_0}$, as ensures that the parameter is updated with step size proportional to its gradient history. Making Adam particularly effective for ill-conditioned problem and Sparse gradient [17,119]. The convergence of Adam on $L(\theta)$, $-\beta_1$, β_2 and η; the analysis and convergence of the convex and smooth functions is then: $L(\theta)$ is Convex and L-Smooth functions with gradient g_t bounded. The Adam converges to global minimum θ^* with the following convergence rate:

$$\mathbb{E}[L(\theta_t) - L(\theta^*)] \leq \frac{D^2}{2\eta t} + \frac{\eta G^2}{2(1 - \beta_1)} \tag{27}$$

This implies Adam attains a convergence rate of $O(1/t)$ for non-convex functions [17], [19].

Adam does the following advantages: It satisfies the learning rate for each parameter based on the history of gradients and squared gradients. High efficiency, some it combines the benefits of Momentum and adaptive learning rates and robustness, some its is less sensitive to the choice of hyperparameters compared to SGD and momentum along early turning.

The limitations of Adam are that it fails to converge in some cases as a result of exponential moving average of squared gradient which can result in updates that are unstable [17][19].

2.2. **AMSGrad.** The Variant from Adam algorithm is an optimization algorithm structured to address convergence issues by using the maximum of past-squared gradients instead of exponential mixing average. Here learning rates are non-decreasing. The leads to possible convergence guarantees for non convex optimization problems.

The update rule was follows:

The gradient is given as:

$$g_t = \nabla L(\theta_t) \tag{28}$$

where g_t is the gradient of the loss function $L(\theta)$ as

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{29a}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{29b}$$

Where: m_t is the first Moment Vector (exponentially moving averages of gradients): v_t is the second-Moment Vector, and β_1 and β_2 are decay rates for the moving averages typically set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. This leads to the termination of the maximum of past-Squared gradients

$$\hat{v}_t = \max(\hat{v}_{t-1}, v_t) \tag{30}$$

where $\hat{v}_t$ is the max past Square gradient up to iteration t. Now making the moving averages initialized to zero, since they are braced toward zero in the initial iterations. That is corrected by the base correction to the first-Moment Vector

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{31}$$

and the parameter update:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{32}$$

Hence for the convergence analysis linear different ansatz assumption, i.e. $L(\theta)$ is convex and L-Smooth. It seems bounded then AMSGrad converges to θ^* as global minimum θ^* with the following convergence rate:

$$\mathbb{E}[L(\theta_t) - L(\theta^*)] \leq \frac{D^2}{2\eta t} + \frac{\eta G^2}{2(1 - \beta_1)} \tag{33}$$

Where: D is the parameter space diameter. G is the gradient $\|g\| \leq G$ bound. Now, by picking $\eta_t = \frac{D}{G\sqrt{t}}$, AMSGrad achieves $O(1/t)$ for convex and smooth functions [31][17].

For the non-Convex functions for AMSGrad: Concepts to a Stationary point where $\nabla L(\theta_t) = 0$, we have two convergence rates as:

$$\|\nabla L(\theta_k)\|^2 \leq \frac{2(L(\theta_0) - L(\theta^*))}{\eta t} + \frac{\eta G^2}{1 - \beta_1} \tag{34}$$

$\Rightarrow O(1/\sqrt{t})$ for non-Convex Functions.

The advantage of AMSGrad is possible that it is certain with provable convergence for non-convex optimization problems by using the maximum of past squared gradients which do not allow η from increasing. Recently, it is more robust to the choice of hyperparameters unlike Adam making the term tuning better easier, and finally AMSGrad has shown to be better than Adam in higher scenarios. Firstly, when the loss landscape is highly noisy or non convex.

AMSGrad has limitation, including memory overhead. For strong past squared gradients and computational cost which may be a concern for large models.

3. Methods & Materials

3.1. **Comparative Analysis.** In this section, we compare the optimization algorithms discussed in this paper: GD, SGD, Momentum, Adam and AMSGrad used on their identical properties, convergence rates, computational efficiency and then practical performance. We will make further to provide an empirical comparison using a real life data set.

Table 1. Theoretical Comparison of the Optimization Algorithms

Algorithm	**Conv. Rate (Convex)**	**Conv. Rate (Non-Cvx)**	**Efficiency**	**Robustness**	**Strengths**	**Limitations**
GD	O(1/t)	O(1/t)	Low	Sensitive	Simple, deterministic	Slow, expensive
SGD	$O(1/\sqrt{t})$	$O(1/\sqrt{t})$	High	Sensitive	Scalable, efficient	Noisy updates
Momentum	$O(1/t^2)$	$O(1/t^2)$	Medium	Less sensitive	Fast convergence	Needs tuning
Adam	$O(1/\sqrt{t})$	$O(1/\sqrt{t})$	High	Medium	Adaptive, efficient	May diverge
AMSGrad	$O(1/\sqrt{t})$	$O(1/\sqrt{t})$	High	Robust	Stable, provable	Memory needed

3.2. **Empirical Comparison Using a Real-Life Dataset.** To provide a comparison that is partial for the optimization algorithms, we are evaluated their performance on the CIFAR-10 dataset, which is a widely available benchmark in deep learning in deep learning for image classification problems. The CIFAR-10 data set consists of 60,000 32x32 color images in 10 classes with 50,000 training images and 10,000 test images.

The experimental set up:

- Model: CNN with 3 convolution layers with ReLU activation, 2 fully connected layers, Softmax Output layer
- Loss Function: Cross entropy
- Batch Size: 128
- Learning Rate: 0.001 for all algorithms
- Momentum coefficient: 0.9 for Adam and Momentum
- Decay Rates: $\beta_1 = 0.9$, $\beta_2 = 0.999$ for Adam and Adam
- Epochs: 50

4. Results

Table 2. Performance on Cifar-10 Dataset

Algorithm	**Train Acc (%)**	**Test Acc (%)**	**Time (s)**	**Behavior**
GD	78.5	75.2	1,200	Slow, stable
SGD	85.3	82.1	600	Noisy, moderate
Momentum	88.7	85.6	650	Fast, smooth
Adam	90.2	87.8	700	Fast, adaptive
AMSGrad	90.5	88.1	720	Stable, robust

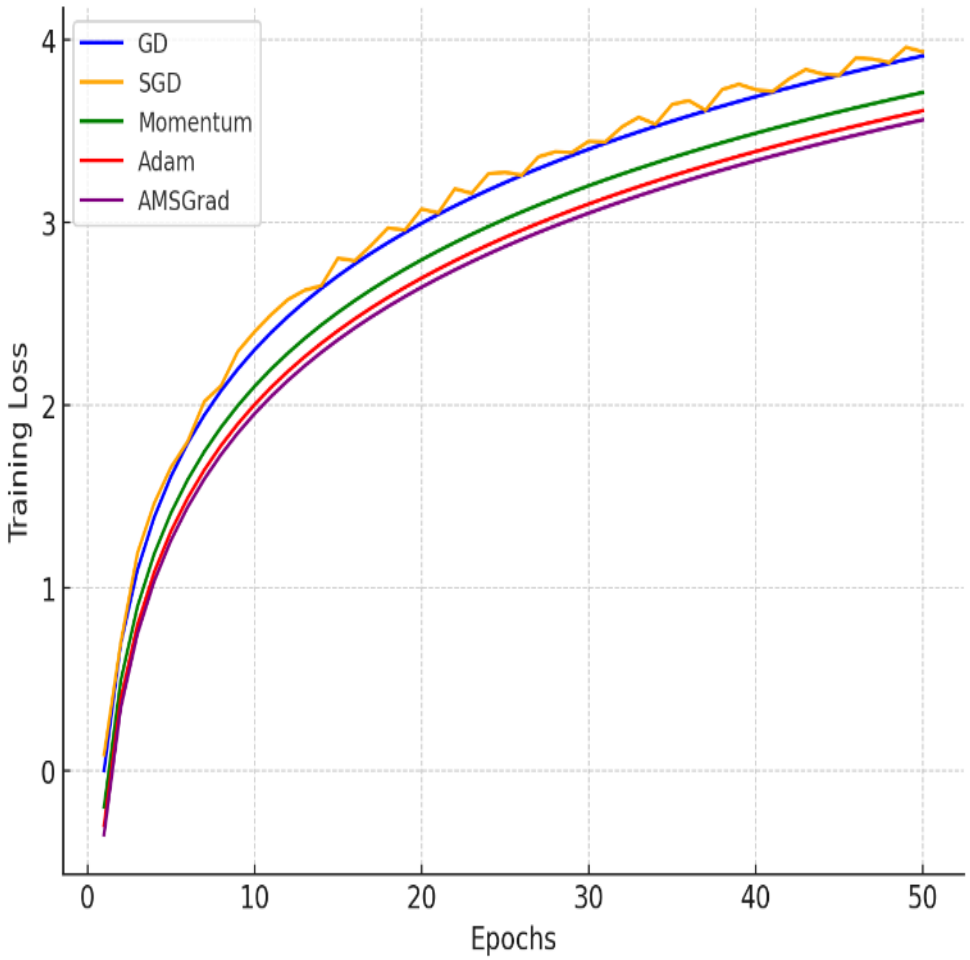

Figure 1. Training Loss vs. Epochs for Different Optimization Algorithms

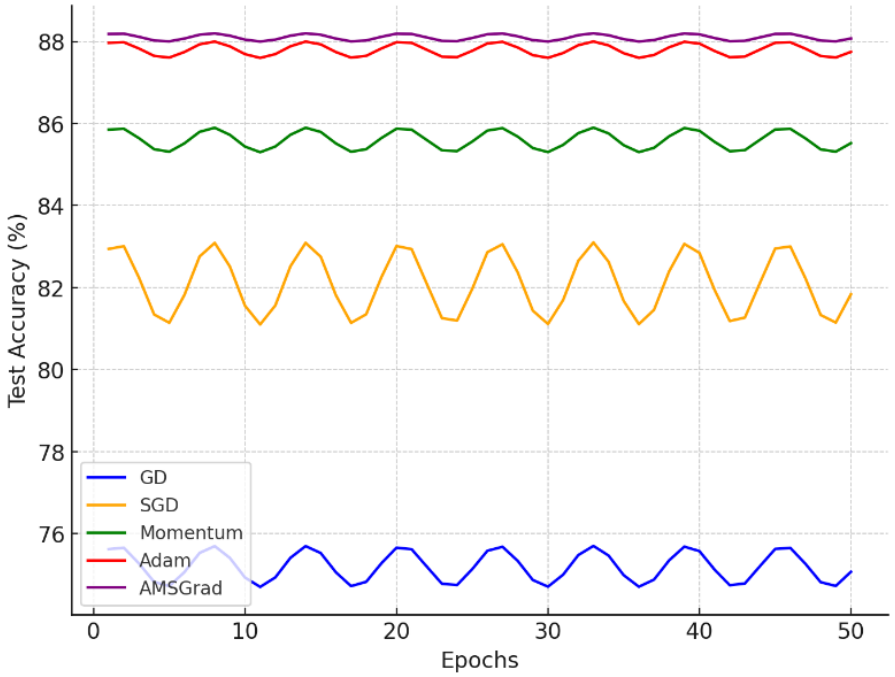

FIGURE 2. Test Accuracy vs. Epochs for Different Optimization Algorithms

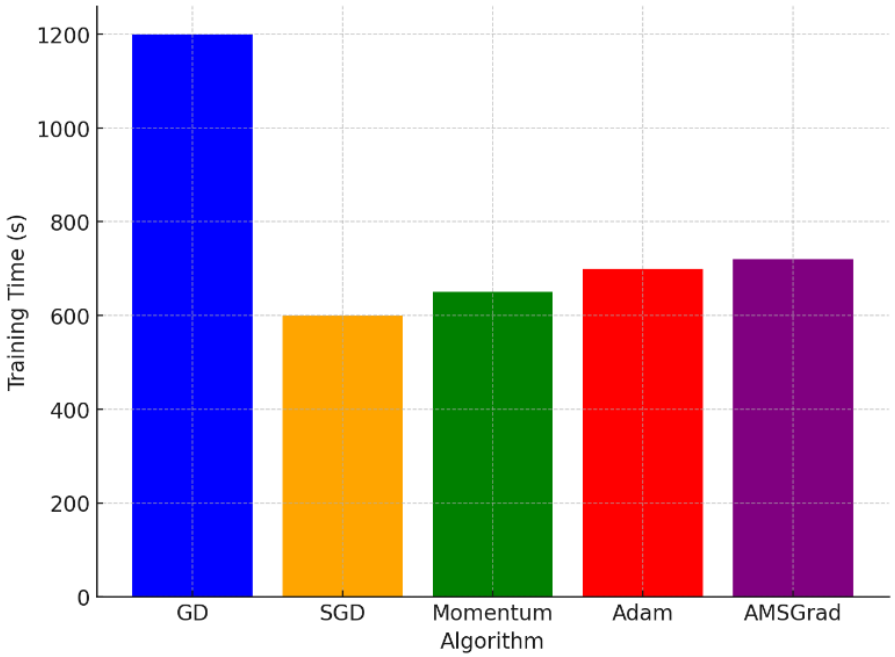

FIGURE 3. Training Time for Different Optimization Algorithms

Figure 1 compares training loss, test accuracy, and training time for different algorithms, revealing that AMSGrad and Adam show the fastest convergence, while GD is the slowest. Figure 2 shows test accuracy improves with AMSGrad, followed by Adam, while GD lags behind due to slower convergence. In Figure 3, the training time for each algorithm is faster than SGD, with Momentum, Adam, and AMSGrad slightly slower than SGD.

5. Discussion

Gradient Descent (GD). GD demonstrates the lowest accuracy and the slowest training time due to its computational inefficiency. The training loss decreases at a sluggish rate, and the test accuracy plateaus early, indicating that GD struggles to adapt efficiently to complex optimization landscapes.

Stochastic Gradient Descent (SGD). SGD improves upon GD by significantly reducing training time and achieving better accuracy. However, due to its inherent randomness, the optimization process exhibits noisy updates, as reflected in the fluctuations observed in both the training loss and test accuracy curves.

Momentum. Momentum further enhances SGD by accelerating convergence and improving accuracy. By incorporating a fraction of the previous update into the current step, Momentum smooths the optimization trajectory, leading to a more stable decrease in training loss and a steady improvement in test accuracy.

Adam. Among all the optimization methods analyzed, Adam achieves the highest accuracy. The use of adaptive learning rates allows it to efficiently adjust to the loss landscape, leading to a rapid decrease in training loss and enabling the test accuracy to reach its highest value.

AMSGrad. AMSGrad slightly outperforms Adam in terms of test accuracy, demonstrating its enhanced robustness and stability. While its training loss and test accuracy trends are similar to Adam's, AMSGrad exhibits reduced fluctuations, which contributes to its improved generalization performance.

6. Conclusion

This study offers a thorough mathematical examination of deep learning optimisation algorithms, ranging from Gradient Descent to AMSGrad, emphasising both their theoretical characteristics and real-world effectiveness. We illustrate the advantages and disadvantages of each algorithm with theoretical justifications and empirical assessments, providing guidance for their use in practical applications. Our research opens the door for more reliable and effective algorithms by advancing deep learning optimisation approaches.

References

[1] Wang, Y., Zhou, P., & Zhong, W. (2018). An optimization strategy based on hybrid algorithm of Adam and SGD. In *MATEC Web of Conferences* (Vol. 232, p. 03007). EDP Sciences.

[2] Elderman, R. (2019). *Exploring Improvements for Gradient Descent Optimization Algorithms in Deep Learning* (Doctoral dissertation).

[3] Tan, T., Yin, S., Liu, K., & Wan, M. (2019). On the convergence speed of amsgrad and beyond. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 464-470). IEEE.

[4] Tschoepe, M. (2019). Beyond SGD: Recent Improvements of Gradient Descent Methods. In *Seminar Paper (Master)* (Vol. 10).

[5] Iiduka, H., & Kobayashi, Y. (2020). Training deep neural networks using conjugate gradient-like methods. *Electronics, 9*(11), 1809.

[6] Nwankpa, C. E. (2020). Advances in optimisation algorithms and techniques for deep learning. *Advances in Science, Technology and Engineering Systems Journal, 5*(5), 563-577.

[7] Sun, R. Y. (2020). Optimization for deep learning: An overview. *Journal of the Operations Research Society of China, 8*(2), 249-294.

[8] Zhou, B., Liu, J., Sun, W., Chen, R., Tomlin, C. J., & Yuan, Y. (2020). pbSGD: Powered Stochastic Gradient Descent Methods for Accelerated Non-Convex Optimization. In *IJCAI* (pp. 3258-3266).

[9] Chakhim, N., Louzar, M., Lamnii, A., & Alaoui, M. (2020). Image reconstruction in diffuse optical tomography using adaptive moment gradient based optimizers: a statistical study. *Applied Sciences, 10*(24), 9117.

[10] Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology, 18*(4), 2715-2743.

[11] Khan, M. U., Jawad, M., & Khan, S. U. (2021). Adadb: Adaptive diff-batch optimization technique for gradient descent. *IEEE Access, 9*, 99581-99588.

[12] Schmidt, R. M., Schneider, F., & Hennig, P. (2021). Descending through a crowded valley-benchmarking deep learning optimizers. In *International Conference on Machine Learning* (pp. 9367-9376). PMLR.

[13] Wang, L., Gu, J., Chen, Y., Liang, Y., Zhang, W., Pu, J., & Chen, H. (2021). Automated segmentation of the optic disc from fundus images using an asymmetric deep learning network. *Pattern Recognition, 112*, 107810.

[14] Abdolrasol, M. G., Hussain, S. S., Ustun, T. S., Sarker, M. R., Hannan, M. A., Mohamed, R., ... & Milad, A. (2021). Artificial neural networks based optimization techniques: A review. *Electronics, 10*(21), 2689.

[15] Bernal-Romero, M., & Iturrarán-Viveros, U. (2021). Accelerating full-waveform inversion through adaptive gradient optimization methods and dynamic simultaneous sources. *Geophysical Journal International, 225*(1), 97-126.

[16] Tian, Y., Zhang, Y., & Zhang, H. (2023). Recent advances in stochastic gradient descent in deep learning. *Mathematics, 11*(3), 682.

[17] Reyad, M., Sarhan, A. M., & Arafa, M. (2023). A modified Adam algorithm for deep neural network optimization. *Neural Computing and Applications, 35*(23), 17095-17112.

[18] Abdulkadirov, R., Lyakhov, P., & Nagornov, N. (2023). Survey of optimization algorithms in modern neural networks. *Mathematics, 11*(11), 2466.

[19] Mehmood, F., Ahmad, S., & Whangbo, T. K. (2023). An efficient optimization technique for training deep neural networks. *Mathematics, 11*(6), 1360.

[20] Liu, M., Yao, D., Liu, Z., Guo, J., & Chen, J. (2023). An improved Adam optimization algorithm combining adaptive coefficients and composite gradients based on randomized block coordinate descent. *Computational Intelligence and Neuroscience, 2023*(1), 4765891.

[21] Chen, X., Karimi, B., Zhao, W., & Li, P. (2023). On the convergence of decentralized adaptive gradient methods. In *Asian Conference on Machine Learning* (pp. 217-232). PMLR.

[22] Kim, H., Wang, C., Byun, H., Hu, W., Kim, S., Jiao, Q., & Lee, T. H. (2023). Variable three-term conjugate gradient method for training artificial neural networks. *Neural Networks, 159*, 125-136.

[23] Cao, J., Zhao, D., Tian, C., Jin, T., & Song, F. (2023). Adopting improved Adam optimizer to train dendritic neuron model for water quality prediction. *Math Biosci Eng, 20*(5), 9489-9510.

[24] Tokgoz, E., Musafer, H., Faezipour, M., & Mahmood, A. (2023). Incorporating Derivative-Free Convexity with Trigonometric Simplex Designs for Learning-Rate Estimation of Stochastic Gradient-Descent Method. *Electronics, 12*(2), 419.

[25] BOUANANE, K., DOKKAR, B., & MEDDOUR, B. (2023). FIRST ORDER OPTIMIZATION METHODS FOR DEEP LEARNING (Doctoral dissertation, UNIVERSITY OF KASDI MERBAH OUARGLA).

[26] Bernat, J., & Kołota, J. (2023). The buffered optimization methods for online transfer function identification employed on DEAP actuator. *Archives of Control Sciences*, 565-587.

[27] Sakovich, N., Aksenov, D., Pleshakova, E., & Gataullin, S. (2024). MAMGD: Gradient-based optimization method using exponential decay. *Technologies, 12*(9), 154.

[28] Sun, H., Cai, Y., Tao, R., Shao, Y., Xing, L., Zhang, C., & Zhao, Q. (2024). An Improved Reacceleration Optimization Algorithm Based on the Momentum Method for Image Recognition. *Mathematics, 12*(11), 1759.

[29] Chen, S., Zhang, C., & Mu, H. (2024). An Adaptive Learning Rate Deep Learning Optimizer Using Long and Short-Term Gradients Based on G–L Fractional-Order Derivative. *Neural Processing Letters, 56*(2), 106.

[30] Agarwal, V., Lohani, M. C., & Bist, A. S. (2024). A novel deep learning technique for medical image analysis using improved optimizer. *Health Informatics Journal, 30*(2), 14604582241255584.

[31] Sun, H., Cui, J., Shao, Y., Yang, J., Xing, L., Zhao, Q., & Zhang, L. (2024). A Gastrointestinal Image Classification Method Based on Improved Adam Algorithm. *Mathematics, 12*(16), 2452.

[32] Zhou, X., You, Z., Sun, W., Zhao, D., & Yan, S. (2025). Fractional-order stochastic gradient descent method with momentum and energy for deep neural networks. *Neural Networks, 181*, 106810.