Stability-Driven CNN Training with Lyapunov-Based Dynamic Learning Rate

Dahao Tang[1 [0009−0005−0999−208X]], Nan Yang[1 [0000−0001−5257−9227]], Yongkun Deng[1 [0009−0002−6002−3485]], Yuning Zhang[1 [0000−−0002−0306−1859]], Abubakar Sadiq Sani[2 [0000−0001−8201−8770]], and Dong Yuan[1 [0000−0003−1130−0888]]

[1] Faculty of Engineering, The University of Sydney, Australia
(dahao.tang, n.yang, yongkun.deng, yuning.zhang1, dong.yuan)@sydney.edu.au

[2] School of Computing and Mathematical Sciences, The University of Greenwich, London SE10 9LS, UK
s.sani@greenwich.ac.uk

Abstract. In recent years, Convolutional Neural Networks (CNNs) have become a cornerstone in computer vision tasks, but ensuring stable training remains a challenge, especially when high learning rates or large datasets are involved, as standard optimization techniques like Stochastic Gradient Descent (SGD) can suffer from oscillations and slow convergence. In this paper, we leverage control theory to propose a novel stability-driven training method by modeling the CNN training process as a dynamic control system where we introduce Lyapunov Stability Analysis, implemented with Quadratic Lyapunov Function, to guide real-time learning rate adjustments, ensuring stability and faster convergence. We provide both theoretical insights and practical guidelines for the implementation of the learning rate adaptation. We examine the effectiveness of this approach in mitigating oscillations and improving training performance by comparing the proposed Lyanpunov-stability-enhanced SGD, termed SGD-DLR (SGD with Lyapunov-based Dynamic Learning Rate), to traditional SGD with a fixed learning rate. We also conduct experiments on the datasets CIFAR-10 and CIFAR-100 to demonstrate that SGD-DLR enhances both stability and performance, outperforming standard SGD. The code used for the experiment has been released on GitHub: https://github.com/DahaoTang/ADC-2024-SGD_DLR.

Keywords: Convolutional Neural Networks · Control Theory · Lyapunov Stability Analysis · Learning Rate.

1 Introduction

Convolutional Neural Networks (CNNs) play an important role in modern deep learning, particularly in image classification, object detection, and other computer vision tasks [1, 2]. Despite their widespread success, CNN training faces significant challenges, including issues like vanishing or exploding gradients, non-convex loss surfaces, and over-fitting [3–5]. These challenges impede convergence and can degrade model performance.

While techniques such as normalization, advanced optimizers like Adam [6] and RMSprop [7], and learning rate schedules have been developed to address these issues, ensuring stability during training remains an unresolved challenge. Current methods lack a theoretical framework that guarantees training stability throughout the process [8]. This paper addresses that gap by introducing a control-theoretic approach to CNN training.

We propose a novel framework by modeling CNN training as a dynamic system, leveraging control theory, specifically, Lyapunov Stability Analysis [9], to guide the process. By viewing the training procedure as a controlled system, we introduce a method to dynamically adjust the learning rate to ensure real-time stability. This approach bridges the gap between CNN training dynamics and control theory, providing a structured solution for stabilizing training.

This paper makes the following key contributions:

- **Control-Theoretic Modeling of CNN Training:** We model CNN training as a dynamic control system, incorporating state-space representation and control inputs to optimize the learning process. This provides a formal structure that goes beyond conventional techniques in stabilizing CNNs.
- **Lyapunov-based Learning Rate Adaptation:** We employ Lyapunov Stability Analysis, implemented using the Quadratic Lyapunov Function, to guide the real-time adjustment of the learning rate during CNN training, ensuring convergence without oscillations or divergence
- **Comprehensive Experiments:** We conduct experiments using datasets CIFAR-10 and CIFAR-100 [10] to demonstrate how our method can empower standard optimizers such as basic Stochastic Gradient Descent (SGD) [5] to demonstrate its effectiveness in enhancing training stability and improving generalization performance.

2 Related Work

In this section, we discuss the challenges causing instability during the training process of CNNs and some existing solutions to tackle them. We then introduce Lyapunov Stability Analysis, a powerful tool used to assess and maintain the stability of a system. Finally, we investigate some existing work that applies Lyapunov Stability Analysis to machine learning and deep learning.

2.1 Challenges in CNN Training

Maintaining stability during the training of a CNN may encounter various challenges, such as vanishing or exploding gradients [3], over-fitting [4], highly non-convex loss surfaces causing the optimizer to get stuck in local minima [11], leading to sub-optimal convergence or even divergence.

To tackle these challenges, a variety of techniques have been invented, with normalization, regularization, and optimization algorithms in adapting learning rates being some of the most popular ones. For instance, as proposed by Ioffe et

al., batch normalization [12] prevents gradients from vanishing. Regularization methods, such as L2 regularization, and dropout[5], can help improve generalization. Advanced optimization algorithms like Adam [6], RMSprop [7], and SGD [13] overcome challenges in non-convex optimization.

2.2 Lyapunov's Methods and Lyapunov Stability Analysis

Traditionally used in control theory for dynamical systems, such as robotic systems, power systems, and autonomous vehicles [9], Lyapunov [14] presents a mathematical method used to assess the stability of a system by constructing a Lyapunov function, which shows that a system remains stable as time progresses if the function decreases along system trajectories.

Recent works have applied Lyapunov Stability Analysis to various machine learning models. For example, Lyapunov functions have been used to guarantee stability in recurrent neural networks under time-varying delays [15], and in Neural ODEs to improve robustness to adversarial attacks [16]. In reinforcement learning, Lyapunov-based stability is employed to stabilize policy learning [17]. Some studies have integrated Lyapunov methods into CNNs, but these typically focus on stability during inference or control applications, not real-time adjustments during training [18].

3 Preliminaries

3.1 Control Theory for CNNs

In this section, we model the training process of CNNs as a dynamic control system and employ Lyapunov Stability Analysis to assess and ensure its stability.
Time Set (T): The time set $T = \{t_0, t_1, \ldots, t_k\}$ represents discrete epochs or iterations during which the CNN's state is updated. Each epoch t marks a time step where hyper-parameters like the learning rate and batch size, can be adjusted to optimize the learning process.
State Space ($\mathcal{X}$): The state space $\mathcal{X}$ consists of all possible configurations of the CNN's weights and biases. The state at any given epoch t is represented by the vector $\mathbf{x}_t = (\mathbf{w}_t, \mathbf{b}_t) \in \mathcal{X}$ where $\mathbf{w}_t$ and $\mathbf{b}_t$ are the weight vector and the bias vector at the epoch t, respectively.
Input Space ($\mathcal{I}$): The input space $\mathcal{I}$ contains all the possible data that can be passed to the network, including the dataset used for training and validation. Each input sample is a tuple $\mathbf{i}_i = (u_i, d_i)$ where u_i is the input data (e.g., an image or feature vector) and d_i is the corresponding true label or output.
Control Input Space ($\mathcal{U}$): The control input space $\mathcal{U}$ consists of the hyper-parameters that influence the learning process of a CNN, such as the learning rate η_t which determines the step size for updating the weights and biases. By adjusting control inputs, the way how the states of the network are transitioned can affected and ideally, controlled.

Performance Metrics ($\mathcal{P}$): The performance metric space $\mathcal{P}$ includes the criteria used to evaluate the network's performance at each epoch, such as accuracy or loss. These metrics provide feedback for adjusting the control inputs and help determine the effectiveness of the learning process.
The transition map (ϕ): The transition map $\phi : \mathbf{X} \times \mathbf{U} \times \mathbf{I} \times \mathbf{T}^2 \to \mathbf{X}$ describes how the state of the CNN evolves, especially during training.

In this paper, we choose the **learning rate** η_t as the only hyper-parameter considered as the control input, thus the transition map for the state can be expressed as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_{\mathbf{x}} L(\mathbf{x}_t) \tag{1}$$

where: $\mathbf{x}_t = (\mathbf{w}_t, \mathbf{b}_t)$ is the state vector at time step t, comprising both weight vector $\mathbf{w}_t$ and bias vector $\mathbf{b}_t$. η_t is the learning rate at time step t. $\nabla_{\mathbf{x}} L(\mathbf{x}_t)$ is the gradient of the loss function $L(\mathbf{x}_t)$ with respect to the state vector $\mathbf{x}_t$. The loss function, whose implementation depends on specific use cases, $L(\mathbf{x})$ measures the error between the predicted and true outputs.

Furthermore, the **Jacobian matrix**, which is defined as:

$$\mathbf{J}_t = \nabla_{\mathbf{x}} L(\mathbf{x}_t) \tag{2}$$

can be used to capture the relationship between the state vector and the loss function. Thus the transition map for the state can be expressed as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_{\mathbf{x}} L(\mathbf{x}_t) = \mathbf{x}_t - \eta_t \mathbf{J}_t \tag{3}$$

which emphasizes the gradient of the loss instead of the native loss itself. Thus the representation of the loss is no longer a scalar function but a matrix (or tensor) so that it can be used as a more useful criterion for measuring the stability of the training process of CNNs when viewing the process as a dynamic system.

3.2 Lyapunov Stability Analysis

As a fundamental method in control theory and dynamical systems, Lyapunov Stability Analysis is commonly used for determining the stability of equilibrium points in dynamic systems [19, 20, 9]. Stability is crucial because it determines whether a system, when subjected to small disturbances, will return to equilibrium or diverge away, leading to potential failure [21]. During CNN training, stability can be used to describe whether the network's parameters will converge to a desirable solution (an equilibrium point or a minimum of the loss function) rather than oscillating or diverging [5, 22].

Lyapunov Function: The core process of applying Lyapunov Stability Analysis involves constructing the Lyapunov Function $V(x)$, which, in essence, is a scalar function that provides a way to evaluate the stability of a system without explicitly solving its differential equations.

Lemma 1. *For a system described as:*

$$\dot{x} = f(x)$$

where $\dot{x}$ is the time derivative of x.

*The system is considered **Lyapunov stable** [9] if a function $V(x)$ can be constructed that satisfies the following three conditions simultaneously:*

1. Positive Definite: *$V(x) > 0$ for all $x \neq x^*$ where x^* is the equilibrium point.*

2. Zero at Equilibrium: *$V(x) = 0$ when $x = x^*$.*

3. Non-increasing Along Trajectories: *$\dot{V}(x) = \frac{dV}{dt} \leq 0$.*

*If $\dot{V}(x) < 0$ strictly, the equilibrium is considered **asymptotically stable.***

Control Lyapunov Function (CLF): While a Lyapunov Function is used to assess the stability of a system [19], a Control Lyapunov Function (CLF) is used in control design to ensure that a system can be made stable by finding an appropriate control input [23, 24].

Lemma 2. *Consider a controlled system with input u:*

$$\dot{x} = f(x) + g(x)u \tag{4}$$

where $f(x)$ represents the natural dynamics of the system and $g(x)$ represents how the control input u affects the system dynamics.

*A CLF $V(x)$ is used to ensure that an appropriate control input $u(x)$ can be found such that the system becomes stable. The CLF must satisfy the condition that for **each** state x, there exists a control input $u(x)$ such that:*

$$\dot{V}(x, u) = \frac{dV}{dt} = \nabla V(x) \cdot [f(x) + g(x)u] \leq 0 \tag{5}$$

This ensures that the function $V(x)$ is non-increasing over time, implying that the system is being stabilized by the control input u.

Quadratic Lyapunov Function (QLF): After investing into the general form and conditions for LF and CLF, now we introduce the Quadratic Lyapunov Function (QLF) which can be used as an actual implementation for the LF and CLF.

Lemma 3. *A Quadratic Lyapunov Function can be expressed in the following general format:*

$$V(\mathbf{x}) = \mathbf{x}^T P \mathbf{x} \tag{6}$$

where: $\mathbf{x}$ is the state vector. P is a symmetric positive definite matrix ($P = P^T$, and $P > 0$).

4 CNN Training Stability Analysis

In this section, we show how to apply Lyapunov Stability Analysis to assess and maintain stability during training.

4.1 CNN Dynamic System Modeling

We choose **loss** as the main system dynamics to invest in when analyzing stability for its nature in representing the state of a CNN during its training process.

By substituting the state vector defined in Equation (6) with the loss represented in Equation (2), we get the Lyapunov function for the dynamic system of CNN training with a focus on the loss (represented using the Jacobian matrix):

$$V(\mathbf{J}_t) = \mathbf{J}_t^T P \mathbf{J}_t \tag{7}$$

where: $\mathbf{J}_t = \nabla_{\mathbf{x}} L(\mathbf{x}_t)$.

Thus, the change in the Lyapunov function from time step t to $t+1$ can be expressed as:

$$\Delta V(\mathbf{J}_t) = V(\mathbf{J}_{t+1}) - V(\mathbf{J}_t) \tag{8}$$

4.2 Lyapunov Stability Analysis

Applying Lemma 1, a system is considered stable if the following three conditions are all satisfied: **positive definiteness**, **zero at equilibrium** and **Non-increasing behavior**. Now let's invest in each of the three conditions.

Zero at Equilibrium: The Lyapunov function would produce zero at equilibrium:

$$V(\mathbf{J}^*) = 0 \text{ where } \mathbf{J}^* \text{ is the Jacobian matrix at the equilibrium point } \mathbf{x}^* \tag{9}$$

We first invest in this second condition from Lemma 1. Such a condition is guaranteed to be satisfied.

Proof. In the context of CNN training, the equilibrium point occurs when the network predicts the output 100% correct in which case the loss and its gradient are all zero. Substitute such a Jacobian matrix into the construction of the Lyapunov function from Equation (7), we get:

$$V(\mathbf{J}^*) = (\mathbf{J}^*)^T P(\mathbf{J}^*) = \mathbf{0} \tag{10}$$

Hence the condition is satisfied. □

Positive Definiteness: The Lyapunov function should produce a positive value if not at equilibrium:

$$V(\mathbf{J}_t) > 0 \text{ for } \mathbf{J}_t \neq \mathbf{J}^* \tag{11}$$

Now we invest in the first condition from Lemma 1. Such a condition is guaranteed to be satisfied.

Proof. Based on the nature of equilibrium point in the context of CNN training, the loss $\mathbf{L}(\mathbf{x}_t) = 0$ and its gradient $\nabla_{\mathbf{x}}\mathbf{L}(\mathbf{x}_t) = 0$ when and only when $\mathbf{x_t} = \mathbf{x}^*$. Since $\mathbf{x} \neq \mathbf{x}^*$, $\mathbf{J}_t = \nabla_{\mathbf{x}} L(\mathbf{x}_t) > 0$.

Because P is a symmetric, positive definite matrix, as defined in Equation (7), the condition $V(\mathbf{J}_t) = \mathbf{J}_t^T P \mathbf{J}_t > 0$ is satisfied for any non-zero $\mathbf{J}_t$.

Thus this condition is also satisfied. □

Thus, the first two conditions from Lemma 1 are naturally satisfied by the nature CNN. Hence if the third condition is also satisfied, the dynamic system for CNN training is considered stable; if we can find a way to adjust some control input, such that for every state $\mathbf{x}$, the last condition **Non-increasing Behavior** is always satisfied, we have found a systematic way to **maintain** the stability of training a CNN.

Non-increasing Behavior: Now we invest in the third condition from Lemma 1. The change in the Lyapunov function should be non-positive:

$$\Delta V(\mathbf{J}_t) = V(\mathbf{J}_{t+1}) - V(\mathbf{J}_t) \leq 0 \tag{12}$$

It is obvious that such a condition cannot be always "naturally" satisfied due to all kinds of challenges that may be encountered during the training of CNNs(mentioned in Section 2.1). Therefore, we are interested in developing a systematic approach to maintaining the stability of CNN training.

4.3 Maintain CNN Training Stability

Theorem 1. *The dynamic system representing the training process of a CNN is considered* ***stable*** *when the learning rate η at the epoch t satisfies:*

$$0 \leq \eta_t \leq \frac{2\mathbf{H}_t P}{\mathbf{H}_t^T P \mathbf{H}_t} \tag{13}$$

where: $\mathbf{J}_t$ is the Jacobian matrix defined as $\mathbf{J}_t = \nabla_{\mathbf{x}} L(\mathbf{x}_t)$. $\mathbf{H}_t$ is the Hessian matrix defined as $\mathbf{H}_t = \nabla^2_{\mathbf{x}} L(\mathbf{x}_t)$. P is a symmetric positive definite matrix ($P = P^T$, and $P > 0$).

Proof. Given the gradient of the loss function and the transition mapping defined in Equation (3):

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_{\mathbf{x}} L(\mathbf{x}_t) = \mathbf{x}_t - \eta_t \mathbf{J}_t \tag{14}$$

The third condition from Lemma 1, as defined in Equation (12), can now be expressed as:

$$\mathbf{J}_{t+1}^T P \mathbf{J}_{t+1} - \mathbf{J}_t^T P \mathbf{J}_t \leq 0 \tag{15}$$

Since instability tends to arise when the learning rate η_t is relatively small, typically smaller than 1, the higher-order terms in the Taylor expansion which are proportional to powers of η_t, such as the second-order term $\nabla_{\mathbf{x}}^2 L(\mathbf{x}_t)$, have very little influence on the first-order term $\nabla_{\mathbf{x}} L(\mathbf{x}_t)$ and on the overall result. Hence we now approximate $\mathbf{J}_t$ using a first-order Taylor expansion around $\mathbf{x}_t$:

$$\mathbf{J}_{t+1} \approx \mathbf{J}_t - \eta_t \mathbf{H}_t \mathbf{J}_t \tag{16}$$

where $\mathbf{H}_t$ is the Hessian matrix defined as $\mathbf{H}_t = \nabla_{\mathbf{x}}^2 L(\mathbf{x}_t)$.
Now we substitute the expression for $\mathbf{J}_t$ into the inequality (15):

$$(\mathbf{J}_t - \eta_t \mathbf{H}_t \mathbf{J}_t)^T P (\mathbf{J}_t - \eta_t \mathbf{H}_t \mathbf{J}_t) - \mathbf{J}_t^T P \mathbf{J}_t \leq 0 \tag{17}$$

through expanding $\mathbf{J}_{t+1}^T P \mathbf{J}_{t+1}$:

$$\mathbf{J}_{t+1}^T P \mathbf{J}_{t+1} = \mathbf{J}_t^T P \mathbf{J}_t - 2\eta_t \mathbf{J}_t^T \mathbf{H}_t P \mathbf{J}_t + \eta_t^2 \mathbf{J}_t^T \mathbf{H}_t^T P \mathbf{H}_t \mathbf{J}_t \tag{18}$$

we get:

$$\mathbf{J}_t^T P \mathbf{J}_t - 2\eta_t \mathbf{J}_t^T \mathbf{H}_t P \mathbf{J}_t + \eta_t^2 \mathbf{J}_t^T \mathbf{H}_t^T P \mathbf{H}_t \mathbf{J}_t - \mathbf{J}_t^T P \mathbf{J}_t \leq 0 \tag{19}$$

which can be simplified as:

$$-2\mathbf{H}_t P + \eta_t \mathbf{H}_t^T P \mathbf{H}_t \leq 0 \tag{20}$$

Thus the learning rate must satisfy:

$$0 \leq \eta_t \leq \frac{2\mathbf{H}_t P}{\mathbf{H}_t^T P \mathbf{H}_t} \tag{21}$$

□

Usage of Theorem 1: Theorem 1 can be used to check whether the dynamic system for training CNNs is stable. Besides, once given how the learning rate can be updated dynamically, even if the system is currently not unstable, we can always keep updating the learning rate until it falls into the range $[0, \frac{2\mathbf{H}_t P}{\mathbf{H}_t^T P \mathbf{H}_t}]$. Thus we have found a systematic way to maintain the stability of the dynamic system.

5 Experiment

Theorem 1 provides a systematic method for ensuring the stability of CNN training by defining conditions for a stable learning rate. In this section, we conduct experiments to demonstrate the practical effectiveness of this criterion. Specifically, we apply the theorem to dynamically adjust the learning rate during CNN training and compare this approach with a traditional fixed learning rate method.

5.1 Learning Rate Adaptation Algorithm for Experiment

In this section, we propose Algorithm 1 that dynamically adapts the learning rate based on Theorem 1 for constructing the experiment. The adaptive mechanism leverages changes in the Lyapunov function $\Delta V(\mathbf{x}_t)$ to ensure stability throughout the training process. If instability is detected (i.e., $\Delta V(\mathbf{x}_t) > 0$), the learning rate is reduced incrementally until stability is restored. Conversely, when the system is stable (i.e., $\Delta V(\mathbf{x}_t) \leq 0$), the learning rate is increased to accelerate convergence while maintaining stability. This balance between reducing the learning rate during instability and increasing it when stable ensures that the model remains within the bounds specified by Theorem 1.

Algorithm 1 Learning Rate Adaptation Based on Lyapunov Stability

Input: Model parameters $\mathbf{x}_0$, initial learning rate η_0, control parameters α, β, minimum learning rate $\eta_{\min}$, maximum learning rate $\eta_{\max}$.
Output: Updated model parameters $\mathbf{x}_T$ after T iterations.

Define Lyapunov function $V(\mathbf{x}_t) = \mathbf{J}_t^\top P \mathbf{J}_t$ and stability criterion
for each training step $t = 1$ to T **do**
  Compute the gradient: $\mathbf{J}_t = \nabla_{\mathbf{x}} L(\mathbf{x}_t)$
  Compute the Lyapunov function value: $V(\mathbf{x}_t) = \mathbf{J}_t^\top P \mathbf{J}_t$
  Update the model parameters: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{J}_t$
  Compute the new gradient: $\mathbf{J}_{t+1} = \nabla_{\mathbf{x}} L(\mathbf{x}_{t+1})$
  Compute the Lyapunov function value: $V(\mathbf{x}_{t+1}) = \mathbf{J}_{t+1}^\top P \mathbf{J}_{t+1}$
  Compute the change in Lyapunov function values: $\Delta V(\mathbf{x}_t) = V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t)$
  while $\Delta V(\mathbf{x}_t) > 0 \Longleftrightarrow 0 \leq \eta_t \leq \frac{2\mathbf{H}_t P}{\mathbf{H}_t^T P \mathbf{H}_t}$ **do** ▷ Indicates instability
    Decrease learning rate: $\eta_{t+1} = \max\left(\frac{\eta_t}{1 + \alpha \Delta V(\mathbf{x}_t)},\ \eta_{\min}\right)$
    Recompute the gradient $\mathbf{J}_{t+1}$ and Lyapunov function $V(\mathbf{x}_{t+1})$
    Recompute the change in Lyapunov function values: $\Delta V(\mathbf{x}_t) = V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t)$
  end while

  if $\Delta V(\mathbf{x}_t) \leq 0$ **then** ▷ Indicates stability
    Increase learning rate: $\eta_{t+1} = \min\left(\eta_t * (1 + \beta),\ \eta_{\max}\right)$
  end if
end for
return Updated model parameters $\mathbf{x}_T$

The algorithm dynamically adjusts the learning rate based on the system's stability, as determined by Theorem 1. This mechanism ensures that training remains stable while maximizing the convergence speed.

5.2 Experimental Design

The experiments aim to assess the effectiveness of dynamic learning rate adjustment in CNN training. By applying Theorem 1, we expect faster convergence and improved generalization, while maintaining stability throughout training. We use two experimental groups: a control group employing Stochastic Gradient Descent (SGD) with a fixed learning rate, and an experimental group using SGD with a dynamic learning rate (SGD-DLR) adjusted according to Theorem 1.

Experimental Setup We conducted experiments on two datasets: CIFAR-10 and CIFAR-100. Data augmentation techniques, including random cropping and horizontal flipping, were applied to minimize over-fitting. The CNN model used (CustomCNN) consists of three convolutional layers with batch normalization and ReLU activations, followed by max-pooling and fully connected layers.

For the control group, we set the learning rate of SGD to a fixed value of 0.01. In the experimental group, the initial learning rate of SGD-DLR was also set to 0.01, and control parameters were set to $\alpha = 0.1$ and $\beta = 0.01$. The learning rate was dynamically bounded between 1×10^{-6} and 0.05.

Both models were trained for 50 epochs on CIFAR-10 and 100 epochs on CIFAR-100, with a batch size of 128 in each case. We tracked the evolution of training/validation loss/accuracy and the dynamic changes in the learning rate of SGD-DLR throughout the training process. In addition, we monitored the Lyapunov function to observe how stability evolved over iterations.

Results and Analysis The results from our experiments are illustrated in Figures 1 and 2, which display the performance comparison between SGD-DLR and fixed-learning-rate SGD when trained on CIFAR-10 and CIFAR-100, respectively.

Across both datasets, SGD-DLR demonstrated faster convergence and improved generalization when compared to basic SGD. The dynamic adjustment of the learning rate, guided by Lyapunov stability, ensured that the system maintained stability throughout training. This resulted in smoother convergence curves and higher accuracy in both CIFAR-10 and CIFAR-100.

The experiments confirm that the control-theoretic framework of Theorem 1 effectively stabilizes CNN training, while also accelerating convergence. The performance improvements observed in these experiments highlight the practical advantages of using this approach for dynamic learning rate adaptation.

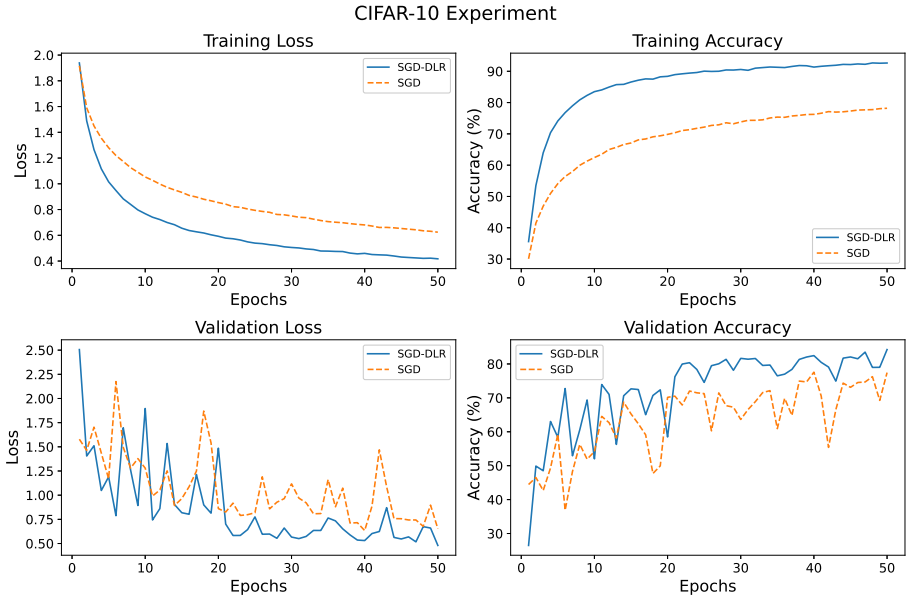

Fig. 1. Performance comparison between SGD-DLR and fixed-learning-rate SGD on CIFAR-10.

5.3 Experiment Conclusion

The experimental results show that using Theorem 1 to adapt the learning rate dynamically enhances CNN training stability and improves performance. When applied to the SGD-DLR, this method outperforms traditional fixed-learning-rate SGD by offering faster convergence and better generalization on both CIFAR-10 and CIFAR-100 datasets.

6 Conclusion

In conclusion, this paper introduces a novel approach to stabilizing CNN training by employing Lyapunov-based dynamic learning rate adaptation. By modeling CNN training as a dynamic control system and applying Lyapunov's method to evaluate its stability, we ensure that the learning rate dynamically adjusts in response to real-time stability metrics. This approach mitigates common training issues such as oscillations and divergence, leading to improved convergence speed and more stable learning. Our experiments demonstrate that this method, when applied to basic SGD, significantly enhances both training stability and accuracy, outperforming the traditional SGD on the CIFAR-10 and CIFAR-100 datasets.

However, this work opens up several directions for future research. One promising avenue is to extend the Lyapunov-based learning rate adjustment to more advanced optimizers like Adam or RMSprop, potentially yielding even

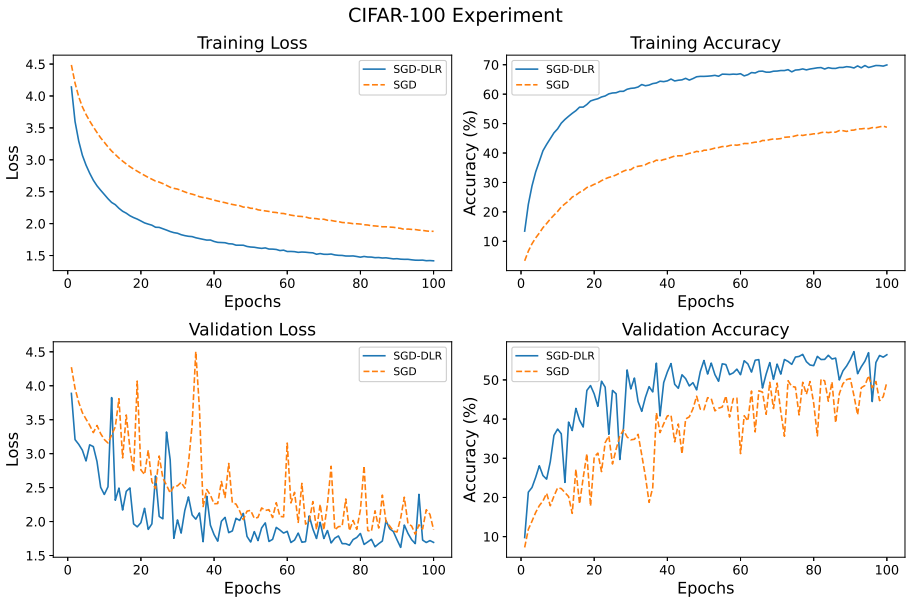

Fig. 2. Performance comparison between SGD-DLR and fixed-learning-rate SGD on CIFAR-100.

better results in scenarios involving non-convex loss surfaces. Additionally, future studies should evaluate the scalability of this approach to larger and more complex datasets, such as ImageNet[1] and larger models, such as VGGNet [26] and ResNet[25], to further validate its robustness. Another important direction would be to investigate the use of other control-theoretic concepts, such as adaptive control or robust control, to enhance the system's ability to handle noisy data or outliers during training.

Acknowledgments. This work is partially supported by the ARC Linkage Project (Grant Number: LP220200893).

References

1. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM (2017).
2. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998).
3. Hochreiter, S., Schmidhuber, J.: Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (1998).
4. Srivastava, N., Hinton, G., et al.: Dropout: A Simple Way to Prevent Neural Networks from over-fitting. Journal of Machine Learning Research (2014).

5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016).
6. Kingma, D. P., Ba, J.: Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (ICLR) (2014).
7. Tieleman, T., Hinton, G.: Lecture 6.5: RMSProp: Divide the Gradient by a Running Average of Its Recent Magnitude. COURSERA: Neural Networks for Machine Learning (2012).
8. Ruthotto, L., Haber, E.: Deep Neural Networks Motivated by Partial Differential Equations. Journal of Mathematical Imaging and Vision 62, 352–364 (2019)
9. Slotine, J. J., Li, W.: Applied Nonlinear Control. Prentice-Hall (1991).
10. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. Technical Report, University of Toronto (2009)
11. Sagun, L., Bottou, L., et al.: Explorations on the Hessian of Overparametrized Neural Networks. International Conference on Learning Representations (ICLR) (2014).
12. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning (ICML) (2015).
13. Brutzkus, A., Globerson, A., Malach, E., Shalev-Shwartz, S.: SGD Learns Over-Parameterized Networks that Provably Generalize on Linearly Separable Data. International Conference on Learning Representations (ICLR) (2018).
14. Lyapunov, A. M.: The General Problem of the Stability of Motion. Taylor & Francis (1992).
15. Zuo, Z., Yang, C., Wang, Y.: A New Method for Stability Analysis of Recurrent Neural Networks With Interval Time-Varying Delay. IEEE Transactions on Neural Networks 21(2), 339-344 (2010).
16. Massaroli, S., Poli, M., et al.: Dissecting Neural ODEs. Advances in Neural Information Processing Systems (2021).
17. Chen, M., Lam, H.K., Shi, Q., Xiao, B.: Reinforcement Learning-Based Control of Nonlinear Systems Using Lyapunov Stability Concept and Fuzzy Reward Scheme. IEEE Transactions on Circuits and Systems II: Express Briefs 67(10), 2059-2063 (2020).
18. Dai, H., Landry, B., Yang, L., Pavone, M., Tedrake, R.: Lyapunov-stable neural-network control. arXiv preprint arXiv:2109.14152 (2021).
19. Khalil, H.K.: Nonlinear Systems, 3rd edn. Prentice Hall, Upper Saddle River (2002).
20. Sastry, S.: Nonlinear Systems: Analysis, Stability, and Control. Springer, New York (1999).
21. Hirsch, M.W., Smale, S., Devaney, R.L.: Differential Equations, Dynamical Systems, and an Introduction to Chaos, 3rd edn. Academic Press, Waltham (2012).
22. Haber, E., Ruthotto, L.: Stable architectures for deep neural networks. Inverse Problems 34(1), 014004 (2017).
23. Sontag, E.D.: A 'universal' construction of Artstein's theorem on nonlinear stabilization. Systems & Control Letters 13(2), 117-123 (1989).
24. Freeman, R.A., Kokotović, P.V.: Robust Nonlinear Control Design: State-Space and Lyapunov Techniques. Birkhäuser, Boston (1996).
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016).
26. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556 (2014).