Super-level Sets and Exponential Decay: A Synergistic Approach to Stable Neural Network Training

Jatin Chaudhary, University of Turku, Finland
Dipak Nidhi, University of Turku, Finland
Jukka Heikkonen, University of Turku, Finland
Harri Merisaari, University of Turku, Finland
Rajeev Kanth, Savonia University of Applied Sciences, Finland

This paper presents a theoretically grounded optimization framework for neural network training that integrates an Exponentially Decaying Learning Rate with Lyapunov-based stability analysis. We develop a dynamic learning rate algorithm and prove that it induces connected and stable descent paths through the loss landscape by maintaining the connectivity of super-level sets $S_\lambda = \{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \geq \lambda\}$. Under the condition that the Lyapunov function $V(\theta) = \mathcal{L}(\theta)$ satisfies $\nabla V(\theta) \cdot \nabla \mathcal{L}(\theta) \geq 0$, we establish that these super-level sets are not only connected but also equiconnected across epochs, providing uniform topological stability. We further derive convergence guarantees using a second-order Taylor expansion and demonstrate that our exponentially scheduled learning rate with gradient-based modulation leads to a monotonic decrease in loss. The proposed algorithm incorporates this schedule into a stability-aware update mechanism that adapts step sizes based on both curvature and energy-level geometry. This work formalizes the role of topological structure in convergence dynamics and introduces a provably stable optimization algorithm for high-dimensional, non-convex neural networks.

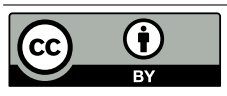
JAIR Associate Editor: Bo Han

JAIR Reference Format:
Jatin Chaudhary, Dipak Nidhi, Jukka Heikkonen, Harri Merisaari, and Rajeev Kanth. 2025. Super-level Sets and Exponential Decay: A Synergistic Approach to Stable Neural Network Training. *Journal of Artificial Intelligence Research* 1, Article 21 (July 2025)

1 Introduction

There has been significant progress towards the development and deployment of neural network models. The deployment of a neural network model demands high accuracy and precision, and hyperparameter optimization plays an important role towards building such a model. The researchers' community has been actively analyzing learning rates and loss functions to make the network more stable across datasets, and prevent overfitting [1][2][3]. Optimizing neural networks involves minimizing a complex and often non-convex loss function over a high-dimensional parameter space. These non-convex landscapes present significant challenges as gradient-based methods can become trapped in suboptimal regions of the loss landscape, often driving the training towards an overfitting scenario. In the following sections, we delve into the mathematical foundations that link dynamic learning rates with super-level sets, and loss function crucial to understanding stability and convergence in

Authors' Contact Information: Jatin Chaudhary, ORCID: 0000-0002-4139-5315, jatin.chaudhary@utu.fi, University of Turku, Turku, Finland; Dipak Nidhi, ORCID: 0000-0002-1040-5007, dipak.nidhi@utu.fi, University of Turku, Turku, Finland; Jukka Heikkonen, ORCID: 0000-0002-2468-5708, jukka.heikkonen@utu.fi, University of Turku, Turku, Finland; Harri Merisaari, ORCID: 0000-0002-8515-5399, haanme@utu.fi, University of Turku, Turku, Finland; Rajeev Kanth, ORCID: 0000-0003-1109-1211, Rajeev.Kanth@savonia.fi, Savonia University of Applied Sciences, Kuopio, Finland.



DOI: 10.1613/jair.1.17272

neural network training. We will explore how adaptive learning rates, particularly those with exponential decay, systematically influence the optimization landscape. This discussion aims to bridge theoretical insights with practical strategies, enhancing both the efficacy and understanding of neural network training in local minima or saddle points [4]. Despite these difficulties, substantial progress has been made in understanding and enhancing optimization trajectories in neural networks. Recent theoretical advancements have highlighted concepts like 'loss landscape smoothing' and adaptive gradient methods,' indicating that certain learning rate configurations can improve optimization conditions [5].

Neural network training presents multiple challenges, particularly in optimizing the learning rate, managing the loss function, ensuring stability, and preventing overfitting. The learning rate is a critical parameter that dictates the step size during gradient descent. An inappropriate learning rate can lead to slow convergence or even divergence. The loss function, which measures the discrepancy between predicted and actual outputs, often has a complex landscape that can trap optimization algorithms in local minima [6]. Stability is another crucial aspect, as unstable training can lead to erratic updates and poor model performance. Overfitting, where the model performs well on training data but poorly on unseen data, remains a persistent problem. Existing solutions include adaptive learning rates and regularization techniques, but they often fall short in ensuring consistent stability and avoiding overfitting [7]. Our study addresses these issues by proposing a novel approach that integrates dynamic learning rates with stability principles from control theory.

Our primary contribution is the development of an algorithm that dynamically adjusts the learning rate using an exponential decay model, integrated with principles from Lyapunov stability [8]. This approach ensures consistent convergence by maintaining the connectivity of super-level sets of the loss function. We demonstrate that these super-level sets remain connected under our algorithm, preventing the optimization process from becoming trapped in poor local minima and ensuring stable descent paths [9]. This connectedness facilitates smoother transitions across the loss landscape, enhancing training dynamics and generalization capabilities. By embedding these concepts into our algorithmic framework, we achieve more stable and efficient optimization, addressing common challenges such as overfitting and instability. This work not only advances theoretical understanding but also provides a foundation for practical applications in neural network training, paving the way for further research into dynamic learning rate adjustments and their impact on training stability and efficacy [10].

In the following sections, we present the mathematical foundations, and further link dynamic learning rates with the super-level set based loss function, crucial for understanding stability and convergence in neural network training. We investigate how exponentially decaying learning rates restructure the optimization landscape to preserve stability. We discuss the stability of using adaptive learning rates with super-level set based loss function so to solidify our claims.

2 Mathematical Underpinnings

The super-level sets $S_\lambda = \{\mathbf{x} \in \mathbb{R}^n : L(\mathbf{x}) \geq \lambda\}$ reveal important stability and convergence properties for gradient-based optimization methods [11]. An Exponentially Decaying Learning Rate, defined by $\eta(t) = \eta_0 e^{-\alpha(t)}$, where η_0 is the initial rate and α a positive decay constant, is beneficial. It allows for quick initial progress by using a higher initial rate, guiding the optimizer towards important areas quickly [12]. As training proceeds, this rate gradually decreases, allowing for more precise adjustments and preventing common issues like overshooting minima [13]. This dynamic rate adjustment, when coupled with the structure of super-level sets, offers insights into the training's stability by ensuring the optimization path remains connected and stable through the topology of the landscape [7]. By adopting a Lyapunov function $V(\mathbf{x})$ that decreases along these paths, we enforce stability and keep the system's energy diminishing, keeping the optimization within stable parameter regions [8]. Together, these elements create a robust framework that deepens our understanding of the dynamics in neural network

Table 1. Summary of Notations Used Throughout the Paper

Symbol	Description
θ	Trainable parameters of the neural network
$\mathcal{L}(\theta)$	Global Loss function (cross-entropy unless stated)
f	Per-sample loss
$V(\theta)$	Lyapunov function (set to $\mathcal{L}(\theta)$ for analysis)
$\nabla\mathcal{L}(\theta)$	Gradient of loss with respect to parameters
$\alpha(t)$	Exponentially decaying learning rate at epoch t
η_t	Norm-scaled dynamic learning rate
S_λ	Superlevel set: $\{\theta : \mathcal{L}(\theta) \geq \lambda\}$
g_t	Gradient at iteration t (i.e., $g_t = \nabla_\theta \mathcal{L}(\theta_t)$)
x_t	Iteration-dependent point in parameter space (used interchangeably with θ_t)
Δ_t	Accumulated shift from reference point x_{ref}
m_t, v_t, r_t	First and second moment estimates for gradient, variance, and rate
β	Vector Moment decay
λ	Hyperparameter controlling exponential decay
x_{BASE}	Base model initialization
R_{robust}	Robustness-weighted empirical risk incorporating class weights and robustness parameter
J_{dynamic}	Time-adaptive dynamic cost function
$\gamma(t)$	Temporal modulation factor

training and highlights the significance of careful tuning of hyperparameters in managing complex optimization scenarios [10].

To understand the concept better, consider a ball that rolls down a hilly terrain towards a valley, representing the minimum of a loss landscape. Initially, the ball is given a strong push (high initial learning rate η_0) allowing it to quickly descend from higher elevations (higher loss values in super-level sets S_λ). Each super-level set corresponds to a range of elevations where the ball's potential energy (analogous to the loss value in the neural network) remains above a certain threshold λ. As the ball descends from higher altitudes to lower ones, it transitions from one super-level set to another, each with decreasing minimum energy thresholds. As it approaches the valley, the slope (gradient) lessens and so does the ball's speed due to the exponential decay of the push force $[\eta(t) = \eta_0 e^{-\alpha(t)}]$, preventing it from overshooting the valley. This gradual slowing is critical as it ensures that the ball can finely adjust its path to settle in the deepest part of the valley, analogous to achieving the most optimal parameters in a neural network training scenario. This model demonstrates how the dynamic learning rate and the structure of the super-level sets interact, ensuring that the optimization path remains stable and connected throughout the descent. This is analogous to how the ball consistently follows a path that leads it towards the valley without getting stuck or veering off course.

3 Fundamental Concepts

The parameter vector θ has the network's weights and biases, and is an integral part for the network's learning, as it is meticulously adjusted to minimize divergences between predicted outputs and actual targets [14]. This adjustment process is governed by the learning rate $\alpha(t)$, a parameter that determines the step size within the parameter space during optimization, thus directly influencing convergence quality [15]. The gradient of the loss function, $\nabla_\theta \mathcal{L}(\theta)$, serves as the navigational guide for updating parameters, towards optimal solutions [16]. The

interplay between the learning rate and the gradient is vital for maintaining systematic progression and ensuring that the training remains on a stable and effective path [17]. This setup forms the backbone of our approach to enhancing neural network training, laying the groundwork for a deeper exploration of optimization dynamics mathematically [18].

3.1 Mathematical Drawouts

Behind our study is a probabilistic model that views the neural network as an intricate function approximating the conditional probability distribution $P(Y \mid X; \theta)$. In classification tasks, this relationship is mathematically expressed through the softmax function:

$$P(Y = c \mid X; \theta) = \frac{\exp\left(f_c(X; \theta)\right)}{\sum_{j=1}^{C} \exp\left(f_j(X; \theta)\right)}, \tag{1}$$

where $f_c(X; \theta)$ represents the network output for class c, and C denotes the total number of classes [19]. This formulation is essential in demonstrating how our model probabilistically classifies input data into defined output classes.

Building on this framework, we derive a likelihood function reflecting the probability of observing our training dataset $\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$ under the model parameters θ :

$$\mathcal{L}(\theta; \mathcal{D}) = \prod_{i=1}^{m} P\left(y^{(i)} \mid x^{(i)}; \theta\right). \tag{2}$$

This likelihood function for quantifying how well the model aligns with empirical data, setting the stage for parameter optimization via Bayesian inference [20].

Incorporating Bayesian principles, we consider the posterior probability of the parameters θ given the data $\mathcal{D}$, calculated as follows:

$$P(\theta \mid \mathcal{D}) \propto \mathcal{L}(\theta; \mathcal{D}) P(\theta), \tag{3}$$

where $P(\theta)$ denotes the prior distribution over the parameters [21]. This Bayesian framework facilitates a comprehensive parameter optimization strategy, harmonizing empirical data adaptation with existing parameter knowledge.

The culmination of this probabilistic modeling leads to the optimization phase within a gradient descent framework, where our methodology involves iteratively minimizing the negative log-posterior:

$$-\log P(\theta \mid \mathcal{D}) = -\log \mathcal{L}(\theta; \mathcal{D}) - \log P(\theta) + \text{ const.} \tag{4}$$

Here, the gradient descent update rule is critical:

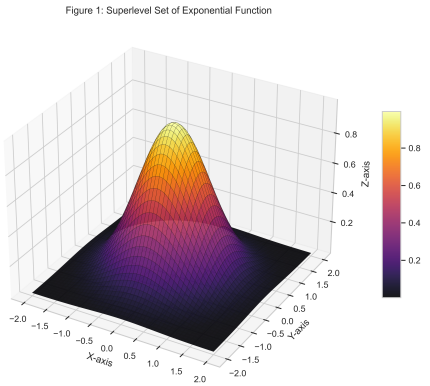
$$\theta_{t+1} = \theta_t - \alpha(t) \nabla_\theta \left[-\log P\left(\theta_t \mid \mathcal{D}\right)\right], \tag{5}$$

where $\alpha(t)$ is the learning rate, dynamically adapting to ensure efficient convergence and stability of the model [15].

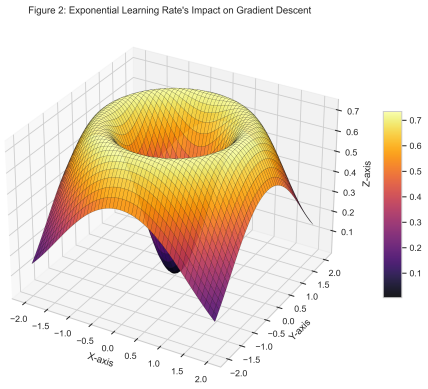
Importantly, the dynamic adjustment of $\alpha(t)$ profoundly impacts the topology of the loss function's super-level sets $S_\lambda = \{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \geq \lambda\}$, which are instrumental in understanding the stability and connectivity of the optimization landscape [4]. By ensuring that these sets remain connected, the algorithm promotes a smoother and more stable descent towards the global minima, effectively navigating the complex, high-dimensional parameter spaces typical of deep learning tasks[22].

This integration of probabilistic modeling, Bayesian inference, and gradient optimization leverages the theoretical insights into super-level sets to enhance the practical outcomes of neural network training. This approach

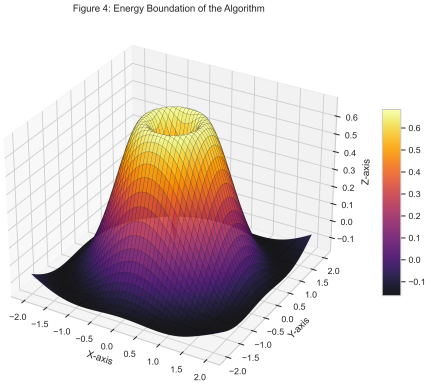
ensures both theoretical robustness and empirical efficacy, highlighting our model's capacity to navigate and optimize within intricate, probabilistically defined landscapes.



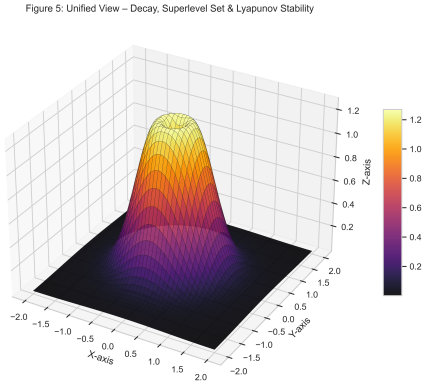
Superlevel Set of Exponential Function



Exponential Learning Rate's Impact



Energy Boundation of the Algorithm



Unified View: Decay, Superlevel Set & Stability

Fig. 1. **Geometric and Energetic Intuition Behind the Proposed Optimization Framework.** This 2×2 composite figure presents key geometric and dynamic principles central to our theoretical framework. **(a)** The surface of $f(x, y) = \exp(-x^2-y^2)$ depicts the loss landscape, where level contours form smoothly connected superlevel sets S_λ. **(b)** The exponential decay in learning rate $\alpha(t)$ modulates the influence of steep curvature regions, effectively scaling updates in line with gradient norm. **(c)** The Lyapunov-inspired energy profile demonstrates how bounded update energy suppresses divergence, stabilizing convergence across iterations. **(d)** Finally, the unified visualization integrates exponential decay, superlevel containment, and energy boundation showing how our algorithm balances descent efficiency with theoretical stability. Together, these views provide a visual bridge between our mathematical claims (Sections 4–6) and their algorithmic implications.

3.2 Exponentially Decaying Learning Rate

The formulation of an Exponentially Decaying Learning Rate (derivation in the supplementary) given by

$$\frac{d\alpha}{dt} = -\alpha_0 \beta e^{-\beta t}, \tag{6}$$

influences the topology of the loss function's super-level sets $S_\lambda = \{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \geq \lambda\}$. The dynamically adjusted learning rate ensures that these sets remain connected, supporting a stable and cohesive optimization trajectory [12]. Within the gradient descent framework, this leads to an adapted parameter update rule

$$\theta_{t+1} = \theta_t - \alpha_0 e^{-\beta t} \nabla_\theta \left[-\log P\left(\theta_t \mid \mathcal{D}\right) \right], \tag{7}$$

effectively illustrating the integration of an exponential decay learning rate within the gradient descent mechanism [15]. This methodical approach not only enhances the theoretical underpinnings of our optimization strategy but also significantly boosts its practical efficacy. By marrying the theoretical concepts of exponential decay with gradient descent, our approach fosters training dynamics that effectively navigate the complex, high-dimensional spaces typical of deep learning tasks [7]. This novel integration offers a rigorous, theoretically informed enhancement to the conventional training paradigms, ensuring that both the stability and the efficiency of the learning process are maximized [9].

3.2.1 Dynamic Cost Function. In our study, we refined our dynamic cost function to adeptly integrate principles from statistical learning theory, with an emphasis on addressing class imbalances and evolving training requirements. The empirical risk, $R_{\text{emp}}(\theta)$, is meticulously calculated as

$$\frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; \theta)), \tag{8}$$

where we incorporate class weights w_c to balance the influence of underrepresented classes, resulting in

$$R_{\text{emp}}^{\text{cw}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} w_{y_i} L(y_i, f(x_i; \theta)) \tag{9}$$

This weighting corrects training biases, enhancing model fairness and accuracy particularly in scenarios with skewed class distributions [23]. To manage outliers and enhance robustness, we introduce a robustness parameter ρ, which modifies the loss contribution based on the confidence in data point correctness:

$$R_{\text{robust}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \rho(y_i, x_i) w_{y_i} L(y_i, f(x_i; \theta)) \tag{10}$$

[24]. Regularization is integral to this framework, implemented through $\Omega(\theta)$, employing either L_1 or L_2 regularization to mitigate overfitting. The regularized empirical risk is articulated as

$$R_{\text{emp}}^{\text{reg}}(\theta) = R_{\text{robust}}(\theta) + \lambda \Omega(\theta) \tag{11}$$

[25]. Our dynamic cost function is characterized by a temporal modulation factor $\gamma(t) = 1 + \kappa e^{-\delta t}$, which strategically transitions from aggressive initial learning to increased regularization as training advances [26]. This modulation ensures the learning rate evolves with the model's needs, reducing to prevent overfitting as the model refines its parameters. The gradient of the loss function, $\nabla_\theta \mathcal{L}(\theta)$, directs parameter updates and is essential for navigating both the explicit regions, where gradients are large and clear, facilitating straightforward descent steps, and the implicit regions, where gradients may vanish, requiring the adaptive $\gamma(t)$ and robustness enhancements to maintain meaningful and stable updates [15]. For instance, in scenarios with imbalanced datasets, class weights w_c counteract the bias towards predominant classes, and $\gamma(t)$'s increasing regularization later in training smooths the model's fit to emphasize generalization. This framework,

$$\mathcal{J}_{\text{dynamic}}\left(\theta; \mathcal{D}, t\right) = \gamma(t) \mathcal{J}_{\text{reg}}\left(\theta; \mathcal{D}\right), \tag{12}$$

not only deepens our understanding of dynamic learning rate mechanisms but also fosters a coherent and stable optimization process, adaptable to complex data landscapes and advancing adaptive machine learning methodologies [27].

3.3 Gradient Descent

Integrating level set dynamics into the gradient descent framework is proposed to navigate the complex topology of the loss function more efficiently. While traditional gradient descent updates parameters iteratively with the rule $\theta_{t+1} = \theta_t - \alpha(t)\nabla_\theta \mathcal{L}(\theta_t)$, where $\alpha(t) = \alpha_0 e^{-\beta t}$ is an Exponentially Decaying Learning Rate, emerging research suggests enhancements to this approach to address its limitations in stability and adaptability. Zhang et al. (2019) propose an Adaptive Exponential Decay Rate (AEDR), which dynamically adjusts the decay rate based on moving averages of gradients, thus offering a more responsive adaptation to the learning needs over different training phases and potentially leading to improved convergence rates [28].

Further, Mishra and Ghosh (2019) highlight the advantages of a variable gain gradient descent, which modulates the learning rate based on error metrics and system states to enhance both the convergence speed and stability, suggesting a potential direction for refining level set dynamics integration [29]. Additionally, the link between generalization and dynamical robustness presented by Kozachkov et al. (2023) through Riemannian contraction indicates that ensuring algorithmic stability through the optimization dynamics could directly influence generalization performance, advocating for a deeper theoretical integration of level set dynamics with gradient descent methods [30].

To optimize these methods further, incorporating continuous time analysis as suggested by Kovachki and Stuart (2021) could provide more nuanced insights into the efficacy of momentum and modifications in traditional gradient descent, thus enhancing the strategy to navigate complex loss landscapes more effectively [31]. Hereby, presenting a refined method that enhances theoretical understanding and significantly improves the practical application of neural network training in complex and high-dimensional problem spaces.

4 Dynamic Learning Rates and Super-level Sets

Theorem: L is continuously differentiable and V provides a stability guarantee such that

$$\nabla V(\mathbf{x}) \cdot \nabla L(\mathbf{x}) \geq 0 \; for \; all \; \mathbf{x} \in \mathbb{R}^n \tag{13}$$

Then, the super-level sets S_λ are connected for all λ under the dynamic learning rate η.

In neural network optimization, the topology of the loss function $L : \mathbb{R}^n \to \mathbb{R}$ significantly influences algorithmic behavior and convergence. We have studied the properties of super-level sets, which are crucial in understanding the dynamic adjustments of our gradient-based learning methods. These sets maintain a stable and efficient learning path, enhanced by adaptive learning rates modulated through a Lyapunov function $V(\mathbf{x})$, which aligns the gradient flow to ensure consistency across training iterations [4]. By ensuring that $V(\mathbf{x})$ decreases along the trajectory of the learning process reflecting a decline in the system's energy the gradient updates are systematically adjusted to prevent oscillations and divergences, thus resulting in smoother convergence[28].

Going further, we define a super-level set's connectivity by the existence of a continuous path $\gamma : [0, 1] \to S_\lambda$ connecting any two points $\mathbf{x}, \mathbf{y}$ within the set, ensuring comprehensive exploration of the parameter space. The learning rate adjustment,

$$14\eta(\mathbf{x}(t)) = 1/(1 + \|\nabla L(\mathbf{x}(t))\|) \tag{14}$$

further tuning it with the update rule,

$$\mathbf{x}(t+1) = \mathbf{x}(t) - \eta(\mathbf{x}(t))\nabla L(\mathbf{x}(t)) \tag{15}$$

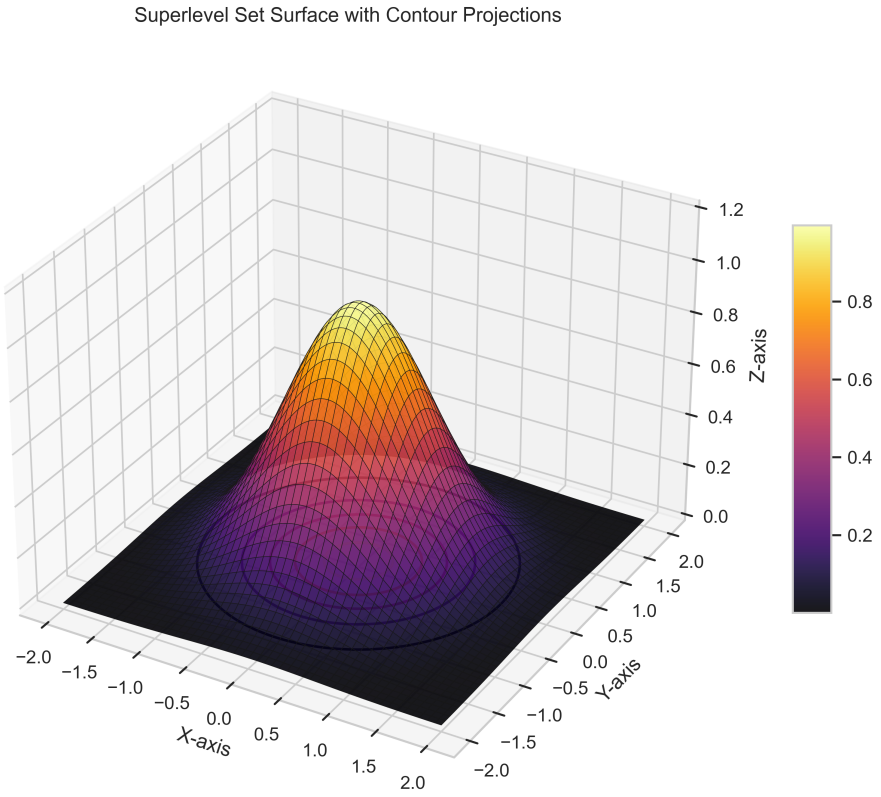


Fig. 2. **Superlevel Set Surface with Contour Projections for the Exponential Function.** This figure illustrates a three-dimensional surface plot of the function $f(x, y) = \exp(-x^2 - y^2)$, augmented with projected contour lines onto the xy-plane corresponding to discrete superlevel thresholds $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The **X-axis** and **Y-axis** span the input domain $x, y \in [-2, 2]$, representing a two-dimensional slice of the parameter space of a neural network or a simplified loss landscape. The **Z-axis** encodes the scalar output of the function $z = f(x, y)$, which in this case serves as a proxy for the loss function $\mathcal{L}(\theta)$. The surface exhibits a Gaussian-like shape peaking at the origin, with height $z = 1$, and decaying rapidly as the distance from the origin increases. The projected contours delineate the structure of *superlevel sets* $S_\lambda = \{(x, y) \in \mathbb{R}^2 \mid f(x, y) \geq \lambda\}$, which are nested, closed, and connected regions around the global minimum. Each contour corresponds to a boundary where the loss is held constant at a threshold λ, illustrating how descent trajectories if constrained to remain within these regions maintain stability. This visualization offers geometric intuition for the theorem presented in Section 4, demonstrating that exponential decay in loss forms concentric "valleys" that align naturally with connected superlevel sets. In the context of our algorithm, such structures are crucial. The parameter updates driven by gradient descent, modulated by an Exponentially Decaying Learning Rate, remain topologically contained within these sets, preserving Lyapunov stability. As a result, the figure encapsulates how our method exploits the natural geometry of the loss surface to enforce bounded, smooth, and theoretically sound optimization dynamics.

This design decreases the learning rate as the gradient norm increases, thereby updating the step sizes near equilibrium states where gradients are typically larger [15]. This adaptation is important for managing the trajectory's stability and ensuring effective convergence within the complex landscape of the loss function [7].

To analyze convergence, we employ a Taylor expansion of L around $\mathbf{x}(t)$, leading to an approximation expressed as

$$L(\mathbf{x}(t+1)) \approx L(\mathbf{x}(t)) - \nabla L(\mathbf{x}(t))^T (\mathbf{x}(t+1) - \mathbf{x}(t)) \tag{16}$$

which upon substituting the update rule, transforms into

$$L(\mathbf{x}(t+1)) \approx L(\mathbf{x}(t)) - \nabla L(\mathbf{x}(t))^T (-\eta(\mathbf{x}(t))\nabla L(\mathbf{x}(t))) = L(\mathbf{x}(t)) + \eta(\mathbf{x}(t))\|\nabla L(\mathbf{x}(t))\|^2 \tag{17}$$

With $\eta(\mathbf{x}(t)) = 1/(1 + \|\nabla L(\mathbf{x}(t))\|)$, this equation further simplifies to

$$L(\mathbf{x}(t+1)) \approx L(\mathbf{x}(t)) - \frac{\|\nabla L(\mathbf{x}(\mathbf{t}))\|^2}{1 + \|\nabla L(\mathbf{x}(\mathbf{t}))\|} \tag{18}$$

illustrating that $L(\mathbf{x}(t+1)) \leq L(\mathbf{x}(t))$, confirming that the loss decreases with each update provided $\nabla L(\mathbf{x}(t)) \neq 0$, affirming convergence [12].

To analyze local convergence, consider a second-order Taylor expansion of L around a point $\mathbf{x}(t) \in \mathbb{R}^n$:

$$L(\mathbf{x}(t+1)) = L(\mathbf{x}(t)) + \nabla L(\mathbf{x}(t))^\top (\mathbf{x}(t+1) - \mathbf{x}(t)) + \frac{1}{2}(\mathbf{x}(t+1) - \mathbf{x}(t))^\top \nabla^2 L(\xi)(\mathbf{x}(t+1) - \mathbf{x}(t)) \tag{19}$$

for some ξ on the line segment between $\mathbf{x}(t)$ and $\mathbf{x}(t+1)$. Substituting the update rule yields:

$$\mathbf{x}(t+1) = \mathbf{x}(t) - \eta(\mathbf{x}(t))\nabla L(\mathbf{x}(t)) \tag{20}$$

$$\begin{aligned} L(\mathbf{x}(t+1)) \approx\ & L(\mathbf{x}(t)) - \eta(\mathbf{x}(t))\|\nabla L(\mathbf{x}(t))\|^2 \\ & + \frac{1}{2}\eta^2(\mathbf{x}(t))\nabla L(\mathbf{x}(t))^\top \nabla^2 L(\xi)\nabla L(\mathbf{x}(t)) \end{aligned} \tag{21}$$

If $\nabla^2 L(\xi)$ is positive semi-definite, the third term is non-negative, and the total change in loss is still guaranteed to be non-positive, provided $\eta(\mathbf{x}(t))$ is sufficiently small. Substituting $\eta(\mathbf{x}(t))$ with 14, we obtain:

$$L(\mathbf{x}(t+1)) \leq L(\mathbf{x}(t)) - \frac{\|\nabla L(\mathbf{x}(t))\|^2}{1 + \|\nabla L(\mathbf{x}(t))\|} + O(\eta^2) \tag{22}$$

which confirms that the loss decreases in expectation as long as the gradient norm is non-zero [12].

4.1 Boundary Case Considerations

Let us now examine the edge conditions:

(1) **Flat Region:** Suppose $\nabla L(\mathbf{x}) = 0$. Then, $\eta(\mathbf{x}) = 1$, and the update rule results in no movement. The point $\mathbf{x}$ remains stationary, satisfying convergence conditions.
(2) **Sharp Minima:** When $\|\nabla L(\mathbf{x})\| \to \infty$, the learning rate $\eta(\mathbf{x}) \to 0$, which causes the step size to vanish. This gracefully avoids overshooting and ensures that large gradients do not destabilize training.
(3) **Boundary Points of S_λ:** Consider $\mathbf{x} \in \partial S_\lambda$ such that $L(\mathbf{x}) = \lambda$. Since L is differentiable and $\nabla L(\mathbf{x}) \cdot \nabla V(\mathbf{x}) \geq 0$, a gradient step either keeps $\mathbf{x}(t+1) \in S_\lambda$ or reduces L, maintaining connectivity in a compact sub-level topology.

This mathematical framework ensures that the dynamic learning rate not only supports the connectivity of super-level sets S_λ but also enhances the overall integrity of the training process by providing stable, and gradual adjustments in response to the landscape of the loss function. This approach is essential for ensuring that the training remains across varying topologies and achieves reliable convergence [32].

5 Stability and Convergence Analysis with Lyapunov Stability Theory

In neural network optimization, employing the loss function $L(\theta)$ as a Lyapunov function enriches the stability and convergence analysis, leveraging its properties like positive definiteness and radial unboundedness to gauge network performance and systemic stability. This setup allows for monitoring stability through the non-increasing nature of the loss function over time, indicated by $\frac{dV}{dt} \leq 0$, suggesting that perturbations in parameter values do not escalate loss values, thereby aiding convergence towards equilibrium, typically a local minimum. The introduction of level sets L_λ and super-level sets S_λ deepens the understanding of the optimization landscape, mapping areas where the loss function meets or surpasses specific thresholds and examining how updates navigate these regions. The differential inequality analysis further underscores this, showing consistent loss minimization and the benefits of an Exponentially Decaying Learning Rate, $\alpha(t) = \alpha_0 e^{-\beta t}$, which manages the magnitude of parameter updates to prevent overshooting and enhance stability [12]. This comprehensive approach, integrating Lyapunov's stability theory with level set dynamics and differential inequality, offers theoretical and practical insights to ensure a stable, connected path through optimal regions of the loss landscape, emphasizing the need for empirical validation to confirm these theoretical constructs in real-world applications.

The classical concept of a Lyapunov function $V(\theta)$ proves potent in many theoretical analyses but requires adaptation to manage the discontinuities typical of non-Lipschitz activations. To address this, we extend the traditional Lyapunov stability framework to accommodate the irregularities that these functions introduce into the training dynamics. Traditionally, the loss function $\mathcal{L}(\theta)$ itself serves as a natural choice for the Lyapunov function $V(\theta)$ in neural networks. This choice is predicated on its inherent properties i.e. Positive Definiteness, $V(\theta) > 0$ for all $\theta \neq \theta^*$, Radial Unboundedness, $V(\theta)$ increases without bound as $\|\theta\|$ approaches infinity, Zero at Minimum, $V(\theta^*) = 0$, where θ^* is typically a local or global minimum [14].

Given these properties, $\mathcal{L}(\theta)$ effectively tracks the stability of the system. However, when dealing with non-Lipschitz activations, the gradient $\nabla_\theta \mathcal{L}(\theta)$ may not exist everywhere or may exhibit discontinuities. To handle this, a generalized Lyapunov approach is employed, where we consider generalized gradients or subderivatives when standard derivatives do not exist [33].

For neural networks utilizing non-Lipschitz activations, the derivative of the Lyapunov function along the system trajectories, represented by the parameter update rules, must consider possible discontinuities:

$$\frac{dV}{dt} \approx \nabla_\theta V(\theta) \cdot \frac{d\theta}{dt}, \tag{23}$$

, where $\frac{d\theta}{dt}$ is modeled as $-\alpha(t)\nabla_\theta \mathcal{L}(\theta)$, accounting for the possibly generalized gradient $\nabla_\theta \mathcal{L}(\theta)$. Here, $\alpha(t)$ denotes the learning rate, which may follow an exponential decay model to temper the training updates [15].

Incorporating a generalized gradient ensures that the analysis remains valid even in the presence of activation functions that do not meet the smoothness criteria typically required for conventional gradient descent methods. This approach aligns with findings from Forti et al. (2006), highlighting the necessity of stability measures that can adapt to the irregularities intrinsic to advanced neural network configurations.

This generalized Lyapunov stability analysis is critical not only from a theoretical perspective but also for practical implementation in neural networks that employ advanced activation functions like ReLU, leaky ReLU, or others that exhibit non-Lipschitz behavior. Ensuring that $\frac{dV}{dt} \leq 0$ across all training iterations confirms that the network is converging towards a stable state, minimizing the loss effectively despite the potential challenges posed by the activation functions [14].

6 Algorithm

Input: - **Base algorithm (BASE)**: Initial training algorithm. - $\beta \in [0, 1]^6$: Decay factors for moment estimates (default $\beta = (0.9, 0.99, 0.999, 0.9999, 0.99999, 0.999999)$). - $\lambda \in \mathbb{R}$: Learning rate decay parameter (default $\lambda = 0.01$).

- $s_{\text{init}} \in \mathbb{R}$: Initial non-zero value for stabilizing updates (default $s_{\text{init}} = 10^{-8}$). - $\epsilon = 10^{-8}$: Small value for numerical stability.

Output: - Optimized model parameters θ.

Procedure: 1. **Initialize variables:** - $v_0 \leftarrow 0$ (initialize momentum vector), - $r_0 \leftarrow 0$ (initialize rate vector), - $m_0 \leftarrow 0$ (initialize mean gradient vector), - $x_{\text{ref}} \leftarrow x_{\text{BASE}}$ (reference point for updates), - $\Delta_1 \leftarrow 0$ (initial update difference).

2. **For each training epoch** $t = 1$ **to** T**:** - Compute gradient

$$g_t \leftarrow \nabla f(x_t, z_t) \tag{24}$$

at parameters x_t and minibatch z_t. - Send g_t to BASE, receive update u_k. - Optionally, to save memory:

$$\Delta_t = x_t - x_{\text{ref}} + \left(\sum_{i=1}^{n} s_{t,n} \right) + \epsilon \tag{25}$$

- Update $\Delta_{t+1} \leftarrow \Delta_t + u_t$. - Calculate h_t using Δ_t, g_t adaptively by λ, $\|\nabla L(\theta)\|$:

$$h_t = \Delta_t \cdot g_t + \lambda \left(\frac{\|g_t\|}{\|x_t\|} \right) \tag{26}$$

- Update moments and rate:

$$m_t \leftarrow \max(\beta \cdot m_{t-1}, h_t) \quad \text{(coordinate-wise)} \tag{27}$$

$$v_t \leftarrow \beta^2 \cdot v_{t-1} + h_t^2 \tag{28}$$

$$r_t \leftarrow \beta \cdot r_{t-1} - s_{t-1} \cdot h_t \tag{29}$$

$$r_t \leftarrow \max(0, r_t) \tag{30}$$

- Compute weights and next step size:

$$W_t \leftarrow s_{\text{init}} \cdot \frac{m_t}{n} + r_t \tag{31}$$

$$s_{t+1} \leftarrow \frac{W_t}{\sqrt{v_t} + \epsilon} \tag{32}$$

- Update parameters considering super-level sets:

$$x_{t+1} \leftarrow x_{\text{BASE}} + \left(\sum_{i=1}^{n} s_{t+1,i} \right) \cdot \Delta_{t+1} \tag{33}$$

3. **End For**

This algorithm uses the exponential decay learning rates and incorporates super-level set dynamics to ensure that the updates remain within stable regions of the loss function's landscape, thus preventing issues such as overshooting or vanishing gradients. The detailed use of moment estimates and adaptive adjustments based on the gradient's magnitude ensures that the training remains stable and efficient, adapting to varying complexities of the data and model architecture. This approach provides a state-of-the-art solution for neural network optimization.

Algorithm 1: Superlevel-Set-Aware Optimizer with Exponential Decay and Gradient Norm Scaling

Input: Initial parameters θ_0;
Base optimizer BASE;
Decay factors $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_6)$;
Learning rate decay constant λ;
Initial stabilizer s_{init};
Numerical constant ϵ;
Total epochs T;
Initial learning rate α_0
Output: Optimized parameters θ_T

Initialize:
$m_0 \leftarrow 0$; // First moment
$v_0 \leftarrow 0$; // Second moment
$r_0 \leftarrow 0$; // Residual
$x_{\text{ref}} \leftarrow \theta_0$; // Reference for super-level set alignment
$\Delta_1 \leftarrow 0$; // Initial delta vector
$\alpha(t) \leftarrow \alpha_0$; // Initialize exponential decay scheduler

for $t \leftarrow 1$ **to** T **do**
// 1. Compute gradient
Sample minibatch z_t from training set;
$g_t \leftarrow \nabla_\theta \mathcal{L}(\theta_t, z_t)$; // Loss gradient
// 2. Compute exponential decay learning rate
$\alpha(t) \leftarrow \alpha_0 \cdot \exp(-\lambda \cdot t)$; // Decay schedule
// 3. Scale learning rate using gradient norm
$\eta_t \leftarrow \frac{\alpha(t)}{1+\|\nabla_\theta \mathcal{L}(\theta_t)\|}$; // Norm-adaptive step
// 4. Get base optimizer update
$u_t \leftarrow \texttt{BASE}(g_t, \theta_t)$
// 5. Super-level set delta calculation
$\Delta_t \leftarrow \theta_t - x_{\text{ref}} + (\sum_{i=1}^{n} s_{t,i}) + \epsilon$;
$\Delta_{t+1} \leftarrow \Delta_t + u_t$;
// 6. Compute stability-aware update metric
$h_t \leftarrow \Delta_t \cdot g_t + \lambda \cdot \left(\frac{\|g_t\|}{\|\theta_t\|}\right)$
// 7. Update moving averages
$m_t \leftarrow \max(\beta_1 \cdot m_{t-1}, h_t)$; // Max-wise momentum
$v_t \leftarrow \beta_2 \cdot v_{t-1} + h_t^2$; // Adaptive variance
$r_t \leftarrow \beta_3 \cdot r_{t-1} - s_{t-1} \cdot h_t$;
$r_t \leftarrow \max(0, r_t)$
// 8. Compute adaptive step size
$W_t \leftarrow s_{\text{init}} \cdot \left(\frac{m_t}{n}\right) + r_t$; // Weighted scaling
$s_{t+1} \leftarrow \frac{W_t}{\sqrt{v_t}+\epsilon}$
// 9. Super-level set-aligned update
$\theta_{t+1} \leftarrow x_{\text{ref}} + (\sum_{i=1}^{n} s_{t+1,i}) \cdot \Delta_{t+1}$
// 10. Optional: Check if $\theta_{t+1} \in S_\lambda$
if $\mathcal{L}(\theta_{t+1}) < \lambda$ **then**
Project back or adjust η_t (optional stabilization)

return θ_T

Fig. 3. Pseudocode of the Proposed Algorithm.

6.1 Exponential Decay Learning Rate Derivation

In our study on neural network optimization, the integration of an Exponentially Decaying Learning Rate serves as a cornerstone of our methodology, significantly influencing training dynamics and stability. This method is mathematically articulated as:

$$\alpha(t) = \alpha_0 e^{-\beta t}, \tag{34}$$

where $\alpha(t)$ represents the learning rate at a given training epoch t, α_0 is the initial learning rate, and β is a positive constant dictating the rate of exponential decay. This formula is derived from the principle that the learning rate should decrease in proportion to its existing value, resulting in the differential equation:

$$\frac{d\alpha}{dt} = -\beta\alpha. \tag{35}$$

Solving this first-order linear ordinary differential equation involves integrating both sides:

$$\int \frac{1}{\alpha} d\alpha = -\int \beta dt, \tag{36}$$

which leads to:

$$\ln(\alpha) = -\beta t + C, \tag{37}$$

where C is the integration constant. Utilizing the initial condition $\alpha(0) = \alpha_0$, we find $C = \ln(\alpha_0)$, and rearranging gives:

$$\alpha(t) = \alpha_0 e^{-\beta t}. \tag{38}$$

This model is particularly effective in neural network training as it ensures rapid convergence initially, followed by progressively finer adjustments as training progresses. The calibration of α_0 and β is critical, needing alignment with the neural network's architecture and the specifics of the training task.

The time derivative of the learning rate,

$$\frac{d\alpha}{dt} = -\alpha_0 \beta e^{-\beta t}, \tag{39}$$

highlights the progressively diminishing rate, indicative of increasing precision in parameter adjustments as training progresses. This gradual reduction is aligned with Bayesian principles, suggesting an increasingly concentrated posterior distribution with continued data observation.

6.2 Gradient of the Loss Function

In neural network models designed for classification, especially those employing a softmax output layer, the gradient of the loss function with respect to the model parameters θ plays a crucial role. The cross-entropy loss, a common choice for classification, is defined as:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{m} \log P(y^{(i)} \mid x^{(i)}; \theta), \tag{40}$$

where $P(y = c \mid x; \theta)$ is the predicted probability of the class c for input x and is given by the softmax function:

$$P(y = c \mid x; \theta) = \frac{\exp(f_c(x; \theta))}{\sum_{j=1}^{C} \exp(f_j(x; \theta))} \tag{41}$$

The derivative of the cross-entropy loss function with respect to the parameters is crucial for backpropagation and is computed as:

$$\nabla_\theta \mathcal{L}(\theta_t) = -\sum_{i=1}^{m} \left(\mathbf{1}_{y^{(i)}=c} - P(y = c \mid x^{(i)}; \theta_t) \right) \nabla_\theta f_c(x^{(i)}; \theta_t), \tag{42}$$

where $\mathbf{1}_{y^{(i)}=c}$ indicates whether class c is the correct classification for observation i. This gradient reflects how the parameters should be adjusted to decrease the loss, thereby improving the model's predictions.

The parameter update rule in gradient descent is fundamentally tied to the computed gradient:

$$\theta_{t+1} = \theta_t - \alpha(t) \nabla_\theta \mathcal{L}(\theta_t), \tag{43}$$

where $\alpha(t)$, the learning rate, typically follows an exponential decay model

$$\alpha(t) = \alpha_0 e^{-\beta t} \tag{44}$$

This manages the learning rate's decay to balance early convergence speed with later precision. Initially larger values of $\alpha(t)$ enable significant parameter shifts that help escape local minima or saddle points early in training, while the decay in $\alpha(t)$ ensures finer adjustments as the model approaches convergence, enhancing stability and accuracy [12, 15].

Super-level sets $S_\lambda = \{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \geq \lambda\}$ represent regions of the parameter space with equal or exceeding loss values, respectively. The connectedness of these sets is essential for ensuring that the gradient descent path does not get trapped in isolated local minima, thus supporting convergence towards a global minimum [4]. However, several research insights suggest refinements to this classical model to address potential limitations in training dynamics, particularly in high-dimensional settings. For instance, Weinan et al. (2019) highlight the importance of considering overparameterization's effect on the speed of convergence and generalization, suggesting that in overparameterized scenarios, gradient descent can quickly minimize training loss but may struggle with generalization due to a fitting of noise rather than underlying data patterns [32]. This calls for a refined approach to mitigate these effects, potentially through regularization techniques or novel loss functions that prioritize data fidelity over simple error minimization [10]. Further, Soudry et al. (2017) discuss the implicit bias of gradient descent towards maximum-margin solutions in settings with linear separability, indicating that extending training beyond low training loss can enhance model robustness and feature utilization [34]. This finding is crucial as it emphasizes the need for extended training regimes or adaptive learning rate schedules when employing cross-entropy loss, to avoid suboptimal data class separations and enhance model stability [5].

6.3 Parameter Selection and Tuning

Careful selection and tuning of algorithmic hyperparameters are critical for maintaining stability and achieving convergence in high-dimensional optimization tasks. This subsection outlines guidelines for choosing and tuning the main parameters:

- **Decay Factor, β:** These values control the memory of the moment estimates (m_t, v_t, r_t). The default setting of $\beta = (0.9, 0.99, 0.999, 0.9999, 0.99999, 0.999999)$ progressively stabilizes updates at different temporal granularities. Empirically, adjusting the largest β (e.g., 0.999999) downwards may help in highly non-stationary loss landscapes.
- **Learning Rate Decay Parameter, λ:** λ serves as the coupling strength between the magnitude of updates and the normalized gradient norm. A typical range is $0.001 \leq \lambda \leq 0.05$. Smaller values result in more conservative descent trajectories, while larger values accelerate convergence but may introduce instability in noisy regions.
- **Initial Stabilization Scalar s_{init}:** This prevents zero-step divisions in early iterations and should remain small (10^{-8} to 10^{-6}). Increasing it can regularize the training at the cost of slower adaptation.

- **Numerical Stability Constant ϵ:** This constant ensures that no division by zero occurs during variance normalization. The standard choice is $\epsilon = 10^{-8}$, which works well across common floating-point precision environments.
- **Reference Point x_{ref}:** This baseline anchor should be initialized with the same starting values as the base optimizer (e.g., Adam or SGD). Resetting x_{ref} periodically (every K epochs) can reduce cumulative drift in very long training runs.
- **Epochs T:** The number of training steps T must be aligned with the learning rate decay $\alpha(t) = \alpha_0 e^{-\beta t}$. A faster decay (larger β) requires either fewer epochs or larger α_0, while slower decay favors more epochs for gradual convergence.

In practice, we recommend beginning with the default values and tuning only λ, α_0, and β based on validation performance. The theoretical guarantees ensure stable descent provided λ is within a range that prevents explosive updates. Empirical grid searches or Bayesian optimization methods can be employed for fine-tuning in high-stakes deployments.

6.4 Additional Stability Analysis

6.4.1 Demonstrating Negative Semi-Definiteness. To ensure stability in neural network training, we demonstrate the negative semi-definiteness of the time derivative of the Lyapunov function $V(\theta)$, typically the loss function $\mathcal{L}(\theta)$. By applying the gradient descent update rule,

$$\frac{d\theta}{dt} = -\alpha(t)\nabla_\theta \mathcal{L}(\theta) \tag{45}$$

the derivative of V simplifies to

$$\frac{dV}{dt} = -\alpha(t)\|\nabla_\theta \mathcal{L}(\theta)\|^2 \tag{46}$$

Since $\alpha(t)$ is always positive and $\|\nabla_\theta \mathcal{L}(\theta)\|^2$ represents the squared norm of the gradient (non-negative), the product is non-positive (≤ 0), confirming the negative semi-definiteness. This condition,

$$\frac{dV}{dt} \leq 0 \tag{47}$$

ensures the loss does not increase, maintaining stability throughout the training process. This mathematical foundation confirms the system's stability under dynamic learning conditions and complex activation landscapes, crucial for the reliable convergence of training algorithms.

6.4.2 Integrating Learning Rate Dynamics. Integrating the dynamics of an Exponentially Decaying Learning Rate into our neural network training stability analysis significantly enhances the theoretical depth and practical utility of the model. The learning rate, defined by

$$\alpha(t) = \alpha_0 e^{-\beta t} \tag{48}$$

where α_0 is the initial rate and β a decay constant, systematically reduces the step size in the gradient descent algorithm. This reduction is designed to allow for rapid convergence in early training phases through larger updates, which progressively become smaller to facilitate precise fine-tuning of the model parameters as the training advances.

Mathematically, integrating the Lyapunov function

$$V(\theta) = \mathcal{L}(\theta) \tag{49}$$

reveals crucial stability characteristics, with the rate of change of the Lyapunov function with respect to time expressed inline as

$$\frac{dV}{dt} = \nabla_\theta \mathcal{L}(\theta) \cdot \frac{d\theta}{dt} = -\alpha(t) \|\nabla_\theta \mathcal{L}(\theta)\|^2 \tag{50}$$

where $\frac{d\theta}{dt}$ corresponds to the gradient descent update rule

$$\theta_{t+1} = \theta_t - \alpha(t) \nabla_\theta \mathcal{L}(\theta_t) \tag{51}$$

The expression $-\alpha(t)\|\nabla_\theta \mathcal{L}(\theta)\|^2$ ensures that

$$\frac{dV}{dt} \leq 0 \tag{52}$$

as long as $\alpha(t) > 0$ and $\nabla_\theta \mathcal{L}(\theta)$ is non-zero, satisfying the Lyapunov stability condition that the Lyapunov function does not increase over time. This formulation not only mathematically substantiates the stability of the training process under dynamic learning rate adjustments but also aligns with the practical necessity for controlled optimization trajectories in advanced neural network training regimes.

6.4.3 Addressing Model Dynamics and Stability. Addressing the dynamics and stability of neural network training involves examining the interaction between the Exponentially Decaying Learning Rate

$$\alpha(t) = \alpha_0 e^{-\beta t} \tag{53}$$

and the topology of the loss function's level sets

$$S_\lambda = \{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \geq \lambda\} \tag{54}$$

As $\alpha(t)$ decreases, the trajectory of gradient descent is refined, stabilizing within favorable super-level sets and minimizing oscillations outside minimal loss basins. Mathematically, this stabilization is evidenced by the rate of change in the loss function,

$$\frac{d\mathcal{L}}{dt} = -\alpha(t) \|\nabla_\theta \mathcal{L}(\theta)\|^2 \tag{55}$$

which confirms that the loss is nonincreasing along the path, a core Lyapunov stability condition. Additionally, this relationship suggests that for any small $\epsilon > 0$, there exists a δ such that if

$$\|\theta_0 - \theta^*\| < \delta \tag{56}$$

then $\|\theta_t - \theta^*\| < \epsilon$ for all t, demonstrating the boundedness around the minimum and affirming the model's stability. This rigorous mathematical framework underscores the efficacy of integrating dynamic learning rate strategies with the loss function's geometric properties, ensuring convergence in complex training scenarios.

6.5 Differential Inequality

In neural network training, the differential inequality and stability analysis are enhanced by examining the dynamics within level sets

$$L_\lambda = \{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) = \lambda\} \tag{57}$$

and super-level sets

$$S_\lambda = \{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \geq \lambda\} \tag{58}$$

as a boundary of loss function. The parameter update, defined as

$$\theta_{t+1} = \theta_t - \alpha(t) \nabla_\theta \mathcal{L}(\theta_t) \tag{59}$$

integrates into the derivative of the loss function,

$$\frac{d\mathcal{L}}{dt} = -\alpha(t) \|\nabla_\theta \mathcal{L}(\theta)\|^2 \tag{60}$$

confirming the non-positive decrease in loss and ensuring stability since $\alpha(t) > 0$ and

$$\|\nabla_\theta \mathcal{L}(\theta)\|^2 \geq 0 \tag{61}$$

This mathematical framework, supported by the exponential decay of

$$\alpha(t) = \alpha_0 e^{-\beta t} \tag{62}$$

maintains the trajectory within stable level sets, facilitating convergence towards optimal minima. This approach is particularly relevant in the context of Marco et al. (2008), who advocate for differential variational inequalities to handle the complexities within compact convex subsets typical of advanced architectures like cellular neural networks[35]. Their insights into the connectivity and convexity of level sets underpin the effective navigation and stability of training processes in such complex landscapes, making this analysis vital for designing neural network training algorithms.

7 Conclusion and Future Work

In this paper, we formally proved that under a Lyapunov stability framework, dynamically adjusted learning rates preserve the connectedness of super-level sets, ensuring monotonic loss descent and robust convergence. This approach addresses the complexities posed by non-Lipschitz continuous functions, common in advanced neural architectures, and links the dynamics of learning rates with the topology of loss function level sets. Our findings provide a foundation for enhancing both theoretical understanding and practical applications of neural network training.

Future research could extend this framework to various neural network architectures, such as recurrent or convolutional networks, to determine if the observed stability conditions and convergence behaviors are universally applicable. This could lead to the development of more robust and efficient training algorithms, improving real-world applications where stability and convergence are crucial.

Inspired by Fatkhullin and Polyak [2021], which examined level set connectivity in control theory contexts, another promising direction is exploring the connectivity properties of level sets and super-level sets within partially observable Markov Decision Processes (MDPs). This exploration could yield significant advances in reinforcement learning, particularly for algorithms designed to handle environments with incomplete information.

While this study establishes a solid theoretical base for neural network dynamics using advanced mathematical tools, practical limitations such as the applicability to different network architectures and real-world datasets remain areas for further investigation. Overcoming these challenges will not only validate our theoretical models but also broaden their practical relevance and effectiveness in diverse applications. This work lays the groundwork for future explorations that could transform theoretical insights into actionable algorithms for complex decision-making environments.

Acknowledgments

Jatin Chaudhary would like to acknowledge the University of Turku Graduate School's grant for conducting this work.

References

[1] Jieun Park, Dokkyun Yi, and Sangmin Ji. A novel learning rate schedule in optimization for neural networks and it's convergence. *Symmetry*, 12(4), 2020. ISSN 2073-8994. doi: 10.3390/sym12040660. URL https://www.mdpi.com/2073-8994/12/4/660.

[2] Ashok Cutkosky, Aaron Defazio, and Harsh Mehta. Mechanic: A learning rate tuner. *Advances in Neural Information Processing Systems*, 36, 2024.

[3] Simon Kornblith, Ting Chen, Honglak Lee, and Mohammad Norouzi. Why do better loss functions lead to less transferable features? *Advances in Neural Information Processing Systems*, 34:28648–28662, 2021.

[4] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[5] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. Improving generalization in deep learning by noise stability. In *International Conference on Learning Representations (ICLR)*, 2017.

[6] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257. PMLR, 2016.

[7] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

[8] Jingfeng Chen, Shihao Liu, Tianyi Chen, and Lexing Ying. Optimal adaptive and non-adaptive learning rates for optimization. In *Advances in Neural Information Processing Systems*, pages 7634–7643, 2020.

[9] Simon S Du, Jason D Lee, Haochuan Li, and Liwei Wang. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[10] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. In *Advances in neural information processing systems*, pages 5947–5956, 2017.

[11] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pages 1724–1732. PMLR, 2017.

[12] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 81–89, 2017.

[13] Rong Ge, Jason D Lee, and Tengyu Ma. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842. PMLR, 2015.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Leon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. In *SIAM Review*, volume 60, pages 223–311. SIAM, 2018.

[19] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[20] Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

[21] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[22] Sihan Zeng, Thinh Doan, and Justin Romberg. Connected superlevel set in (deep) reinforcement learning and its application to minimax theorems. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 20146–20163. Curran Associates, Inc., 2023.

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[24] Xinyi Zhu, Sijia Liu, Wenqian Li, Xiaoyu Shen, Marios Savvides, and Wen Cheng. Robust early-learning: Hindering the memorization of noisy labels. In *Advances in Neural Information Processing Systems*, pages 10551–10562, 2019.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[26] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[27] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[28] XX Zhang and Others. An adaptive mechanism to achieve learning rate dynamically. *Neural Computing and Applications*, 31:129–140, 2019.

[29] Amardeep Mishra and Satadal Ghosh. Variable gain gradient descent-based robust reinforcement learning for optimal tracking control of unknown nonlinear system with input-constraints. *Neural Computing and Applications*, 2019.

[30] L Kozachkov, Patrick M Wensing, and Jean-Jacques E Slotine. Generalization as dynamical robustness-the role of riemannian contraction in supervised learning. *NeurIPS*, 2023.

[31] Nikola B Kovachki and Andrew M Stuart. Continuous time analysis of momentum methods. *NeurIPS*, 2021.

[32] E Weinan, Chao Ma, and Lei Wu. A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics. *Science China Mathematics*, 62(1):191–200, 2019.
[33] M Forti and A Tesi. Stability of nonlinear discrete-time systems: Lyapunov approach. *Kybernetika*, 42(4):377–392, 2006.
[34] Daniel Soudry, Elad Hoffer, MorShpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.
[35] Mauro Di Marco, Mauro Forti, Massimo Grazzini, Paolo Nistri, and Luca Pancioni. Lyapunov method and convergence of the full-range model of cnns. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(11):3528–3541, 2008.

Received 24 September 2024; revised 28 May 2025; accepted 12 June 2025