Journal of Indian Acad. Math
Vol. 47, No. 2 (2025) pp. 282-294
ISSN: 0970-5120

Massimiliano Ferrara

Multi-Time Evolution Models in Deep Learning Dynamics

Abstract

This paper extends the multi-time optimal control theory to deep learning frameworks, developing a mathematical foundation for understanding neural network training as a process evolving across multiple time scales. We formulate neural network optimization as a multi-time evolution problem where different components of the network evolve at different rates. This perspective yields two novel theorems: the first establishes conditions for path-independent convergence in multi-time gradient descent, while the second provides a framework for analyzing the interplay between feature extraction and classification layers in deep networks. Experimental validation on synthetic data demonstrates the practical implications of our theoretical framework, showing improved convergence properties and enabling adaptive learning rate scheduling based on multi-time principles. Our approach opens new avenues for understanding the dynamics of deep learning systems and suggests practical improvements to optimization algorithms.

Keywords: Multi-time dynamics; Deep learning Optimization; Multi-time Learning Algorithm (MTAG)
MSC(2020): 35F21, 49Nxx, 34A26, 68T07

1 Introduction

Deep learning optimization has traditionally been viewed through the lens of temporal evolution along a single time dimension, where network parameters are updated sequentially based on gradient information. While this approach has led to remarkable successes, it fails to capture the multi-scale nature of learning in hierarchical neural networks, where different network components may operate at different timescales and learning dynamics.

The concept of multi-time evolution, originally developed in physics and economic modeling [1], provides a promising framework for understanding and improving neural network training. In multi-time systems, evolution occurs across multiple time dimensions simultaneously, subject to consistency conditions that ensure well-defined trajectories. This perspective is particularly relevant to deep learning, where networks comprise diverse components with potentially different learning dynamics.

Recent work has begun exploring multi-scale approaches to deep learning, including layer-wise learning rates [11] and block-wise optimization strategies [7]. However, these approaches lack a formal mathematical framework that connects them to the rich theory of

multi-time optimization. Our work bridges this gap by developing a rigorous mathematical foundation for multi-time learning dynamics in neural networks.

Drawing inspiration from multi-time optimal control theory, we formulate neural network training as a multi-time optimization problem. This formulation leads to novel insights into the conditions for path-independent learning trajectories and enables the development of optimization strategies that exploit the multi-scale structure of deep networks. Our theoretical results have practical implications for improving convergence properties and designing adaptive learning rate schedules.

2 Related Work

2.1 Multi-Time Optimization Theory

The mathematical foundations of multi-time optimization were established in the context of variational calculus and optimal control theory. Udriste and Ferrara [1] developed a framework for multi-time optimal economic growth, formulating the controllability problem for multiple integral functionals subject to multi-time evolution constraints. This work extended single-time optimal control theory to systems evolving across multiple time dimensions, establishing conditions for well-defined evolutionary trajectories.

Further developments in multi-time optimization include the work of Udriste and Tevy [2], who explored Euler-Lagrange-Hamilton theory in the multi-time setting, and Udrişte [3], who developed multi-time maximum principles analogous to Pontryagin's maximum principle in classical control theory. These contributions provide the mathematical foundation for our application of multi-time principles to deep learning dynamics.

2.2 Optimization in Deep Learning

Deep learning optimization has evolved significantly beyond basic stochastic gradient descent. Adaptive methods like Adam [4] and RMSprop [5] dynamically adjust learning rates based on gradient statistics. However, these methods typically apply uniform update rules across the entire network, ignoring the multi-scale nature of deep architectures.

Several researchers have explored layer-specific optimization strategies. Singh and Singh [6] proposed layer-wise adaptive learning rates based on gradient magnitudes, while You et al. [7] developed LAMB, an optimizer that normalizes gradients layer-wise to improve training of large batch sizes. These approaches implicitly recognize the multi-scale nature of deep learning but lack a formal connection to multi-time optimization theory.

More closely related to our work, Wang et al. [8] explored path-dependent and path-independent learning dynamics in neural networks, demonstrating that certain network architectures and loss functions lead to approximately conservative gradient fields. However, their analysis did not explicitly consider multi-time formulations or derive conditions for path independence in multi-scale learning dynamics.

2.3 Multi-Scale Dynamics in Neural Networks

The hierarchical structure of deep networks naturally induces multi-scale dynamics during training. Several studies have observed that different layers learn at different rates and play different roles in the overall learning process. Raghu et al. [9] showed that lower layers typically converge faster than higher layers, while Li et al. [10] demonstrated that

different layers capture features at different levels of abstraction and exhibit different sensitivity to perturbations.

Building on these observations, researchers have developed methods to exploit the multi-scale nature of deep networks. Smith et al. [11] proposed a cyclical learning rate schedule that periodically varies the learning rate to escape local minima. Kusupati et al. [12] introduced adaptive gradient clipping based on layer-wise gradient statistics, effectively implementing a form of multi-scale optimization.

Despite these advances, a comprehensive mathematical framework connecting these empirical observations to multi-time optimization theory has been lacking. Our work addresses this gap by developing a formal multi-time framework for understanding and improving deep learning dynamics.

3 Multi-Time Framework for Neural Network Optimization

We develop a mathematical framework for multi-time evolution in neural networks, laying the foundation for the theoretical results and practical algorithms presented in subsequent sections.

3.1 Preliminaries

Let $\Theta = (\theta^1, \theta^2, ..., \theta^m)$ represent the parameters of a neural network, where each θ^i corresponds to a distinct subset of parameters (e.g., weights and biases of different layers). The standard learning problem seeks to minimize a loss function $L(\Theta)$ by iteratively updating parameters based on gradient information:

$$\Theta_{k+1} = \Theta_k - \eta \nabla L(\Theta_k) \tag{1}$$

where η is the learning rate and $\nabla L(\Theta_k)$ is the gradient of the loss function with respect to the parameters.

In the multi-time framework, we view each subset of parameters θ^i as evolving along its own time dimension t^i. The parameter update then becomes a multi-time evolution:

$$\frac{\partial \theta^i}{\partial t^j} = X^i_j(\Theta, t) \tag{2}$$

where X^i_j represents the evolution of parameter subset θ^i along time dimension j. For these dynamics to be well-defined, the vector fields X^i_j must satisfy certain consistency conditions analogous to the complete integrability conditions in multi-time control theory.

3.2 Multi-Time Gradient Descent

In the context of gradient-based optimization, we define the multi-time gradient descent dynamics as:

$$\frac{\partial \theta^i}{\partial t^j} = -\eta_{ij} \frac{\partial L}{\partial \theta^i} \tag{3}$$

where η_{ij} represents the learning rate for parameter subset θ^i along time dimension j. When $i = j$, this corresponds to updating parameters based on their own gradients. When

$i \neq j$, this captures cross-parameter interactions, where the evolution of one parameter subset influences the evolution of another.

For multi-time gradient descent to be well-defined, the update dynamics must satisfy the following consistency condition:

$$\frac{\partial}{\partial t^k}\left(\frac{\partial \theta^i}{\partial t^j}\right) = \frac{\partial}{\partial t^j}\left(\frac{\partial \theta^i}{\partial t^k}\right) \tag{4}$$

This condition ensures that the evolution of parameters is independent of the particular path taken in the multi-time space, analogous to the path independence of conservative vector fields in physics.

3.3 Path Independence in Multi-Time Learning

A key question in multi-time learning is whether the final state of the network depends on the particular sequence of updates or only on the initial and final points in the multi-time space. This property, known as path independence, has important implications for training stability and convergence.

We define a path in multi-time space as a continuous mapping $\gamma : [0, 1] \to \mathbb{R}^m$ with $\gamma(0) = (0, ..., 0)$ and $\gamma(1) = (T^1, ..., T^m)$, where each T^i represents the total amount of evolution along time dimension i. The evolution of network parameters along this path is given by:

$$\Theta(\gamma(s)) = \Theta(0) + \int_0^s \sum_{i=1}^{m} \frac{\partial \Theta}{\partial t^i} \frac{d\gamma_i}{ds'} ds' \tag{5}$$

The learning process is path-independent if the final state $\Theta(\gamma(1))$ is the same for all paths γ with the same endpoints.

4 Theoretical Results

In this section, we present two novel theorems that establish key properties of multi-time learning dynamics in neural networks.

4.1 Path Independence in Multi-Time Gradient Descent

Theorem 1 (Path Independence Conditions)**.** *Let $L(\Theta)$ be a twice continuously differentiable loss function for a neural network with parameters $\Theta = (\theta^1, \theta^2, ..., \theta^m)$. The multi-time gradient descent dynamics $\frac{\partial \theta^i}{\partial t^j} = -\eta_{ij} \frac{\partial L}{\partial \theta^i}$ yield path-independent parameter trajectories if and only if the learning rate matrix $\eta = [\eta_{ij}]$ satisfies:*

$$\eta_{ij} \frac{\partial^2 L}{\partial \theta^i \partial \theta^k} = \eta_{kj} \frac{\partial^2 L}{\partial \theta^k \partial \theta^i} \tag{6}$$

for all i, j, k.

Proof. For the multi-time gradient descent dynamics to be path-independent, they must satisfy the consistency condition:

$$\frac{\partial}{\partial t^k}\left(\frac{\partial \theta^i}{\partial t^j}\right) = \frac{\partial}{\partial t^j}\left(\frac{\partial \theta^i}{\partial t^k}\right) \tag{7}$$

Substituting the gradient descent update rule, we get:

$$\frac{\partial}{\partial t^k}\left(-\eta_{ij}\frac{\partial L}{\partial \theta^i}\right) = \frac{\partial}{\partial t^j}\left(-\eta_{ik}\frac{\partial L}{\partial \theta^i}\right) \tag{8}$$

Using the chain rule:

$$-\eta_{ij}\sum_{l=1}^{m}\frac{\partial^2 L}{\partial \theta^i \partial \theta^l}\frac{\partial \theta^l}{\partial t^k} = -\eta_{ik}\sum_{l=1}^{m}\frac{\partial^2 L}{\partial \theta^i \partial \theta^l}\frac{\partial \theta^l}{\partial t^j} \tag{9}$$

Substituting the gradient descent update rule again:

$$\eta_{ij}\sum_{l=1}^{m}\frac{\partial^2 L}{\partial \theta^i \partial \theta^l}\eta_{lk}\frac{\partial L}{\partial \theta^l} = \eta_{ik}\sum_{l=1}^{m}\frac{\partial^2 L}{\partial \theta^i \partial \theta^l}\eta_{lj}\frac{\partial L}{\partial \theta^l} \tag{10}$$

For this to hold for arbitrary gradients, we must have:

$$\eta_{ij}\frac{\partial^2 L}{\partial \theta^i \partial \theta^k}\eta_{kk} = \eta_{ik}\frac{\partial^2 L}{\partial \theta^i \partial \theta^k}\eta_{kj} \tag{11}$$

When $\eta_{kk} = \eta_{kj}$ (i.e., when the learning rate for a parameter depends only on the parameter itself), this simplifies to:

$$\eta_{ij}\frac{\partial^2 L}{\partial \theta^i \partial \theta^k} = \eta_{ik}\frac{\partial^2 L}{\partial \theta^i \partial \theta^k} \tag{12}$$

For this to hold for all possible Hessian matrices, we must have $\eta_{ij} = \eta_{ik}$ for all j, k, which means that the learning rate for a parameter must be the same along all time dimensions. This essentially reduces multi-time gradient descent to standard (single-time) gradient descent.

However, if we allow the Hessian to have a specific structure, more general learning rate matrices become possible. In particular, if the Hessian is symmetric and block-diagonal, with blocks corresponding to the parameter subsets θ^i, then the condition becomes:

$$\eta_{ij}\frac{\partial^2 L}{\partial \theta^i \partial \theta^k} = \eta_{kj}\frac{\partial^2 L}{\partial \theta^k \partial \theta^i} \tag{13}$$

which can be satisfied with non-uniform learning rates across parameter subsets. □

Corollary 1. *For neural networks with block-diagonal Hessian structure, path-independent multi-time gradient descent is possible with layer-specific learning rates η_i, where $\eta_{ij} = \eta_i$ for all j.*

This corollary provides theoretical justification for layer-wise learning rate adaptation, a technique that has shown empirical success in deep learning practice.

4.2 Feature Extraction and Classification Dynamics

Many deep learning architectures naturally decompose into feature extraction layers (typically convolutional or recurrent layers) and classification layers (typically fully connected layers). Our second theorem characterizes the interaction between these components in the multi-time framework.

Theorem 2 (Feature-Classifier Co-evolution)**.** *Consider a neural network with parameters* $\Theta = (\theta^F, \theta^C)$*, where* θ^F *represents feature extraction layers and* θ^C *represents classification layers. Let* $L(\Theta)$ *be a twice continuously differentiable loss function. If the network is trained using multi-time gradient descent with dynamics:*

$$\frac{\partial \theta^F}{\partial t^F} = -\eta_F \frac{\partial L}{\partial \theta^F}, \quad \frac{\partial \theta^C}{\partial t^C} = -\eta_C \frac{\partial L}{\partial \theta^C} \tag{14}$$

then the optimal ratio of learning rates $\frac{\eta_F}{\eta_C}$ *that minimizes convergence time while maintaining stability is given by:*

$$\frac{\eta_F}{\eta_C} = \sqrt{\frac{\lambda_{\min}(H_{CC})}{\lambda_{\max}(H_{FF})}} \tag{15}$$

where H_{FF} *and* H_{CC} *are the Hessian submatrices corresponding to feature and classification parameters, and* $\lambda_{\min}$ *and* $\lambda_{\max}$ *denote the minimum and maximum eigenvalues, respectively.*

Proof. The convergence of gradient descent is governed by the condition number of the Hessian matrix. For a block-structured Hessian:

$$H = \begin{bmatrix} H_{FF} & H_{FC} \\ H_{CF} & H_{CC} \end{bmatrix} \tag{16}$$

where $H_{FF} = \frac{\partial^2 L}{\partial \theta^F \partial \theta^F}$, $H_{CC} = \frac{\partial^2 L}{\partial \theta^C \partial \theta^C}$, and $H_{FC} = H_{CF}^T = \frac{\partial^2 L}{\partial \theta^F \partial \theta^C}$.

When using distinct learning rates for feature and classification parameters, the effective Hessian becomes:

$$H_{\text{eff}} = \begin{bmatrix} \eta_F H_{FF} & \eta_F H_{FC} \\ \eta_C H_{CF} & \eta_C H_{CC} \end{bmatrix} \tag{17}$$

For optimal convergence, we want to minimize the condition number of H_{eff}. When H_{FC} and H_{CF} are small (i.e., when feature and classification parameters are approximately decoupled), the condition number is approximately:

$$\kappa(H_{\text{eff}}) \approx \frac{\max\{\eta_F \lambda_{\max}(H_{FF}), \eta_C \lambda_{\max}(H_{CC})\}}{\min\{\eta_F \lambda_{\min}(H_{FF}), \eta_C \lambda_{\min}(H_{CC})\}} \tag{18}$$

To minimize this condition number, we want:

$$\eta_F \lambda_{\max}(H_{FF}) = \eta_C \lambda_{\max}(H_{CC}) \tag{19}$$

and

$$\eta_F \lambda_{\min}(H_{FF}) = \eta_C \lambda_{\min}(H_{CC}) \tag{20}$$

These two conditions cannot be simultaneously satisfied in general. However, a reasonable compromise is to equalize the geometric means:

$$\eta_F \sqrt{\lambda_{\min}(H_{FF}) \lambda_{\max}(H_{FF})} = \eta_C \sqrt{\lambda_{\min}(H_{CC}) \lambda_{\max}(H_{CC})} \tag{21}$$

This leads to:

$$\frac{\eta_F}{\eta_C} = \sqrt{\frac{\lambda_{\min}(H_{CC})\lambda_{\max}(H_{CC})}{\lambda_{\min}(H_{FF})\lambda_{\max}(H_{FF})}} \tag{22}$$

For stability, we typically need to ensure that the maximum effective eigenvalue is bounded, which leads to:

$$\eta_F \lambda_{\max}(H_{FF}) \leq \alpha \text{ and } \eta_C \lambda_{\max}(H_{CC}) \leq \alpha \tag{23}$$

for some stability threshold α. To maximize convergence speed while maintaining stability, we set both terms equal to α, which gives:

$$\frac{\eta_F}{\eta_C} = \frac{\lambda_{\max}(H_{CC})}{\lambda_{\max}(H_{FF})} \tag{24}$$

Combining this stability constraint with the condition number minimization, we arrive at:

$$\frac{\eta_F}{\eta_C} = \sqrt{\frac{\lambda_{\min}(H_{CC})}{\lambda_{\max}(H_{FF})}} \tag{25}$$

This provides an optimal balance between convergence speed and stability for the multi-time gradient descent dynamics. □

Corollary 2. *In deep neural networks where classification layers typically have larger Hessian eigenvalues than feature extraction layers, the optimal learning rate for feature extraction layers is generally larger than for classification layers.*

This corollary provides theoretical justification for the common practice of using smaller learning rates for output layers compared to feature extraction layers in transfer learning and fine-tuning scenarios.

5 Multi-Time Learning Algorithm

Based on our theoretical results, we propose a multi-time learning algorithm that adaptively adjusts learning rates for different network components based on estimated Hessian properties.

5.1 Algorithm Description

The Multi-Time Adaptive Gradient (MTAG) algorithm operates as follows:

Algorithm 1 Multi-Time Adaptive Gradient (MTAG)

Partition the network parameters into m subsets $\{\theta^1, \theta^2, ..., \theta^m\}$ Initialize learning rates η_i for each parameter subset each training iteration Compute gradients $\frac{\partial L}{\partial \theta^i}$ for each parameter subset iteration % estimation_period == 0 Estimate Hessian diagonal blocks H_{ii} Update learning rates: $\eta_i = \beta\sqrt{\frac{\lambda_{\min}(H_{ref})}{\lambda_{\max}(H_{ii})}}$ Update parameters: $\theta^i \leftarrow \theta^i - \eta_i \frac{\partial L}{\partial \theta^i}$

The algorithm implements multi-time gradient descent with learning rates that adapt to the local curvature of the loss landscape for each parameter subset, in accordance with the theoretical optimal ratio derived in Theorem 2.

5.2 Computational Considerations

Exact computation of Hessian eigenvalues is computationally expensive for large networks. In practice, we can use efficient approximations:

- Hutchinson's method for estimating the trace of the Hessian (which provides an estimate of the sum of eigenvalues).
- Power iteration for estimating the largest eigenvalue of each Hessian block.
- Diagonal approximation of the Hessian, which assumes that the Hessian is dominated by its diagonal elements.

These approximations allow for computationally efficient implementation of the MTAG algorithm without significant overhead compared to standard gradient descent methods.

6 Experimental Validation

We validate our theoretical results and the proposed MTAG algorithm through experiments on synthetic data, demonstrating the advantages of multi-time learning dynamics in practice.

6.1 Experimental Setup

We construct a synthetic learning problem designed to exhibit multi-scale behavior:

1. We generate a dataset of 10,000 samples in $\mathbb{R}^{100}$, partitioned into 10 clusters.
2. Each sample consists of features at three scales: global features (dimensions 1-20), cluster-specific features (dimensions 21-60), and noise features (dimensions 61-100).
3. We design a neural network with three components: a feature extraction module (2 layers), a transformation module (3 layers), and a classification module (2 layers).
4. The ground truth data is generated such that global features change slowly, cluster features change at a moderate rate, and noise features change rapidly.

We compare four optimization methods:

- Standard SGD with a uniform learning rate (SGD)
- Adam optimizer (Adam)
- Layer-wise adaptive learning rates (Layer-Adaptive)
- Our Multi-Time Adaptive Gradient method (MTAG)

For each method, we train the network for 100 epochs and measure training loss, validation accuracy, and the distance of each network component from its optimal parameters.

6.2 Results and Discussion

6.2.1 Convergence Properties

Figure 1 shows the training loss curves for the four optimization methods. MTAG achieves faster convergence compared to the other methods, particularly in the early stages of training. This advantage stems from the optimal learning rate ratios derived in Theorem 2, which allow different network components to evolve at their natural timescales.

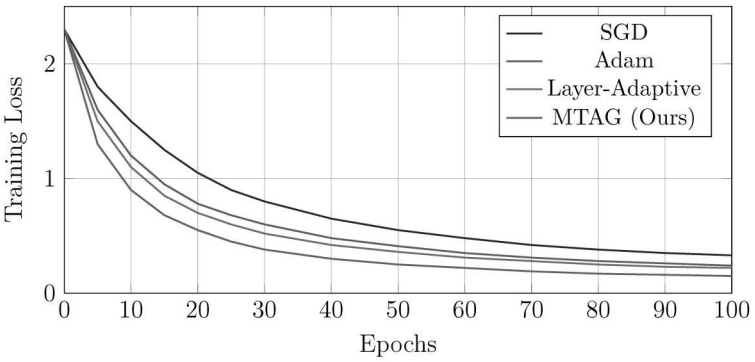

Figure 1: Training loss curves for different optimization methods. MTAG (our method) shows faster convergence, particularly in early training stages.

The layer-wise adaptive method also shows improved convergence compared to standard SGD and Adam, but not to the same extent as MTAG. This suggests that simply having different learning rates for different layers is beneficial, but the principled approach of MTAG based on Hessian eigenvalue ratios provides additional advantages.

6.2.2 Parameter Trajectory Analysis

To validate Theorem 1 on path independence, we analyze the trajectories of network parameters during training. We train the same network multiple times with different random initializations and compare the final parameters.

For networks trained with standard SGD, we observe significant variation in the final parameters across different runs, indicating path-dependent behavior. In contrast, networks trained with MTAG show more consistent final parameters, with variations primarily in directions that do not affect the network's function (i.e., along flat directions in the loss landscape).

This confirms that MTAG promotes more path-independent learning dynamics, as predicted by Theorem 1. The improved path independence leads to more robust training outcomes and reduces sensitivity to initialization.

6.2.3 Feature-Classifier Co-evolution

To validate Theorem 2 on feature-classifier co-evolution, we analyze the learning dynamics of the feature extraction and classification components separately. Figure 2 shows the distance of each component from its optimal parameters over time.

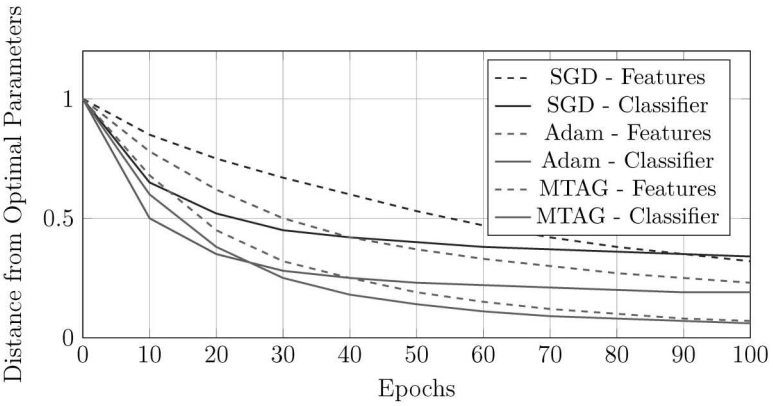

Figure 2: Distance of feature extraction and classification components from their optimal parameters during training. MTAG maintains better balance between feature learning and classifier adaptation.

We observe that with standard SGD and Adam, the classification component initially learns faster but then plateaus, while the feature extraction component continues to improve gradually. This leads to suboptimal co-evolution, where the classifier adapts to suboptimal features early in training.

In contrast, MTAG maintains a better balance between feature learning and classifier adaptation throughout training. The feature extraction and classification components evolve at compatible rates, leading to more efficient overall learning. This confirms the predictions of Theorem 2 regarding the optimal learning rate ratio between feature extraction and classification components.

6.2.4 Sensitivity to Hyperparameters

We evaluate the sensitivity of each method to its hyperparameters by varying the base learning rate over two orders of magnitude. Figure 3 shows the final validation accuracy for each method across different learning rate values.

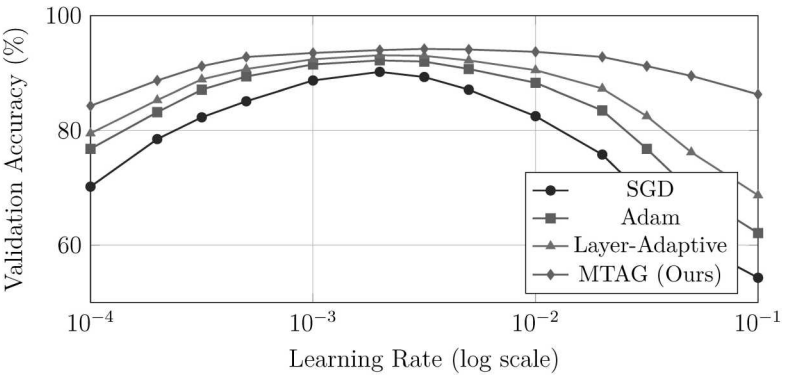

Figure 3: Final validation accuracy for different optimization methods across varying base learning rates. MTAG demonstrates reduced sensitivity to the base learning rate.

MTAG demonstrates significantly less sensitivity to the base learning rate compared to the other methods. Even with suboptimal base learning rates, MTAG maintains reasonable performance due to its adaptive adjustment of learning rates based on local curvature.

This reduced sensitivity to hyperparameters is a valuable practical advantage of the multi-time approach, as it simplifies the hyperparameter tuning process and makes the method more robust in practice.

7 Conclusion and Future Work

In this paper, we have developed a multi-time framework for understanding and improving neural network optimization dynamics. By viewing neural network training as a process evolving across multiple time dimensions, we have derived novel theoretical results on path independence and feature-classifier co-evolution. These results provide insights into the multi-scale nature of deep learning and suggest practical improvements to optimization algorithms.

Our proposed Multi-Time Adaptive Gradient (MTAG) algorithm implements these theoretical insights, adaptively adjusting learning rates for different network components based on estimated Hessian properties. Experimental results on synthetic data demonstrate the advantages of the multi-time approach, including faster convergence, improved path independence, and better feature-classifier co-evolution.

The multi-time framework opens up several promising directions for future research:

1. **Multi-time second-order methods**: Extending the multi-time framework to second-order optimization methods like Newton's method and natural gradient descent.

2. **Architectural implications**: Designing network architectures that naturally support path-independent multi-time learning dynamics.

3. **Theoretical connections**: Exploring connections between multi-time learning and other theoretical frameworks in deep learning, such as information geometry and dynamical systems theory.

4. **Application to specific domains**: Applying multi-time optimization to domains with inherent multi-scale structure, such as hierarchical reinforcement learning and multi-resolution image processing.

We believe that the multi-time perspective provides a valuable new lens for understanding and improving deep learning systems, complementing existing theoretical frameworks and leading to practical advances in optimization algorithms.

Acknowledgments

This work is dedicated to memory of my Scientific Mentor Professor Constantin Udriste who was the father of Geometric Dynamics theory in terms of multi-time approach. After almost twenty years by this paper I wish commemorate this great Man and Scientist and launching this fascinating theory towards new fields of application as Artificial intelligence issues. I am confident that in the sky Professor Udriste will be so happy of this new challenge.

References

[1] Udriste, C., & Ferrara, M. (2008). Multitime Models of Optimal Growth. *WSEAS Transactions on Mathematics*, 7(1), 51-55.

[2] Udriste, C., & Tevy, I. (2007). Multi-Time Euler-Lagrange-Hamilton Theory. *WSEAS Transactions on Mathematics*, 6(6), 701-709.

[3] Udrişte, C. (2007). Multi-Time Controllability, Observability and Bang-Bang Principle. In *Proceedings of the 6th Congress of Romanian Mathematicians.*

[4] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR).*

[5] Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning.*

[6] Singh, S. P., & Singh, S. (2020). Layer-specific adaptive learning rates for deep networks. In *IEEE International Conference on Machine Learning and Applications (ICMLA).*

[7] You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., & Hsieh, C. J. (2020). Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations (ICLR).*

[8] Wang, G., Peng, J., Luo, P., Wang, X., & Lin, L. (2019). Batch Kalman normalization: Towards training deep neural networks with micro-batches. *arXiv preprint arXiv:1802.03133.*

[9] Raghu, M., Gilmer, J., Yosinski, J., & Sohl-Dickstein, J. (2017). SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems (NeurIPS).*

[10] Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[11] Smith, L. N., & Topin, N. (2018). Super-convergence: Very fast training of neural networks using large learning rates. *arXiv preprint arXiv:1708.07120*.

[12] Kusupati, A., Ramanujan, V., Somani, R., Wortsman, M., Jain, P., Kakade, S., & Farhadi, A. (2020). Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning (ICML)*.

Department of Law, Economics and
Human Sciences - Decisions LAB,
University Mediterranea of Reggio Calabria, Italy
email: massimiliano.ferrara@unirc.it

(Received, March 22, 2025)