Adam Optimizer with Hierarchical and Dynamic Hyperparameters (Adam-HD)

Cesar H. S. França , Pedro A. B. de Oliveira

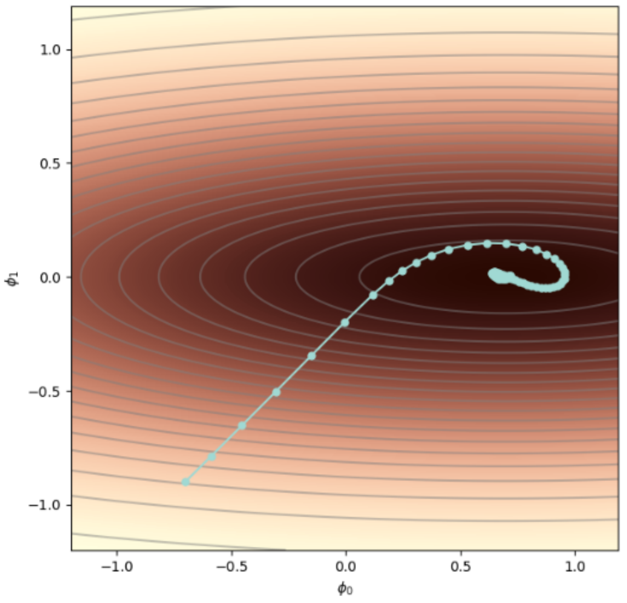


Figure 1. Adam-HD optimized trajectory over synthetic loss landscape (Prince, 2023).

Abstract

We propose Adam-HD, an adaptive optimizer that provides hierarchical and coordinated scheduling for the learning rate (α), momentum (β), and adaptive memory coefficient (γ). Leveraging real-time feedback from the loss landscape—such as gradient magnitude and directional consistency—Adam-HD dynamically shifts between aggressive and cautious states. This coordinated adjustment improves convergence, mitigates overshooting, and enhances robustness, formalizing a class of more holistic schedulers for autonomous and stable training.

1. Introduction

The **Adam Optimizer with Hierarchical and Dynamic Hyperparameters (Adam-HD)** is a modification of the standard Adam algorithm (Kingma & Ba, 2015). Its goal is to improve convergence and training stability by dynamically adjusting its key hyperparameters — the learning rate α, the first-moment momentum coefficient β, and the second-moment momentum coefficient γ — in response to the local geometry of the loss landscape.

The strategy is based on a hierarchical control system that defines a general *optimization state* and coordinates the behavior of the hyperparameters to achieve the common goal of finding a minimum quickly and accurately, while avoiding the oscillations and *overshooting* commonly observed in simpler optimizers.

Research Question

Can the Learning Rate be dynamically tuned?

Research Statement

Development of a Coordinated and Hierarchical Hyperparameter Scheduler for Adaptive Optimizers.

2. The Problem and the Existing Gap

Tuning key hyperparameters remains a major challenge in training deep neural networks. The performance of optimizers such as Adam depends heavily on the learning rate (α) and momentum coefficient (β). Poor settings may cause slow convergence, instability, or failure to reach a good minimum (Brunton & Kutz, 2022).

Proposed solutions, such as *learning rate schedulers*, address this only partially by updating α based on fixed schedules (e.g., by epoch) (PyTorch Developers, 2025), (Smith, 2017), (Li, 2022) while Bengio (2012) provides a set of practical recommendations for hyperparameter tuning, treating other hyperparameters as static. This neglects the critical interdependence between step size (α), the loss landscape and trajectory inertia (β), resulting in suboptimal adaptation which also requires time-consuming manual adjustments.

3. The Proposed Solution: The Adam-HD Scheduler

This research proposes a new paradigm: a **Coordinated Hyperparameter Scheduler**, named **Adam-HD**. Instead of adjusting a single parameter in isolation, Adam-HD implements a hierarchical control system that monitors multiple real-time signals from the loss landscape and uses them to simultaneously and coherently modulate all major optimizer hyperparameters (α, β, γ).

The core innovation lies in assigning specific control signals to each hyperparameter based on its functional role in the optimization process, i.e., The **learning rate** (α) is controlled by the gradient magnitude, an indicator of the local "slope", defining a general state of *aggressiveness*; The **momentum** (β) is adjusted based on a combination of directional consistency of the trajectory and the current aggressiveness level, allowing inertia to increase in stable directions and decrease near minima to avoid *overshooting*; The **adaptive memory** (γ) is modulated by training stability (i.e., loss variance), ensuring that the adaptation scale becomes more conservative during periods of high volatility.

4. Contributions and Expected Impact

This research introduces a new type of scheduler that coordinates multiple hyperparameters, not just the learning rate. It presents the **Adam-HD** algorithm, which improves training stability and reduces sensitivity to initial settings. The study also analyzes how signals from the loss landscape guide optimizer behavior. The goal is to enhance optimization efficiency and autonomy, reducing the dependency on a manual tuning process that requires time and expertise.

5. Methodology

5.1. Problem Definition

The performance of optimizers like Adam depends heavily on hyperparameter configuration. Static values for learning rate (α) and momentum (β) can lead to instability or slow convergence depending on the loss landscape. This work addresses the limitation of fixed hyperparameters, proposing that dynamically and jointly adjusting them improves training robustness, speed, and stability.

5.2. Proposed Method

Adam-HD introduces a hierarchical control system that dynamically adjusts the hyperparameters α, β, and γ at each training step. The method relies on three signals: the gradient magnitude ($\|g_t\|$), directional consistency of the trajectory ($\bar{c}_t$), and the recent loss variance (v_t^{loss}). The gradient magnitude defines an aggressiveness factor (f_t^{aggress}), which directly modulates the learning rate α_t. The momentum coefficient β_t is adjusted according to the directional consistency but is dampened by f_t^{aggress} to prevent instability. The memory factor γ_t is tuned conservatively, considering both aggressiveness and training stability. This coordination leads to more refined control over the optimization process, improving responsiveness to loss landscape dynamics beyond what traditional learning rate schedulers typically offer.

6. Experiments

To validate the effectiveness and analyze the behavior of Adam-HD, experiments were conducted in two distinct environments.

Test Environment 1: A synthetic 2D loss function was used to allow for direct visualization of the optimization trajectory (Figure 1). The qualitative analysis of the trajectory demonstrated the algorithm's stability: it began with large steps far from the minimum and progressively reduced its step size as it approached the target, navigating the loss landscape smoothly and without significant oscillations.

Test Environment 2: The optimizer's performance was evaluated on the classic image classification task of the MNIST dataset, using a deep Multi-Layer Perceptron (MLP) neural network architecture. In this practical scenario, Adam-HD demonstrated robust and stable training, achieving a high final accuracy.

Evaluation Metrics: Performance was measured quantitatively through the loss and validation accuracy curves per epoch, in addition to the qualitative analysis of the optimization trajectory in the 2D environment.

Results: Adam-HD demonstrated stable convergence in both environments. On the synthetic function, the trajectory was visibly smooth. On the MNIST benchmark, the optimizer reached a high final accuracy and demonstrated stable learning, validating the effectiveness of its dynamic control system. The implementation code is publicly available for reproducibility purposes.[1]

7. Conclusions

We introduced Adam-HD, an optimizer that implements a coordinated and hierarchical hyperparameter scheduler. Unlike existing methods that adjust only the learning rate, Adam-HD dynamically modulates the learning rate, momentum, and adaptive memory based on multiple real-time signals extracted from the loss landscape.

The proposed methodology enables the optimizer to intelligently transition between an *aggressive* state in high-curvature or steep-gradient regions and a *cautious* state near minima or in low-curvature plateaus, coordinating the reduction of inertia with the shrinking of the step size. The proposed experiments aim to demonstrate that this approach leads to more stable, faster, and robust training.

The concept of coordinated scheduling represents a promising advancement over traditional learning rate schedulers. Adam-HD serves as an effective proof of concept, paving the way for future research on more autonomous and geometry-aware optimizers.

[1] `https://github.com/cesarhsfranca/Adam-HD`

References

Bengio, Y. Practical Recommendations for Gradient-Based Training of Deep Architectures. *arXiv preprint arXiv:1206.5533*, 2012. URL `https://arxiv.org/abs/1206.5533`.

Brunton, S. L. and Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, second edition, 2022.

Goh, G. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006. URL `http://distill.pub/2017/momentum`.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL `https://arxiv.org/abs/1412.6980`.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the Loss Landscape of Neural Nets. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018. URL `https://arxiv.org/abs/1712.09913`.

Li, K. Y. How to choose a learning rate scheduler for neural networks. `https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler`, 2022. Accessed: August 4, 2025.

Murphy, K. P. *Probabilistic Machine Learning: An Introduction*. Adaptive Computation and Machine Learning series. The MIT Press, 2022.

Prince, S. J. D. *Understanding Deep Learning*. The MIT Press, 2023.

PyTorch Developers. Learning rate scheduling. `https://docs.pytorch.org/docs/stable/search.html?q=learning+rate+schedule&check_keywords=yes&area=default`, 2025. Accessed: August 4, 2025.

Smith, L. N. Cyclical learning rates for training neural networks. *arXiv preprint arXiv:1506.01186*, 2017. URL `https://arxiv.org/abs/1506.01186`.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. *Dive into Deep Learning*. Cambridge University Press, 2023. `https://D2L.ai`.

Appendix

A. Related Work

Research in optimization for machine learning has a rich history, thoroughly documented in foundational works such as *Deep Learning* (Goodfellow et al., 2016), *Understanding Deep Learning* (Prince, 2023), (Murphy, 2022) and (Zhang et al., 2023). Early methods such as Stochastic Gradient Descent (SGD) laid the groundwork but suffered from slow convergence rates. The concept of momentum, explored in depth in the article *Why Momentum Really Works* (Goh, 2017), was introduced to accelerate convergence by adding inertia to the optimization trajectory.

To address sensitivity to feature scale, adaptive methods emerged. Adagrad introduced per-parameter learning rates, which were later refined by RMSProp to prevent the aggressive and monotonic decay of the learning rate. The Adam optimizer became a milestone by combining the benefits of RMSProp with momentum, and has since become the default choice in many applications (Kingma & Ba, 2015).

However, Adam itself relies on static hyperparameters. The research community has tackled the rigidity of the learning rate α through a variety of learning rate schedulers. While often effective, these methods do not address the fundamental limitation: they adjust only α, typically following a predefined schedule and disregarding the real-time state of the loss landscape.

In parallel, works such as *Visualizing the Loss Landscape of Neural Nets* (Li et al., 2018) have shown that the geometry of the loss landscape is critical for generalization (Zhang et al., 2023). Wide and flat minima are correlated with better model performance (Zhang et al., 2023). This insight motivates the need for optimizers that not only converge to a minimum, but do so in a way that favors more stable regions (Zhang et al., 2023).

The Adam-HD proposal sits at the intersection of these research lines. It fundamentally differentiates itself from prior work by introducing a coordinated control system. Whereas previous methods focus on adapting a single hyperparameter, Adam-HD leverages a diverse set of signals (slope, curvature, stability) to simultaneously modulate multiple hyperparameters. For example, its ability to reduce momentum (β) near a minimum can be interpreted as an explicit strategy to "settle" into wide minima, rather than overshooting due to excessive inertia, aligning with the findings of Li et al. (2018).

This work, therefore, is not merely an incremental improvement, but a proposal for a new class of schedulers — more holistic and geometry-aware.

B. Mathematical Description of the Adam-HD Optimizer

B.1. Introduction and Purpose

The Adam optimizer with Hierarchical and Dynamic Hyperparameters (Adam-HD), as implemented in the reference code, is a modification of the standard Adam algorithm. Its goal is to improve convergence and training stability by dynamically adjusting its main hyperparameters — the learning rate α, the first moment momentum coefficient β, and the second moment momentum coefficient γ — in response to the local geometry of the loss landscape.

The strategy is based on a hierarchical control system that uses two primary control signals to define an ”optimization state” and coordinate hyperparameter actions.

B.2. Notation and Assumptions

$$g_t = \nabla L(\boldsymbol{\phi}_t) \in \mathbb{R}^d \tag{1}$$

$$g_t^{\text{norm}} = \|\nabla L(\boldsymbol{\phi}_t)\|_2 \in \mathbb{R}^d \tag{2}$$

Small constant $\epsilon = 10^{-8}$ used for numerical stabilization.

User parameters (ranges):

- $\alpha_{\text{range}} = (\alpha_{\min}, \alpha_{\max}); \quad T_{\text{range}} = (T_{\min}, T_{\max})$.
- $\beta \in [\beta_{\min}, \beta_{\max}]; \quad \gamma \in [\gamma_{\min}, \gamma_{\max}]$.

Smoothing coefficients for exponential moving averages (EMA):

- $\beta_{\text{norm}} \in [0, 1]; \quad \beta_{\text{smooth}} \in [0, 1]$.

B.3. Dynamic Control Signals

At each optimization step t, with model parameters at ϕ_t, the algorithm computes two control signals from the loss landscape.

B.3.1. Signal 1: Gradient Magnitude (Slope)

We define the L2 norm of the gradient at step t as:

$$g_t^{\text{norm}} = \|\nabla L(\phi_t)\|_2 \equiv ||\nabla \mathcal{L}_t|| \tag{3}$$

Purpose of the Formula: This primary signal measures the ”steepness” of the loss landscape at the current point. It serves as the main indicator to define the optimizer’s overall ”aggressiveness” state. A high slope suggests we are far from a minimum while a low one suggests we are close. In short, it is the basis for the optimizer’s adaptive behavior: it measures the slope of the terrain to decide how fast to move.

B.3.2. Extraction of Signals from the Loss Landscape:

The optimizer computes two real-time signals to understand the local geometry.

Normalized “Slope” Signal: To make the optimizer robust across different models and loss scales, the gradient magnitude is not directly used. Instead, it is normalized based on its own recent trend, which is captured by an exponential moving average ($\bar{g}_t$). This process provides a scale-invariant measure of the landscape’s steepness which involves two preliminary calculations:

- Current gradient norm: g_t^{norm}
- Moving average update: The EMA provides a stable, less noisy baseline of the recent gradient magnitude.

$$\bar{g}_t = \beta_{\text{norm}} \cdot \bar{g}_{t-1} + (1 - \beta_{\text{norm}}) \cdot g_t^{\text{norm}} \tag{4}$$

From Equations (2) and (4), the normalized error signal, z_t, which measures the relative gradient variation, is calculated. This signal measures the relative variation of the current gradient against its recent trend. A positive z_t indicates an acceleration of the descent (entering a steeper region), while a negative z_t indicates a deceleration:

$$z_t = \frac{g_t^{\text{norm}}}{\bar{g}_t + \epsilon} - 1 \tag{5}$$

B.3.3. Signal 2: Directional Consistency (Trajectory Curvature)

Instantaneous consistency:

$$c_t = \frac{\nabla L(\phi_t) \cdot \nabla L(\phi_{t-1})}{\|\nabla L(\phi_t)\|_2 \|\nabla L(\phi_{t-1})\|_2} \tag{6}$$

Smoothed consistency:

$$\bar{c}_t = \beta_{\text{smooth}} \cdot \bar{c}_{t-1} + (1 - \beta_{\text{smooth}}) \cdot c_t \tag{7}$$

Purpose of the Formulas: Measure the smoothness of the trajectory. The first formula (c_t) computes the instantaneous directional consistency. The second ($\bar{c}_t$) smooths this signal over time to avoid abrupt reactions, providing a more robust estimate of recent directional trends. The hyperparameter β_{smooth} is the decay rate for an exponential moving average (EMA) that smooths the directional consistency signal, c_t. It acts as a low-pass filter, where a high value (e.g., 0.9) ensures that adjustments to β_t and γ_t are based on stable trajectory trends rather than noisy gradient fluctuations.

B.4. Hierarchical Control Logic and Hyperparameter Computation

Adam-HD uses the above signals to adjust α_t, β_t, and γ_t at each step.

B.4.1. Calculation of Hierarchical Control Factors:

The extracted signals are mapped to normalized control factors in the range [0, 1]:

- **Directional Factor (f_t^{dir}):** Maps the smoothed consistency $\bar{c}_t$ (which ranges from -1 to 1) to the interval $[0, 1]$. A value near 1 indicates a stable (straight) trajectory, while a value near 0 indicates instability (a sharp curve):

$$f_t^{\text{dir}} = \frac{\bar{c}_t + 1}{2} \tag{8}$$

- **Aggressiveness Factor (f_t^{aggress}):** The normalized error signal z_t is passed through a `softsign` function whose flattening factor T is also dynamic:

- First, the dynamic T (T_t) is calculated. It is high for unstable trajectories ($f_t^{\text{dir}} \approx 0$) and low for stable ones ($f_t^{\text{dir}} \approx 1$).

$$T_t = T_{\max} - (T_{\max} - T_{\min}) \cdot f_t^{\text{dir}} \tag{9}$$

- Next, the aggressiveness factor is calculated using the `softsign` with this dynamic T_t.

$$f_t^{\text{aggress}} = \frac{1}{2}\left(\frac{z_t}{T_t + |z_t|} + 1\right) \tag{10}$$

Purpose of the Formula: To map the gradient norm (slope signal) into an “aggressiveness factor” between 0 and 1, equation (10) uses a Generalized Softsign function. The behavior of this function is controlled by the *dynamic* flattening factor T_t, which delays the saturation point and creates a smoother, more proportional response zone, adapting at each step according to the directional factor f_t^{dir} as illustrated in Figure 2. This design allows T_t to be larger for unstable trajectories ($f_t^{\text{dir}} \approx 0$) and smaller for stable ones ($f_t^{\text{dir}} \approx 1$), effectively tuning the softsign's saturation threshold in real time. The practical benefit of this approach is visualized in Figure 3, where the raw input signal z_t (a) is transformed into a controlled output (b), leading to a more stable final aggressiveness factor f_t^{agress} (c).

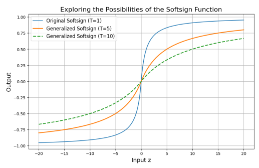
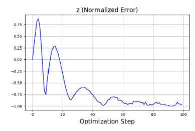
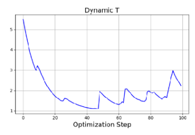


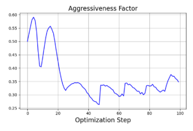
Figure 2. Exploring the Possibilities of the Generalized Softsign Function. This plot illustrates how the output of the Generalized Softsign function varies with the input z_t for different values of the *dynamic* flattening factor T_t. Note that as T_t increases, the function becomes flatter and more linear around z = 0.



(a) Input signal z_t, representing the raw error signal based on the gradient norm.



(b) Output of the Generalized Softsign function, showing the controlled and non-saturating signal.



(c) Final aggressiveness factor f_t^{agress}, scaled between 0 and 1, used to modulate the learning rate.

Figure 3. Visualization of the signal transformation process within the Adam-HD controller for the 2D optimization task. The raw input signal z_t (a) is passed through the Generalized Softsign function with dynamic T_t to produce a controlled, non-saturating output (b), which is then scaled to the final aggressiveness factor (c) used to modulate the learning rate (α).

Purpose of the Formula: To map the gradient norm (slope signal) into an ”aggressiveness factor” between 0 and 1, using a softsign function.

B.4.2. Computation of α_t (Learning Rate)

$$\alpha_t = \alpha_{\text{min}} + (\alpha_{\text{max}} - \alpha_{\text{min}}) \cdot f_t^{\text{agress}} \tag{11}$$

Purpose of the Formula: To control the learning rate directly through the aggressiveness state, using linear interpolation.

B.4.3. Computation of β_t (Momentum)

$$f_t^{\text{dir}} = \frac{\bar{c}_t + 1}{2} \tag{12}$$

$$\beta_t^{\text{target}} = \beta_{\text{min}} + (\beta_{\text{max}} - \beta_{\text{min}}) \cdot f_t^{\text{dir}} \tag{13}$$

$$\beta_t = f_t^{\text{agress}} \cdot \beta_t^{\text{target}} + (1 - f_t^{\text{agress}}) \cdot \beta_{\text{min}} \tag{14}$$

Purpose of the Formulas: To coordinate momentum with both the aggressiveness and directional consistency. The logic is nested: the direction (f_t^{dir}) defines a momentum ”target”. The aggressiveness factor (f_t^{agress}) then acts as a ”safety gate”, deciding whether to use this target or force momentum to a safe minimum value near minima.

B.4.4. Computation of γ_t (Adaptation Memory)

$$\gamma_t = \gamma_{\text{min}} + (\gamma_{\text{max}} - \gamma_{\text{min}}) \cdot f_t^{\text{dir}} \tag{15}$$

Purpose of the Formulas: To coordinate γ (second moment memory, analogous to β_2 in Adam) with the directional consistency of the trajectory. If direction is consistent (i.e., straight path, $f_t^{\text{dir}} \approx 1$), the landscape is considered stable, and the optimizer should use long memory (γ_t high). If direction changes (i.e., curved path, $f_t^{\text{dir}} \approx 0$), the geometry is considered unstable, and the optimizer should use short memory (γ_t low) to adapt more quickly.

B.5. Final Update Rule

With the dynamic hyperparameters computed, the algorithm proceeds with the standard Adam steps and all operations on the moments m_t and v_t are performed elementwise for each component of $\boldsymbol{\phi}$, except for the signals and factors that the algorithm defines as scalars (norms and consistency), which are global scalars computed from all parameters.

B.5.1. Update of Moving Averages

$$m_t = \beta_t m_{t-1} + (1 - \beta_t)\nabla L(\phi_t) \tag{16}$$

$$v_t = \gamma_t v_{t-1} + (1 - \gamma_t)(\nabla L(\phi_t))^2 \tag{17}$$

B.5.2. Bias Correction

$$\hat{m}_t = \frac{m_t}{1 - \beta_t^{t+1}} \tag{18}$$

$$\hat{v}_t = \frac{v_t}{1 - \gamma_t^{t+1}} \tag{19}$$

Detailed Analysis The code uses an **approximation** for bias correction by applying the current dynamic value raised to the power of $t + 1$. The exact theoretical formula for variable hyperparameters would involve the product of all past values.

B.5.3. Parameter Update

$$\phi_{t+1} = \phi_t - \alpha_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{20}$$

B.6. Image Descriptions

Image 1: Loss Landscape and Optimization Trajectory

Description This visualization would show a 2D contour plot. The colors represent the value of the loss function, with darker regions indicating lower loss (the "valleys") and lighter regions indicating higher loss (the "hills"). Overlaid on this landscape is a line with points, representing the path (ϕ_0, ϕ_1) taken by the optimizer from its starting position to the final position over all steps.

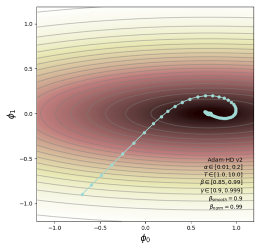
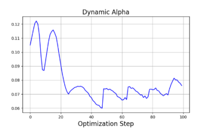


Figure 4. Detailed view of the Adam-HD optimization trajectory on the synthetic 2D loss landscape. In this visual representation for the qualitative analysis, the path starts in a high-loss region and smoothly converges to the minimum without significant oscillation or overshooting, demonstrating the stability of the dynamic hyperparameter adjustments (Prince, 2023).

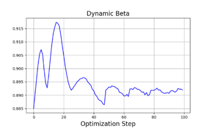
Interpretation A successful optimization is represented by a trajectory that moves efficiently from a high-loss region to the center of the darkest region (the minimum). The smoothness of the path is critical. A jagged, oscillating path indicates instability, whereas a smooth, direct path indicates a stable and efficient optimizer. This plot is essential for qualitatively assessing phenomena like *overshooting*, where the path overshoots the minimum and has to turn back.

Image 2: Dynamic Hyperparameter History

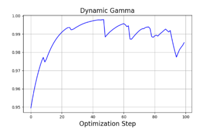
Description This visualization would consist of three separate line plots, one for each dynamic hyperparameter $(\alpha_t, \beta_t, \gamma_t)$. The x-axis for all plots is the optimization step, and the y-axis is the value of the respective hyperparameter at that step.



(a) Alpha Dynamics (α_t). The learning rate starts high and decreases as the optimizer approaches the minimum.



(b) Beta Dynamics (β_t). Momentum is coordinated with the learning rate, decreasing near the minimum to avoid overshooting.



(c) Gamma Dynamics (γ_t). The memory coefficient adapts to the trajectory's stability.

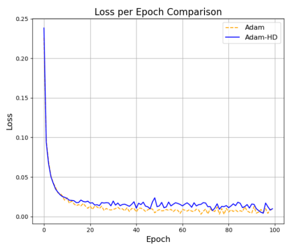
Figure 5. Dynamic history of the Adam-HD hyperparameters $(\alpha_t, \beta_t, \gamma_t)$ during optimization. Each plot reveals a different aspect of the Adam-HD adaptive dynamics.

Interpretation This image reveals the adaptive dynamics of the Adam-HD scheduler, visually confirming its hierarchical coordination among α, β and γ. The plots show the transition from an initial, aggressive exploration phase (volatile α and β) to a cautious, fine-tuning phase where both hyperparameters decrease and stabilize. This showcases the hierarchical control, where momentum is dampened near the minimum to prevent overshooting, while the steady, high value of γ indicates that the optimizer perceives the trajectory as stable and opts for long-term memory.

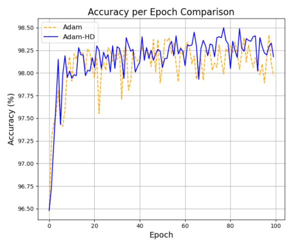
- **Alpha (α_t) Plot:** Typically, one would expect to see a high initial alpha during the "aggressive" exploration phase, which then decreases and stabilizes as the optimizer gets closer to the minimum and enters a "cautious" fine-tuning phase.
- **Beta (β_t) Plot:** The beta plot should show a strong correlation with the alpha plot. During the aggressive phase, beta might be high to build momentum. Crucially, as alpha drops, a well-coordinated scheduler should also force beta to drop, demonstrating the logic of reducing inertia near the minimum.
- **Gamma (γ_t) Plot:** This plot shows how the memory of the second moment adapts. A high, stable gamma indicates the optimizer perceives the path as predictable, while fluctuations in gamma would suggest it is adapting to a more chaotic or changing landscape.

Image 3: Adam-HD versus Adam (baseline) Comparison

Description The plots compare Adam-HD and Adam on MNIST over 100 epochs. Both optimizers quickly reduce training loss and stabilize at low values; Adam's curve is smoother, while Adam-HD shows minor oscillations. In test accuracy, both perform well, but Adam-HD averages slightly higher, maintaining ~98%–98.5%, whereas Adam sometimes dips below this range. From epoch 20 onwards, average Loss was 0.0082 (Adam) and 0.0143 (Adam-HD), and average Accuracy was 98.15% (Adam) and 98.25% (Adam-HD).



(a) Adam-HD versus Adam (Baseline) Loss.



(b) Adam-HD versus Adam (Baseline) Accuracy.

Figure 6. Adam-HD vs Adam comparison. Loss (a) shows stable convergence; Accuracy (b) shows robust learning with slightly higher performance for Adam-HD.

Algorithm: Adam-HD

Algorithm: Adam-HD, our proposed optimizer with a dynamic flattening factor and normalized gradient signal.

```
Require: Good default settings are:
    $\beta_{\text{range}} = (0.85, 0.99)$, $\gamma_{\text{range}} = (0.9, 0.999)$, $\beta_{\text{smooth}} = 0.9$, $\beta_{\text{norm}} = 0.99$, $\epsilon = 10^{-8}$.
    The following fixed ranges were optimized via Optuna:
Require: $\alpha_{\text{range}} = (0.000014, 0.001403)$
Require: $T_{\text{range}} = (1.0, 83.1290)$ (Dynamic Flattening Factor Range)
Require: $f(\phi)$ (Stochastic objective function)
Require: $\phi_0$ (Initial parameter vector)
 1: $m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$
 2: $g_0 \leftarrow 0, \bar{g}_0 \leftarrow 0, z_0 \leftarrow 0, c_0 \leftarrow 0, \bar{c}_0 \leftarrow 0$
 3: while $\phi_t$ not converged do
 4:     $t \leftarrow t + 1$
 5:     $g_t \leftarrow \nabla_\phi f_t(\phi_{t-1})$
        {— Adam-HD: Control signal calculations —}
 6:     $\bar{g}_t \leftarrow \beta_{\text{norm}} \cdot \bar{g}_{t-1} + (1 - \beta_{\text{norm}}) \cdot g_t^{\text{norm}}$ {Update moving avg of grad norm}
 7:     $z_t \leftarrow (g_t^{\text{norm}} / (\bar{g}_t + \epsilon)) - 1$ {Compute normalized error signal}
 8:     $c_t \leftarrow \frac{g_t \cdot g_{t-1}}{g_t^{\text{norm}} \cdot g_{t-1}^{\text{norm}}}$ {Instantaneous directional consistency}
 9:     $\bar{c}_t \leftarrow \beta_{\text{smooth}} \cdot \bar{c}_{t-1} + (1 - \beta_{\text{smooth}}) \cdot c_t$
        {— Hierarchical control factors —}
10:     $f_t^{\text{dir}} \leftarrow (\bar{c}_t + 1)/2$
11:     $T_t \leftarrow T_{\max} - (T_{\max} - T_{\min}) \cdot f_t^{\text{dir}}$ {Compute dynamic flattening factor}
12:     $f_t^{\text{aggress}} \leftarrow 0.5 \cdot (z_t / (T_t + |z_t|) + 1)$
        {— Dynamic hyperparameter computation —}
13:     $\alpha_t \leftarrow \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot f_t^{\text{aggress}}$
14:     $\beta_t^{\text{target}} \leftarrow \beta_{\min} + (\beta_{\max} - \beta_{\min}) \cdot f_t^{\text{dir}}$
15:     $\beta_t \leftarrow f_t^{\text{aggress}} \cdot \beta_t^{\text{target}} + (1 - f_t^{\text{aggress}}) \cdot \beta_{\min}$
16:     $\gamma_t \leftarrow \gamma_{\min} + (\gamma_{\max} - \gamma_{\min}) \cdot f_t^{\text{dir}}$
        {— Adam update steps with dynamic values —}
17:     $m_t \leftarrow \beta_t \cdot m_{t-1} + (1 - \beta_t) \cdot g_t$
18:     $v_t \leftarrow \gamma_t \cdot v_{t-1} + (1 - \gamma_t) \cdot g_t^2$
19:     $\hat{m}_t \leftarrow m_t / (1 - \beta_t^t)$
20:     $\hat{v}_t \leftarrow v_t / (1 - \gamma_t^t)$
21:     $\phi_t \leftarrow \phi_{t-1} - \alpha_t \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
22:     $g_{t-1} \leftarrow g_t$
23: end while
24: return $\phi_t$
```

Adapted by the authors from (Kingma & Ba, 2015).