Connectome-Guided Optimization for Deep Networks

Peilin He
Duke Kunshan University

Tananun Songdechakraiwut
Duke University

Abstract

The human brain is highly adaptive: its functional connectivity reconfigures on multiple timescales during cognition and learning, enabling flexible information processing. By contrast, artificial neural networks typically rely on manually-tuned learning-rate schedules or generic adaptive optimizers whose hyperparameters remain largely agnostic to a model's internal dynamics. In this paper, we propose Connectome-Guided Automatic Learning Rate (CG-ALR) that dynamically constructs a functional connectome of the neural network from neuron co-activations at each training iteration and adjusts learning rates online as this connectome reconfigures. This connectomics-inspired mechanism adapts step sizes to the network's dynamic functional organization, slowing learning during unstable reconfiguration and accelerating it when stable organization emerges. Our results demonstrate that principles inspired by brain connectomes can inform the design of adaptive learning rates in deep learning, with particularly consistent improvements over traditional SGD-based schedules and competitive performance against Adam-family scheduled baselines and recent adaptive methods.

Code — `https://github.com/peilin128/CG-ALR`

1 INTRODUCTION

Stochastic gradient descent (SGD) (Robbins and Monro, 1951), RMSProp (Tieleman and Hinton, 2012), and Adam (Kingma and Ba, 2015) are indispensable optimization tools in modern machine learning. Yet their effectiveness hinges critically on the choice of learning rate. If the step size is too large, updates may become unstable and diverge; if too small, training can stagnate. Moreover, there is no universal "one-size-fits-all" schedule, as different regions of the parameter space or different phases of training may require drastically different learning rate magnitudes.

To address this challenge, many handcrafted learning-rate schedules have been developed, such as exponential decay, step decay, cosine annealing, and plateau-based reduction. While effective in some scenarios, these schedules are time-driven or loss-triggered heuristics that do not directly reflect the intrinsic state of the evolving representation. More recently, parameter-free adaptive methods have been introduced, such as Distance-over-Gradients (DoG) (Ivgi and Globerson, 2023), which sets the step size as the ratio between the running maximum of the parameter distance from initialization and the root of the accumulated squared gradient norms. While reducing manual tuning, such methods rely solely on first-order, gradient-magnitude-based signals and are agnostic to the evolving topology of internal representations, potentially missing important dynamics in deep networks.

In this work, we propose a novel *Connectome-Guided Automatic Learning Rate* (CG-ALR) framework that leverages persistent homology to analyze the topological reconfiguration of intermediate functional connectomes and adapt step sizes accordingly. Rather than relying on the loss-based triggers or time-driven decay rules, CG-ALR drives the learning rate by persistent-homology-quantified representation change. By applying persistent homology to the evolving connectivity graphs, we obtain a topological time series that summarizes functional changes in the network over time.

We show that monitoring the dynamics of this topological time series, quantified via Wasserstein distances between persistence diagrams (Skraba and Turner, 2020), provides an efficient schedule for adaptively changing the learning rate. Our experiments on image and graph classification benchmarks demon-

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

strate that CG-ALR achieves competitive or superior performance relative to a broad range of optimization baselines, including traditional SGD-based schedules (SGD-Constant, SGD-Cosine, SGD-Step, SGD-Exp, and SGD-Plateau (Goodfellow et al., 2016; PyTorch, 2023)), Adam-family scheduled baselines, namely Adam-Cosine, Adam-OneCycle, AdamW-Cosine, and AdamW-OneCycle, formed by combining Adam (Kingma and Ba, 2015) or AdamW (Loshchilov and Hutter, 2019) with cosine annealing (Loshchilov and Hutter, 2017) or the one-cycle policy (Smith and Topin, 2018), and DoG, a recent parameter-free adaptive method. Across these comparisons, CG-ALR is often competitive with or superior to the baselines, with particularly consistent gains over SGD-based schedules.

Our main contributions are as follows:

- We propose *Connectome-Guided Automatic Learning Rate* (CG-ALR), a topology-aware adaptive mechanism that leverages persistent homology to capture the representational dynamics of the network's changing internal functional connectomes.
- We design a robust change-detection signal for CG-ALR with hysteresis and a late-phase multiplier, reducing jitter and preventing premature annealing.
- We provide extensive empirical validation across diverse datasets and architectures, showing that CG-ALR is competitive with or superior to a diverse set of optimization baselines, including traditional SGD-based schedules, Adam-family scheduled baselines, and advanced parameter-free adaptive methods.

2 METHODS

2.1 Functional Connectomes

Consider a training set $\mathcal{X} = \{x^{(1)}, \ldots, x^{(N)}\}$, where each sample $x^{(i)}$ is passed through a neural network. We fix a probe set $\mathcal{X}_{\text{probe}} \subset \mathcal{X}$ of size P_{probe}. The probe set is sampled once per run from the training set (stratified by class) and reused at every epoch. We focus on a collection of P *activation signals*, which may correspond to neurons in a fully connected layer, channels in a convolutional layer, or even units drawn from multiple layers together. For probe input $x^{(i)} \in \mathcal{X}_{\text{probe}}$ and activation index $j \in \{1, \ldots, P\}$, let the scalar response be s_{ij}. Collecting these responses forms the matrix $A = [s_{ij}] \in \mathbb{R}^{P_{\text{probe}} \times P}$, where row i corresponds to a probe sample $x^{(i)}$. The activation profile of signal j across the probe set is then $\mathbf{a}_j = [s_{1j}, \ldots, s_{P_{\text{probe}},j}]^\top$.

To study relationships among activations, we ask whether two signals tend to vary together across the probe set. This correlation provides a measure of statistical dependency between their activation profiles. Following conventions in neuroscience and connectomics, we adopt the widely-used Pearson correlation coefficient: $\rho_{pq} = \text{Cov}(\mathbf{a}_p, \mathbf{a}_q)/\big(\sigma(\mathbf{a}_p)\sigma(\mathbf{a}_q)\big)$, where Cov denotes covariance and σ denotes standard deviation. The *functional connectome* is then defined as the weighted adjacency matrix $M \in \mathbb{R}^{P \times P}$, with entries $M_{pq} = |\rho_{pq}|$ for $p \neq q$ and $M_{pp} = 0$, so that M captures pairwise statistical dependencies among activation signals, excluding self-loops.

When the activation signals are drawn from a fully connected layer, each corresponds to a neuron applying an affine transformation followed by a nonlinearity. Its activation profile $\mathbf{a}_j$ is simply the vector of activations across samples.

This construction extends naturally to other architectures. For example, in a convolutional layer, each activation signal corresponds to a filter producing activation maps $z_{ij} \in \mathbb{R}^{H \times W}$ for sample $x^{(i)}$, where H and W are the spatial dimensions after convolution. A reduction operator (e.g., mean pooling, max pooling, or global norm) maps each z_{ij} to a scalar s_{ij}, yielding activation profiles $\mathbf{a}_j$ from which correlations are computed as before.

2.2 Topological Connectome Distances

Given two functional connectomes $M^{(t)}, M^{(t+1)} \in \mathbb{R}^{P \times P}$ obtained at consecutive training epochs, we quantify their dissimilarity using *persistent homology* (PH) and its graph-based variant, *persistent graph homology* (PGH) (Songdechakraiwut et al., 2021; Songdechakraiwut and Chung, 2023). These methods capture higher-order topological features of connectomes (e.g., loops and components) beyond pairwise edge comparisons, and the stability theorem guarantees robustness to small perturbations (Skraba and Turner, 2020). We treat each connectome as a weighted undirected graph and define a distance $D(M^{(t)}, M^{(t+1)})$ from topological signatures of their edge weights, measuring how much the network's functional connectivity patterns change from one epoch to the next.

Formally, the *p-Wasserstein distance* between two persistence objects X and Y (e.g., diagrams or vectors) is defined as

$$W_p(X, Y) = \left(\inf_{\gamma \in \Gamma(X,Y)} \sum_{(x,y) \in \gamma} \|x - y\|^p \right)^{1/p},$$

where X and Y are multisets of points and $\Gamma(X, Y)$

is the set of all valid matchings, including optional diagonal projections for unmatched features.

PGH. Starting from the adjacency matrix $M^{(t)}$, we extract its maximum spanning tree T. The remaining edges outside T form 1-cycles, whose weights are regarded as death times. Sorting these values yields a persistence vector $\mathbf{d}^{(t)} = [d_1^{(t)} \leq \cdots \leq d_k^{(t)}]$ (Songdechakraiwut et al., 2023). The distance between consecutive persistence vectors, referred to as TOP, is defined using the 1-Wasserstein distance (Songdechakraiwut and Wu, 2025; Wu et al., 2025):

$$\mathrm{TOP}(\mathbf{d}^{(t)}, \mathbf{d}^{(t+1)}) = \sum_{j=1}^{k} \left| d_j^{(t)} - d_j^{(t+1)} \right|.$$

Thus, $D(M^{(t)}, M^{(t+1)}) = \mathrm{TOP}(\mathbf{d}^{(t)}, \mathbf{d}^{(t+1)})$ provides one way to compare connectomes.

PH. We perform a Vietoris–Rips filtration on the standard correlation-distance matrix $D^{(t)}$, where $D_{pq}^{(t)} = \sqrt{2\big(1-\rho_{pq}^{(t)}\big)}$, yielding a persistence diagram $\mathrm{PD}^{(t)} = \{(b_i^{(t)}, d_i^{(t)})\}_i$, where each point $(b_i^{(t)}, d_i^{(t)})$ represents the birth and death of a topological feature (e.g., connected component or loop). The distance between consecutive diagrams, denoted WD, is defined as the 2-Wasserstein distance:

$$\mathrm{WD}\big(\mathrm{PD}^{(t)}, \mathrm{PD}^{(t+1)}\big) = \left(\inf_{\gamma} \sum_{i} \left\| (b_i^{(t)}, d_i^{(t)}) - \gamma(b_i^{(t+1)}, d_i^{(t+1)}) \right\|_2^2 \right)^{1/2}.$$

Thus, $D(M^{(t)}, M^{(t+1)}) = \mathrm{WD}(\mathrm{PD}^{(t)}, \mathrm{PD}^{(t+1)})$ provides another way to compare connectomes.

Beyond TOP and WD, we also consider the bottleneck distance (BD) (Cohen-Steiner et al., 2007), the heat kernel distance (HK) (Reininghaus et al., 2015), and the sliced Wasserstein kernel (SWK) (Carriere et al., 2017). Implementation details are provided in the Appendix.

2.3 Topological Connectome Dynamics

We construct a time series of distances to capture the dynamics of functional connectomes during training. Let the number of training epochs be $t = 1, 2, \ldots, T-1$. At each step, we compute $\delta_t = D(M^{(t)}, M^{(t+1)})$, which quantifies the topological change from epoch t to $t+1$.

To smooth fluctuations and emphasize underlying trends, we apply an exponential moving average:

$$\tilde{\delta}_t = (1-\lambda)\,\delta_t + \lambda \tilde{\delta}_{t-1}, \quad \tilde{\delta}_1 = \delta_1,$$

Algorithm 1 Online Connectome-Guided Learning Rate

Require: initial learning rate $\eta^\star$; RM envelope ($t_0 \geq 1$, $\alpha \in (\frac{1}{2}, 1]$); global steps counter s; multipliers $(\gamma_\downarrow, \gamma_\uparrow, \gamma_{\text{late}})$; late split N_{late}; epochs T; batches per epoch B.
Ensure: learning-rate schedule $\{\eta_{t,b}\}$; controller $\{\psi_t\}$

$\psi_0 \leftarrow 1$; $s \leftarrow 0$;
$\eta_0 \leftarrow \eta^\star \cdot t_0^\alpha$
for $t = 1$ to T **do**
  for $b = 1$ to B **do**
    $\bar{\eta} \leftarrow \eta_0/(s+t_0)^\alpha$ *// RM envelope*
    $\eta_{t,b} \leftarrow \bar{\eta} \cdot \psi_{t-1}$
    $s \leftarrow s + 1$
  end for
  $z_t \leftarrow \mathrm{ComputeTopoSignal}(t)$ *// Eq. (1)*
  if still in early epochs **then**
    $u_t \leftarrow 1$
  else
    $\varepsilon_t \leftarrow \mathrm{AdaptiveThreshold}(t)$ *// Eq. (3)*
    if cooldown active **then**
      $u_t \leftarrow 1$
    else if $z_t > \varepsilon_t$ **then**
      $u_t \leftarrow \gamma_\downarrow$ *// Aggressive downscale*
    else
      if $t \leq N_{\text{late}}$ **then**
        $u_t \leftarrow \gamma_\uparrow$ *// Moderate upscale*
      else
        $u_t \leftarrow \gamma_{\text{late}}$ *// Milder downscale in late phase*
      end if
    end if
  end if
  $\psi_t \leftarrow \mathrm{clip}(\psi_{t-1} \cdot u_t,\ \psi_{\min},\ \psi_{\max})$
end for
return $\{\eta_{t,b}\},\ \{\psi_t\}$

with smoothing factor $\lambda \in [0, 1)$.

Finally, we normalize the smoothed distances using a rolling median–MAD scaling, which recenters the signal and rescales it by a robust measure of variability, allowing stable thresholding across epochs and runs. Here, the statistics are computed using all epochs observed up to time t:

$$z_t = \frac{\tilde{\delta}_t - \mathrm{median}(\{\tilde{\delta}_u\}_{u=1}^t)}{\mathrm{MAD}(\{\tilde{\delta}_u\}_{u=1}^t) + \tau}, \tag{1}$$

where $\tau > 0$ is a small constant for numerical stability and

$$\mathrm{MAD}\Big(\{\tilde{\delta}_u\}_{u=1}^t\Big) = \mathrm{median}\big(|\tilde{\delta}_u - \mathrm{median}(\{\tilde{\delta}_r\}_{r=1}^t)|\big).$$

The resulting normalized signal z_t provides a compact and robust measure that adapts online, and we later use it in Section 2.4 to guide adaptive learning rate updates.

2.4 Connectome-Guided Adaptive Learning Rate

Conventional learning-rate schedules rely on loss or accuracy curves, which are insensitive to changes in

the internal representation topology. To address this limitation, we propose *Connectome-Guided Adaptive Learning Rate* (CG-ALR), which use the normalized connectome topological signal z_t as feedback to adapt the learning rate in a stable yet responsive manner.

Large z_t values signal unstable representation topology, warranting slower learning to avoid destabilization. Small values indicate stability, permitting faster learning supported by more reliable gradients. We therefore update the next-step rate by

$$\eta_{t+1} = \begin{cases} \gamma_\downarrow \, \eta_t, & \text{if } z_t > \varepsilon_t, \\ \gamma_\uparrow \, \eta_t, & \text{if } z_t \le \varepsilon_t \text{ and } t \le N_{\text{late}}, \\ \gamma_{\text{late}} \, \eta_t, & \text{if } z_t \le \varepsilon_t \text{ and } t > N_{\text{late}}. \end{cases} \tag{2}$$

The adaptive threshold is defined by

$$\varepsilon_t = \text{median}(\{z_u\}_{u=1}^{t}) + k_{\text{MAD}} \cdot \text{MAD}(\{z_u\}_{u=1}^{t}), \tag{3}$$

where $k_{\text{MAD}} > 0$ controls the sensitivity. This threshold adapts through a cumulative median–MAD statistic, ensuring robustness by responding to relative rather than absolute fluctuations in topology. Here, $\gamma_\downarrow < 1$ is the downscale factor, $\gamma_\uparrow > 1$ is the upscale factor, and in the late phase we apply an additional decay $\gamma_{\text{late}} \in (\gamma_\downarrow, 1)$ to encourage steady convergence.

Equivalently, the update rules in Eq. (2) can be expressed using a multiplicative controller ψ_t:

$$\psi_t = \text{clip}(\psi_{t-1} \cdot u_t, \ \psi_{\min}, \ \psi_{\max}) \, ,$$

with $u_t \in \{\gamma_\downarrow, \gamma_\uparrow, \gamma_{\text{late}}, 1\}$, and $\text{clip}(x, a, b) = \min\{b, \max\{a, x\}\}$. The baseline schedule $\bar{\eta}(s)$ evolves at the batch level, while the controller ψ_t evolves at the epoch level. The effective learning rate for batch b in epoch t is

$$\eta_{t,b} = \bar{\eta}(s) \cdot \psi_{t-1},$$

where s is the cumulative batch index (i.e., $s = tB+b$ if there are B batches per epoch). The Robbins–Monro (RM) baseline (Robbins and Monro, 1951) is

$$\bar{\eta}(s) = \frac{\eta_0}{(s + t_0)^\alpha}, \quad \alpha \in \left(\tfrac{1}{2}, 1\right] ,$$

with $\eta_0 = \eta^\star \cdot t_0^\alpha$ chosen so that the initial step size is calibrated relative to the target $\eta^\star$. Thus, $\bar{\eta}(s)$ provides fine-grained decay across mini-batches, while ψ_t introduces slower, topology-driven adjustments across epochs. To further avoid chattering, we require that the condition on z_t hold for several consecutive epochs before triggering an update, and we impose a cooldown period of a few epochs during which no further adjustments are allowed.

By coupling the optimizer with the topology-driven signal z_t, CG-ALR adapts step sizes by slowing learning during unstable reconfiguration and accelerating it when stable organization emerges, improving both training stability and generalization (see Algorithm 1). Reference code implementing CG-ALR is provided in the supplementary material.

Table 1: Model Architecture for CIFAR-10/100 and Mini-ImageNet. Each layer is shown as [*input dimension, output dimension*].

Layer	**CIFAR-10/100/Mini-ImageNet**
Backbone	ResNet-18
Flatten	[512]
FC1	$[512, 512]$
FC2	$[512, 256]$
FC3	$[256, C]$
Output	$C = 10/100/100$

Theorem 1. *Our controller preserves the Robbins–Monro envelope, and thus the proposed CG-ALR algorithm enjoys the same convergence guarantees: the iterates almost surely approach stationary points, and under PL/KL conditions converge to minimizers.*

Proof. See Appendix for the complete proof. □

3 EXPERIMENTS

Datasets. We conducted experiments on six benchmark datasets. For images, we use CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and Mini-ImageNet (Vinyals et al., 2016), which contain natural images spanning multiple object categories. For graphs, we use MUTAG, PROTEINS, and ENZYMES (Morris et al., 2020), which involve molecular and protein structures for classification. Further dataset details are provided in the Appendix.

Architecture and Optimization. For image datasets (CIFAR-10/100 and Mini-ImageNet), we used a ResNet-18 (He et al., 2016) backbone with a three-layer fully connected (FC) classifier head; the configuration is summarized in Table 1. For graph datasets (MUTAG, PROTEINS, ENZYMES), we used three graph convolutional layers with GCN (Kipf and Welling, 2016) and GAT (Velčković et al., 2017) backbones, followed by pooled features fed into a FC head with one, two, or three layers to study the effect of classifier depth; the configuration is summarized in Table 2.

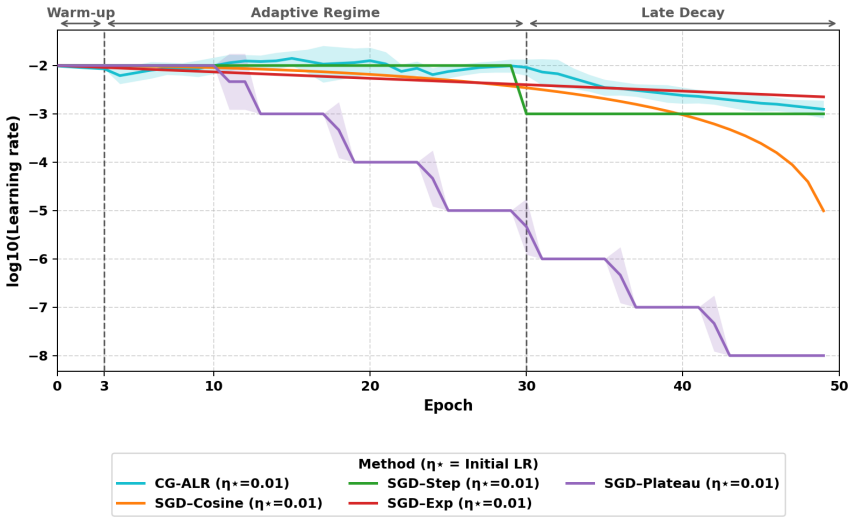

Figure 1: Learning Rate Dynamics Across Epochs on CIFAR-10. Comparison of our proposed CG-ALR with TOP against standard schedules that require initial learning rates: SGD-Cosine, SGD-Step, SGD-Exp, and SGD-Plateau.

Table 2: Model Architectures for Graph Datasets (MUTAG, PROTEINS, ENZYMES). Each layer is shown as [*input dimension, output dimension*].

Layer	**MUTAG**	**PROTEINS/ENZYMES**
Backbone	GCN/GAT	GCN/GAT
Pooling	[64]	[128]
FC1	$[64, 64]/[64, C]$	$[128, 128]/[128, C]$
FC2 (optional)	$[64, 64]/[64, C]$	$[128, 128]/[128, C]$
FC3 (optional)	$[64, C]$	$[128, C]$
Output	$C = 2$	$C = 2/6$

3.1 Study 1: Dynamics of Adaptive Learning Rates

This study analyzes the trajectory of adaptive learning rate dynamics with an initial value of $\eta^\star = 0.01$. In this study, we incorporate TOP into our CG-ALR and compare it against four standard SGD-based schedules that also require initial learning rates. As shown in Figure 1, SGD-Step and SGD-Plateau reduce the learning rate by orders of magnitude early in training, while SGD-Cosine and SGD-Exp shrink it monotonically toward minimal values, slowing late-stage progress. In contrast, CG-ALR tracks representation reorganization via the connectome signal and produces a gradual, on-demand trajectory, with the learning rate remaining flat during the warm-up stage (first 3 epochs), adapting in the intermediate stage, and decaying only in the late stage (after 30 epochs), thereby avoiding premature annealing.

3.2 Study 2: Ablation Analyses

In this study, we conduct ablation analyses to understand how design choices affect the performance of CG-ALR. We focus on three factors: (i) the number of FC layers used in the classifier head; (ii) the probe set size used to compute the connectome, denoted P_{probe}; and (iii) the sensitivity to key controller hyperparameters in the learning-rate update.

Effect of FC Layer Depth. We vary how connectomes are extracted by adjusting the number of FC layers used in the GCN model. To compare depths, we aggregate accuracy, generalization stability, and convergence efficiency into a single composite score. For each dataset and FC depth $d \in \{1, 2, 3\}$, let $\overline{m}_d$ denote the mean of metric m across CG-ALR methods, seeds, and initial learning-rate values. We standardize each metric across depths within a dataset as

$$z(\overline{m}_d) = \frac{\overline{m}_d - \frac{1}{3}\sum_{d'=1}^{3} \overline{m}_{d'}}{\sqrt{\frac{1}{3}\sum_{d'=1}^{3}\left(\overline{m}_{d'} - \frac{1}{3}\sum_{d''=1}^{3} \overline{m}_{d''}\right)^2}},$$

with $z(\overline{m}_d) = 0$ if the denominator is zero. The composite score for depth d is then

$$\text{Composite}(d) = \sum_{p \in \mathcal{P}} z(\overline{p}_d) - \sum_{g \in \mathcal{G}} z(\overline{g}_d) - \sum_{c \in \mathcal{C}} z(\overline{c}_d),$$

where

- $\mathcal{P}$ (performance): test accuracy at best-validation epoch, test accuracy at final epoch, and maximum validation accuracy;

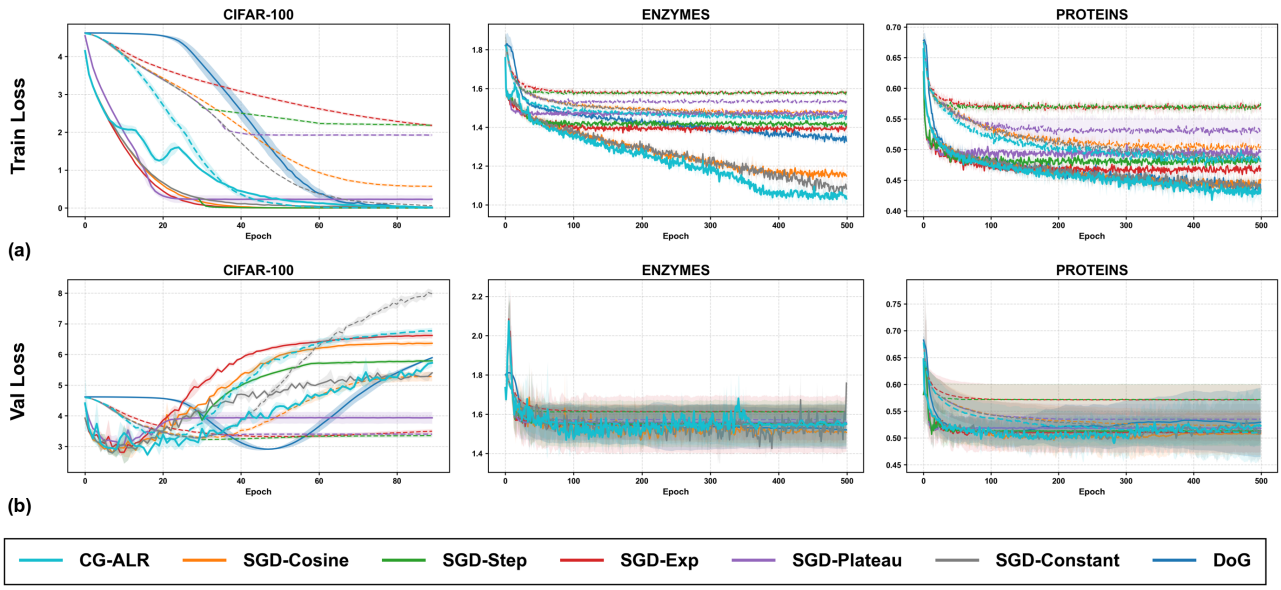

Figure 2: (a) Training Loss on Three Datasets (**CIFAR-100**, **ENZYMES**, **PROTEINS**). Solid lines are the averages across seeds for the *best* learning-rate choice from the predefined grid ($\eta^\star \in \{0.1, 0.01, 0.001\}$ for CG-ALR and SGD-based schedules; $\eta \in \{0.1, 0.01, 0.001\}$ for SGD-Constant; default hyperparameters for DoG). Dashed lines are the averages across seeds for the *worst* choice from the same grid. Shaded bands denote the min–max range across three seeds. For CG-ALR, we use **TOP** on CIFAR-100 and **SWK** on ENZYMES/PROTEINS. (b) Validation Loss on the Same Three Datasets for the Same Set of Optimizers.

- $\mathcal{G}$ (generalization): absolute test–val gap at best-validation epoch, test–val gap at final epoch, train–val gap (accuracy) at final epoch, train–val gap (loss) at final epoch;
- $\mathcal{C}$ (convergence): wall-clock time to reach max validation accuracy.

Higher values indicate better overall performance (higher accuracy, smaller generalization gaps, and faster convergence).

Across datasets, Table 3 shows that a single FC layer is optimal for MUTAG, while two FC layers consistently perform best on ENZYMES and PROTEINS. Adding a third FC layer does not provide additional gains, indicating that efficient training does not require deeper stacks. Instead, constructing the target connectome with the appropriate FC depth is sufficient to capture representational dynamics and enable strong, efficient performance.

Table 3: Composite Scores Across Datasets and FC Layers. Higher values mean better overall performance; **bold** marks the best score for each dataset.

	FC Layer(s) Depth		
Dataset	1	2	3
MUTAG	**7.043**	3.957	-10.999
ENZYMES	-3.290	**4.721**	-1.432
PROTEINS	-2.160	**3.383**	-1.224

Effect of Probe Set Size. We investigate the role of probe set size in CG-ALR, as using more samples to construct the connectomes improves the precision of the topology-change signal z_t, but also increases computational cost. To compare sizes, we fix the default training configuration (ResNet-18 backbone with FC depth d=3), and vary only the probe set size $P_{\text{probe}} \in \{64, 256, 1024, 2096\}$. We then use the same composite score to compare the performance on different probe set sizes. As shown in Table 4, the performance reaches a clear peak when $P_{\text{probe}} = 1024$. This indicates that efficient training does not require a larger probe set. Based on this, we set $P_{\text{probe}} = 1024$ as the default in subsequent experiments.

Effect of Hyperparameters. We evaluate the sensitivity of CG-ALR to the controller hyperparameters governing the multiplicative state ψ_t, which is updated online as $\psi_t = \text{clip}(\psi_{t-1} \cdot u_t, \psi_{\min}, \psi_{\max})$, with $u_t \in \{\gamma_{\text{down}}, 1, \gamma_{\text{up}}, \gamma_{\text{late}}\}$ selected from the smoothed topology signal z_t. We vary the main controller param-

Table 4: Composite Scores Across Probe Set Sizes for CIFAR-10. Higher values mean better overall performance; **bold** marks the best score.

	Probe Set Size			
Dataset	64	256	1024	2096
CIFAR-10	-2.50	-1.55	**7.55**	-3.50

eters over broad ranges on all datasets, including $\gamma_{\text{up}} \in [1.08, 1.22]$, $\gamma_{\text{down}} \in [0.50, 0.88]$, $N_{\text{ratio}} \in [0.60, 0.90]$, and $\psi_{\max} \in [2, 20]$, while also perturbing β, τ, and the robust window length. Across these settings, test accuracy, RED, and learning-rate trajectories remain broadly stable.

3.3 Study 3: Performance Comparisons

This study evaluates the overall effectiveness of CG-ALR across image and graph classification tasks, benchmarking it against a range of widely-used optimization baselines. For the image datasets, we train ResNet-18 from scratch; for the graph datasets, we train GCN and GAT from scratch; and then compute connectomes from the neuron co-activations of an FC layer. For the methods described in Section 3.1, we consider initial learning rates $\eta^\star \in \{0.1, 0.01, 0.001\}$, a widely-adopted range for these tasks. For SGD-Constant, we use $\eta \in \{0.1, 0.01, 0.001\}$. For the Adam-family baselines, namely Adam-Cosine, Adam-OneCycle, AdamW-Cosine, and AdamW-OneCycle, we sweep learning rates $\text{lr} \in \{0.1, 0.01, 0.001\}$. Adam and AdamW use $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The cosine variants employ cosine annealing with $T_{\max}$ set to the total number of training epochs, whereas the one-cycle variants use OneCycleLR with $\max_{\text{l}}\text{r}$ set to the chosen learning rate and $\text{steps}_{\text{p}}\text{er}_{\text{e}}\text{poch}$ determined by the training loader. For DoG, we adopt the default hyperparameters $r_\varepsilon = 10^{-6}$ and $\varepsilon = 10^{-8}$. Further implementation details are provided in the Appendix.

Figure 2 reports training and validation loss for three representative datasets (CIFAR-100, ENZYMES, and PROTEINS), with the remaining datasets shown in the Appendix. CG-ALR consistently accelerates optimization: its training loss nearly always decreases the fastest and reaches the lowest level across schedules. On the two graph datasets, CG-ALR also achieves the lowest validation loss both throughout training and at convergence. Moreover, for CG-ALR the gap between the best and worst initial learning rates (i.e., the strongest and weakest choices within $\{0.1, 0.01, 0.001\}$) is consistently smaller than for the baselines, suggesting that CG-ALR provides more robust performance across different initial learning rates.

Since raw performance metrics vary substantially across methods, aggregate comparisons are challenging. To address this, we use the *relation error difference* (RED) (Ivgi and Globerson, 2023) as a normalized measure of performance differences. Given the test accuracy *acc* (measured at the epoch with the highest validation accuracy), we define the test error as $err = 1 - acc$. Let $err_{\text{CG-ALR}}$ denote the error when trained with CG-ALR. Then,

$$\text{RED}(\text{err}_x, \text{err}_{\text{CG-ALR}}) = \frac{\text{err}_{\text{CG-ALR}} - \text{err}_x}{\text{err}_{\text{CG-ALR}}},$$

where x represents a baseline method. Negative RED indicates that CG-ALR's generalization performance outperforms the baseline (lower test error, higher test accuracy), while positive RED indicates the opposite. We also report confidence intervals (CIs) to quantify uncertainty and assess reliability. Specifically, we compute 95% CIs for the median RED using a seed-level percentile bootstrap with 1,000 resamples (resampling seeds with replacement and recomputing the median). CIs entirely below 0 indicate that CG-ALR is statistically significantly outperforming the baselines, CIs entirely above 0 indicate the baseline is statistically significantly better, and CIs overlapping 0 are inconclusive.

As summarized in Table 5 and Table 6, when CG-ALR outperforms the baselines (negative RED), many CIs lie entirely below 0, indicating statistically significant improvements in generalization. By contrast, when baselines achieve higher average RED values (positive RED), the corresponding CIs are often inconclusive, although statistically significant baseline advantages do occur in some settings, particularly on certain graph tasks. Overall, CG-ALR achieves superior generalization performance to baselines in most cases, with stronger statistical support for its advantages. Additional experimental results are provided in the Appendix.

We also examine the computational cost of CG-ALR. From a theoretical standpoint, the topology-guided update adds $O(P^2 \log P)$ time and $O(P^2)$ memory per call for a probe set of size P, independent of the full training-set size. Empirically, CG-ALR remain substantially faster than DoG, although they still incur additional wall-clock cost relative to the fastest SGD-based and Adam-family baselines; full timing and memory-usage results are provided in the Appendix.

4 BROADER IMPACT

Our work demonstrates that a network's internal functional connectome can serve as a principled and interpretable signal for guiding optimizer learning-rate de-

Table 5: Median RED with 95% Percentile-Bootstrap CIs Across Seeds. **Bold** entries indicate cases where CG-ALR achieves better generalization performance (RED < 0), or statistically significant improvements when the 95% CI lies entirely below 0. Results are shown side by side for CIFAR-10 and CIFAR-100.

		CIFAR-10	**CIFAR-100**
CG-ALR	**Baseline**	RED (95% CI)	RED (95% CI)
BD	DoG	**-0.255** (-0.421, 0.046)	**-0.143** (**-0.192**,**-0.015**)
BD	SGD-Cosine	**-0.125** (**-0.238**, **-0.032**)	**-0.096** (**-0.175**,**-0.009**)
BD	SGD-Exp	**-0.143** (**-0.226**, **-0.028**)	**-0.141** (**-0.180**,**-0.021**)
BD	SGD-Plateau	**-0.165** (**-0.194**, **-0.023**)	**-0.107** (**-0.150**,**-0.013**)
BD	SGD-Step	**-0.102** (**-0.211**, **-0.021**)	**-0.090** (**-0.157**,**-0.021**)
BD	SGD-Constant	**-0.142** (**-0.229**, **-0.101**)	**-0.107** (**-0.164**,**-0.058**)
HK	DoG	**-0.202** (-0.448, 0.025)	**-0.135** (**-0.186**,**-0.010**)
HK	SGD-Cosine	**-0.094** (**-0.260**, **-0.047**)	**-0.075** (**-0.177**,**-0.006**)
HK	SGD-Exp	**-0.106** (**-0.225**, **-0.068**)	**-0.121** (**-0.190**,**-0.012**)
HK	SGD-Plateau	**-0.122** (**-0.213**, **-0.058**)	**-0.093** (**-0.136**,**-0.002**)
HK	SGD-Step	**-0.074** (**-0.233**, **-0.055**)	**-0.073** (**-0.147**,**-0.018**)
HK	SGD-Constant	**-0.120** (**-0.252**, **-0.058**)	**-0.081** (**-0.146**,**-0.034**)
SWK	DoG	**-0.241** (**-0.415**, **-0.014**)	**-0.153** (**-0.189**,**-0.012**)
SWK	SGD-Cosine	**-0.112** (**-0.216**, **-0.027**)	**-0.096** (**-0.182**,**-0.006**)
SWK	SGD-Exp	**-0.124** (**-0.289**, **-0.023**)	**-0.141** (**-0.188**,**-0.021**)
SWK	SGD-Plateau	**-0.152** (**-0.218**, **-0.019**)	**-0.110** (**-0.158**,**-0.010**)
SWK	SGD-Step	**-0.081** (**-0.237**, **-0.017**)	**-0.096** (**-0.163**,**-0.019**)
SWK	SGD-Constant	**-0.133** (**-0.256**, **-0.102**)	**-0.107** (**-0.171**,**-0.052**)
SWK	Adam-Cosine	**-0.006** (-0.700, 0.202)	**-0.051** (-0.723, 0.125)
SWK	Adam-OneCycle	0.101 (-0.343, 0.137)	0.042 (-0.386, 0.064)
SWK	AdamW-Cosine	**-0.008** (-0.730, 0.202)	**-0.051** (-0.723, 0.125)
SWK	AdamW-OneCycle	0.103 (-0.377, 0.125)	0.042 (-0.379, 0.064)
TOP	DoG	**-0.184** (-0.410, 0.025)	**-0.153** (**-0.189**,**-0.006**)
TOP	SGD-Cosine	**-0.078** (**-0.216**, **-0.041**)	**-0.096** (**-0.160**,**-0.027**)
TOP	SGD-Exp	**-0.101** (**-0.289**, **-0.019**)	**-0.141** (**-0.166**,**-0.021**)
TOP	SGD-Plateau	**-0.105** (**-0.218**, **-0.015**)	**-0.110** (**-0.136**,**-0.010**)
TOP	SGD-Step	**-0.059** (**-0.237**, **-0.013**)	**-0.096** (**-0.144**,**-0.015**)
TOP	SGD-Constant	**-0.117** (**-0.256**, **-0.040**)	**-0.107** (**-0.149**,**-0.034**)
TOP	Adam-Cosine	0.003 (-0.730, 0.199)	**-0.052** (-0.723, 0.125)
TOP	Adam-OneCycle	0.104 (-0.343, 0.138)	0.055 (-0.386, 0.056)
TOP	AdamW-Cosine	0.001 (-0.730, 0.208)	**-0.052** (-0.723, 0.125)
TOP	AdamW-OneCycle	0.114 (-0.377, 0.131)	0.055 (-0.372, 0.056)
WD	DoG	**-0.241** (-0.415, 0.025)	**-0.153** (**-0.189**,**-0.015**)
WD	SGD-Cosine	**-0.112** (**-0.265**, **-0.027**)	**-0.096** (**-0.182**,**-0.006**)
WD	SGD-Exp	**-0.124** (**-0.289**, **-0.023**)	**-0.141** (**-0.188**,**-0.021**)
WD	SGD-Plateau	**-0.152** (**-0.218**, **-0.019**)	**-0.110** (**-0.136**,**-0.010**)
WD	SGD-Step	**-0.081** (**-0.237**, **-0.017**)	**-0.096** (**-0.163**,**-0.019**)
WD	SGD-Constant	**-0.133** (**-0.189**, **-0.095**)	**-0.107** (**-0.146**,**-0.052**)
WD	Adam-Cosine	**-0.004** (-0.730, 0.202)	**-0.052** (-0.720, 0.125)
WD	Adam-OneCycle	0.104 (-0.323, 0.137)	0.048 (-0.386, 0.064)
WD	AdamW-Cosine	**-0.008** (-0.730, 0.203)	**-0.052** (-0.723, 0.125)
WD	AdamW-OneCycle	0.103 (-0.339, 0.132)	0.048 (-0.386, 0.056)

cisions, providing an alternative to hand-crafted schedules and loss-only heuristics. By linking topological signals to learning-rate updates, we introduce a mechanism whose behavior is directly traceable to representation reorganization, making it easier to understand why step sizes change across training. Our experiments on image and graph benchmarks validate the effectiveness of this approach and point toward the broader potential of connectome-guided signals for principled training across diverse domains. Looking forward, this line of research has the potential to make optimization not only more efficient, but also more explainable and reliable, with broad impact on the development of future machine learning systems.

References

Mathieu Carriere, Marco Cuturi, and Steve Oudot. Sliced wasserstein kernel for persistence diagrams. In *International Conference on Machine Learning*, pages 664–673. PMLR, 2017.

David Cohen-Steiner, Herbert Edelsbrunner, and John

Table 6: Median RED with 95% Percentile-Bootstrap CIs Across Seeds. **Bold** entries indicate cases where CG-ALR achieves better generalization performance (RED < 0), or statistically significant improvements when the 95% CI lies entirely below 0. Results are shown for GCN and GAT, with ENZYMES and PROTEINS reported beneath each method.

		GCN		**GAT**	
		ENZYMES	**PROTEINS**	**ENZYMES**	**PROTEINS**
CG-ALR	**Baseline**	RED (95% CI)	RED (95% CI)	RED (95% CI)	RED (95% CI)
BD	DoG	0.090 (0.060, 0.218)	**-0.014** (-0.104, 0.053)	0.106 (0.013, 0.248)	0.037 (-0.053, 0.114)
BD	SGD-Cosine	**-0.034** (-0.119, 0.026)	0.011 (-0.014, 0.052)	0.000 (-0.052, 0.082)	**-0.016** (-0.200, 0.049)
BD	SGD-Exp	**-0.167** (**-0.230**, **-0.090**)	**-0.095** (**-0.125**, **-0.061**)	**-0.099** (**-0.190**, **-0.040**)	**-0.037** (-0.233, 0.024)
BD	SGD-Plateau	**-0.128** (**-0.304**, **-0.022**)	**-0.050** (-0.125, 0.000)	**-0.110** (-0.190, 0.000)	**-0.071** (-0.233, 0.000)
BD	SGD-Step	**-0.192** (**-0.299**, **-0.101**)	**-0.070** (-0.104, 0.000)	**-0.110** (-0.222, 0.000)	**-0.098** (**-0.200**, **-0.012**)
BD	SGD-Constant	0.000 (-0.026, 0.095)	0.013 (-0.026, 0.056)	0.040 (-0.012, 0.047)	0.000 (-0.037, 0.040)
BD	Adam-Cosine	0.053 (-0.235, 0.211)	0.013 (-0.250, 0.150)	**-0.075** (-0.128, 0.124)	**-0.059** (-0.169, 0.182)
BD	Adam-OneCycle	0.187 (-0.074, 0.343)	0.055 (-0.111, 0.212)	0.011 (-0.061, 0.227)	**-0.013** (-0.070, 0.160)
BD	AdamW-Cosine	0.096 (-0.222, 0.333)	0.040 (-0.250, 0.188)	**-0.057** (-0.143, 0.176)	0.026 (-0.169, 0.182)
BD	AdamW-OneCycle	0.218 (-0.012, 0.281)	0.053 (-0.100, 0.123)	0.011 (-0.023, 0.227)	0.114 (-0.016, 0.205)
HK	DoG	0.088 (0.042, 0.203)	**-0.082** (-0.214, 0.014)	0.088 (0.035, 0.248)	0.037 (0.014, 0.132)
HK	SGD-Cosine	**-0.054** (-0.155, 0.000)	**-0.032** (-0.061, 0.011)	0.000 (-0.023, 0.049)	0.011 (-0.100, 0.047)
HK	SGD-Exp	**-0.167** (**-0.282**, **-0.038**)	**-0.125** (**-0.214**, **-0.094**)	**-0.085** (**-0.190**, **-0.010**)	**-0.023** (-0.139, 0.012)
HK	SGD-Plateau	**-0.158** (-0.304, 0.000)	**-0.121** (**-0.200**, **-0.026**)	**-0.148** (-0.188, 0.000)	**-0.047** (-0.125, 0.011)
HK	SGD-Step	**-0.167** (**-0.254**, **-0.101**)	**-0.125** (-0.214, 0.000)	**-0.165** (-0.198, 0.000)	**-0.071** (**-0.129**, **-0.025**)
HK	SGD-Constant	0.000 (-0.036, 0.154)	**-0.014** (-0.113, 0.011)	0.012 (-0.020, 0.071)	0.015 (0.000, 0.037)
HK	Adam-Cosine	0.108 (-0.351, 0.366)	**-0.043** (-0.216, 0.160)	**-0.099** (-0.163, 0.171)	**-0.059** (-0.127, 0.182)
HK	Adam-OneCycle	0.200 (-0.108, 0.333)	0.025 (-0.081, 0.222)	0.000 (-0.070, 0.248)	0.029 (-0.143, 0.176)
HK	AdamW-Cosine	0.069 (-0.338, 0.329)	0.000 (-0.216, 0.198)	**-0.070** (-0.163, 0.228)	**-0.041** (-0.098, 0.182)
HK	AdamW-OneCycle	0.171 (-0.024, 0.292)	0.000 (-0.031, 0.099)	0.000 (-0.070, 0.257)	0.088 (-0.040, 0.205)
SWK	DoG	0.137 (-0.046, 0.165)	**-0.014** (-0.145, 0.053)	0.136 (0.073, 0.248)	0.028 (-0.078, 0.122)
SWK	SGD-Cosine	**-0.034** (-0.176, 0.026)	0.011 (-0.041, 0.072)	0.000 (-0.010, 0.114)	**-0.016** (-0.114, 0.035)
SWK	SGD-Exp	**-0.152** (**-0.309**, **-0.066**)	**-0.070** (-0.174, 0.014)	**-0.040** (-0.169, 0.000)	**-0.023** (-0.286, 0.000)
SWK	SGD-Plateau	**-0.128** (**-0.324**, **-0.011**)	**-0.049** (-0.151, 0.012)	**-0.080** (-0.148, 0.000)	**-0.059** (-0.139, 0.000)
SWK	SGD-Step	**-0.192** (**-0.338**, **-0.077**)	**-0.049** (-0.126, 0.012)	**-0.088** (-0.149, 0.000)	**-0.071** (**-0.286**, **-0.023**)
SWK	SGD-Constant	0.011 (-0.024, 0.110)	0.013 (-0.013, 0.027)	0.040 (-0.012, 0.080)	0.000 (0.000, 0.013)
SWK	Adam-Cosine	0.129 (-0.235, 0.366)	**-0.028** (-0.268, 0.160)	**-0.107** (-0.158, 0.147)	**-0.059** (-0.139, 0.182)
SWK	Adam-OneCycle	0.160 (-0.012, 0.333)	0.000 (-0.085, 0.143)	**-0.013** (-0.060, 0.227)	0.000 (-0.091, 0.159)
SWK	AdamW-Cosine	0.054 (-0.158, 0.333)	0.013 (-0.250, 0.065)	**-0.108** (-0.171, 0.196)	0.038 (-0.139, 0.097)
SWK	AdamW-OneCycle	0.194 (-0.012, 0.258)	0.029 (-0.097, 0.099)	0.000 (-0.095, 0.227)	0.077 (0.000, 0.188)
TOP	DoG	0.150 (-0.076, 0.250)	**-0.091** (-0.214, 0.027)	0.170 (0.012, 0.248)	0.055 (-0.013, 0.132)
TOP	SGD-Cosine	**-0.055** (-0.242, 0.025)	**-0.014** (-0.076, 0.030)	0.000 (-0.066, 0.060)	0.000 (-0.071, 0.035)
TOP	SGD-Exp	**-0.152** (**-0.379**, **-0.066**)	**-0.086** (**-0.266**, **-0.027**)	**-0.072** (-0.190, 0.000)	**-0.026** (-0.154, 0.000)
TOP	SGD-Plateau	**-0.190** (**-0.318**, **-0.023**)	**-0.121** (-0.200, 0.000)	**-0.067** (-0.202, 0.000)	**-0.064** (**-0.154**, **-0.023**)
TOP	SGD-Step	**-0.153** (**-0.348**, **-0.036**)	**-0.061** (**-0.214**, **-0.011**)	**-0.112** (-0.190, 0.000)	**-0.071** (-0.154, 0.000)
TOP	SGD-Constant	0.000 (-0.024, 0.167)	**-0.025** (-0.109, 0.025)	0.024 (-0.020, 0.146)	0.000 (0.000, 0.068)
TOP	Adam-Cosine	0.159 (-0.297, 0.352)	**-0.028** (-0.233, 0.090)	**-0.114** (-0.149, 0.124)	0.000 (-0.139, 0.114)
TOP	Adam-OneCycle	0.227 (-0.169, 0.333)	0.028 (-0.055, 0.222)	**-0.034** (-0.046, 0.072)	0.000 (-0.147, 0.159)
TOP	AdamW-Cosine	0.082 (-0.275, 0.319)	0.040 (-0.233, 0.198)	**-0.103** (-0.185, 0.196)	0.044 (-0.139, 0.182)
TOP	AdamW-OneCycle	0.227 (-0.176, 0.313)	0.051 (-0.068, 0.099)	**-0.034** (-0.048, 0.227)	0.099 (-0.039, 0.205)
WD	DoG	0.137 (0.074, 0.209)	**-0.014** (-0.118, 0.056)	0.136 (0.057, 0.198)	0.060 (-0.113, 0.132)
WD	SGD-Cosine	**-0.047** (-0.171, 0.164)	0.014 (-0.038, 0.072)	0.000 (-0.010, 0.114)	**-0.023** (-0.071, 0.035)
WD	SGD-Exp	**-0.123** (-0.300, 0.051)	**-0.070** (-0.118, 0.014)	**-0.040** (-0.169, 0.000)	**-0.048** (-0.286, 0.000)
WD	SGD-Plateau	**-0.106** (**-0.324**, **-0.023**)	**-0.042** (**-0.118**, **-0.013**)	**-0.080** (-0.148, 0.000)	**-0.051** (**-0.286**, **-0.016**)
WD	SGD-Step	**-0.153** (-0.271, 0.000)	**-0.023** (-0.095, 0.000)	**-0.088** (-0.149, 0.000)	**-0.071** (**-0.274**, **-0.023**)
WD	SGD-Constant	0.011 (-0.015, 0.110)	0.013 (0.000, 0.041)	0.040 (-0.012, 0.080)	0.000 (-0.097, 0.013)
WD	Adam-Cosine	0.065 (-0.280, 0.379)	**-0.028** (-0.304, 0.117)	**-0.099** (-0.158, 0.171)	**-0.059** (-0.139, 0.182)
WD	Adam-OneCycle	0.187 (-0.155, 0.347)	0.028 (-0.026, 0.182)	**-0.013** (-0.083, 0.174)	0.000 (-0.068, 0.159)
WD	AdamW-Cosine	0.039 (-0.282, 0.308)	0.027 (-0.221, 0.156)	**-0.082** (-0.171, 0.220)	0.038 (-0.139, 0.182)
WD	AdamW-OneCycle	0.200 (-0.155, 0.281)	0.039 (0.027, 0.052)	0.000 (-0.145, 0.143)	0.088 (0.000, 0.205)

Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL `http://`

www.deeplearningbook.org.

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 (CVPR)*, pages 770–778, 2016.

Maor Ivgi and Amir Globerson. Dog: Diversity over gradient for better neural network training. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 14382–14407. PMLR, 2023.

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

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

PyTorch. Learning rate schedulers, 2023. URL `https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate`.

Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4741–4748, 2015.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Primož Skraba and Katharine Turner. Wasserstein stability for persistence diagrams. *arXiv preprint arXiv:2006.16824*, 2020.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2018.

Tananun Songdechakraiwut and Moo K Chung. Topological learning for brain networks. *The Annals of Applied Statistics*, 17(1):403–433, 2023.

Tananun Songdechakraiwut and Yutong Wu. Functional connectomes of neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20558–20566, 2025.

Tananun Songdechakraiwut, Li Shen, and Moo Chung. Topological learning and its application to multimodal brain network integration. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 166–176, 2021.

Tananun Songdechakraiwut, Bryan M Krause, Matthew I Banks, Kirill V Nourski, and Barry D Van Veen. Wasserstein distance-preserving vector space of persistent homology. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 277–286, 2023.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude, 2012. COURSERA: Neural Networks for Machine Learning.

Petar Velčković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 3637–3645, 2016.

Yutong Wu, Peilin He, and Tananun Songdechakraiwut. Data-efficient neural training with dynamic connectomes. *arXiv preprint arXiv:2508.06817*, 2025.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] We give a detailed description in Section 2.4.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] We have theoretical complexity analysis, wall-clock timing, and memory-overhead results.

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] We provide the Python code in the supplementary material.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] See Appendix.
   (b) Complete proofs of all theoretical results. [Yes] See Appendix.
   (c) Clear explanations of any assumptions. [Yes] See Appendix.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] See supplementary materials.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] See Appendix.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Materials

A RELATED WORK

Learning rate control in stochastic optimization has long relied on handcrafted schedules such as step decay, exponential decay, cosine annealing, and reduction on plateau; these schedules are primarily time-based or loss-triggered and usually require careful tuning to the model and dataset (Goodfellow et al., 2016; PyTorch, 2023). Adaptive optimizers such as RMSProp and Adam use gradient statistics to temper this sensitivity; however, they remain hyperparameter dependent and do not directly track how internal representations evolve during training (Tieleman and Hinton, 2012; Kingma and Ba, 2015). Parameter-free methods such as Distance over Gradients (DoG) scale the step size using the ratio of parameter displacement to accumulated gradient norms. This removes explicit schedules while still relying on first-order magnitudes and may miss important phases of internal reorganization (Ivgi and Globerson, 2023).

Topological data analysis provides a complementary view. Persistent homology yields stable summaries of structure with Wasserstein and related metrics, and kernelized constructions such as the sliced Wasserstein and heat kernels enable efficient comparisons of evolving topological signals (Cohen-Steiner et al., 2007; Reininghaus et al., 2015; Carriere et al., 2017; Skraba and Turner, 2020). Recent work introduces functional connectomes for artificial networks, bringing coactivation graphs and persistent graph homology to the study of representation dynamics (Songdechakraiwut and Wu, 2025; Songdechakraiwut et al., 2023). Our approach builds at this intersection. We couple a Robbins and Monro envelope (Robbins and Monro, 1951) with a topology-driven controller that monitors connectome change through persistent homology distances, for example, Wasserstein comparisons between successive connectomes, so that learning rate adjustments respond to representation reconfiguration rather than to time or loss alone.

B TOPOLOGICAL CONNECTOME DISTANCES

B.1 Bottleneck Distance (BD)

The bottleneck distance measures the worst-case discrepancy between two persistence diagrams. A persistence diagram D is a multiset of points (b, d) encoding birth and death times of topological features. To compare diagrams D_1 and D_2, consider matchings γ that, for every point $p \in D_1 \cup D_2$, pair p either with a point in the other diagram or with its projection onto the diagonal $\Delta = \{(x, x)\}$. Each point is matched exactly once, and images under γ are unique.

The bottleneck distance is

$$d_B(D_1, D_2) = \inf_{\gamma} \sup_{p \in D_1 \cup D_2} \|p - \gamma(p)\|_\infty,$$

where $\|(b_1, d_1) - (b_2, d_2)\|_\infty = \max\{|b_1 - b_2|, |d_1 - d_2|\}$ for point-to-point matches, and for a match to the diagonal one has

$$\big\|(b, d) - \Pi_\Delta(b, d)\big\|_\infty = \tfrac{1}{2}(d - b),$$

which is the minimal ℓ_∞ distance from (b, d) to Δ. Intuitively, d_B reports the largest single adjustment needed to transform one diagram into the other and is a standard stability metric in persistent homology.

B.2 Heat Kernel (HK)

The heat kernel defines a positive-definite similarity on persistence diagrams (Reininghaus et al., 2015). For diagrams D_1 and D_2, and points $p = (b_p, d_p) \in D_1$, $q = (b_q, d_q) \in D_2$, denote their reflections across the diagonal

by $\bar{p} = (d_p, b_p)$ and $\bar{q} = (d_q, b_q)$. With bandwidth $\sigma > 0$, the kernel is

$$k_\sigma(D_1, D_2) = \frac{1}{8\pi\sigma} \sum_{\substack{p\in D_1 \\ q\in D_2}} \big[e^{-\frac{\|p-q\|^2}{8\sigma}} - e^{-\frac{\|p-\bar{q}\|^2}{8\sigma}} - e^{-\frac{\|\bar{p}-q\|^2}{8\sigma}} + e^{-\frac{\|\bar{p}-\bar{q}\|^2}{8\sigma}} \big].$$

The four terms jointly account for interactions with diagonal reflections, which suppress the influence of low-persistence features and yield a symmetric, positive-definite kernel in a reproducing kernel Hilbert space.

B.3 Sliced Wasserstein Kernel (SWK)

The sliced Wasserstein construction combines optimal transport with kernel methods while remaining computationally tractable Carriere et al. (2017). Given diagrams D_1, D_2 and order $p \geq 1$, the sliced Wasserstein distance is

$$\mathrm{SW}_p^p(D_1, D_2) = \frac{1}{\pi} \int_0^\pi W_p^p\big(\theta_* D_1,\ \theta_* D_2\big)\, d\theta,$$

where $\theta_* D$ denotes the projection of all points in D onto the line through the origin with angle θ, and W_p is the p-Wasserstein distance on the real line. A Gaussian kernel built from this distance is

$$k_{\mathrm{SW}}(D_1, D_2) = \exp\left(-\frac{\mathrm{SW}_p^p(D_1, D_2)}{2\tau} \right),$$

with scale parameter $\tau > 0$.

B.4 Implementation Details for Topological Connectome Distances

Feature tap and probe set. Each epoch uses a fixed probe size $P =$ `PROBE_P` for {TOP, WD, SWK, BD, HK} (baselines set P=0). Layer/features are obtained via `activations` (graphs) or `extract_features` (images). We form a robust absolute correlation $\mathbf{S} \in [0,1]^{P\times P}$ using `robust_corr` and zero the diagonal via `correlation_graph`.

TOP (graph–filtration vector). From $\mathbf{S}$, we compute an MST on $\mathbf{1} - \mathbf{S}$, remove its edges, take the strict upper triangle of the remainder, keep positive entries, and sort descending; implemented by `adj2pers`.

WD/SWK/BD/HK (Vietoris–Rips on H_1). We set $\mathbf{D} = \mathbf{1} - |\mathrm{corr}|$ (zero diagonal) and run Vietoris–Rips with `compute_persistence_diagram_from_correlation` (H_1 only). Epoch distances use `persim_wasserstein` (WD, $p =$ `pd_w_p`), `sliced_wasserstein` (SWK, $M =$ `swk_K`, $p =$ `swk_p`), `bottleneck` (BD), and `heat` (HK); we fix H_1 with unit weight.

Topological time series. Let $\mathcal{S}_e$ be the epoch-e signature (GF vector or H_1 PD). We set W_0=0 and for TOP compute $W_e = W_1(\text{sorted GF}_e, \text{sorted GF}_{e-1})$ via `wasserstein_distance`; for PDs, W_e is the chosen `persim` distance above. The series $\{W_e\}$ is consumed within `train_model`.

C THEOREM 1 PROOFS

Setup. We minimize the population objective

$$f(\theta)\mathbb{E}_\xi[\ell(\theta;\xi)]$$

over $\theta \in \mathbb{R}^d$. At (global) SGD step $t \in \mathbb{N}_0$, the iterate is updated by

$$\theta_{t+1} = \theta_t - \eta_t\, g_t, \qquad g_t = \nabla\ell(\theta_t;\xi_t), \tag{4}$$

with a stepsize of the form

$$\eta_t = \psi_{e(t)-1}^{-1}\, \tilde{\eta}_t, \qquad \tilde{\eta}_t = \frac{\eta_0}{(t+t_0)^\alpha}, \tag{5}$$

where $\eta_0 > 0$, $t_0 \geq 1$, and $\alpha \in (1/2, 1]$. Training is organized in epochs indexed by $e \in \mathbb{N}_0$, with integer boundaries $0 = T_0 < T_1 < T_2 < \cdots$ and epoch membership map $e(t)$ defined by $e(t) = e$ iff $T_e \leq t < T_{e+1}$. At all

t in epoch $e \geq 1$, the (clipped) factor ψ_{e-1} computed from epoch $e-1$ is used; for $e = 0$ we set a deterministic $\psi_{-1} \in [\psi_{\min}, \psi_{\max}]$. Let $(\mathcal{F}_t)_{t\geq 0}$ be the canonical filtration

$$\mathcal{F}_t \sigma\big(\theta_0,\ \xi_0, \ldots, \xi_{t-1}\big),$$

so that g_t depends on ξ_t and is *not* $\mathcal{F}_t$-measurable, while any quantity decided before sampling ξ_t must be $\mathcal{F}_t$-measurable ("predictable").

Assumptions.

(A1) (Lower boundedness) $f_\star \inf_\theta f(\theta) > -\infty$.

(A2) (Smoothness) f is L-smooth: for all x, y, $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$.

(A3) (Unbiased stochastic gradient with bounded second moment) For some constants $A, B \geq 0$,

$$\mathbb{E}[g_t \mid \mathcal{F}_t] = \nabla f(\theta_t), \qquad \mathbb{E}\big[\|g_t\|^2 \mid \mathcal{F}_t\big] \leq A + B\,\|\nabla f(\theta_t)\|^2.$$

(A4) (Bounded iterates) $(\theta_t)_{t\geq 0}$ is almost surely bounded.

(A5) (Predictable, bounded controller) For every epoch $e \geq 0$, ψ_e is $\mathcal{F}_{T_{e+1}}$-measurable and satisfies $0 < \psi_{\min} \leq \psi_e \leq \psi_{\max} < \infty$ almost surely. Equivalently, for any t with $e(t) = e + 1$, ψ_e is $\mathcal{F}_t$-measurable.

(A6) (Robbins–Monro envelope) $\alpha \in (1/2, 1]$ in (5).

The analysis uses only (A5), i.e., that the controller produces an *a priori* bounded, predictable multiplicative factor per epoch. The specific statistic used to form ψ_e (e.g., a 1D Wasserstein distance between empirical similarity distributions on $[0, 1]$) is immaterial to the proofs.

Technical lemmas. [Robbins–Monro conditions preserved] Under (A5)–(A6), the stepsizes are predictable and satisfy

$$\sum_{t=0}^{\infty} \eta_t = \infty, \qquad \sum_{t=0}^{\infty} \eta_t^2 < \infty, \qquad \eta_t \text{ is } \mathcal{F}_t\text{-measurable for all } t, \qquad \eta_t \to 0.$$

Proof. By (A5), $\psi_{e(t)-1} \in [\psi_{\min}, \psi_{\max}]$ and is decided before sampling ξ_t, hence $\mathcal{F}_t$-measurable. Thus η_t is predictable as the product of an $\mathcal{F}_t$-measurable random variable and the deterministic $\tilde{\eta}_t$. Comparison gives

$$\frac{1}{\psi_{\max}} \sum_t \tilde{\eta}_t \ \leq\ \sum_t \eta_t \ \leq\ \frac{1}{\psi_{\min}} \sum_t \tilde{\eta}_t, \qquad \sum_t \eta_t^2 \ \leq\ \frac{1}{\psi_{\min}^2} \sum_t \tilde{\eta}_t^2.$$

Since $\alpha \in (1/2, 1]$, $\sum_t \tilde{\eta}_t = \infty$ and $\sum_t \tilde{\eta}_t^2 < \infty$; the same then holds for (η_t), and moreover $\eta_t \to 0$. □

[One-step expected descent] Under (A2)–(A3), the update (4) satisfies

$$\mathbb{E}[f(\theta_{t+1}) \mid \mathcal{F}_t] \leq f(\theta_t) - \big(\eta_t - \tfrac{LB}{2}\eta_t^2\big)\|\nabla f(\theta_t)\|^2 + \tfrac{LA}{2}\eta_t^2.$$

Proof. L-smoothness gives $f(\theta_{t+1}) \leq f(\theta_t) - \eta_t\langle \nabla f(\theta_t), g_t\rangle + \frac{L}{2}\eta_t^2\|g_t\|^2$. Condition on $\mathcal{F}_t$ and use (A3). □

[Robbins–Siegmund, almost supermartingale] Let (X_t) be nonnegative and adapted. Suppose there exist nonnegative adapted (Y_t) and (Z_t) such that

$$\mathbb{E}[X_{t+1} \mid \mathcal{F}_t] \leq X_t - Y_t + Z_t, \qquad \sum_{t=0}^{\infty} \mathbb{E}[Z_t] < \infty.$$

Then (X_t) converges almost surely to a finite random variable and $\sum_{t=0}^{\infty} Y_t < \infty$ almost surely.

Main results.

Theorem 2 (Convergence to stationary points)**.** *Under* (A1)–(A6)*, the following hold:*

1. $f(\theta_t)$ *converges almost surely to a finite random variable;*
2. $\sum_{t=0}^{\infty} \eta_t \|\nabla f(\theta_t)\|^2 < \infty$ *almost surely, hence* $\liminf_{t\to\infty} \|\nabla f(\theta_t)\| = 0$ *almost surely;*
3. *Every almost-sure limit point of* (θ_t) *is a stationary point of* f*.*

Proof. Define

$$X_t f(\theta_t) - f_\star \geq 0, \quad Y_t\Big(\eta_t - \tfrac{LB}{2}\eta_t^2\Big)\,\|\nabla f(\theta_t)\|^2, \quad Z_t \tfrac{LA}{2}\eta_t^2.$$

By Lemmas C and C, $\mathbb{E}[X_{t+1} \mid \mathcal{F}_t] \leq X_t - Y_t + Z_t$ and $\sum_t \mathbb{E}[Z_t] \leq \frac{LA}{2}\sum_t \eta_t^2 < \infty$. Since $\eta_t \to 0$, there exists an almost surely finite time T such that for all $t \geq T$, $\eta_t - \frac{LB}{2}\eta_t^2 \geq \frac{1}{2}\eta_t$. Applying Lemma C from T onward (the finite prefix being harmless) yields the convergence of $(f(\theta_t))_t$ and $\sum_t Y_t < \infty$ almost surely. In particular, $\sum_t \eta_t \|\nabla f(\theta_t)\|^2 < \infty$ almost surely.

By Lemma C, $\sum_t \eta_t = \infty$; thus the only way for $\sum_t \eta_t \|\nabla f(\theta_t)\|^2$ to be finite is that $\liminf_t \|\nabla f(\theta_t)\| = 0$ almost surely. Boundedness of (θ_t) (A4) implies the existence of almost-sure accumulation points; by continuity of ∇f and the previous display, every limit point $\bar{\theta}$ satisfies $\nabla f(\bar{\theta}) = 0$. □

Theorem 3 (Convergence under PL)**.** *Suppose, in addition, that* f *satisfies the Polyak–Łojasiewicz (PL) inequality*

$$\frac{1}{2}\|\nabla f(\theta)\|^2 \;\geq\; \mu\big(f(\theta) - f_\star\big) \qquad \text{for some } \mu > 0 \text{ and all } \theta.$$

Then $f(\theta_t) \to f_\star$ *almost surely and* $\operatorname{dist}(\theta_t, \mathcal{X}_\star) \to 0$ *almost surely, where* $\mathcal{X}_\star \arg\min f$*. Consequently, every limit point of* (θ_t) *is a (global) minimizer. If* $\mathcal{X}_\star$ *is a singleton, then* $\theta_t \to \theta_\star$ *almost surely.*

Proof. Under PL, $\|\nabla f(\theta_t)\|^2 \geq 2\mu\,(f(\theta_t) - f_\star)$, hence

$$\sum_t \eta_t\,(f(\theta_t) - f_\star) \;\leq\; \frac{1}{2\mu}\sum_t \eta_t \|\nabla f(\theta_t)\|^2 \;<\; \infty \quad \text{a.s.}$$

By Theorem 2, $f(\theta_t)$ converges almost surely and $\sum_t \eta_t = \infty$ (Lemma C); therefore $f(\theta_t) \to f_\star$ almost surely (otherwise the above sum would diverge). PL then implies $\|\nabla f(\theta_t)\| \to 0$ almost surely and, by standard error-bound arguments under PL (e.g., quadratic growth), $\operatorname{dist}(\theta_t, \mathcal{X}_\star) \to 0$ almost surely. If $\mathcal{X}_\star$ is a singleton, the whole sequence converges. □

Why the controller cannot break convergence. The controller forms ψ_e from epoch-e data and *clips* it into $[\psi_{\min}, \psi_{\max}]$. Thus ψ_e is $\mathcal{F}_{T_{e+1}}$-measurable and uniformly bounded (A5). Any additional heuristics (e.g., cooldowns, multiplicative up/down rules, late-phase boosts) that transform ψ_e by $\mathcal{F}_{T_{e+1}}$-measurable operations and re-clip to $[\psi_{\min}, \psi_{\max}]$ preserve (A5). Consequently, Lemma C holds, which is the only property of the controller used by the proofs of Theorems 2 and 3.

Notes on predictability. At each step t, the random variables $\psi_{e(t)-1}$ and η_t are $\mathcal{F}_t$-measurable (decided before sampling ξ_t), while g_t is only $\mathcal{F}_{t+1}$-measurable. This ensures the conditional expectations in Lemma C are well-defined and that the Robbins–Siegmund lemma applies with $Z_t = \frac{LA}{2}\eta_t^2$ and $\sum_t \mathbb{E}[Z_t] < \infty$ by Lemma C.

D EXPERIMENT DETAILS

Additional Dataset Details. MUTAG consists of graphs representing nitroaromatic compounds, labeled according to whether they exhibit mutagenic effect on the *Salmonella typhimurium* bacterium. PROTEINS contains protein graphs labeled as enzymes or non-enzymes. ENZYMES includes protein graphs categorized into 6 predefined classes according to enzyme commission numbers (Morris et al., 2020).

Table 7: Training configurations for CIFAR-10, CIFAR-100, and Mini-ImageNet experiments.

Parameter	Symbol / Key	CIFAR-10	CIFAR-100	Mini-ImageNet
Epochs	`epochs`	50	90	90
Optimizer	`optimizer`	SGD	SGD	SGD
Exponential decay γ	`exp_gamma`	0.97	0.97	0.97
Weight decay	`weight_decay`	5×10^{-4}	7×10^{-4}	5×10^{-4}
Momentum	`momentum`	0.9	0.9	0.9
RM offset	t_0 (`t0`)	1600	2000	1600
RM exponent	α (`alpha`)	0.52	0.50	0.60
Upscale multiplier	$\gamma_\uparrow$ (`gamma_up`)	1.20	1.22	1.08
Downscale multiplier	$\gamma_\downarrow$ (`gamma_down`)	0.85	0.88	0.82
Late-phase multiplier	γ_{late} (`gamma_late`)	0.985	0.98	0.95
Cooldown length	`cooldown`	4	5	3
Trigger count	`n_trigger`	6	7	3
Late split ratio	N_{ratio} (`N_ratio`)	0.88	0.90	0.76
Probe set size	P_{probe} (`probe_P`)	1024	1024	10000
$\psi_{\min}$	`psi_min`	0.65	0.70	0.60
$\psi_{\max}$	`psi_max`	6.0	10.0	1.70
Smoothing factor	β (`beta`)	0.96	0.97	0.94
Threshold step size	τ (`tau`)	0.002	0.003	0.002
Robust window	w (`robust_w`)	13	15	17
MAD multiplier	k (`mad_k`)	3.6	3.8	3.2
Warm-up epochs	K_{warm} (`K_warm`)	4	6	12

Table 8: Training configurations for MUTAG, PROTEINS, and ENZYMES experiments.

Parameter	Symbol / Key	MUTAG	PROTEINS	ENZYMES
Epochs	`epochs`	300	500	500
Optimizer	`optimizer`	SGD	SGD	SGD
Exponential decay γ	`exp_gamma`	0.97	0.97	0.97
Weight decay	`weight_decay`	5×10^{-4}	5×10^{-4}	5×10^{-4}
Momentum	`momentum`	0.9	0.9	0.9
RM offset	t_0 (`t0`)	800	1300	1200
RM exponent	α (`alpha`)	0.56	0.56	0.57
Upscale multiplier	$\gamma_\uparrow$ (`gamma_up`)	1.10	1.08	1.08
Downscale multiplier	$\gamma_\downarrow$ (`gamma_down`)	0.80	0.84	0.84
Late-phase multiplier	γ_{late} (`gamma_late`)	0.95	0.96	0.96
Cooldown length	`cooldown`	3	4	4
Trigger count	`n_trigger`	3	4	4
Late split ratio	N_{ratio} (`N_ratio`)	0.70	0.80	0.78
$\psi_{\min}$	`psi_min`	0.62	0.60	0.60
$\psi_{\max}$	`psi_max`	1.8	1.8	2.0
Smoothing factor	β (`beta`)	0.94	0.95	0.95
Threshold step size	τ (`tau`)	0.001	0.002	0.002
Robust window	w (`robust_w`)	12	16	15
MAD multiplier	k (`mad_k`)	3.3	3.2	3.2
Warm-up epochs	K_{warm} (`K_warm`)	12	16	16

System Configuration. All experiments were performed on the system with the following specifications:

- **CPU:** Intel Xeon Gold 6226, 2.70 GHz
- **GPU:** NVIDIA RTX 5000 Ada Generation, 32 GB VRAM
- **DRAM:** 755 GB
- **Architecture:** x86_64
- **OS:** Ubuntu 22.04.5 LTS

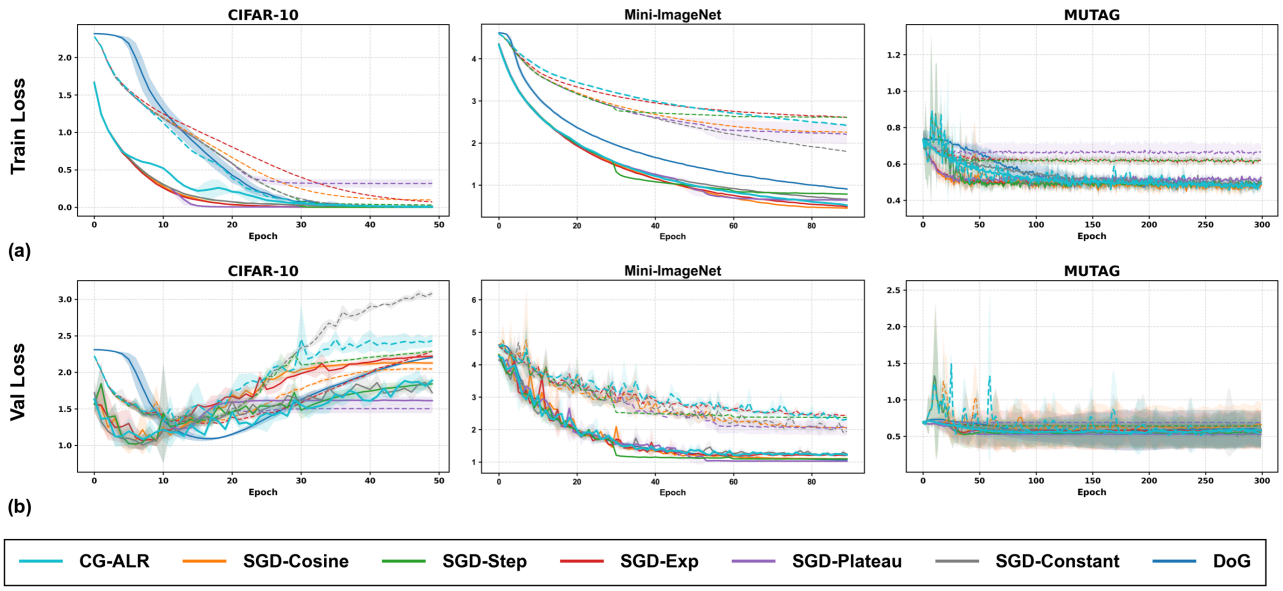

Figure 3: (a) **Training Loss** on three datasets (**CIFAR-10**, **Mini-ImageNet**, and **MUTAG**). (b) **Validation Loss** on the same datasets. Solid lines indicate the averages across three seeds using the *best* learning-rate configuration from the predefined grid ($\eta^\star \in \{0.1, 0.01, 0.001\}$ for CG-ALR and SGD-based schedules; $\eta \in \{0.1, 0.01, 0.001\}$ for SGD-Constant; and default hyperparameters for DoG). Dashed lines denote the corresponding *worst* configurations. Shaded regions represent the min–max range across seeds. All methods share the same model backbone and training protocol.

- **Kernel:** Linux 5.15
- **CUDA Toolkit:** 12.8
- **NVIDIA Driver:** 570.148.08

Hyperparameters in Algorithm 1. The hyperparameters used in all experiments are listed in Table 7.

E ADDITIONAL EXPERIMENT RESULTS

E.1 RED Results

Figure 3 reports training and validation loss for additional datasets (CIFAR-10, Mini-ImageNet, MUTAG). For MUTAG, we employ a single fully connected (FC) layer and compute the connectomes from this first FC layer. For CIFAR-10 and Mini-ImageNet, the models include three FC layers, and the connectomes are computed from the second FC layer.

For Mini-ImageNet, as shown in Table 9, CG-ALR delivers statistically significant generalization gains in several comparisons. For MUTAG with GCN and GAT backbones, we employ a single FC layer and compute the connectomes from this first FC layer. This design choice follows from the small graph size and limited sample count of MUTAG ($n = 188$), where deeper architectures tend to overfit rapidly and provide limited representational diversity. As shown in Table 10, our CG-ALR outperforms some baselines, though the gains are not uniform across all optimizer families and are not always statistically significant. This is largely due to the simplicity and small scale of MUTAG, where performance differences are easily saturated and variance across seeds can obscure consistent trends.

Table 9: Median RED with 95% Percentile-Bootstrap CIs Across Seeds for Mini-ImageNet (ResNet-18). **Bold** entries indicate cases where CG-ALR achieves better generalization performance (RED < 0), or statistically significant improvements when the 95% CI lies entirely below 0.

CG-ALR	Baseline	RED (95% CI)
SWK	DoG	0.074 (-0.129, 0.487)
SWK	SGD-Cosine	0.085 (0.026, 0.107)
SWK	SGD-Exp	**-0.018** (**-0.041**, **-0.008**)
SWK	SGD-Plateau	0.052 (-0.004, 0.157)
SWK	SGD-Step	**-0.027** (**-0.048**, **-0.001**)
SWK	SGD-Constant	0.033 (-0.046, 0.184)
SWK	Adam-Cosine	**-0.627** (-2.411, 0.489)
SWK	Adam-OneCycle	0.012 (-1.902, 0.591)
SWK	AdamW-Cosine	**-0.585** (-2.411, 0.498)
SWK	AdamW-OneCycle	**-0.152** (-1.921, 0.564)
TOP	DoG	0.074 (-0.129, 0.487)
TOP	SGD-Cosine	0.084 (0.026, 0.107)
TOP	SGD-Exp	**-0.021** (**-0.041**, **-0.018**)
TOP	SGD-Plateau	0.052 (-0.004, 0.156)
TOP	SGD-Step	**-0.028** (**-0.048**, **-0.008**)
TOP	SGD-Constant	0.033 (-0.043, 0.176)
TOP	Adam-Cosine	**-0.642** (-2.411, 0.490)
TOP	Adam-OneCycle	0.049 (-1.916, 0.587)
TOP	AdamW-Cosine	**-0.573** (-2.411, 0.482)
TOP	AdamW-OneCycle	**-0.118** (-1.966, 0.563)
WD	DoG	0.074 (-0.129, 0.487)
WD	SGD-Cosine	0.076 (0.026, 0.107)
WD	SGD-Exp	**-0.021** (**-0.046**, **-0.008**)
WD	SGD-Plateau	0.052 (-0.004, 0.144)
WD	SGD-Step	**-0.028** (**-0.048**, **-0.022**)
WD	SGD-Constant	0.033 (-0.046, 0.183)
WD	Adam-Cosine	**-0.642** (-2.476, 0.482)
WD	Adam-OneCycle	0.049 (-2.034, 0.591)
WD	AdamW-Cosine	**-0.576** (-2.476, 0.488)
WD	AdamW-OneCycle	**-0.118** (-2.028, 0.564)

E.2 Training Efficiency: Wall-Clock Time to Reach the 95% Threshold

On CIFAR-100 with ResNet-18, Table 11 summarizes the wall-clock time required for each method to reach 95% of its own best test accuracy. Among the CG-ALR variants, TOP, WD, and SWK are consistently much more efficient than BD and HK, reaching the threshold in a few hundred seconds across all three values of η_{init}. These stronger CG-ALR variants are also substantially faster than DoG. Relative to the SGD-based baselines, CG-ALR does not achieve the smallest raw wall-clock times, as aggressive schedule heuristics such as SGD-Plateau, SGD-Step, and SGD-Cosine typically reach the threshold earlier. The Adam/AdamW baselines, especially Adam-Cosine and AdamW-Cosine at $\eta_{\text{init}} = 0.1$, are the fastest overall. Overall, these results suggest that the advantage of CG-ALR lies not in minimizing time-to-threshold alone, but in offering competitive training efficiency while using a topology-guided control mechanism instead of a hand-crafted learning-rate schedule.

E.3 Memory Overhead Across Probe Sizes

To quantify how the probe size affects memory usage, Table 12 reports the mean additional CPU and GPU memory recorded during the topology-probing step for $P_{\text{probe}} \in \{64, 256, 1024, 2096\}$. As the probe size increases,

Table 10: Median RED with 95% Percentile-Bootstrap CIs Across Seeds. **Bold** entries indicate cases where CG-ALR achieves better generalization performance (RED < 0), or statistically significant improvements when the 95% CI lies entirely below 0. Results are shown for **MUTAG** with GCN and GAT.

		GCN	**GAT**
CG-ALR	**Baseline**	RED (95% CI)	RED (95% CI)
BD	DoG	**-0.111** (**-0.500**, **-0.091**)	0.000 (-0.625, 0.364)
BD	SGD-Cosine	0.111 (-0.500, 0.182)	**-0.182** (-0.429, 0.000)
BD	SGD-Exp	0.111 (-0.500, 0.182)	**-0.182** (-0.300, 0.077)
BD	SGD-Plateau	**-0.333** (-0.500, 0.182)	**-0.182** (-0.857, 0.000)
BD	SGD-Step	0.111 (-0.500, 0.182)	**-0.182** (-0.857, 0.000)
BD	SGD-Constant	**-0.333** (-0.500, 0.182)	0.000 (-0.004, 0.091)
BD	Adam-Cosine	0.000 (-0.222, 0.100)	0.167 (-0.100, 0.308)
BD	Adam-OneCycle	0.111 (0.000, 0.200)	0.100 (-0.250, 0.231)
BD	AdamW-Cosine	0.000 (-0.143, 0.000)	0.167 (-0.125, 0.308)
BD	AdamW-OneCycle	0.000 (0.000, 0.222)	0.125 (-0.125, 0.231)
HK	DoG	0.000 (-0.200, 0.077)	0.000 (-0.625, 0.462)
HK	SGD-Cosine	0.200 (-0.200, 0.308)	**-0.100** (-0.250, 0.000)
HK	SGD-Exp	0.200 (-0.200, 0.308)	**-0.182** (-0.300, 0.077)
HK	SGD-Plateau	**-0.200** (-0.200, 0.308)	**-0.182** (-0.308, 0.000)
HK	SGD-Step	0.200 (-0.200, 0.308)	**-0.182** (-0.625, 0.000)
HK	SGD-Constant	**-0.200** (-0.200, 0.308)	0.000 (0.000, 0.182)
HK	Adam-Cosine	0.000 (-0.222, 0.100)	0.125 (0.083, 0.308)
HK	Adam-OneCycle	0.111 (0.000, 0.167)	0.125 (0.000, 0.333)
HK	AdamW-Cosine	0.000 (-0.167, 0.000)	0.091 (0.000, 0.308)
HK	AdamW-OneCycle	0.000 (0.000, 0.222)	0.125 (0.091, 0.250)
SWK	DoG	0.000 (-0.333, 0.231)	0.000 (-0.625, 0.000)
SWK	SGD-Cosine	0.000 (0.000, 0.385)	**-0.182** (-0.429, 0.000)
SWK	SGD-Exp	0.000 (0.000, 0.385)	**-0.182** (-0.300, 0.077)
SWK	SGD-Plateau	0.000 (0.000, 0.077)	**-0.182** (-0.857, 0.000)
SWK	SGD-Step	0.000 (0.000, 0.385)	**-0.182** (-0.857, 0.000)
SWK	SGD-Constant	0.000 (0.000, 0.077)	0.000 (0.000, 0.000)
SWK	Adam-Cosine	0.000 (-0.167, 0.100)	0.167 (-0.222, 0.308)
SWK	Adam-OneCycle	0.111 (0.000, 0.200)	0.100 (-0.250, 0.308)
SWK	AdamW-Cosine	0.000 (-0.167, 0.000)	0.167 (-0.125, 0.300)
SWK	AdamW-OneCycle	0.000 (0.000, 0.222)	0.125 (-0.125, 0.231)
TOP	DoG	**-0.250** (-0.500, 0.000)	0.000 (-0.625, 0.304)
TOP	SGD-Cosine	0.000 (-0.125, 0.000)	**-0.100** (-0.375, 0.000)
TOP	SGD-Exp	0.000 (-0.125, 0.000)	**-0.300** (-0.429, 0.077)
TOP	SGD-Plateau	**-0.125** (-0.500, 0.000)	**-0.300** (-0.625, 0.000)
TOP	SGD-Step	0.000 (-0.125, 0.000)	**-0.300** (-0.625, 0.000)
TOP	SGD-Constant	**-0.125** (-0.500, 0.000)	0.000 (0.000, 0.100)
TOP	Adam-Cosine	0.000 (-0.222, 0.100)	0.222 (-0.222, 0.300)
TOP	Adam-OneCycle	0.111 (0.000, 0.167)	0.111 (-0.111, 0.308)
TOP	AdamW-Cosine	0.000 (-0.167, 0.000)	0.167 (-0.125, 0.308)
TOP	AdamW-OneCycle	0.000 (0.000, 0.222)	0.167 (-0.111, 0.231)
WD	DoG	**-0.200** (-0.200, 0.000)	0.000 (-0.625, 0.300)
WD	SGD-Cosine	0.100 (-0.200, 0.200)	**-0.250** (-0.429, 0.000)
WD	SGD-Exp	0.100 (-0.200, 0.200)	**-0.250** (-0.300, 0.077)
WD	SGD-Plateau	**-0.200** (-0.200, 0.100)	**-0.300** (-0.857, 0.000)
WD	SGD-Step	0.100 (-0.200, 0.200)	**-0.300** (-0.857, 0.000)
WD	SGD-Constant	**-0.200** (-0.200, 0.100)	0.000 (-0.004, 0.000)
WD	Adam-Cosine	0.000 (-0.222, 0.100)	0.222 (-0.222, 0.308)
WD	Adam-OneCycle	0.111 (0.000, 0.200)	0.111 (-0.111, 0.308)
WD	AdamW-Cosine	0.000 (-0.143, 0.000)	0.167 (-0.125, 0.308)
WD	AdamW-OneCycle	0.000 (0.000, 0.222)	0.167 (-0.111, 0.231)

the extra CPU memory required to store the connectome grows from 0.0436 MB at $P_{\text{probe}} = 64$ to 0.1536 MB, 4.6387 MB, and 23.5678 MB at $P_{\text{probe}} = 256, 1024$, and 2096, respectively. By contrast, the additional GPU memory remains negligible throughout, staying within roughly 0.01–0.04 MB across all probe sizes. Overall,

Table 11: Wall-clock time to reach 95% of each method's own best test accuracy on CIFAR-100 with ResNet-18. Each entry reports the mean wall-clock time (in seconds) with 95% percentile-bootstrap confidence intervals across three seeds. Lower is better. DoG does not use an initial learning rate, so its result is reported once across all columns.

Method	$\eta_{\text{init}} = 0.001$	$\eta_{\text{init}} = 0.01$	$\eta_{\text{init}} = 0.1$
DoG		1428.9 (1002.0, 1650.8)	
SGD-Cosine	196.3 (177.6, 207.7)	226.5 (217.4, 239.7)	246.2 (239.2, 257.0)
SGD-Exp	398.1 (258.5, 645.7)	284.0 (176.0, 493.6)	503.1 (192.5, 1073.5)
SGD-Plateau	210.6 (186.0, 235.7)	111.7 (106.7, 118.4)	112.8 (103.5, 118.4)
SGD-Step	183.9 (167.4, 194.5)	201.6 (198.8, 203.3)	197.4 (194.6, 200.1)
SGD-Constant	598.0 (451.0, 872.0)	665.9 (606.5, 745.7)	409.1 (207.6, 559.9)
BD	2426.7 (2223.2, 2608.1)	3812.7 (3260.2, 4206.3)	2125.8 (1693.1, 2385.9)
HK	2034.9 (1830.6, 2176.7)	2531.5 (2487.7, 2597.5)	1510.2 (1460.7, 1594.3)
SWK	503.4 (458.3, 563.7)	485.7 (463.4, 500.8)	410.5 (377.3, 438.6)
TOP	424.1 (407.5, 445.3)	424.8 (389.4, 453.1)	323.8 (309.9, 335.1)
WD	491.6 (455.6, 560.9)	498.2 (462.2, 551.7)	395.9 (365.3, 418.1)
Adam-Cosine	796.7 (290.8, 1792.8)	356.6 (220.9, 535.2)	14.7 (6.5, 31.1)
Adam-OneCycle	425.0 (388.0, 455.9)	387.2 (371.7, 415.8)	51.8 (45.5, 55.2)
AdamW-Cosine	863.0 (768.7, 959.5)	371.5 (283.0, 527.8)	21.1 (15.5, 31.0)
AdamW-OneCycle	1935.8 (1861.0, 1982.6)	1628.9 (1089.8, 1977.3)	208.9 (168.1, 241.4)

Table 12: Memory overhead of the topology-probing step across probe sizes. Entries report mean additional memory in MB. Lower is better.

P_{probe}	**Extra CPU Memory (MB)**	**Extra GPU Memory (MB)**
64	0.0436	0.0150
256	0.1536	0.0411
1024	4.6387	0.0389
2096	23.5678	0.0095

these results indicate that the memory overhead of larger probe sets is dominated by CPU-side storage rather than GPU allocation.