TopoAdamW: Topology-Guided Learning Rate Adaptation via Loss Landscape Geometry

Congkai Peng
kelvinpeng2004@outlook.com

Abstract

We introduce TopoAdamW, a learning rate controller built on top of AdamW that adapts the step size at each interval by probing the local geometry of the loss landscape. Every T optimisation steps the method evaluates a sparse 25-point neighbourhood in a two-dimensional random subspace of parameter space, extracts *sharpness* and *variance* features, and scales the learning rate according to a geometric heuristic: accelerate in flat, smooth regions; slow down near sharp or noisy regions. Optionally, a small convolutional network (TopoCNN) trained online on topological persistence images replaces the hand-coded heuristic once sufficient training data has been collected. On CIFAR-10 and CIFAR-100 the heuristic mode outperforms a tuned AdamW baseline by +1.43% and +1.50% accuracy respectively, with approximately 14% additional per-epoch wall-clock time. The TopoCNN variant achieves +1.51% and +1.46% at 22% overhead. TopoAdamW is available as an open-source PyTorch package at `https://github.com/SVAH-X/topoadamw` and on PyPI (`pip install topoadamw`).

1 Introduction

The learning rate is the single most consequential hyperparameter in deep learning optimisation. Too large and training diverges; too small and convergence is needlessly slow. Despite decades of research on adaptive optimisers [Kingma and Ba, 2014, Loshchilov and Hutter, 2017b] and learning rate schedules [Smith, 2015, Loshchilov and Hutter, 2017a], the dominant paradigm remains *fixed or schedule-driven*: the step size is determined before training begins, without reference to what the loss landscape actually looks like at any given point in the optimisation trajectory.

A body of work in loss landscape visualisation [Li et al., 2018, Goodfellow et al., 2014] shows that the curvature of the landscape changes substantially over the course of training and differs qualitatively across architectures. Sharp minima correlate with poor generalisation [Hochreiter and Schmidhuber, 1997, Keskar et al., 2016]; flat regions can be traversed faster without risk. These observations suggest an opportunity: if we can *detect* the current geometric regime on-the-fly, we can adjust the learning rate accordingly.

Topological Data Analysis (TDA) provides a principled framework for characterising the shape of a function from a finite set of samples [Edelsbrunner and Harer, 2010, Carlsson, 2009]. Persistent homology, in particular, captures multi-scale connectivity and loop structure in a representation (the *persistence diagram*) that is stable under small perturbations [Cohen-Steiner et al., 2007]. Recent work has applied TDA to neural network loss landscapes [Rieck et al., 2019] and to characterising the training dynamics [TODO: (add further citations here)].

We make the following contributions:

1. **TopoAdamW (heuristic mode).** A lightweight LR controller that probes 25 points in a 2D filter-normalised random subspace every T steps and applies a sharpness/variance heuristic to scale the LR. Total overhead per probe is $\mathcal{O}(25 \cdot C_{\text{fwd}})$ where C_{fwd} is the cost of one forward pass.

2. **TopoAdamW-TDA (TopoCNN mode).** An extension that computes a persistence image of the local loss surface and feeds it to a small CNN classifier, TopoCNN, which is bootstrapped online from the heuristic labels. This replaces the hand-coded decision boundary with a learned one once enough samples are available.

3. **Open-source release.** A drop-in PyTorch replacement for `torch.optim.AdamW` with no new required dependencies (`gudhi` is optional).

2 Related Work

Adaptive optimisers. Adam [Kingma and Ba, 2014] and AdamW [Loshchilov and Hutter, 2017b] maintain per-parameter second-moment estimates to adapt effective step sizes. AMSGrad [Reddi et al., 2018] fixes a convergence issue in Adam. Adagrad [Duchi et al., 2011] and RMSProp accumulate squared gradients. All of these are *gradient-based* adaptations: the geometry signal comes entirely from the gradient sequence, not from explicit landscape probing. TopoAdamW is orthogonal—it wraps any base optimiser and adjusts the global LR using geometric information from forward-pass evaluations.

Learning rate schedules. Cosine annealing [Loshchilov and Hutter, 2017a], warmup, and cyclical schedules [Smith, 2015] modulate the LR as a function of time, not landscape geometry. ReduceLROnPlateau monitors validation loss and halves the LR on plateaus. TopoAdamW instead reacts to the *current* local geometry, independent of wall-clock time or plateau detection.

Sharpness-Aware Minimisation (SAM). SAM [Foret et al., 2021] explicitly seeks flat minima by perturbing parameters in the gradient direction and computing a second backward pass. Unlike TopoAdamW, SAM modifies the gradient rather than the learning rate, and requires two backward passes per step versus our one forward pass per probe step. The two approaches are complementary and could in principle be combined.

Loss landscape geometry. Li et al. [2018] introduced filter-wise normalisation for 2D loss landscape visualisation, which we adopt for our random subspace directions. Keskar et al. [2016] showed empirically that large-batch training converges to sharper minima with worse generalisation, motivating sharpness as a useful proxy.

TDA in machine learning. Persistent homology has been used to analyse the topology of data manifolds [Carlsson, 2009], to regularise neural networks [Chen et al., 2019], and to study loss landscapes [Rieck et al., 2019]. To our knowledge, TopoAdamW is the first work to close the loop by using persistence images as an *online* control signal for learning rate adaptation.

3 Method

3.1 Loss Landscape Probing

Let $\theta^* \in \mathbb{R}^D$ denote the current parameter vector. We sample two filter-normalised random directions $\mathbf{d}_1, \mathbf{d}_2 \in \mathbb{R}^D$ following Li et al. [2018]: for each layer ℓ with weight tensor W_ℓ, a random tensor of the same shape is drawn and rescaled filter-wise so that each filter's norm matches that of the corresponding filter in W_ℓ. The directions are concatenated into flat vectors and the loss is evaluated on a mini-batch $(\mathbf{x}, \mathbf{y})$ at perturbed parameters

$$\theta(u, v) = \theta^* + u\,\mathbf{d}_1 + v\,\mathbf{d}_2, \quad (u, v) \in \mathcal{C}, \tag{1}$$

where $\mathcal{C}$ is a set of evaluation coordinates.

Sparse probe (heuristic mode). We evaluate at 25 coordinates: the centre $(0, 0)$, eight axis-aligned and diagonal neighbours at step size $h = 2\delta/(G-1)$ (matching one grid cell width for reference grid size $G = 15$ and span $\delta = 0.12$), and $n_s = 16$ uniform random samples within $[-\delta, \delta]^2$. Total cost: 25 forward passes.

Dense probe (TDA mode). For the persistence image we evaluate a $G_{\text{tda}} \times G_{\text{tda}}$ regular grid (default $G_{\text{tda}} = 7$, i.e. 49 forward passes).

In both cases the model parameters are restored exactly after probing (no gradient computation; executed under `torch.no_grad()`).

3.2 Geometric Feature Extraction

From the sparse probe we extract two scalar features:

$$\text{Sharpness} = \frac{\bar{\ell}_{\text{nbr}} - \ell_0}{\ell_0 + \epsilon}, \tag{2}$$

$$\text{Variance} = \frac{\sigma(\{\ell_i\}_{i=1}^{25})}{\mu(\{\ell_i\}_{i=1}^{25}) + \epsilon}, \tag{3}$$

where ℓ_0 is the centre loss, $\bar{\ell}_{\text{nbr}}$ is the mean of the eight neighbour losses, σ and μ are sample standard deviation and mean over all 25 losses, and $\epsilon = 10^{-8}$. Sharpness measures how much the immediate neighbourhood rises above the current position; variance measures overall roughness of the sampled region.

3.3 Heuristic Learning Rate Controller

Given the features above, the LR scaling factor ρ is determined by:

$$\rho = \begin{cases} 1.15 & \text{if sharpness} < 0.1 \text{ and variance} < 0.3 \quad \text{(flat \& smooth)} \\ 0.80 & \text{if sharpness} > 0.5 \quad \text{(very sharp)} \\ 0.85 & \text{if variance} > 0.5 \quad \text{(high variance)} \\ 0.95 & \text{if sharpness} > 0.2 \quad \text{(moderately sharp)} \\ 1.00 & \text{otherwise (neutral)} \end{cases} \tag{4}$$

A *divergence brake* overrides Eq. (4): if the current centre loss exceeds $2\times$ the exponential moving average $\hat{\ell} \leftarrow 0.2\,\ell_0 + 0.8\,\hat{\ell}$, then $\rho = 0.8$ regardless. The new LR for group i is clipped to $[\alpha_i^0 \cdot r_{\min}^{(i)},\ \alpha_i^0 \cdot r_{\max}^{(i)}]$ where α_i^0 is the initial LR and $r_{\min}^{(i)}, r_{\max}^{(i)}$ are per-group floor and ceiling ratios (defaults 0.2 and 1.0).

3.4 Persistence Image and TopoCNN

Persistence image. Given the $G_{\text{tda}} \times G_{\text{tda}}$ loss grid, we run a cubical complex persistent homology computation [Edelsbrunner and Harer, 2010] using GUDHI [GUDHI Project, 2021] and extract the H_0 (connected component) birth–death pairs (b_k, d_k). We map each pair to image coordinates via log-ratio transforms centred on the current loss ℓ_0:

$$x_k = \frac{\log(\ell_0/(b_k + \epsilon))}{\log \kappa}, \quad y_k = \frac{\log(d_k/(\ell_0 + \epsilon))}{\log \kappa}, \tag{5}$$

clipped to $[0, 1]^2$, with cap $\kappa = 5$. We render a 50×50 Gaussian persistence image [Adams et al., 2017] weighted by distance from the origin, stacked as a 2-channel tensor (I, I) for TopoCNN input.

TopoCNN architecture. TopoCNN is a small CNN that maps $(2, 50, 50)$ persistence images to three landscape regime logits: FLAT (LR $\times 1.15$), NEUTRAL (LR $\times 1.00$), and DECEL (LR $\times 0.85$). The architecture is: Conv(2→16, 3×3) → ReLU → MaxPool → Conv(16→32) → ReLU → MaxPool → Conv(32→64) → ReLU → AdaptiveAvgPool → Linear(64→32) → ReLU → Linear(32→3).

Online bootstrapping. Supervision is obtained from the geometric heuristic (Section 3.3). A ring buffer of capacity 500 stores (I_t, y_t) pairs where y_t is the heuristic label. Once the buffer reaches 100 samples, TopoCNN is trained for 20 gradient steps; thereafter it is retrained every 25 new samples. Before training is complete, the optimizer falls back to the heuristic.

3.5 Full Algorithm

Algorithm 1 TopoAdamW training step

Require: model f_θ, base AdamW optimizer, batch $(\mathbf{x}, \mathbf{y})$, criterion $\mathcal{L}$, interval T, warmup W

Compute $\hat{y} = f_\theta(\mathbf{x})$; $\ell = \mathcal{L}(\hat{y}, \mathbf{y})$
ℓ.backward(); AdamW.step(); $t \leftarrow t + 1$
if $t > W$ **and** $t \bmod T = 0$ **then**
    Sample directions $\mathbf{d}_1, \mathbf{d}_2$ (filter-normalised)
    Evaluate losses at $\mathcal{C}$ under `no_grad`; restore θ
    Compute sharpness, variance (Eqs. 2–3)
    if use_topo_cnn **and** TopoCNN is ready **then**
        $\rho \leftarrow$ TopoCNN(persistence image)
    else
        $\rho \leftarrow$ heuristic (Eq. 4)
    end if
    if $\ell_0 > 2\hat{\ell}$ **then** $\rho \leftarrow 0.8$
    end if ▷ divergence brake
    Update $\alpha \leftarrow \text{clip}(\alpha \cdot \rho,\ \alpha^0 r_{\min},\ \alpha^0 r_{\max})$
end if

4 Experiments

4.1 Setup

Model. We use CifarNet, a five-block convolutional network (Conv → BatchNorm → ReLU → MaxPool, followed by a two-layer MLP classifier) with approximately [TODO: X]M parameters.

Datasets. CIFAR-10 and CIFAR-100 [Krizhevsky, 2009], split into 50,000 training and 10,000 test images of size 32×32. Standard data augmentation: random horizontal flip and random crop with padding 4.

Training details. Batch size 128; weight decay 5×10^{-4}; 50 epochs for CIFAR-10 (lr 10^{-3}), 100 epochs for CIFAR-100 (lr 5×10^{-4}). TopoAdamW parameters: $T = 50$, $W = 150$, $r_{\max} = 1.0$, $r_{\min} = 0.2$, span $\delta = 0.12$, $n_s = 16$. All experiments run on [TODO: hardware spec]. [TODO: Report mean ± std over N seeds.]

4.2 Main Results

Table 1: Validation accuracy and loss on CIFAR-10 (50 epochs) and CIFAR-100 (100 epochs). [TODO: Add mean ± std over 5 seeds.]

	CIFAR-10		CIFAR-100	
Optimizer	Acc (%)	Loss	Acc (%)	Loss
AdamW (baseline)	84.76	0.4983	57.26	1.9423
TopoAdamW (heuristic)	86.19	0.4523	58.76	1.7980
TopoAdamW-TDA (TopoCNN)	**86.27**	**0.4559**	58.72	1.8520
Δ (heuristic vs. AdamW)	+1.43	−0.046	+1.50	−0.144
Δ (TDA vs. AdamW)	+1.51	−0.042	+1.46	−0.090

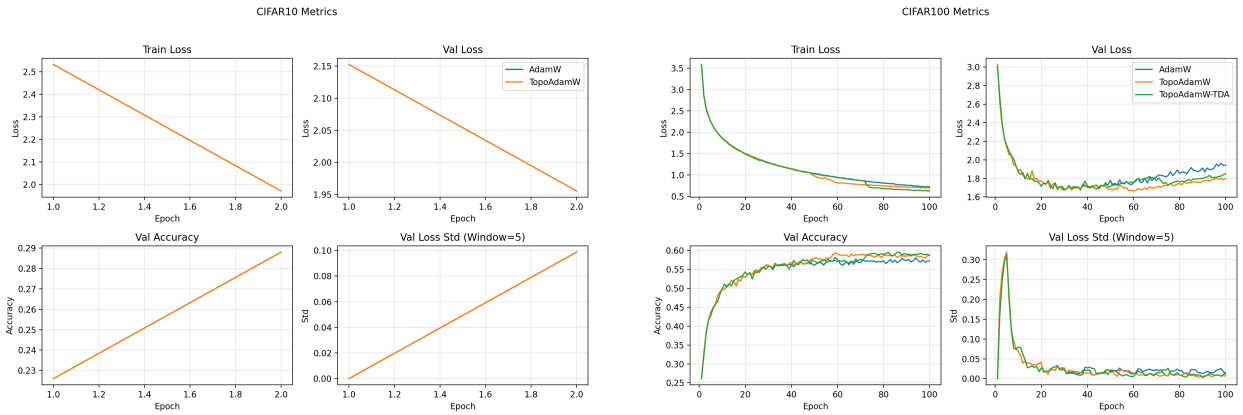

Figure 1: Training and validation curves on CIFAR-10 (left) and CIFAR-100 (right). TopoAdamW reaches higher accuracy faster in both cases.

CIFAR-10. Both TopoAdamW variants outperform AdamW by over 1.4 percentage points. The TopoCNN variant marginally exceeds the heuristic (+0.08%), suggesting the learned classifier adds modest benefit at 50 epochs.

CIFAR-100. Over 100 epochs the TopoCNN mode keeps the LR higher for longer (epochs 1–36 at full LR vs. heuristic which begins cutting at epoch 30), then cuts decisively, squeezing additional training progress before regularising. This behaviour emerges automatically from the online bootstrapping, without any schedule engineering.

4.3 Overhead Analysis

Table 2: Per-epoch wall-clock overhead relative to AdamW baseline. [TODO: Report on GPU as well.]

Mode	Forward passes / probe	Overhead
AdamW (baseline)	0	—
TopoAdamW (heuristic)	25	$\approx 14\%$
TopoAdamW-TDA (TopoCNN)	49	$\approx 22\%$
Naive 15×15 grid	225	$\approx 9\times$ heuristic

The sparse probe (25 points) achieves a $9\times$ reduction in probe cost versus the naive 15×15 grid while preserving sufficient geometric information for the heuristic.

4.4 Ablation Studies

[TODO: The following ablations should be run and reported.]

Probe density. Compare $n_s \in \{0, 4, 8, 16, 32\}$ random samples to assess sensitivity of the variance estimate.

Probe interval. Compare $T \in \{10, 25, 50, 100, 200\}$ to study the trade-off between overhead and adaptation frequency.

Heuristic thresholds. Vary the sharpness and variance thresholds in Eq. (4) to evaluate robustness.

TDA bootstrapping speed. Compare `min_samples` $\in \{50, 100, 200\}$ for the TopoCNN warmup to assess how quickly the CNN takes over from the heuristic.

5 Discussion

Why does a 2D subspace probe capture useful geometry? The filter-normalised directions ensure that each direction has meaningful scale relative to the current parameters. While a 2D slice is a coarse approximation of the full-dimensional landscape, Li et al. [2018] showed that such slices preserve qualitative geometric structure. Our sharpness metric (Eq. 2) is closely related to the trace of the Hessian restricted to the probed subspace, which has been linked to generalisation performance [Keskar et al., 2016, Foret et al., 2021].

Limitations.

- **Scale of experiments.** Current results are on CIFAR with a small convolutional network. Behaviour on larger architectures (ResNet, ViT) and datasets (ImageNet) remains to be validated.
- **Uniform LR factor.** All parameter groups receive the same scaling factor. Layer-wise LR adaptation (e.g., different factors for early vs. late layers) could be more effective.
- **TopoCNN warmup.** With default settings, TopoCNN activates after ≈13 epochs on CIFAR-10. For short training runs (<20 epochs) the optimizer remains in heuristic mode.
- **Overhead on GPU.** The probe's forward passes benefit from GPU parallelism but are not batched; batching perturbations could further reduce overhead.

Future work. Layer-wise probing; combining with SAM-style gradient sharpness; applying to transformer language model fine-tuning; theoretical convergence analysis.

6 Conclusion

We presented TopoAdamW, a topology-guided learning rate controller that probes a sparse neighbourhood of the current parameter vector at regular training intervals and adjusts the learning rate based on the detected geometric regime. The heuristic mode requires no additional dependencies and adds only 25 forward passes per probe. The optional TopoCNN extension learns to classify persistence images of the loss surface online, bootstrapped from the heuristic labels. Both modes improve over AdamW on CIFAR-10 and CIFAR-100 with moderate overhead. The package is publicly available at `https://github.com/SVAH-X/topoadamw`.

References

H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18:1–35, 2017.

G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

C. Chen, X. Ni, Q. Bai, and Y. Wang. A topological regularizer for classifiers via persistent homology. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Discrete & Computational Geometry*, volume 37, pages 103–120, 2007.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

GUDHI Project. GUDHI library. `https://gudhi.inria.fr`, 2021.

S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017a.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017b.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

B. Rieck, M. Togninalli, C. Bock, M. Moor, M. Horn, T. Gumbsch, and K. Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *International Conference on Learning Representations*, 2019.

L. N. Smith. Cyclical learning rates for training neural networks. *arXiv preprint arXiv:1506.01186*, 2015.