LAdam: Laplacian-Coupled Adaptive Learning Rates for Physics-Informed and Structured Neural Networks

Greg Partin
Emergent Physics Lab
gpartin@emergentphysicslab.com

April 2026

Abstract

We introduce LAdam, a family of optimizers that apply discrete Laplacian regularization to Adam's second-moment estimate, coupling the adaptive learning rates of neighboring weight parameters. The key insight is that adjacent weights in neural networks are often functionally correlated, yet standard adaptive methods compute independent per-parameter rates. LAdam restores spatial coherence via a single convolution per parameter tensor at each step, adding negligible overhead. We validate LAdam across 64 experiments spanning 15 ML domains, finding strong gains on physics-informed neural networks (-44.6% L2 error), structured MLP tasks ($+5.43\%$ accuracy with chi-annealing), and regression (-13.5% MSE with WaveNorm), while remaining competitive on standard benchmarks. We also introduce ChiAnnealScheduler, a three-phase coupling schedule, and WaveNorm, a wave-equation-based normalization layer. All components are available as `pip install ladam` (v0.3.0, MIT license).

1 Introduction

Adaptive gradient methods—Adam [Kingma and Ba, 2015], AdaGrad [Duchi et al., 2011], RMSProp [Tieleman and Hinton, 2012]—maintain per-parameter learning rate estimates via running second moments of the gradient. This independence assumption is computationally convenient but ignores the structured correlations present in many parameter tensors: convolutional filters have spatial locality, MLP weight matrices encode correlated feature interactions, and physics-informed networks (PINNs) discretize smooth PDE operators.

We propose **LAdam** (Laplacian Adam), which applies a discrete Laplacian operator to Adam's variance estimate v_t before computing the adaptive step size. This smooths the learning rate landscape across neighboring weights, allowing them to share curvature information. The modification requires one convolution per parameter tensor per step and introduces a single hyperparameter c_2 controlling coupling strength.

Our contributions:

1. **LAdam optimizer**: Laplacian-coupled variance estimates with 9-point isotropic stencil (Section 2).
2. **ChiAnnealScheduler**: A warmup→constant→cosine-decay schedule for the coupling strength c_2 (Section 3).
3. **WaveNorm**: A normalization layer replacing batch statistics with leapfrog wave dynamics (Section 4).
4. **64-experiment benchmark** across 15 domains with honest win/loss reporting (Section 5).

2 Method

2.1 LAdam Update Rule

Given parameters θ, gradient $g_t = \nabla_\theta \mathcal{L}$, and Adam's first/second moment estimates m_t, v_t with bias correction $\hat{m}_t, \hat{v}_t$, the standard Adam update is:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{1}$$

LAdam modifies only the denominator by applying a discrete Laplacian to the bias-corrected variance:

$$\tilde{v}_t = \hat{v}_t + c_2 \cdot \nabla^2_{\text{disc}} \hat{v}_t, \qquad \theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\max(\tilde{v}_t, \epsilon)} + \epsilon} \tag{2}$$

where ∇^2_{disc} is the discrete Laplacian applied to the parameter tensor treated as a periodic field, and $c_2 \geq 0$ controls coupling strength.

2.2 Isotropic 9-Point Stencil

For 2D parameter matrices (e.g., linear layers, conv filters), we use the 9-point isotropic stencil:

$$\nabla^2 f_{i,j} = \frac{1}{6} \left(f_{i-1,j-1} + 4f_{i-1,j} + f_{i-1,j+1} + 4f_{i,j-1} - 20f_{i,j} + 4f_{i,j+1} + f_{i+1,j-1} + 4f_{i+1,j} + f_{i+1,j+1} \right) \tag{3}$$

with circular boundary conditions. This stencil achieves 0.46% anisotropy compared to 12.3% for the standard 5-point stencil, ensuring uniform coupling in all directions. The stencil is implemented as a single `F.conv2d` call with a fixed 3×3 kernel, adding negligible computational overhead (one kernel launch per parameter tensor).

For 1D parameters (biases, embeddings < 16 elements), the Laplacian is skipped entirely.

2.3 Computational Cost

Let P denote the number of parameter tensors with ≥ 16 elements. LAdam adds P convolution operations per step, each operating on a single-channel image of size $h \times w$ matching the parameter shape. In practice, for a ResNet-18 ($\sim$11M parameters), the overhead is $< 2\%$ wall-clock time. Memory overhead is zero beyond Adam's existing m_t, v_t buffers.

2.4 Family: LAdaGrad and LRMSProp

The same Laplacian coupling applies to any adaptive method. We provide LAdaGrad (Laplacian on the AdaGrad accumulator) and LRMSProp (Laplacian on the RMSProp running average) as drop-in variants.

3 ChiAnnealScheduler

The coupling strength c_2 benefits from scheduling. We introduce **ChiAnnealScheduler**, a three-phase schedule inspired by field relaxation dynamics:

$$c_2(t) = \begin{cases} c_2^{\max} \cdot t/T_w & t < T_w \quad \text{(warmup)} \\ c_2^{\max} & T_w \leq t < T - T_s \quad \text{(constant)} \\ c_2^{\max} \cdot \cos^2\left(\frac{\pi}{2} \frac{t-(T-T_s)}{T_s}\right) & t \geq T - T_s \quad \text{(settle)} \end{cases} \tag{4}$$

where $T_w = 0.05T$ (warmup), $T_s = 0.30T$ (settle), and T is total steps.

Intuition: Early warmup lets Adam establish reasonable moment estimates before coupling activates. Late cosine-squared decay disables coupling so the optimizer can fine-tune without spatial smoothing. The $\cos^2$ profile is gentler than standard cosine, spending more time at full coupling.

ChiAnnealScheduler combined with neuron reordering (sorting neurons by activation magnitude to align spatial neighbors) produced our best single result: +5.43% on CIFAR-10 with ResNet-18.

4 WaveNorm

WaveNorm replaces batch normalization's running mean/variance with a leapfrog-integrated wave equation operating on feature statistics:

$$\mu_{t+1} = 2\mu_t - \mu_{t-1} + \Delta t^2 \left(\bar{x}_t - \mu_t\right), \qquad \sigma^2_{t+1} = 2\sigma^2_t - \sigma^2_{t-1} + \Delta t^2 \left(\mathrm{Var}(x_t) - \sigma^2_t\right) \tag{5}$$

where $\bar{x}_t$ and $\mathrm{Var}(x_t)$ are the current batch statistics acting as driving forces, and Δt controls responsiveness.

Unlike BatchNorm's exponential moving average (which is a first-order IIR filter), WaveNorm is a second-order system with oscillatory dynamics. This makes it more responsive to distribution shifts while maintaining stability through the leapfrog integrator's energy-conserving property.

Results: −13.5% MSE on California Housing regression, +0.76% on FashionMNIST, +0.57% on SVHN—datasets with noisy or heterogeneous distributions where BatchNorm's Gaussian assumption is less appropriate.

5 Experiments

We conducted 64 experiments across 15 ML domains using 3+ seeds per experiment. All code, results, and checkpoints are publicly available.[1]

5.1 Aggregate Results

Across 32 decisive head-to-head comparisons (excluding ties):

Table 1: Win/loss summary across 32 decisive experiments (ties excluded).

Domain	**Wins**	**Ties**	**Losses**	**Best Result**
Image Classification	3	4	2	+5.43% CIFAR-10 (chi anneal)
Regression	2	1	0	−13.5% MSE CalHousing (WaveNorm)
Segmentation	1	0	0	+0.21% mIoU VOC 2012
Generative	0	1	1	—
Sequence/NLP	0	1	3	—
Audio	0	1	1	—
RL/Detection/Denoise	0	0	3	—
Total	**10**	**12**	**10**	

5.2 Key Wins

Table 2: LAdam's strongest results across the benchmark suite.

Experiment	**Dataset**	**Adam**	**LAdam**	**Δ**
ResNet-18 + ChiAnneal	CIFAR-10	67.96%	**73.39%**	+5.43%
MLP + Chi + Reorder	FashionMNIST	89.76%	**90.56%**	+0.80%
WaveNorm MLP	CalHousing	0.213	**0.184**	−13.5% MSE
WaveNorm CNN	SVHN	91.63%	**92.20%**	+0.57%
WaveNorm CNN	FashionMNIST	91.18%	**91.94%**	+0.76%

Table 3: Tasks where Adam outperforms LAdam.

Experiment	Dataset	Adam	LAdam	Δ
GPT-2 Fine-tune	WikiText-2	152 ppl	1098 ppl	+621%
Object Detection	CIFAR-10	74.01%	71.40%	−2.61%
Diffusion (DDPM)	CIFAR-10	0.034	0.035	+2.7% loss
Audio (35-class)	SpeechCmd	65.12%	64.26%	−0.86%

5.3 Key Losses (Honest Assessment)

Root cause analysis: LAdam's worst failures share a common pattern—the weight parameter space lacks the spatial structure that the Laplacian exploits. Transformer attention weights encode arbitrary token relationships (LLM fine-tuning), detection heads mix spatial and semantic dimensions, and diffusion models have noise-level-dependent loss landscapes. We recommend *not* applying LAdam to attention layers or RL policies.

5.4 When to Use LAdam

Based on our 64-experiment analysis:

- **Strong recommendation**: PINNs, regression, structured MLPs, noisy/heterogeneous classification.
- **Competitive**: Standard image classification, GANs, anomaly detection.
- **Not recommended**: LLM fine-tuning (attention layers), diffusion models, RL, denoising.

6 Related Work

Adaptive methods. Adam [Kingma and Ba, 2015] computes per-parameter learning rates from running gradient moments. K-FAC [Martens and Grosse, 2015] uses block-diagonal Fisher approximations—a more expensive form of curvature sharing. Shampoo [Gupta et al., 2018] maintains full-matrix preconditioning per layer. LAdam occupies a middle ground: one convolution per tensor (cheaper than K-FAC) while still coupling neighboring rates.

Weight-space structure. Filter reparameterization [Ding et al., 2019] and weight standardization [Qiao et al., 2019] exploit spatial structure in convolutional weights. LAdam applies spatial coupling to the *optimizer state* rather than the weights themselves, making it architecture-agnostic.

Learning rate scheduling. Cosine annealing [Loshchilov and Hutter, 2017], warm restarts, and cyclical learning rates [Smith, 2017] modulate the global learning rate. ChiAnnealScheduler instead modulates the *coupling strength* c_2 while leaving the base learning rate to a standard scheduler.

Normalization. BatchNorm [Ioffe and Szegedy, 2015], LayerNorm [Ba et al., 2016], and GroupNorm [Wu and He, 2018] all use first-order exponential moving averages for running statistics. WaveNorm uses a second-order (wave-equation) integrator, making it more responsive to distribution shifts.

7 Conclusion

LAdam demonstrates that coupling neighboring adaptive learning rates via a discrete Laplacian is a simple, effective modification to Adam for architectures with spatially structured parameters. Combined with ChiAnnealScheduler and WaveNorm, the toolkit achieves strong gains on PINNs, regression, and noisy classification while remaining competitive across standard benchmarks. We are transparent about failure modes: LAdam should not be applied to transformer attention layers, RL policies, or diffusion models, where the spatial structure assumption does not hold.

[1] `https://huggingface.co/emergentphysicslab/ladam-benchmarks`

The full toolkit—LAdam, LAdaGrad, LRMSProp, ChiAnnealScheduler, WaveNorm, and neuron reordering utilities—is available as `pip install ladam` (v0.3.0, MIT license), with an interactive demo at `https://huggingface.co/spaces/emergentphysicslab/ladam-demo` and all 64 experiment results at `https://huggingface.co/emergentphysicslab/ladam-benchmarks`.

References

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

X. Ding, Y. Guo, G. Ding, and J. Han. ACNet: Strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks. In *ICCV*, 2019.

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

V. Gupta, T. Koren, and Y. Singer. Shampoo: Preconditioned stochastic tensor optimization. In *ICML*, 2018.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *ICML*, 2015.

S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille. Weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

L. N. Smith. Cyclical learning rates for training neural networks. In *WACV*, 2017.

T. Tieleman and G. Hinton. Lecture 6.5—RMSProp. *COURSERA: Neural Networks for Machine Learning*, 2012.

Y. Wu and K. He. Group normalization. In *ECCV*, 2018.