NEUROPLASTIC: PLASTICITY-INSPIRED MODULATION OF GRADIENT-DERIVED OPTIMIZATION STATISTICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimization algorithms are fundamental to modern deep learning, yet most widely used methods construct updates from the current gradient and statistics of its history. We introduce **NeuroPlastic**, a plasticity-modulated optimizer that augments gradient-based updates with an adaptive multi-signal modulation mechanism inspired by multi-factor synaptic plasticity, a concept from neurobiology. NeuroPlastic dynamically scales gradient updates using interacting components that capture gradient, activity-like, and memory-like statistics, forming a lightweight modulation layer compatible with standard deep learning training pipelines. Across image classification benchmarks, NeuroPlastic consistently improves over a controlled gradient-only ablation, with more pronounced gains on the Fashion-MNIST benchmark and in reduced-data regimes. In transfer experiments on CIFAR-10 with ResNet-18, the method remains stable and competitive without retuning. These results suggest that multi-signal plasticity-inspired modulation can provide a useful extension to conventional gradient-driven optimization, particularly when learning signals are limited or noisy, and offer a promising direction for gradient-based methods in deep learning. Code is available at: https://github.com/Sunnylincc/NeuroPlastic-Optimizer.

1 INTRODUCTION

Optimization algorithms are a foundation of modern deep learning. The ability of neural networks to generalize depends not only on model architecture and training data, but critically on how parameters are updated during training (Goodfellow et al., 2016). Over the past decade, stochastic gradient descent and adaptive methods such as Adam and AdamW have become the dominant tools for this purpose (Robbins & Monro, 1951; Kingma & Ba, 2015; Loshchilov & Hutter, 2019). They demonstrate strong empirical performance across vision, language, and large-scale foundation model training (He et al., 2016; Brown et al., 2020; Vaswani et al., 2017). These methods share a common design principle: parameter updates are computed as functions of local gradient statistics, either raw stochastic gradients with momentum, or first- and second-order exponential moving averages (Polyak, 1964; Rumelhart et al., 1986; Kingma & Ba, 2015). This design is computationally efficient and broadly effective, but it leaves open how different gradient-derived statistics operating over different timescales should be combined within the update rule. In particular, current gradient magnitude and statistics summarizing recent optimization history may provide complementary information about how strongly a parameter update should be expressed. NeuroPlastic explores this design question through a plasticity-inspired modulation mechanism (Frémaux & Gerstner, 2016).

Unlike stochastic gradients, biological learning systems like the human brain organize learning through multiple interacting components. Classical Hebbian learning describes synaptic strengthening as a function of correlated pre- and post-synaptic activity (Hebb, 1949). Subsequent work in computational neuroscience extended this idea into three-factor or multi-factor plasticity rules, where synaptic updates depend not only on co-activation but also on additional signals (Gerstner, 2011; Lillicrap et al., 2020). These signals may include neuromodulators, activity traces reflecting the history of a neuron's firing, and network-level regulatory mechanisms that maintain stability over time (Hebb, 1949; Frémaux & Gerstner, 2016). Together, these factors allow learning to be modu-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

lated by context: what has recently occurred, what the network is currently doing, and signals that reflect the broader relevance of a given update. A feature relevant to optimizer design is therefore the interaction between an instantaneous learning signal and quantities that summarize its recent history. We use this principle as a structural inspiration rather than a literal biological mapping: the activity-like and memory-like quantities in NeuroPlastic are derived from the stochastic gradient sequence rather than from independent biological signals.

However, translating these neurobiological principles into scalable deep learning optimizers remains challenging, as neural plasticity mechanisms are not readily compatible with gradient-based training in modern deep learning architectures. Therefore, we ask a focused question: can multi-factor plasticity be implemented in a lightweight optimizer for standard deep learning settings?

To investigate this question, we introduce **NeuroPlastic**, a gradient-based optimizer that constructs a modulation coefficient from three normalized, gradient-derived statistics: current gradient magnitude, an exponential moving average of gradient magnitude, and a memory-like statistic derived from first- and second-moment traces. These quantities jointly determine a modulation factor that scales the gradient before a bounded stabilization step. Importantly, NeuroPlastic does not introduce an information source beyond the stochastic gradient sequence; instead, it combines gradient-derived statistics with different temporal characteristics within a single update rule.

We evaluate NeuroPlastic on MNIST (LeCun et al., 1998) and Fashion-MNIST (Xiao et al., 2017) using a controlled gradient-magnitude-based ablation, including reduced-data settings. We further examine transfer behavior on CIFAR-10 with ResNet-18 (Krizhevsky, 2009; He et al., 2016) using a configuration selected on the lightweight benchmarks. In this transferred setting, NeuroPlastic remains trainable but does not outperform tuned SGD, Adam, or AdamW; we therefore use the CIFAR-10 experiment primarily to characterize transfer stability and optimization sensitivity rather than tuning-parity performance.

Our main contributions can be summarized as follows:

1. We introduce **NeuroPlastic**, a biologically inspired optimizer that modulates gradient updates through a plasticity signal combining instantaneous and historical gradient-derived statistics into activity- and memory-like modulation components.
2. We design a stabilization mechanism that regulates update magnitudes and maintains stable optimization dynamics across learning rates.
3. We develop a reproducible benchmarking framework and diagnostic tools for analyzing plasticity-modulated optimization behavior during training.
4. We empirically evaluate NeuroPlastic across standard benchmarks and data regimes, demonstrating consistent improvements over gradient-only updates, with the advantage scaling with task difficulty and data scarcity.

2 Related Work

Prior work has explored richer optimizer state, curvature information, learned update rules, and biologically motivated learning mechanisms. NeuroPlastic introduces no information source beyond the gradient; instead, it asks whether several gradient-derived statistics computed over different timescales can be combined within a bounded modulation rule.

2.1 Gradient-Based and Alternative Optimizers

SGD and momentum-based methods remain widely used for neural-network optimization, while Adam, AdamW, AdaGrad, RMSProp, and AdaDelta adapt updates using accumulated gradient statistics (Kingma & Ba, 2015; Loshchilov & Hutter, 2019; Duchi et al., 2011; Zeiler, 2012; Tieleman & Hinton, 2012). Other approaches enrich the optimizer through curvature information, as in K-FAC and Shampoo (Martens & Grosse, 2015; Gupta et al., 2018; Eschenhagen et al., 2023), or learn and search for update rules directly (Andrychowicz et al., 2016; Chandra et al., 2022; Chen

et al., 2023). NeuroPlastic differs in that its update structure is fixed in advance and uses only current and historical gradient-derived signals.

2.2 Biologically Inspired Learning Rules

Hebbian and three-factor plasticity rules describe learning as the interaction of local activity, historical traces, and modulatory signals (Hebb, 1949; Gerstner, 2011; Frémaux & Gerstner, 2016; Gerstner et al., 2018). Related machine-learning work has explored biologically plausible credit assignment, including feedback alignment and target propagation (Lillicrap et al., 2016; Nøkland, 2016; Lee et al., 2015; Meulemans et al., 2020), as well as differentiable and neuromodulated plasticity (Miconi et al., 2018; 2019).

NeuroPlastic shares the idea of factorized modulation but differs in objective and implementation. It does not replace backpropagation or introduce trainable plastic connections; instead, it uses plasticity as an organizational analogy for combining instantaneous gradient magnitude, magnitude history, and moment-based history within a standard optimizer. To our knowledge, this particular bounded combination has not previously been evaluated under matched ablations.

3 Methods

3.1 Overview

We propose **NeuroPlastic**, a gradient-based optimizer in which the raw gradient is scaled by a coefficient constructed from several normalized, gradient-derived statistics and then stabilized before being applied. Let θ_t denote the model parameters at step t, and let

$$g_t = \nabla_\theta \mathcal{L}(\theta_t) \tag{1}$$

denote the stochastic gradient. The update takes the form

$$\theta_{t+1} = \theta_t - \eta_t \, \widetilde{u}_t, \tag{2}$$

where η_t is the effective learning rate and $\widetilde{u}_t$ is the stabilized, plasticity-modulated update.

In the default rule-based configuration, NeuroPlastic first computes a per-parameter modulation factor α_t, forms the modulated update

$$u_t = \alpha_t \odot g_t, \tag{3}$$

and then applies a stabilization operator

$$\widetilde{u}_t = \mathcal{S}(u_t). \tag{4}$$

Figure 1 summarizes the full procedure.

3.2 Activity and Memory Signals

NeuroPlastic maintains three internal statistics derived from the gradient: a gradient-magnitude trace, a first-moment trace, and a second-moment trace. Throughout, the terms "activity-like" and "memory-like" describe computational roles only; these quantities are not measurements or mechanistic models of neuronal activity, eligibility traces, or neuromodulatory signals.

Gradient-magnitude trace. The activity-like trace is an exponential moving average of gradient magnitudes:

$$a_t = \beta_a a_{t-1} + (1 - \beta_a)|g_t|. \tag{5}$$

Momentum and variance traces. We also maintain Adam-style first- and second-moment statistics:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{6}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t \odot g_t). \tag{7}$$

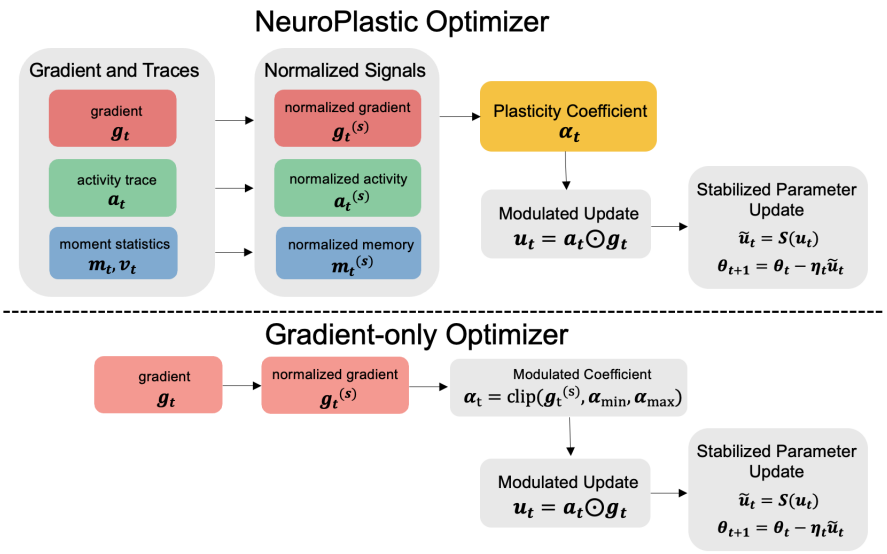


Figure 1: **Overview of the NeuroPlastic optimizer.** NeuroPlastic combines normalized instantaneous and historical gradient-derived signals into a bounded plasticity coefficient that modulates the gradient before stabilization. The gradient-magnitude-only ablation retains only instantaneous gradient-magnitude modulation.

3.3 Normalized Plasticity Signals

From the quantities above, the optimizer constructs three normalized signals. For a parameter tensor with d scalar coordinates, all means, RMS values, and norms below are computed over the coordinates of that tensor, independently of every other parameter tensor; that is, $\text{mean}(x) = d^{-1}\sum_{i=1}^{d} x_i$ and $\|x\|_2 = (\sum_{i=1}^{d} x_i^2)^{1/2}$. All operations are element-wise unless scalar modulation is explicitly enabled.

Gradient signal.

$$g_t^{(s)} = \frac{|g_t|}{\text{mean}(|g_t|) + \epsilon}. \tag{8}$$

Activity signal.

$$a_t^{(s)} = \frac{a_t}{\text{mean}(a_t) + \epsilon}. \tag{9}$$

Memory signal. The memory signal combines the first- and second-moment traces:

$$m_t^{(s)} = \frac{|m_t|/(\sqrt{v_t} + \epsilon)}{\text{mean}\big(|m_t|/(\sqrt{v_t} + \epsilon)\big) + \epsilon}, \tag{10}$$

where m_t and v_t are the raw (non-bias-corrected) moment estimates defined in Equations equation 6–equation 7, and $\epsilon > 0$ is a small constant added for numerical stability.

We omit the usual Adam bias-correction factors because they contribute timestep-dependent common scale factors that the subsequent within-tensor normalization removes, up to their interaction with the numerical constant ϵ. The equations and the implementation therefore both use raw moments.

3.4 Plasticity Coefficient

Gradient-magnitude-only ablation. The ablation used in our experiments does *not* set $\alpha_t = 1$. It removes the magnitude-history and moment-history contributions while retaining instantaneous

gradient-magnitude modulation:

$$\alpha_t^{\text{grad}} = \text{clip}\Big(g_t^{(s)}, \alpha_{\min}, \alpha_{\max}\Big). \tag{11}$$

This ablation therefore isolates the contribution of the two history terms rather than the contribution of modulation as a whole.

Full plasticity rule. In the full rule-based configuration, the plasticity coefficient is a weighted combination of the three normalized signals,

$$\alpha_t^{\text{full}} = w_a a_t^{(s)} + w_g g_t^{(s)} + w_m m_t^{(s)}, \tag{12}$$

followed by clipping:

$$\alpha_t^{\text{full}} \leftarrow \text{clip}\big(\alpha_t^{\text{full}}, \alpha_{\min}, \alpha_{\max}\big). \tag{13}$$

The biological rules that motivate this design combine their factors multiplicatively, whereas Equation equation 12 combines the three signals additively and the resulting coefficient then multiplies the gradient. The correspondence is therefore organizational rather than functional, and we do not claim that α_t implements a three-factor product.

Ignoring coefficient clipping and holding the within-tensor normalization denominator fixed, the instantaneous gradient-magnitude contribution to coordinate i of the update is

$$w_g g_{t,i}^{(s)} g_{t,i} = \frac{w_g |g_{t,i}| g_{t,i}}{\text{mean}(|g_t|) + \epsilon}. \tag{14}$$

This component therefore amplifies coordinates that are large relative to the tensor mean and attenuates those that are small: it changes coordinate-wise conditioning rather than applying a single scalar to the whole gradient. Clipping α_t bounds the size of this effect.

Warmup interpolation. The final plasticity coefficient interpolates between the gradient-magnitude-only and full rules:

$$\alpha_t = \alpha_t^{\text{grad}} + \lambda_t \Big(\alpha_t^{\text{full}} - \alpha_t^{\text{grad}}\Big), \tag{15}$$

where

$$\lambda_t = s_{\text{plast}} \cdot \gamma_t^{\text{warmup}} \tag{16}$$

and s_{plast} is a plasticity scaling factor. For a linear warmup of T_{warmup} optimizer steps, the gate is

$$\gamma_t^{\text{warmup}} = \begin{cases} 1, & T_{\text{warmup}} = 0, \\ \min(1, t/T_{\text{warmup}}), & T_{\text{warmup}} > 0. \end{cases} \tag{17}$$

Note that the interpolation starts from the gradient-magnitude-only rule rather than from a neutral coefficient of one.

Parameter-wise versus scalar modulation. By default, α_t is computed per parameter. If parameter-wise modulation is disabled, the implementation replaces α_t by its mean and broadcasts the resulting scalar across all coordinates of the tensor.

3.5 Homeostatic Stabilization

Given the plasticity-modulated update $u_t = \alpha_t \odot g_t$ of Equation equation 3, NeuroPlastic applies a stabilization operator $\mathcal{S}(\cdot)$, which we call homeostatic by analogy with activity regulation in biological circuits rather than because it implements any such mechanism. The operator consists of two steps.

Step 1: Pre-gain norm clipping. Let τ denote the clipping threshold applied to the intermediate update. If $\|u_t\|_2 > \tau$, the update is rescaled:

$$u_t \leftarrow u_t \cdot \frac{\tau}{\|u_t\|_2 + \epsilon}. \tag{18}$$

Step 2: Bounded RMS-dependent gain. The update is then adjusted according to its root-mean-square (RMS) magnitude,

$$\mathrm{RMS}(u_t) = \sqrt{\mathrm{mean}(u_t^2) + \epsilon}. \tag{19}$$

Given a target RMS value r_{target}, the stabilizer computes a gain

$$\gamma_t^{\mathrm{stab}} = 1 + \rho\left(r_{\mathrm{target}} - \mathrm{RMS}(u_t)\right), \tag{20}$$

and clips it to a bounded interval:

$$\gamma_t^{\mathrm{stab}} \leftarrow \mathrm{clip}\big(\gamma_t^{\mathrm{stab}}, \gamma_{\min}, \gamma_{\max}\big). \tag{21}$$

The stabilized update is therefore

$$\widetilde{u}_t = \gamma_t^{\mathrm{stab}} u_t. \tag{22}$$

This stateless proportional rule provides a bounded correction rather than exact regulation to r_{target}, so in general $\mathrm{RMS}(\widetilde{u}_t) \neq r_{\mathrm{target}}$. Moreover, because the gain is applied after norm clipping and may exceed one, τ is not a strict upper bound on the final update; the implemented ordering gives only the looser bound $\|\widetilde{u}_t\|_2 \leq \gamma_{\max}\tau$. We retain this ordering for consistency with the reported experiments.

The implementation additionally supports layer-dependent scaling of the target RMS through a multiplicative factor; when all scaling factors are equal, the same target RMS applies across parameter groups, which is the setting used in all experiments reported here.

3.6 Weight Decay and Effective Learning Rate

The implementation supports both coupled and decoupled weight decay. In the coupled setting, the decay term is added directly to the update; in the decoupled setting, parameters are shrunk separately before the stabilized update is applied, analogous to AdamW-style regularization.

The optimizer also provides an optional learning-rate controller that rescales the base learning rate according to the mean discrepancy between α_t and α_t^{grad}. Defining

$$d_t = \mathrm{mean}\Big(\alpha_t - \alpha_t^{\mathrm{grad}}\Big), \tag{23}$$

the controller sets

$$c_t = \mathrm{clip}\big(1 + \alpha_{\mathrm{ctrl}} d_t, \eta_{\min}^{\mathrm{ctrl}}, \eta_{\max}^{\mathrm{ctrl}}\big), \qquad \eta_t = \eta c_t. \tag{24}$$

This controller is disabled in the default configuration and in all main experiments, so $\eta_t = \eta$ throughout Section 4.

4 Experiments

We evaluate NeuroPlastic in controlled experiments designed to characterize its behavior relative to the gradient-magnitude-only ablation and to selected standard optimizers. Because the present evaluation uses only three seeds and lightweight vision benchmarks, we report all performance differences descriptively and do not attach statistical claims to them. Architectures, training pipelines, and hyperparameters are given in Appendix B.

4.1 MNIST and Fashion-MNIST

We first evaluate the optimizer on two standard image-classification benchmarks, using MNIST as a lightweight sanity check and Fashion-MNIST as a harder follow-up task. Pairing the two lets us ask whether the contribution of the history terms changes as task difficulty increases. On both benchmarks the comparison is against the gradient-magnitude-only ablation alone; standard optimizers are compared on CIFAR-10 in Section 4.3.

On MNIST, NeuroPlastic differs from the gradient-magnitude-only ablation by approximately $+0.001$ in mean final test accuracy (0.1 percentage points) across three seeds. Given the size of this difference and the small number of runs, we treat it as descriptive rather than as evidence of an improvement. Both conditions optimize stably throughout training (Figure 2A).

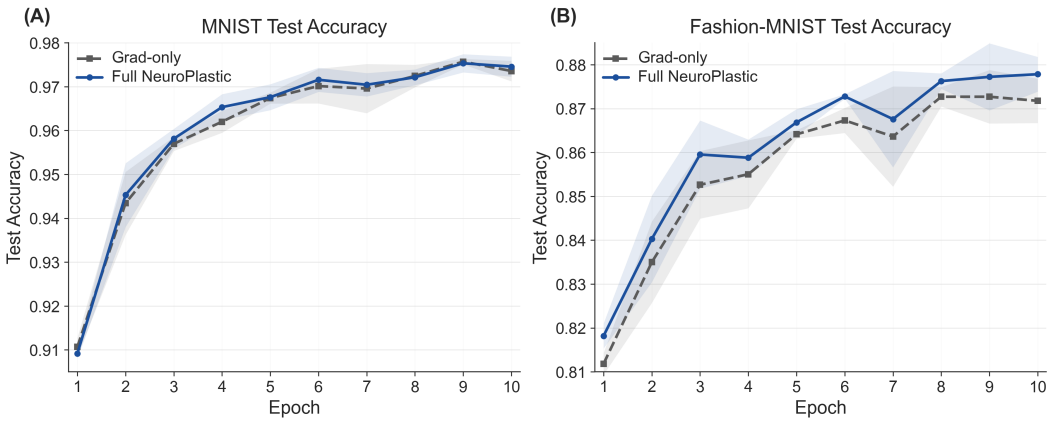


Figure 2: **Test accuracy across training epochs on MNIST and Fashion-MNIST.** (A) MNIST test accuracy for the full NeuroPlastic optimizer and the gradient-magnitude-only ablation. (B) Fashion-MNIST test accuracy under the same training configuration. Shaded regions denote the standard deviation across three independent runs.

On Fashion-MNIST, the mean final-accuracy difference is approximately $+0.006$ (0.6 percentage points) across three seeds, accompanied by lower mean training loss. This difference is larger than on MNIST, although more seeds would be needed to quantify its uncertainty reliably (Figure 2B).

To characterize how the two update rules differ on Fashion-MNIST, we examine internal optimizer dynamics during training. As detailed in Appendix F, full NeuroPlastic maintains a higher mean coefficient and follows a distinct effective-update-norm trajectory relative to the ablation, whereas the raw gradient norms of the two conditions are similar. These diagnostics are consistent with a difference in post-gradient transformation, but they do not establish that the update-norm trajectory causally explains the accuracy difference.

4.2 Low-Data Regime Analysis

We next run a controlled reduced-data study on Fashion-MNIST, training on fractions $\{0.1, 0.25, 0.5, 1.0\}$ of the training set while holding the test set fixed. Across these fractions, final-accuracy differences relative to the gradient-magnitude-only ablation range from approximately $+0.004$ to $+0.006$ (0.4–0.6 percentage points), together with lower mean training loss. The differences are larger in some reduced-data settings, but they do not vary monotonically with training-set size. These experiments manipulate training-set size only; they do not speak to robustness under label noise or gradient noise (Figure 3).

4.3 Cross-Dataset Transfer and Learning-Rate Sensitivity on CIFAR-10

We evaluate NeuroPlastic on CIFAR-10 (Krizhevsky, 2009) with ResNet-18 and compare it against SGD, Adam, and AdamW. NeuroPlastic deliberately reuses a configuration transferred from MNIST, whereas the standard optimizers use tuned learning rates. This is therefore not a tuning-parity comparison: it characterizes transfer stability and learning-rate sensitivity, not comparative peak performance.

As shown in Figure 4, the tuned standard optimizers reach higher final accuracy. NeuroPlastic converges without numerical divergence at every learning rate evaluated, but is markedly sensitive to learning-rate scaling. These results do not establish competitiveness with tuned standard optimizers; they show that the transferred formulation can remain numerically stable while incurring a performance gap.

Appendix E reports an additional transfer validation using the fixed MNIST-tuned configuration, in which full NeuroPlastic and the gradient-magnitude-only ablation perform comparably and both train stably. Appendix G then explores the design space around the proposed rule. Performance

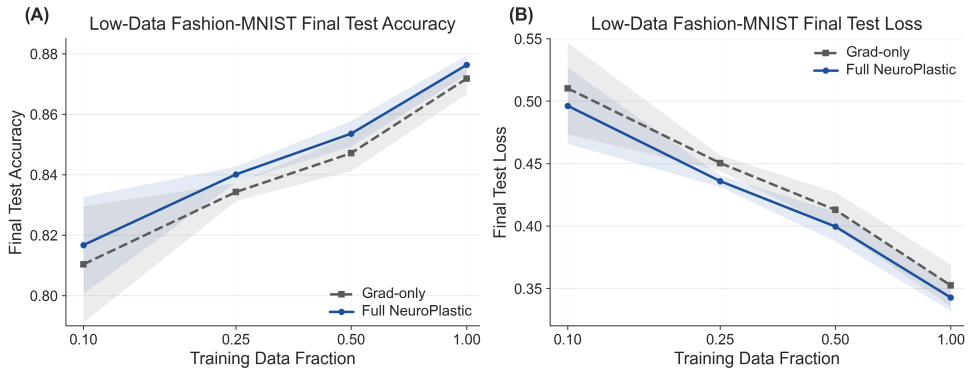


Figure 3: **Reduced-data analysis on Fashion-MNIST.** (A) Final test accuracy and (B) final test loss across training-set fractions for full NeuroPlastic and the gradient-magnitude-only ablation. Shaded regions indicate the standard deviation across three independent runs.

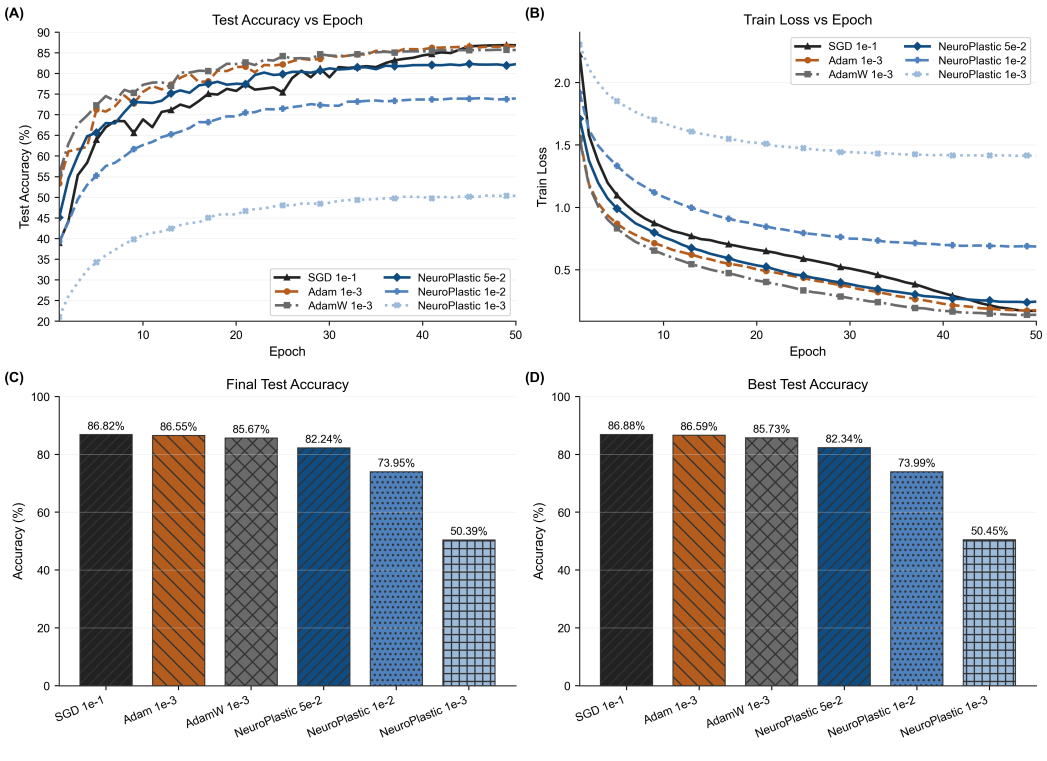


Figure 4: **Comparison of standard and plasticity-modulated optimizers on CIFAR-10 with ResNet-18.** (A) Test accuracy over training epochs. (B) Training loss over epochs. (C) Final test accuracy after 50 epochs. (D) Best test accuracy reached during training. The tuned standard optimizers achieve higher final accuracy. NeuroPlastic converges without numerical divergence at all evaluated learning rates but remains sensitive to the choice of learning rate. Because NeuroPlastic was not tuned with the same budget, this figure characterizes transfer behavior and sensitivity rather than comparative peak performance.

varies substantially across formulations, and naive variants generally underperform standard optimizers, which supports the view that the specific modulation structure matters and that plasticity is not a uniformly beneficial modification to gradient-based updates.

5 CONCLUSIONS

NeuroPlastic combines gradient statistics across three timescales to modulate parameter updates. It produced modest gains over a gradient-magnitude-only ablation on MNIST and Fashion-MNIST, but did not outperform tuned SGD, Adam, or AdamW on CIFAR-10 with ResNet-18. These findings support further investigation rather than a claim of superiority over established optimizers. The evaluation is limited by its small benchmarks, three random seeds, incomplete baseline and tuning parity, and unmeasured runtime and peak-memory costs. NeuroPlastic also requires three parameter-sized state tensors, approximately 50% more optimizer-state memory than AdamW at equal precision. Future work should test larger vision and Transformer tasks under matched tuning budgets, measure computational costs, and use component-wise ablations and controlled noise experiments to establish when its gradient statistics are useful.

AI ASSISTANCE DISCLOSURE

Large language models were used to assist with language editing and manuscript organization. All scientific claims, equations, experimental results, analyses, citations, and the final manuscript were reviewed and verified by the authors.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Kartik Chandra, Audrey Xie, Jonathan Ragan-Kelley, and Erik Meijer. Gradient descent: The ultimate optimizer. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems*, 2023.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of Machine Learning Research*, volume 12, 2011.

Runa Eschenhagen, Alexander Immer, Richard E. Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. In *Advances in Neural Information Processing Systems*, 2023.

Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9, 2016.

Wulfram Gerstner. Hebbian learning and plasticity. In *From Neuron to Cognition via Computational Neuroscience*, chapter 9. MIT Press, 2011.

Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility traces and plasticity on behavioral time scales: Experimental support of NeoHebbian three-factor learning rules. *Frontiers in Neural Circuits*, 12, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning (ICML)*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Donald O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, 1949.

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

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

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015.

Timothy P Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, 2016.

Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21, 2020.

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

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 2015.

Alexander Meulemans, Francesco S. Carzaniga, Johan A.K. Suykens, Jo ao Sacramento, and Benjamin F. Grewe. A theoretical framework for target propagation. In *Advances in Neural Information Processing Systems*, 2020.

Thomas Miconi, Kenneth O. Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O. Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *International Conference on Learning Representations*, 2019.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5), 1964.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Matthew D Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

APPENDIX

A OPTIMIZER HYPERPARAMETERS

This appendix summarizes the default optimizer and stabilization hyperparameters of the NeuroPlastic implementation. Where an experiment departs from these defaults, the value used is stated in Appendix B.

A.1 PLASTICITY CONFIGURATION

The default plasticity configuration is

$$w_a = 0.4, \qquad w_g = 0.4, \qquad w_m = 0.2, \tag{25}$$

$$s_{\text{plast}} = 1.0, \qquad \alpha_{\min} = 0.2, \qquad \alpha_{\max} = 2.0, \tag{26}$$

with numerical stability constant

$$\epsilon = 10^{-8}. \tag{27}$$

By default, the optimizer uses:

- rule-based plasticity mode,
- parameter-wise modulation,
- warmup disabled (`warmup_epochs` $= 0$),
- no learning-rate controller,
- no bounded residual branch,
- no orthogonal residual branch,
- no hybrid AdamW or SGD-momentum base.

The default implementation stores three parameter-sized state tensors: the magnitude trace a_t, the first-moment trace m_t, and the second-moment trace v_t. At equal precision this is 1.5 times the optimizer-state storage of AdamW, excluding parameters, gradients, activations, and framework overhead.

A.2 TRACE AND STATE STATISTICS

The default activity-trace decay is

$$\beta_a = 0.95, \tag{28}$$

and the default momentum and variance decay coefficients are

$$\beta_1 = 0.9, \qquad \beta_2 = 0.99. \tag{29}$$

A.3 HOMEOSTATIC STABILIZATION

The stabilization hyperparameters are

$$\tau = 1.0, \qquad r_{\text{target}} = 0.02, \qquad \rho = 0.01, \tag{30}$$

and the gain is clipped to

$$\gamma_t^{\text{stab}} \in [0.5,\ 1.5]. \tag{31}$$

The implementation also supports layer-dependent target-RMS scaling. The default scaling factors for early, middle, and late parameter groups are all set to 1.0, so the same target RMS applies to every group in the reported experiments.

A.4 Weight Decay and Learning Rate Control

The optimizer supports both coupled and decoupled weight decay, with a default weight decay of 0.0.

The optional learning-rate controller of Equations equation 23–equation 24 is disabled by default. When enabled, its default parameters are

$$\alpha_{\text{ctrl}} = 0.1, \qquad \eta_{\min}^{\text{ctrl}} = 0.8, \qquad \eta_{\max}^{\text{ctrl}} = 1.2. \tag{32}$$

B Training Hyperparameters

Within each experimental setting, the full NeuroPlastic optimizer and the gradient-magnitude-only ablation use identical model architectures and training pipelines.

Lightweight benchmarks (MNIST, Fashion-MNIST)

- Optimizer learning rate: $\eta = 10^{-3}$
- Batch size: 128
- Training epochs: 10
- Number of random seeds: 3
- Plasticity coefficient bounds: $\alpha_{\min} = 0.9$, $\alpha_{\max} = 1.1$

Note that these benchmarks deliberately use tighter coefficient bounds than the implementation defaults reported in Appendix A ($\alpha_{\min} = 0.2$, $\alpha_{\max} = 2.0$), restricting modulation to a narrow band around one.

CIFAR-10 (ResNet-18)

- Learning rates: $\{10^{-3}, 10^{-2}, 5 \times 10^{-2}\}$
- Training epochs: 50
- Batch size: 128
- Optimizers compared: SGD, Adam, AdamW, NeuroPlastic
- Number of random seeds: 3

Datasets

- MNIST
- Fashion-MNIST
- CIFAR-10

C Reproducibility

To support reproducibility, we record the following protocol details.

- Within each experimental setting, all optimizers were trained with identical architectures and training pipelines.
- Random seeds were fixed for dataset splitting, weight initialization, and training order.
- Every experiment was repeated across three independent seeds, and results are reported as the mean with standard deviation.
- The gradient-magnitude-only ablation is implemented by removing the magnitude-history and moment-history contributions while retaining instantaneous gradient-magnitude modulation, as in Equation equation 11.

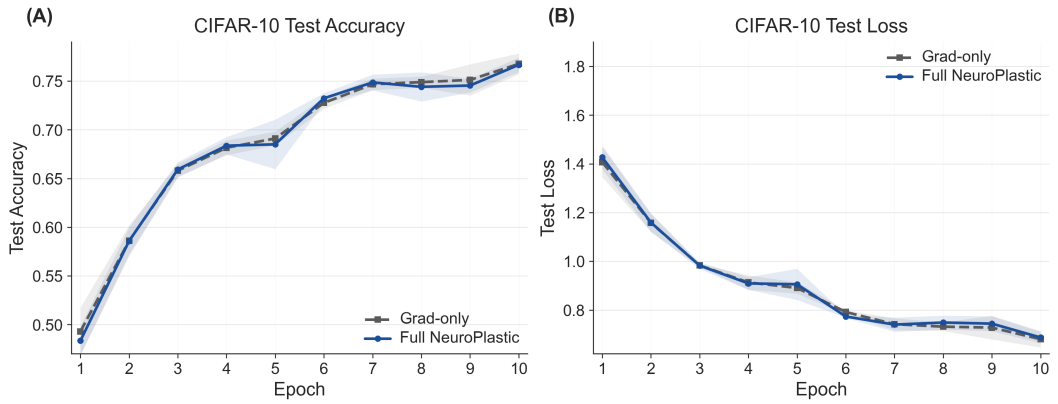


Figure 5: **Transfer validation on CIFAR-10.** (A) Test accuracy across training epochs when the MNIST-tuned optimizer configuration is applied without retuning. The gradient-magnitude-only ablation reaches slightly higher final accuracy. (B) Test loss across epochs. Both conditions converge stably, indicating that the NeuroPlastic mechanism does not destabilize training when transferred across datasets. Shaded regions denote the standard deviation across three independent runs.

- NeuroPlastic is implemented in PyTorch without modifying model architectures or introducing additional trainable parameters. It nevertheless maintains three parameter-sized optimizer state tensors.
- The plasticity coefficient α_t is computed per parameter and normalized within each parameter tensor, as defined in Section 3.
- Stabilization is applied to every update to limit instability, particularly at larger learning rates.
- Learning-rate sensitivity is evaluated explicitly, and results at multiple learning rates are reported to cover different optimization regimes.

D Compute Resources

All experiments were run on CPU or NVIDIA GPU in a workstation environment using PyTorch. Depending on the dataset and model configuration, runtime ranged from 1 to 24 hours. Exact hardware, software versions, peak memory, throughput, and optimizer-step timing will be reported with the released benchmark configuration.

E Additional Transfer Validation on CIFAR-10

We further examine transfer behavior on CIFAR-10 using the fixed hyperparameter configuration obtained by tuning on MNIST. This experiment asks whether the plasticity-modulated optimizer generalizes to a more complex dataset without retuning.

Under this setting, the gradient-magnitude-only ablation reaches slightly higher final accuracy than full NeuroPlastic, with a mean difference of approximately 0.001 (0.1 percentage points; Figure 5). Full NeuroPlastic occasionally matches or exceeds the ablation at intermediate epochs, but this advantage is not sustained at convergence.

These results indicate that the MNIST-derived configuration does not transfer into improved CIFAR-10 performance. Training converges without numerical divergence under the transferred configuration, but the experiment does not establish tuning-parity competitiveness against standard optimizers.

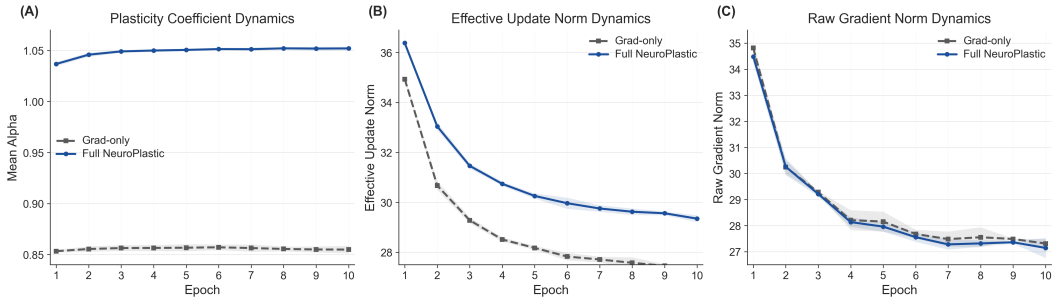


Figure 6: **Optimizer diagnostics on Fashion-MNIST.** (A) Mean plasticity coefficient over training. Full NeuroPlastic maintains a higher mean coefficient than the gradient-magnitude-only condition. (B) Effective update norm over training. Full NeuroPlastic follows a distinct trajectory, with slightly larger early effective updates and a smoother decay. (C) Raw gradient norm over training. The raw gradient scale is similar in the two conditions, indicating that the difference in optimization behavior is not driven by larger gradients but by how updates are transformed before they are applied. Solid blue lines denote full NeuroPlastic; dashed gray lines denote gradient-magnitude-only modulation. Shaded bands indicate ± 1 standard deviation over three random seeds.

F Analysis of Plasticity-Modulated Optimization Behavior

To characterize the two update rules, we analyze optimizer-state dynamics during training on Fashion-MNIST (Figure 6). NeuroPlastic maintains a higher mean coefficient than the gradient-magnitude-only ablation, and its effective update norm follows a distinct trajectory, slightly larger early updates and a smoother later decay, while the raw gradient norms of the two conditions remain similar. These observations are consistent with different post-gradient transformations, but they do not establish a causal explanation for the accuracy difference.

We also observe that removing normalization or stabilization can destabilize training, particularly at larger learning rates, where the plasticity coefficient may grow large enough to produce exploding updates. This underscores the role of bounded modulation and normalization in maintaining stable optimization.

Taken together, these diagnostics are consistent with NeuroPlastic acting as a secondary modulation layer on top of gradient-based optimization, altering effective update dynamics without substantially changing the underlying gradient scale.

G Design Space Exploration of Plasticity-Modulated Optimization

To understand how alternative formulations of plasticity-modulated updates affect optimization behavior, we evaluate a set of variants derived from the proposed framework: residual-style updates, hybrid modulation schemes, and an interpretation of plasticity as a learning-rate controller.

As shown in Figure 7, performance varies substantially across formulations. Naive variants such as residual or hybrid modulation generally underperform standard optimizers, indicating that injecting plasticity signals directly into the update does not reliably improve optimization.

By contrast, a controller-style interpretation, in which plasticity modulates the effective learning rate rather than acting as an additive update component, performs close to well-tuned SGD. This suggests that the usefulness of plasticity-modulated updates depends critically on how the modulation signal is integrated into the optimization process.

These observations indicate that plasticity is better understood as a modulation mechanism whose impact is strongly formulation-dependent than as a standalone improvement over existing optimizers. Identifying formulations that align with stable and effective optimization dynamics remains an important direction for future work.

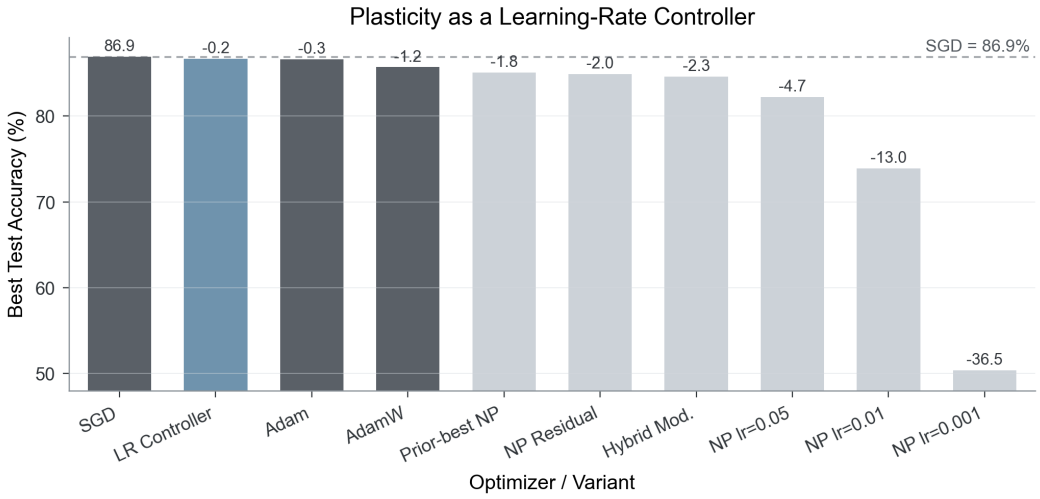


Figure 7: **Design space exploration of plasticity-modulated optimization variants on CIFAR-10 (ResNet-18).** Each bar shows the best test accuracy reached by a given optimizer or formulation, with differences relative to the SGD baseline annotated above the bars. Several naive variants underperform, whereas a controller-style formulation of plasticity comes close to SGD, suggesting that the effectiveness of plasticity modulation depends strongly on how it enters the update rule.