NATURAL GRADIENT LEARNING IN PREDICTIVE CODING THROUGH LATERAL CIRCUITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Predictive coding (PC) trains neural networks through local energy-minimization dynamics without depending on a global error signal that backpropagation (BP) requires. Although PC is a promising alternative algorithm for efficient learning, the computational cost of inference dynamics currently outweighs the resulting learning benefits in practice. In this work, we take inspiration from optimization theory to extract more learning benefit from each inference phase. We introduce *Natural PC*, which augments standard PC with a lateral circuit, enabling exact and efficient natural gradient learning locally in space and time. Our lateral circuit learns the activity correlations that determine the Fisher information matrix and implicitly computes its inverse, avoiding expensive matrix inversion. Across all tested vision benchmarks, Natural PC achieves state of the art in terms of sample efficiency and accuracy compared to existing PC approaches and even surpassing BP trained with AdamW. Most notably, Natural PC outperforms BP on ImageNet reconstruction at a comparable computational cost, while using an order of magnitude fewer training samples. For the first time, our results establish PC as a practical alternative to BP.

1 INTRODUCTION

Predictive coding (PC) is a prominent theory of cortical computation, where higher cortical areas predict lower-level sensory inputs, unifying perception and learning as the local minimization of prediction errors (Rao & Ballard, 1999; Friston, 2005; 2008; 2010; Spratling, 2010; Bastos et al., 2012; Shipp, 2016). The local computation of prediction errors in PC provides a mechanism to solve the credit assignment problem in the brain (Rumelhart et al., 1986; Lillicrap et al., 2020) and has therefore motivated the development of algorithms for training artificial neural networks. Although PC can approximate or exactly implement backpropagation (BP) under specific conditions (Whittington & Bogacz, 2017; Song et al., 2020; Salvatori et al., 2021; Rosenbaum, 2022; Millidge et al., 2023; Innocenti et al., 2026; Seely & Gould, 2026), recent work suggests that PC can also provide computational advantages over BP, including fewer weight updates (Alonso et al., 2022; 2024), improved online and continual learning (Song et al., 2024), faster escape from saddle points (Innocenti et al., 2024), and greater sample efficiency (Oliviers et al., 2026). Together, these properties make PC a promising foundation for efficient learning in neural networks.

Despite this promise, standard PC remains difficult to apply in practice, since even relatively simple problems can require substantial computational resources (Pinchetti et al., 2025). Unlike BP, which straightforwardly computes synaptic updates by differentiating a global loss function, PC treats the network as a dynamical system and requires an inference process to solve the dynamics before synaptic weights are updated (Song et al., 2024; Aceituno et al., 2025). Once inference has minimized prediction errors and the dynamics have reached equilibrium, the synaptic weights are updated to reduce the residual prediction errors at that equilibrium (Whittington & Bogacz, 2017). However, reaching equilibrium often requires hundreds or thousands of inference iterations per sample, and PC becomes practical only when the trade-off between inference costs and learning benefits is favorable.

Recent work has sought to improve this trade-off by reducing inference costs or increasing the learning benefit from inference. To reduce inference costs, some approaches learn activity estimates from the input or initialize activities from progress on earlier samples to provide inference with a more informative starting point (Song et al., 2019; Tschantz et al., 2023; Pinchetti et al., 2026). Other approaches improve the learning benefit from inference by mitigating ill-conditioned inference

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.

(Innocenti et al., 2025), balancing prediction errors across layers (Qi et al., 2025), preventing prediction errors from vanishing or exploding (Ha et al., 2026), or improving credit assignment through local constraints (Seely & Gould, 2026). Although these existing approaches improve inference and learning, their synaptic updates do not follow the local geometry of the model distribution (Amari, 1987; 1998), leaving open an information-geometric route to more efficient learning.

In this work, we introduce *Natural PC*, an extension of standard PC with a lateral interneuron circuit enabling natural gradient learning. We first derive an exact layerwise form of the natural gradient for PC, identifying the activity correlations required for geometry-aware synaptic updates. We then show that lateral weights learn these activity correlations and that the dynamics of the interneuron circuit implicitly invert the correlations during inference, removing the need for explicit and expensive matrix inversion. We theoretically show that the natural gradient of the PC learning objective coincides with Newton and Gauss–Newton updates, explaining how Natural PC can obtain more learning benefit from each costly inference phase. Across multiple datasets spanning image reconstruction and classification, Natural PC achieves state of the art in terms of sample efficiency and accuracy compared to existing PC approaches and even BP trained with AdamW. Crucially, Natural PC enables efficient scaling to ImageNet, and outperforms BP on reconstruction at a comparable computational cost while using an order of magnitude fewer training samples. Together, these results establish PC as a credible alternative to BP and suggest a direct computational role for lateral interneurons in the brain.

2 Natural Predictive Coding

We now introduce *Natural PC*, which extends PC with a lateral interneuron circuit that implements natural gradient learning locally in space and time. We first provide a background on standard PC (Sec. 2.1), and derive the natural gradient of the PC learning objective (Sec. 2.2). We then introduce a lateral circuit that implements natural gradient learning (Sec. 2.3), and show that natural gradient learning in PC coincides with Newton and Gauss–Newton optimization (Sec. 2.4). Finally, we derive an adaptive learning rate from the lateral weights (Sec. 2.5).

2.1 Predictive Coding Background

We formalize the PC network as a directed Gaussian generative model of depth L, following the standard formulation (Rao & Ballard, 1999; Friston, 2005; Bogacz, 2017; Buckley et al., 2017). Let z_l denote the activity at layer l, with z_0 denoting the observed input; let ϕ denote a smooth element-wise nonlinearity; let W_l contain the top-down weights from layer l to layer $l-1$; and let Π_l denote the precision, or inverse residual covariance, of the corresponding conditional distribution. The PC network (see Fig. 1a) factorizes as,

$$p(z_{0:L} \mid \theta) = p(z_L) \prod_{l=1}^{L} p(z_{l-1} \mid z_l; W_l, \Pi_l), \;\; p(z_{l-1} \mid z_l; W_l, \Pi_l) = \mathcal{N}\big(z_{l-1}; W_l \phi(z_l), \Pi_l^{-1}\big), \quad (1)$$

with learnable weights $\theta = \{W_l\}_{l=1}^{L}$. Intuitively, the precision Π_l specifies the weight assigned to each prediction-error component, leading to larger penalties along low-variance dimensions than along noisy dimensions. For simplicity and to isolate the learning geometry, we hold the precision parameters fixed, instead of inferring or learning these parameters (Friston, 2008; Ofner et al., 2021).

PC learns by alternating the inference of the activities while keeping the weights fixed, and the learning of the weights once the activities have settled. For a clamped input z_0, inference minimizes the precision-weighted energy $\mathcal{E}$ by gradient descent on the activities (Friston, 2005),

$$z^*_{1:L} = \arg\min_{z_{1:L}} \mathcal{E}(z_{0:L}, \theta), \qquad \mathcal{E}(z_{0:L}, \theta) = \tfrac{1}{2} \sum_{l=1}^{L} \varepsilon_l^\top \Pi_l \varepsilon_l, \qquad \varepsilon_l = z_{l-1} - W_l \phi(z_l). \quad (2)$$

where ε_l is the top-down prediction error of z_{l-1}. Once inference has reached equilibrium (ε^*, z^*), the weights are updated following the local Hebbian rule (Whittington & Bogacz, 2017; Millidge et al., 2021),

$$\Delta W_l^{\mathrm{PC}} = -\eta \, \nabla_{W_l} \mathcal{E}|_{z^*} = \boxed{\eta \, \Pi_l \varepsilon_l^* \phi(z_l^*)^\top}. \tag{3}$$

Minimizing the energy with respect to the weights of each layer with the equilibrium activities held fixed is equivalent to maximizing $\log p(z_{l-1}^* \mid z_l^*; W_l, \Pi_l)$ under the Gaussian model in Eq. 1. Intuitively, each layer treats the equilibrium activity $\phi(z_l^*)$ as a fixed input and the equilibrium activity z_{l-1}^* of the layer below as a noisy target, such that learning in each layer is a linear regression of the layer below onto the layer above, with the prediction error as the residual.

2.2 Natural Gradient Learning for Predictive Coding

We now identify a limitation of the standard PC learning rule that restricts the effectiveness of learning after inference. The learning rule in Eq. 3 follows the Euclidean gradient and does not account for differences in curvature across parameters. Because the objective can be much more curved in some directions than in others, a descent step must be small enough to avoid overshooting along the most strongly curved directions, leading to slow progress along flatter ones (Amari, 1998). The single Euclidean weight update performed after inference generally does not fully maximize the likelihood given the equilibrium activities, and completing the maximization would generally require further weight updates, thereby limiting the learning benefit obtained from each costly inference phase. We observe that the natural metric of the model would allow a single appropriately scaled natural gradient step to reproduce the exact maximization (Amari, 1995), eliminating the need for repeated weight updates (see Fig. 1b).

To derive the natural gradient for PC, we rescale the energy gradient by the inverse Fisher information matrix (Amari, 1987; 1998),

$$\widetilde{\nabla}_\theta \mathcal{E} = \mathcal{F}^*(\theta)^{-1} \nabla_\theta \mathcal{E}, \qquad \mathcal{F}^*(\theta) = \mathbb{E}\left[\nabla_\theta \log p \, \nabla_\theta \log p^\top\right], \tag{4}$$

where $\mathcal{F}^*(\theta)$ is the Fisher of the layerwise Gaussian regressions. Intuitively, the Fisher measures how strongly parameter changes alter the model distribution in each direction, and the inverse Fisher therefore rescales update magnitudes to give smaller steps along sensitive directions and larger steps along insensitive directions, addressing the curvature imbalance that limits learning in the Euclidean metric.

To compute the Fisher, we hold z_l^* fixed, average over $z_{l-1} \sim p(z_{l-1} \mid z_l^*; W_l, \Pi_l)$, and then average over observed inputs. We obtain (see App. A),

$$\mathcal{F}_l^* = G_l^* \otimes \Pi_l, \qquad G_l^* \triangleq \mathbb{E}_{z_0 \sim p_{\mathrm{data}}}\left[\phi(z_l^*)\phi(z_l^*)^\top\right], \tag{5}$$

where $\otimes$ denotes the Kronecker product and G_l^* represents the second moment of the equilibrium presynaptic activities, which we refer to as the Gram matrix. The learning objective thus factorizes across layers, making the Fisher block diagonal with one matrix $\mathcal{F}_l^*$ for each layer. Substituting into the learning rule of Eq. 3 gives,

$$\Delta W_l^{\mathrm{NG}} = -\eta \, \Pi_l^{-1} \left(-\Pi_l \varepsilon_l^* \phi(z_l^*)^\top\right) G_l^{*-1} = \boxed{\eta \, \varepsilon_l^* \phi(z_l^*)^\top G_l^{*-1}}, \tag{6}$$

where the precision factor from the learning rule cancels with the corresponding factor of the inverse Fisher. Intuitively, the natural geometry rescales the standard PC update according to correlations among presynaptic activities, allowing larger steps along directions where Euclidean learning progresses slowly without overshooting along more strongly curved directions. The resulting weight update can therefore make more progress toward the likelihood maximum after each inference phase without requiring additional inference iterations. We now show how a lateral circuit learns the Gram matrix and applies the inverse through local dynamics.

2.3 Lateral Circuits Implement Natural Gradient Learning

In order to implement natural gradient learning, we need to estimate the presynaptic correlations across training samples to construct the requisite Gram matrix, and apply its inverse to the presynaptic

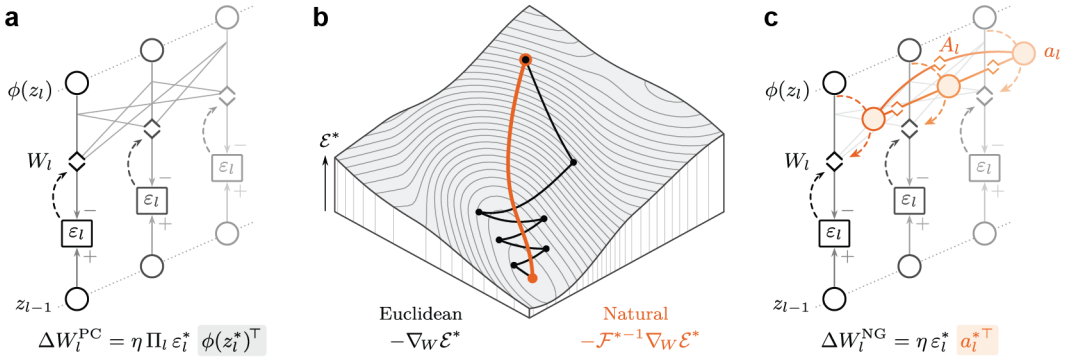


Figure 1: Natural Predictive Coding. **(a)** Standard PC. Layer l sends $\phi(z_l)$ through W_l (diamonds) to predict the layer below, and error neurons ε_l compare the prediction with the activity. Dashed arrows carry plasticity signals. **(b)** Energy $\mathcal{E}$ over a slice of weight space at equilibrium activity. Euclidean descent (black) takes many steps across the valley, while the natural gradient (orange) reaches the minimum in one step. **(c)** Natural PC adds interneurons with activities a_l that receive $\phi(z_l)$ and interact through lateral weights A_l to obtain the natural geometry.

activities. We introduce a lateral circuit consisting of a population of interneurons with activities a_l and lateral weights A_l for every layer, governed by,

$$\tau_a \dot{a}_l = \phi(z_l) - A_l a_l, \qquad \tau_A \dot{A}_l = \phi(z_l^*)\phi(z_l^*)^\top - A_l, \tag{7}$$

where the interneurons receive the activities from the hidden neurons, and the lateral weights connect the interneurons within each layer (Fig. 1c). The interneuron dynamics evolve inside the inference phase, while the lateral weights update at equilibrium on the slower timescale of synaptic learning together with the feedforward weights.

The lateral weights accumulate presynaptic correlations across samples over time (Földiák, 1990; Pehlevan & Chklovskii, 2015; Pehlevan et al., 2015), and thereby estimate the Gram matrix as an exponential moving average. The interneuron dynamics apply the inverse of the learned matrix through a dynamical inversion (Podlaski & Machens, 2020). For stationary activity statistics and symmetric positive-definite A_l, the lateral dynamics and the interneuron equilibrium satisfy (see App. B),

$$A_l \longrightarrow G_l^*, \qquad a_l^* = A_l^{-1}\phi(z_l^*). \tag{8}$$

The local synaptic rule of Natural PC uses the interneuron activities as the presynaptic factor when inference reaches the new equilibrium $(\varepsilon^*, z^*, a^*)$,

$$\Delta W_l^{\mathrm{NPC}} = \eta\, \varepsilon_l^* a_l^{*\top} = \boxed{\eta\, \varepsilon_l^* \phi(z_l^*)^\top A_l^{-1}}. \tag{9}$$

We thereby recover the natural gradient learning rule of Eq. 6 exactly when $A_l = G_l^*$. The lateral circuit thus implements natural gradient learning locally in space and time, where the approximation during training is determined by how closely A_l tracks G_l^*, as we empirically verify in Sec. 3.

2.4 Natural PC Implements Newton Optimization

We now ask which optimization principle Natural PC implements. Unlike first-order gradient descent using gradients computed by BP, curvature-based second-order optimization accounts for how quickly the gradient changes along different weight directions, allowing larger steps toward a minimum along directions where gradient descent progresses slowly. To recap, rescaling the gradient by the inverse Hessian, or by the inverse of a curvature approximation obtained by linearizing the predictions, is called Newton or Gauss–Newton optimization, respectively (Nocedal & Wright, 2006).

As we have seen, weight learning in PC minimizes the Gaussian negative log-likelihood, where the learning objective and its curvature described by the Hessian $\mathcal{H}$ are,

$$\bar{\mathcal{E}}_l(W_l) = -\mathbb{E}\big[\log p(z^*_{l-1} \mid z^*_l; W_l, \Pi_l)\big], \qquad \mathcal{H}^*_l = \nabla^2 \bar{\mathcal{E}}_l = \mathbb{E}\big[J^{*\top}_l \Pi_l J^*_l\big], \tag{10}$$

where $J^*_l = \phi(z^*_l)^\top \otimes I$ is the Jacobian of the layer predictions with respect to the weights. Since the predictions are linear in the weights, differentiating the objective twice gives exactly the Jacobian product defining the Gauss–Newton matrix $\mathcal{G}$. Additionally, for the Gaussian likelihood with fixed precision, the Fisher equals the same Jacobian product (Martens, 2020). Consequently, we obtain the following unification (see App. C),

$$\mathcal{H}^*_l = \mathcal{F}^*_l = \mathcal{G}^*_l = G^*_l \otimes \Pi_l. \tag{11}$$

Natural PC therefore implements Newton and Gauss–Newton updates in addition to natural gradient learning.

2.5 Lateral Weights Determine the Learning Rate

The equality between the Fisher and the Hessian explains why an exact natural gradient step can complete likelihood maximization after inference. Since the predictions are linear in the weights, the Gaussian objective is quadratic, and a unit Newton step reaches the likelihood maximum (App. D), allowing Natural PC to maximize the learning benefit obtained from each costly inference phase.

The connection to Newton optimization also allows us to determine the learning rate analytically from the lateral weights and the activity statistics. After preconditioning by the inverse of A, the eigenvalues of $A_l^{-1} G^*_l$ describe the curvature governing progress along each weight direction. We choose the largest step that approaches the likelihood maximum without overshooting in any direction,

$$\eta_l = \frac{1}{\lambda_{\max}\big(A_l^{-1} G^*_l\big)}, \tag{12}$$

where $\lambda_{\max}(A_l^{-1} G^*_l)$ is the Lipschitz constant of the gradient in the geometry defined by the lateral weights and measures the strongest remaining curvature. We can prove that this learning rate guarantees descent and reaches the optimum along the direction with the largest curvature for the fixed-activity quadratic objective (App. E). Intuitively, the step size reflects how closely the learned geometry matches the natural gradient geometry, approaching a unit Newton step as the lateral weights recover the Gram matrix. Setting $A_l = I$ gives the corresponding curvature-based rate for standard PC. As $A_l \longrightarrow G^*_l$, the curvature becomes uniform and a unit update maximizes the likelihood across all directions. The lateral weights therefore determine both the geometry of the synaptic update and the learning rate.

3 Empirical Validation of the Learning Theory

We now empirically show how the lateral circuit of Natural PC aligns with the predictions of the learning theory by answering three questions. Do the lateral weights recover the natural gradient during training (Sec. 3.1)? Do the lateral weights align with the full data distribution required for the natural gradient (Sec. 3.2)? Does the lateral circuit support unit natural gradient steps (Sec. 3.3)?

To answer these questions, we train a three-layer generative model on the 50,000 training images of CIFAR-10, with 128 hidden neurons per layer, $\phi = \tanh$, isotropic precisions $\Pi_l = \sigma^{-2} I$ with $\sigma^2 = 0.1$, and a unit Gaussian prior on the top layer. Mini-batches contain $B = 200$ images, giving 250 weight updates per epoch. The lateral weights are initialized at $A_l = I$ and integrate the activities with $\tau_A = 20$ (see Sec. 3.2). Further experimental details are provided in App. F.

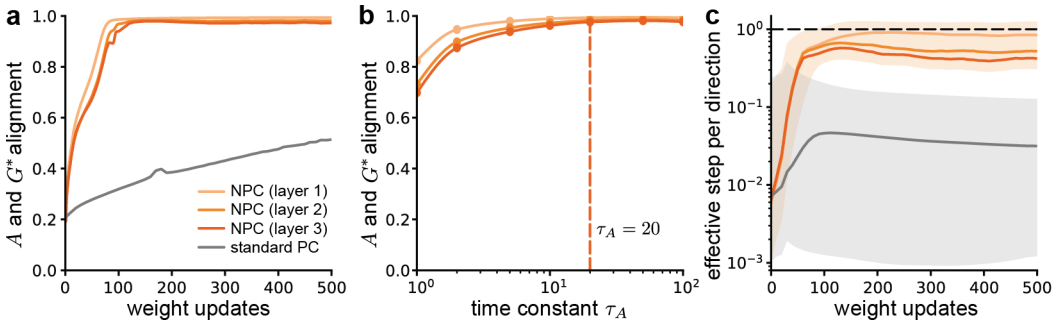


Figure 2: The lateral circuit learns the geometry of natural gradient updates. A three-layer generative model is trained on CIFAR-10. **(a)** Alignment of the lateral weights A_l with the Gram matrix G_l^* over training. **(b)** Alignment after training as a function of the lateral time constant τ_A. The dashed line marks $\tau_A = 20$. **(c)** Effective step size over training. Median and 5th to 95th percentile across directions and layers. The black line indicates the unit step.

3.1 The Lateral Weights Recover Natural Gradient Directions

We first test whether the learned lateral weights produce the directions required by the natural gradient. We measure alignment using an activity-weighted cosine similarity between the directions produced by A_l^{-1} and G_l^{*-1},

$$\text{align}(A_l, G_l^*) = \frac{\langle A_l^{-1}, G_l^{*-1} \rangle}{\|A_l^{-1}\| \, \|G_l^{*-1}\|}, \tag{13}$$

where the inner product and norms are weighted by G_l^*, measuring directional agreement independently of step size. We observe that alignment starts near 0.2 at random network initialization and becomes nearly perfect across layers within 100 weight updates, reaching 0.94–0.99 and increasing to 0.97–0.99 after 500 updates (Fig. 2a). Standard PC without the lateral circuit never reaches comparable alignment and remains below 0.6 averaged across layers. The lateral weights therefore recover the natural gradient directions almost perfectly during training.

3.2 The Lateral Weights Align with the Full Data Distribution

The natural gradient depends on activity correlations across the full data distribution p_{data}, as shown in Eq. 5. However, each lateral weight update observes only a single batch, so the time constant τ_A controls an exponential moving average across batches to estimate the full-data activity statistics. We therefore vary τ_A and measure the alignment of A_l and G_l^* after training. We observe that at $\tau_A = 1$, where the lateral weights track only the current batch, alignment reaches approximately 0.70–0.82 across layers. Increasing to $\tau_A = 20$ raises alignment to approximately 0.98–0.99, with similar values up to $\tau_A = 100$ (Fig. 2b). Smaller time constants adapt more quickly as the feedforward weights learn, whereas larger time constants average over more batches reduce fluctuations in the estimated activity correlations. The lateral weights can thus recover the activity statistics of the full data distribution required for the natural gradient by integrating local correlations across batches.

3.3 The Lateral Circuit Enables Near-Unit Learning Steps

We finally test whether the lateral circuit approaches unit natural gradient steps during training. We estimate the adaptive learning rate from Eq. 12 using the current batch Gram $\hat{G}_l$, computed from the same activity correlations that update the lateral weights. We compute the effective step size for every direction i,

$$s_{l,i} = \eta_l \lambda_i \left(A_l^{-1} \hat{G}_l^* \right), \tag{14}$$

where $s_{l,i} = 1$ denotes an exact unit step with λ_i being the i-th eigenvalue as a function of $A_l^{-1}G_l^*$. We plot the median and the 5th to 95th percentile across directions of the effective step size for Natural PC and standard PC in Fig. 2c, normalized for the batch estimate of the curvature (App. G). Starting near 0.01, the layer medians of Natural PC reach 0.4 to 0.9 and the spread across directions narrows to a factor of five within 100 updates, while standard PC retains a broad spread, indicative of the uncorrected strongly curved and flat directions from the Euclidean metric. The lateral circuit therefore approaches unit learning steps in natural gradient learning.

4 BENCHMARKS

We now test whether the natural gradient of the lateral circuit translates into faster and better learning. We first compare Natural PC against existing PC methods and BP on five image reconstruction benchmarks and on supervised classification (Sec. 4.1), and then scale the model to ImageNet-1k, where we compare against a BP-trained autoencoder (Sec. 4.2).

4.1 NATURAL PC ACHIEVES FAST AND SAMPLE EFFICIENT LEARNING

We now test Natural PC on five computer vision benchmarks. We follow Pinchetti et al. (2025) for benchmarking PC networks and compare Standard PC and incremental-PC (iPC) which updates weights after each inference step instead of letting the activity settle before updating (Salvatori et al., 2024).

All methods train the generative model of Eq. 1 with three hidden layers of 128 units, $\phi = \tanh$ and $\Pi_l = 10\,I$, and share the same $1{,}000$ inference steps for both training and evaluation. We compare across five datasets comprising MNIST, FMNIST, CIFAR-10, STL-10, and Tiny ImageNet, using mini-batches of 200 samples. We report the held-out reconstruction error from 512 samples together with the epochs and weight updates required to reach within 2% of each run's own plateau. We train every method to its own plateau because the methods converge at different errors, as a shared target error would not be reached by all methods equally, and a shared update budget would compare methods at different stages of training. Every learning rate is selected for the fastest plateau within 0.5% of their best error, and the selected setting is ran over five seeds with identical initialization and batch order across methods. BP similarly performs 1000 inference steps and then updates the weights using AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). App. H further details all experimental settings.

Table 1: Reconstruction benchmarks.

Metric	Method	MNIST	FMNIST	CIFAR	STL-10	Tiny ImageNet
MSE ($\times 10^{-3}$)	Backprop (AdamW)	$4.42^{\pm 0.01}$	$6.25^{\pm 0.01}$	$5.40^{\pm 0.01}$	$27.68^{\pm 1.84}$	$15.75^{\pm 0.10}$
	Standard PC	$4.52^{\pm 0.01}$	$6.37^{\pm 0.01}$	$5.51^{\pm 0.01}$	$13.41^{\pm 0.24}$	$15.26^{\pm 0.03}$
	iPC	$4.58^{\pm 0.01}$	$6.57^{\pm 0.01}$	$6.35^{\pm 0.02}$	$14.90^{\pm 0.04}$	$15.49^{\pm 0.01}$
	Natural PC	$\mathbf{4.39^{\pm 0.01}}$	$\mathbf{6.21^{\pm 0.01}}$	$\mathbf{5.34^{\pm 0.01}}$	$\mathbf{13.13^{\pm 0.03}}$	$\mathbf{15.17^{\pm 0.02}}$
Epochs	Backprop (AdamW)	$10.6^{\pm 0.2}$	$14.0^{\pm 0.6}$	$10.9^{\pm 0.1}$	$3.93^{\pm 3.5}$	$4.69^{\pm 0.5}$
	Standard PC	$15.0^{\pm 0.8}$	$16.6^{\pm 2.5}$	$19.4^{\pm 2.1}$	$2.1^{\pm 0.8}$	$3.9^{\pm 0.2}$
	iPC	$12.3^{\pm 0.3}$	$8.1^{\pm 0.1}$	$26.8^{\pm 5.9}$	$\mathbf{0.32^{\pm 0.0}}$	$2.9^{\pm 0.2}$
	Natural PC	$\mathbf{2.6^{\pm 0.1}}$	$\mathbf{2.8^{\pm 0.0}}$	$\mathbf{3.8^{\pm 0.0}}$	$0.43^{\pm 0.1}$	$\mathbf{0.29^{\pm 0.0}}$
W-updates	Backprop (AdamW)	$531^{\pm 8}$	$700^{\pm 28}$	$547^{\pm 4}$	$1{,}965^{\pm 1.8k}$	$2{,}344^{\pm 236}$
	Standard PC	$748^{\pm 40}$	$831^{\pm 126}$	$972^{\pm 104}$	$1{,}024^{\pm 403}$	$1{,}924^{\pm 117}$
	iPC	$12.3k^{\pm 344}$	$8{,}060^{\pm 120}$	$26.8k^{\pm 5.9k}$	$3{,}248^{\pm 396}$	$28.8k^{\pm 1.6k}$
	Natural PC	$\mathbf{129^{\pm 6}}$	$\mathbf{139^{\pm 0}}$	$\mathbf{189^{\pm 1}}$	$\mathbf{214^{\pm 27}}$	$\mathbf{143^{\pm 2}}$

Our results are summarized in Tab. 1 with mean $\pm$ standard deviation over five random seeds. Natural PC reaches the lowest test error on every dataset requiring the fewest amount of weight updates, needing $3\times$ to $18\times$ fewer updates than BP and $3\times$ to $16\times$ fewer than standard PC. Strikingly, on STL-10 and Tiny ImageNet, Natural PC converges after 0.43 and 0.29 epochs, meaning before the network has seen every training image. We also tested network-depth scaling in reconstruction, where Natural PC reaches the lowest error compared to existing PC methods at almost every depth and

width, and is the only PC method that trains under ReLU at depth 32 and beyond that we tested (App. I).

Additionally, we tested the discriminative setting of PC in supervised classification (Whittington & Bogacz, 2017; Pinchetti et al., 2025). We observe that Natural PC gains 3% to 19% accuracy over standard PC and reaches the reference accuracy in $12\times$ to $20\times$ fewer epochs. Moreover, Natural PC also matches or exceeds BP's accuracy on all three tested datasets by 0.48% to 2.88% (App. J, Tab. 6), while requiring up to $5\times$ less weight updates. Furthermore, Natural PC's classification accuracy also scales with depth. At 32 and 64 layers, Natural PC reaches 78% and 51% on MNIST, whereas AdamW diverges and standard PC falls to 35% and 25%, respectively (App. J, Tab. 7).

4.2 Natural PC results scale to ImageNet

We now test Natural PC on ImageNet-1k at 224×224 resolution with a convolutional generative model. We implemented top-down VGG-10 decoder of 32.7M parameters that maps a $1{,}024$-dimensional code through nine 3×3 convolutional layers with tanh to the image. The image is clamped the latents and all hidden layers are inferred with 300 energy steps, and every layer carries lateral weights (2M units total, App. K) summing to 34.7M parameters. As the BP reference, we train an autoencoder end-to-end where both the encoder and decoder are a normal and flipped VGG-10 (Simonyan & Zisserman, 2015) respectively, numbering 65.5M parameters in total, employing AdamW with a warm-up and cosine schedule as well as gradient clipping, following Pinchetti et al. (2025); Oliviers et al. (2026). Both models train for 10 epochs at batch size 800, and we report the reconstruction error of $5{,}000$ validation images decoded from the inferred or encoded latents. Further specifications of the architecture and the full training curves, FLOPs, and GPU-hours are reported in App. L.

Table 2: ImageNet reconstruction.

		Converged	**To reach converged BP**		
Method	**Params**	**MSE** ($\times10^{-3}$)	**W-updates**	**Epochs**	**Wall-clock hours**
Backprop	65.5M	8.62	16,010	10	0.73 (44 min)
Natural PC	**34.7M**	**7.95**	**1,800**	**1.125**	**1.44** (86 min)

Strikingly, Natural PC reaches the final error of the autoencoder after 1.125 epochs and $1{,}800$ weight updates, $9\times$ fewer than the $16{,}010$ updates of the autoencoder in comparable wall-clock time despite the 300 inference steps per update. Furthermore, Natural PC ends at 7.95×10^{-3} against 8.62×10^{-3} while only requiring half of the parameters.

PC networks at this scale have so far been reported as impractical (Pinchetti et al., 2025). Kerjan et al. (2026) were the first, and as far as we know only other, to train a PC network on ImageNet using a VGG-10 to near-BP accuracy, however training took 18 days on an A100 GPU against 36 hours for BP. Corrected for hardware, the gap between Natural PC and BP is 6 times smaller, as Natural PC reaches the error of its BP baseline in twice the training time on the same GPUs, where the network of Kerjan et al. (2026) needed 12 times the time of its baseline. We emphasize the BP implementation is GPU-optimized and there is room for optimization for PC networks on modern GPUs.

5 Discussion

Standard predictive coding wastes expensive inference by taking a single Euclidean step over the energy landscape, causing networks to require many inference phases before weights fit the equilibrium activities. Natural PC replaces the Euclidean learning step with a natural gradient step by adding a lateral circuit of interneurons that encode and implicitly invert the presynaptic correlations that define the natural gradient. We thus improve the trade-off between inference and learning and show that biologically inspired local computation can compete with backpropagation in both accuracy and computational cost. Our results establish PC as a practical alternative to BP and our framework makes training PC networks viable at scale.

Although Natural PC enables training at scale in generative settings, scaling discriminative learning remains a challenge. We observe that in the generative setting every layer predicts the high-dimensional activity of the layer below and thereby receives a dense prediction error, while in the discriminative setting, the target is a one-hot label which carries sparse error information to deeper layers after relaxation. Here, the top layer can sufficiently explain the target, thereby blocking the credit propagation to subsequent layers, which we coin the *target absorption problem*. While our initial discriminative results on small scale are promising, when we test deeper networks of 32 and 64 layers accuracy falls significantly (App. J, Tab. 7), aligning with previous observations (Pinchetti et al., 2025; Innocenti et al., 2025). Additionally, Natural PC on the discriminative VGG-10 does not converge on ImageNet (App. L) without additional mechanisms (Kerjan et al., 2026). To support discriminative PC at scale, we suggest combining the lateral circuit with mechanisms that improve prediction error propagation across layers (Qi et al., 2025; Ha et al., 2026; Seely & Gould, 2026; Kerjan et al., 2026). A second direction is to learn the precisions, as precision weighting has been related to natural gradient descent during inference (Millidge et al., 2021; Ofner et al., 2021).

Moreover, our results suggest a computational role for cortical interneurons in learning and motivate predictions about interneuron function, connectivity, and population size. Lateral inhibition is often associated with decorrelation or whitening (Wiechert et al., 2010; King et al., 2013; Chettih & Harvey, 2019; Znamenskiy et al., 2024). However, we find that whitening equilibrium activities improves standard PC, however guiding synaptic learning without altering the representation as implemented by Natural PC performs best (App. M). Considering a cortical circuit that similarly regulates synaptic learning, leads us to two experimental predictions. First, disrupting interneuron activity should impair learning efficiency without abolishing learning. We demonstrate this effect in our model by silencing the lateral circuit, which restores the standard PC update, and show that reactivating the circuit restores alignment with the natural gradient (App. N). Second, lateral connectivity should reflect neuronal co-activity even between neurons with dissimilar feature preferences, because the lateral learning rule tracks joint activity without requiring shared feature preferences. Consistent with this distinction, lateral weights show little correlation with feature similarity in our models ($r \leq 0.16$ across datasets), and paired recordings and connectivity mapping could test whether co-activity predicts lateral connectivity independently of feature similarity. Finally, cortical inhibitory interneurons are substantially less numerous than principal neurons (Beaulieu, 1993; Sahara et al., 2012), raising the question of whether natural gradient learning requires equally sized neuronal populations. Our analysis provides an indication that smaller populations may suffice, as the lateral weights reach an effective rank of only 17 in layers of width 64 (App. N), yet whether a smaller interneuron circuit preserves the learning benefits of Natural PC remains to be tested.

In conclusion, Natural PC improves the trade-off between inference cost and learning benefit through a lateral circuit that implements natural gradient learning. By enabling PC networks to compete with backpropagation in accuracy and computational cost, Natural PC establishes a foundation for efficient learning through local computation. These results open opportunities to investigate the proposed benefits of PC for online, continual, and sample-efficient learning, as well as to extend PC to larger vision, sequence, and language models.

6 AI USAGE DISCLOSURE

Generative AI tools were used in the preparation of this work for text editing and formatting, as well as for assistance with core implementation. In particular, AI tools were used to help implement standard components such as data loading and preprocessing utilities, plotting and visualization code, and implementations or adaptations of standard methods. AI tools were also used to check author-written derivations for errors and to improve the clarity and notation of mathematical statements and research hypotheses.

Generative AI tools were not used to interpret the experimental results, or formulate the scientific claims of this work. They were also not used to generate synthetic data or to produce proofs or mathematical claims.

All AI-assisted text and code were reviewed by the authors. AI-assisted code was inspected, tested, and validated as part of the same experimental pipeline as code written without AI assistance. The authors take full responsibility for the final content of the paper, including all text, claims, results, and software produced with the aid of generative AI.

REFERENCES

Pau Vilimelis Aceituno, Sander de Haan, Reinhard Loidl, Lhea Beumer, and Benjamin F. Grewe. Challenging backpropagation: Evidence for target learning in the neocortex. *bioRxiv*, 2025.

Nick Alonso, Beren Millidge, Jeffrey Krichmar, and Emre Neftci. A theoretical framework for inference learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 37335–37348, 2022.

Nick Alonso, Jeffrey Krichmar, and Emre Neftci. Understanding and improving optimization in predictive coding networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10812–10820, 2024.

Shun-ichi Amari. Differential geometrical theory of statistics. In *Differential Geometry in Statistical Inference*, volume 10 of *IMS Lecture Notes–Monograph Series*, pp. 19–94. Institute of Mathematical Statistics, Hayward, CA, 1987.

Shun-ichi Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Andre M. Bastos, W. Martin Usrey, Rick A. Adams, George R. Mangun, Pascal Fries, and Karl J. Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.

Clermont Beaulieu. Numerical data on neocortical neurons in adult rat, with special reference to the GABA population. *Brain Research*, 609(1–2):284–292, 1993.

Rafal Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *Journal of Mathematical Psychology*, 76:198–211, 2017.

Christopher L. Buckley, Chang Sub Kim, Simon McGregor, and Anil K. Seth. The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81:55–79, 2017.

S. N. Chettih and C. D. Harvey. Single-neuron perturbations reveal feature-specific competition in V1. *Nature*, 567(7748):334–340, 2019.

Peter Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64(2):165–170, 1990.

Karl Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1456):815–836, 2005.

Karl Friston. Hierarchical models in the brain. *PLoS Computational Biology*, 4(11):e1000211, 2008.

Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2): 127–138, 2010.

Roger Grosse and James Martens. A kronecker-factored approximate Fisher matrix for convolution layers. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48, pp. 573–582. PMLR, 2016.

Myoung Hoon Ha, Hyunjun Kim, Yoondo Sung, Youngha Jo, Min S. Kang, and Sang Wan Lee. Stable and scalable deep predictive coding networks with meta-prediction errors. In *International Conference on Learning Representations*, 2026.

Francesco Innocenti, El Mehdi Achour, Ryan Singh, and Christopher L. Buckley. Only strict saddles in the energy landscape of predictive coding networks? In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Francesco Innocenti, El Mehdi Achour, and Christopher L. Buckley. μPC: Scaling predictive coding to 100+ layer networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Francesco Innocenti, El Mehdi Achour, and Rafal Bogacz. On the infinite width and depth limits of predictive coding networks. In *International Conference on Machine Learning*, 2026.

Tugdual Kerjan, Rasmus Høier, and Benjamin Scellier. Training a predictive coding network on ImageNet using equilibrium propagation. *arXiv*, 2026.

P. D. King, J. Zylberberg, and M. R. DeWeese. Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of V1. *The Journal of Neuroscience*, 33(13): 5475–5485, 2013.

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

Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

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

Vladimir A. Marchenko and Leonid A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457–483, 1967.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

Beren Millidge, Anil Seth, and Christopher L. Buckley. Predictive coding: a theoretical and experimental review. *arXiv*, 2021.

Beren Millidge, Yuhang Song, Tommaso Salvatori, Thomas Lukasiewicz, and Rafal Bogacz. Backpropagation at the infinitesimal inference limit of energy-based models: Unifying predictive coding, equilibrium propagation, and contrastive Hebbian learning. In *International Conference on Learning Representations*, 2023.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2nd edition, 2006.

Andre Ofner, Raihan Kabir Ratul, Suhita Ghosh, and Sebastian Stober. Predictive coding, precision and natural gradients. *arXiv*, 2021.

Gaspard Oliviers, Elene Lominadze, and Rafal Bogacz. Understanding sample efficiency in predictive coding. *arXiv*, 2026.

Cengiz Pehlevan and Dmitri B. Chklovskii. Optimization theory of Hebbian/anti-Hebbian networks for PCA and whitening. In *53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1458–1465. IEEE, 2015.

Cengiz Pehlevan, Tao Hu, and Dmitri B. Chklovskii. A Hebbian/anti-Hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *Neural Computation*, 27(7):1461–1495, 2015.

Luca Pinchetti, Chang Qi, Oleh Lokshyn, Cornelius Emde, Amine M'Charrak, Mufeng Tang, Simon Frieder, Bayar Menzat, Gaspard Oliviers, Rafal Bogacz, Thomas Lukasiewicz, and Tommaso Salvatori. Benchmarking predictive coding networks – made simple. In *International Conference on Learning Representations*, 2025.

Luca Pinchetti, Simon Frieder, Thomas Lukasiewicz, and Tommaso Salvatori. Faster predictive coding networks via better initialization. *arXiv*, 2026.

William F. Podlaski and Christian K. Machens. Biological credit assignment through dynamic inversion of feedforward networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 10065–10076, 2020.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Chang Qi, Matteo Forasassi, Thomas Lukasiewicz, and Tommaso Salvatori. Towards the training of deeper predictive coding neural networks. *arXiv*, 2025.

Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

Robert Rosenbaum. On the relationship between predictive coding and backpropagation. *PLOS ONE*, 17(3):e0266102, 2022.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Setsuko Sahara, Yuchio Yanagawa, Dennis D. M. O'Leary, and Charles F. Stevens. The fraction of cortical GABAergic neurons is constant from near the start of cortical neurogenesis to adulthood. *The Journal of Neuroscience*, 32(14):4755–4761, 2012.

Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, Rafal Bogacz, and Zhenghua Xu. Predictive coding can do exact backpropagation on convolutional and recurrent neural networks. *arXiv*, 2021.

Tommaso Salvatori, Yuhang Song, Yordan Yordanov, Beren Millidge, Lei Sha, Cornelius Emde, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. A stable, fast, and fully automatic learning algorithm for predictive coding networks. In *International Conference on Learning Representations*, 2024.

Jeffrey Seely and Julian Gould. Augmented Lagrangian predictive coding. *arXiv*, 2026.

Stewart Shipp. Neural elements for predictive coding. *Frontiers in Psychology*, 7:1792, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation? exact implementation of backpropagation in predictive coding networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22566–22579, 2020.

Yuhang Song, Beren Millidge, Tommaso Salvatori, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nature Neuroscience*, 27(2):348–358, 2024.

Zengjie Song, Jiangshe Zhang, Guang Shi, and Junmin Liu. Fast inference predictive coding: A novel model for constructing deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4):1150–1165, 2019.

Michael W. Spratling. Predictive coding as a model of response properties in cortical area V1. *The Journal of Neuroscience*, 30(9):3531–3543, 2010.

Alexander Tschantz, Beren Millidge, Anil K. Seth, and Christopher L. Buckley. Hybrid predictive coding: Inferring, fast and slow. *PLOS Computational Biology*, 19(8):e1011280, 2023.

James C. R. Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*, 29(5): 1229–1262, 2017.

M. T. Wiechert, B. Judkewitz, H. Riecke, and R. W. Friedrich. Mechanisms of pattern decorrelation by recurrent neuronal circuits. *Nature Neuroscience*, 13(8):1003–1010, 2010.

Petr Znamenskiy, Mean-Hwan Kim, Dylan R. Muir, M. Florencia Iacaruso, Sonja B. Hofer, and Thomas D. Mrsic-Flogel. Functional specificity of recurrent inhibition in visual cortex. *Neuron*, 112(6):991–1000.e8, 2024.

A Layerwise Fisher Derivation

For the Gaussian conditional distribution in Eq. 1, the score of the vectorized layer weights is

$$\nabla_{\mathrm{vec}(W_l)} \log p(z_{l-1} \mid z_l; W_l, \Pi_l) = \phi(z_l) \otimes \Pi_l \varepsilon_l. \tag{15}$$

The conditional residual covariance is $\mathbb{E}[\varepsilon_l \varepsilon_l^\top \mid z_l] = \Pi_l^{-1}$. Taking the outer product of Eq. 15 and averaging over the model distribution gives

$$\begin{aligned} \mathcal{F}_l &= \mathbb{E}\big[\big(\phi(z_l)\phi(z_l)^\top\big) \otimes \big(\Pi_l \varepsilon_l \varepsilon_l^\top \Pi_l\big)\big] && (16)\\ &= \mathbb{E}\big[\phi(z_l)\phi(z_l)^\top\big] \otimes \Pi_l = G_l \otimes \Pi_l. && (17) \end{aligned}$$

The factorization is exact because the Gaussian residual covariance cancels one factor of Π_l.

B Lateral Circuit Equilibrium

The lateral learning rule in Eq. 7 averages outer products of settled presynaptic activities. The equilibrium Gram matrix is

$$G_l^* \triangleq \mathbb{E}_{z_0 \sim p_{\mathrm{data}}}\big[\phi(z_l^*)\phi(z_l^*)^\top\big]. \tag{18}$$

Taking the expectation over clamped data inputs therefore yields

$$\begin{aligned} \tau_A \, \mathbb{E}[\dot{A}_l] &= \mathbb{E}_{z_0 \sim p_{\mathrm{data}}}\big[\phi(z_l^*)\phi(z_l^*)^\top - A_l\big] \\ &= G_l^* - A_l. \end{aligned} \tag{19}$$

At convergence over training, the stationary mean of the lateral dynamics satisfies

$$A_l \longrightarrow G_l^*. \tag{20}$$

For fixed positive-definite A_l, the equilibrium interneuron activity satisfies

$$0 = \phi(z_l^*) - A_l a_l^*, \qquad a_l^* = A_l^{-1}\phi(z_l^*). \tag{21}$$

Substituting Eq. 21 into the local weight update yields Eq. 9.

C Deriving the Newton, Gauss–Newton, and Fisher Identities

All derivatives below hold equilibrium activities and precision matrices fixed, and all data averages use $z_0 \sim p_{\mathrm{data}}$. Write $w_l = \mathrm{vec}(W_l)$ and define the prediction Jacobian by

$$J_l^* = \frac{\partial W_l \phi(z_l^*)}{\partial w_l} = \phi(z_l^*)^\top \otimes I. \tag{22}$$

The Gaussian negative log-likelihood in Eq. 10 is

$$\bar{\mathcal{E}}_l(W_l) = \tfrac{1}{2}\mathbb{E}_{z_0}\big[\varepsilon_l^{*\top} \Pi_l \varepsilon_l^*\big] + c_l, \qquad \varepsilon_l^* = z_{l-1}^* - W_l \phi(z_l^*), \tag{23}$$

where c_l is independent of W_l. Since $\partial \varepsilon_l^* / \partial w_l = -J_l^*$ and J_l^* is independent of the weights, direct differentiation gives

$$\begin{aligned} \nabla_{w_l} \bar{\mathcal{E}}_l &= -\mathbb{E}_{z_0}[J_l^{*\top} \Pi_l \varepsilon_l^*], \\ \mathcal{H}_l^* = \nabla_{w_l}^2 \bar{\mathcal{E}}_l &= \mathbb{E}_{z_0}[J_l^{*\top} \Pi_l J_l^*] = \mathcal{G}_l^*. \end{aligned} \tag{24}$$

The middle expression is the Gauss–Newton definition for the precision-weighted least-squares objective. For the Fisher, the layer output is sampled from the Gaussian conditional with z_l^* fixed, giving $\varepsilon_l = z_{l-1} - W_l\phi(z_l^*)$ and $\mathbb{E}[\varepsilon_l\varepsilon_l^\top \mid z_l^*] = \Pi_l^{-1}$. The score is $\nabla_{w_l} \log p = J_l^{*\top}\Pi_l\varepsilon_l$, so

$$\begin{aligned}\mathcal{F}_l^* &= \mathbb{E}_{z_0}\left[J_l^{*\top}\Pi_l\mathbb{E}[\varepsilon_l\varepsilon_l^\top \mid z_l^*]\Pi_l J_l^*\right] \\ &= \mathbb{E}_{z_0}[J_l^{*\top}\Pi_l J_l^*] = \mathbb{E}_{z_0}[\phi(z_l^*)\phi(z_l^*)^\top] \otimes \Pi_l = G_l^* \otimes \Pi_l.\end{aligned} \tag{25}$$

Thus $\mathcal{H}_l^* = \mathcal{F}_l^* = \mathcal{G}_l^* = G_l^* \otimes \Pi_l$. The identities do not include derivatives through the equilibrium activities as the weights change.

D Exact Maximization at Fixed Equilibrium Activities

Hold the equilibrium activities and precisions fixed in Eq. 10, and assume G_l^* and Π_l are positive definite. All expectations below use the same data distribution. Differentiating the Gaussian objective gives

$$\nabla_{W_l}\bar{\mathcal{E}}_l = \Pi_l\left(W_l G_l^* - \mathbb{E}[z_{l-1}^*\phi(z_l^*)^\top]\right). \tag{26}$$

Setting the gradient to zero gives the unique minimizing weights, which also maximize the likelihood,

$$W_l^{\mathrm{opt}} = \mathbb{E}[z_{l-1}^*\phi(z_l^*)^\top]G_l^{*-1}, \qquad \nabla_{W_l}\bar{\mathcal{E}}_l = \Pi_l(W_l - W_l^{\mathrm{opt}})G_l^*. \tag{27}$$

The Fisher $G_l^* \otimes \Pi_l$ therefore gives the natural gradient update with unit learning rate,

$$\begin{aligned}\Delta W_l^{\mathrm{NG}} &= -\Pi_l^{-1}\nabla_{W_l}\bar{\mathcal{E}}_l\, G_l^{*-1} \\ &= -\Pi_l^{-1}\Pi_l(W_l - W_l^{\mathrm{opt}})G_l^* G_l^{*-1} = W_l^{\mathrm{opt}} - W_l.\end{aligned} \tag{28}$$

Adding the update to W_l reaches W_l^{opt} and completes the likelihood maximization. In contrast, a Euclidean update leaves the weight error

$$W_l - \eta\nabla_{W_l}\bar{\mathcal{E}}_l - W_l^{\mathrm{opt}} = (W_l - W_l^{\mathrm{opt}}) - \eta\Pi_l(W_l - W_l^{\mathrm{opt}})G_l^*, \tag{29}$$

which generally cannot vanish for a single scalar learning rate when the remaining error spans directions with different curvature. Standard PC does not perform additional updates to complete the maximization at fixed activities. Exact one-step maximization requires the gradient and Fisher to use the same data average. A stochastic gradient combined with a running Gram estimate does not generally satisfy the one-step identity.

E Monotonicity of the Adaptive Learning Rate

Fix the equilibrium activities and write $\varepsilon_l^* = z_{l-1}^* - W_l\phi(z_l^*)$ for the prediction error. The objective of Eq. 10 is then the quadratic

$$\begin{aligned}\bar{\mathcal{E}}_l(W_l) &= \tfrac{1}{2}\,\mathbb{E}\left[\varepsilon_l^{*\top}\Pi_l\varepsilon_l^*\right] + \text{const}, \\ \nabla_{W_l}\bar{\mathcal{E}}_l &= -\Pi_l M_l, \\ M_l &\triangleq \mathbb{E}\left[\varepsilon_l^*\phi(z_l^*)^\top\right],\end{aligned} \tag{30}$$

with Hessian $G_l^* \otimes \Pi_l$, and the Natural PC update of Eq. 9 is $\Delta W_l = \eta\, M_l A_l^{-1}$, where the precision has cancelled. Because the objective is quadratic, its change under the update is exact,

$$\bar{\mathcal{E}}_l(W_l + \Delta W_l) - \bar{\mathcal{E}}_l(W_l) = -\eta\,\mathrm{tr}\left(S_l A_l^{-1}\right) + \frac{\eta^2}{2}\,\mathrm{tr}\left(A_l^{-1}S_l A_l^{-1}G_l^*\right), \qquad S_l \triangleq M_l^\top\Pi_l M_l \succeq 0. \tag{31}$$

Writing $\tilde{S}_l = A_l^{-1/2} S_l A_l^{-1/2}$ and $\tilde{G}_l = A_l^{-1/2} G_l^* A_l^{-1/2}$ for the matrices in the coordinates whitened by the symmetric positive-definite A_l, the two traces become $\mathrm{tr}(\tilde{S}_l)$ and $\mathrm{tr}(\tilde{S}_l \tilde{G}_l) \le \lambda_{\max}(\tilde{G}_l)\,\mathrm{tr}(\tilde{S}_l)$, and $\tilde{G}_l$ has the eigenvalues of $A_l^{-1} G_l^*$. Therefore

$$\bar{\mathcal{E}}_l(W_l + \Delta W_l) - \bar{\mathcal{E}}_l(W_l) \le -\eta\,\mathrm{tr}(\tilde{S}_l)\left(1 - \frac{\eta}{2}\lambda_{\max}(A_l^{-1} G_l^*)\right), \tag{32}$$

so the objective decreases for every $0 < \eta < 2/\lambda_{\max}(A_l^{-1} G_l^*)$, and $\lambda_{\max}(A_l^{-1} G_l^*)$ is the Lipschitz constant of the gradient in the metric A_l. The rate of Eq. 12 sits at the midpoint of this interval and guarantees a decrease of at least half the first-order decrease. The rate is also the largest that overshoots along no direction. Writing the weight error $D_l = W_l - W_l^{\mathrm{opt}}$ relative to the minimizer, the update gives $D_l^+ = D_l\,(I - \eta G_l^* A_l^{-1})$, whose eigenvalues $1 - \eta\lambda_i(A_l^{-1} G_l^*)$ lie in $[0, 1)$ exactly when $\eta \le 1/\lambda_{\max}(A_l^{-1} G_l^*)$. When $A_l = G_l^*$ all eigenvalues equal one, the rate is one, and the update is the exact maximization of Appendix D. The same argument holds for the objective of a single batch, with the expectation replaced by the batch mean and G_l^* by the batch Gram matrix $\hat{G}_l$, which gives the online rule implemented in Section 3.3.

F Adaptive Learning Rate and Experimental Details

The experiments use 1000 activity-inference iterations per batch with an inference rate of 0.002 and 2000 interneuron-relaxation iterations. The lateral time constant is $\tau_A = 20$ weight updates unless stated otherwise. The reference statistics are computed on the first 10,000 training images, which also participate in training. The reference set is therefore a larger training-set estimate of G_l^*, not a held-out set or an exact expectation over all 50,000 images. Alignment is logged at every update through update 100 in the main run and every ten updates thereafter. The time-constant sweep reports the mean alignment over the final ten logged measurements of 3000-update runs. The figure uses one seed, and the spectral bands describe variation across directions, not uncertainty across independent runs.

The implementation uses precision-weighted prediction errors, so a code learning rate η_l^{code} corresponds to $\eta_l = \eta_l^{\mathrm{code}}/\sigma^2$ in Eqs. 12 and 14. For standard PC, the same conversion applies when expressing the update using $A_l = I$ in the effective-step diagnostic. The plotted standard PC alignment baseline uses $\eta^{\mathrm{code}} = 10^{-2}$, and the effective-step baseline uses $\eta^{\mathrm{code}} = 2 \cdot 10^{-2}$. The latter rate was selected from the fixed-rate sweep. Both methods also use weight decay with coefficient 0.03 in the implementation. The effective-step calculation isolates the likelihood update and does not include weight decay or changes in inferred activities.

For numerical stability, the circuit uses the regularized lateral matrix

$$A_{l,\rho} = A_l + \delta_l I, \qquad \delta_l = \max\left(\rho\,\frac{\mathrm{tr}(A_l)}{d_l}, 10^{-12}\right), \qquad \rho = 10^{-2}. \tag{33}$$

The reference Gram uses the same relative-ridge rule when inverted for alignment. The weighted matrix inner product in Eq. 13 is $\langle X, Y\rangle = \mathrm{tr}(X G_l^* Y^\top)$, with $\|X\|^2 = \langle X, X\rangle$. The activity-weighted cosine similarity used in Figure 2a is

$$\mathrm{align}(A_l, G_l^*) = \frac{\mathrm{tr}(A_{l,\rho}^{-1} G_l^* G_{l,\rho}^{*-1})}{\sqrt{\mathrm{tr}(A_{l,\rho}^{-1} G_l^* A_{l,\rho}^{-1})\ \mathrm{tr}(G_{l,\rho}^{*-1} G_l^* G_{l,\rho}^{*-1})}}. \tag{34}$$

The similarity equals one when the inverse actions are positively proportional and measures directional agreement independently of their overall scale. The effective-step calculation uses the eigenvalues of $A_{l,\rho}^{-1} G_l^*$, with the reference Gram left unregularized.

The layerwise reference rate in Eq. 12 follows from the quadratic batch objective. For a fixed preconditioner, a mode with eigenvalue λ_i has error multiplier $1 - \eta_l \lambda_i$. Choosing $\eta_l = 1/\lambda_{\max}$ keeps every multiplier between zero and one. The guarantee applies to the fixed batch objective with an exact eigenvalue, not automatically to the changing stochastic objective.

The online rule replaces G_l^* in Eq. 12 by the Gram matrix of the current batch, $\hat{G}_l = \frac{1}{B}\sum \phi(z_l^*)\phi(z_l^*)^\top$. We estimate $\lambda_{\max}(A_{l,\rho}^{-1}\hat{G}_l)$ with five power iterations, warm-started from the previous update. Each iteration multiplies by the batch activities and solves a linear system with the regularized lateral matrix. The reference estimate uses the lateral matrix before the current batch update, whereas the synaptic update uses the lateral matrix after the batch update. Writing the estimated eigenvalue at update t as $\hat{\lambda}_{l,t}$, the implementation uses

$$\bar{\lambda}_{l,t} = (1-\tau_A^{-1})\bar{\lambda}_{l,t-1} + \tau_A^{-1}\hat{\lambda}_{l,t}, \qquad \eta_{l,t} = \min\Big(1, \max\Big(10^{-2}, \bar{\lambda}_{l,t}^{-1}\Big)\Big), \tag{35}$$

with the first running estimate initialized to the first measured eigenvalue. In code units, the learning-rate floor is 10^{-3} and the cap is $\sigma^2 = 0.1$. Smoothing, finite power iteration, clipping, and the changing lateral matrix make the implemented rule an approximation to the batch reference rate. The rate calculation reuses the inferred activities but requires additional numerical operations and is not a proof of synapse-local learning-rate control.

For evaluation, the main run uses an exponential weight average with rate $\kappa = 0.05$. Weight averaging is motivated by stochastic-approximation methods (Polyak & Juditsky, 1992), although exponential averaging differs from the uniform averaging in the classical result. After update t, the evaluation weights are

$$S_{l,t} = (1-\kappa)S_{l,t-1} + \kappa W_{l,t}, \qquad \bar{W}_{l,t} = \frac{S_{l,t}}{1-(1-\kappa)^t}, \qquad S_{l,0} = 0. \tag{36}$$

The bias correction removes the contribution of the initial zero accumulator. Training always uses the current weights, and the evaluation average does not feed back into inference or learning during training. The alignment run in panel (a) evaluates reference activities using the averaged weights, whereas panel (c) uses a separate run with unaveraged weights to measure the update geometry. The distinction means that panel (a) compares the learned lateral matrix with the geometry of the evaluation model.

G Effective Step Analysis

Consider the likelihood update with equilibrium activities and the preconditioner held fixed, without weight decay. Let W_l^{opt} minimize the reference objective and let $D_l = W_l - W_l^{\mathrm{opt}}$. After cancelling the isotropic precision through the learning-rate convention above, the matrix error evolves as

$$D_l^+ = D_l\left(I - \eta_l G_l^* A_{l,\rho}^{-1}\right). \tag{37}$$

The eigenvalues of $G_l^* A_{l,\rho}^{-1}$ equal those of the symmetric matrix $A_{l,\rho}^{-1/2} G_l^* A_{l,\rho}^{-1/2}$. In the corresponding transformed coordinates, a mode with eigenvalue λ_i has error multiplier $1-\eta_l\lambda_i$. The unnormalized effective step $s_{l,i} = \eta_l\lambda_i$ is therefore the fraction of the remaining distance covered along that mode. A positive-curvature mode contracts for $0 < s_{l,i} < 2$, reaches the optimum for $s_{l,i} = 1$, and overshoots for $s_{l,i} > 1$. Without the ridge, $A_l = G_l^*$ makes every eigenvalue one and recovers the exact unit step of Appendix D. A ridge or an imperfect Gram estimate generally prevents exact one-step maximization. For a stochastic update, the spectral calculation describes the reference likelihood curvature and does not include the additional error from a minibatch gradient.

The batch Gram introduces another distinction between the reference rate and an exact unit step. For independent, zero-mean Gaussian activities whitened to identity covariance, the upper Marchenko–Pastur edge is

$$c_l = \left(1 + \sqrt{d_l/B}\right)^2. \tag{38}$$

The Gaussian sample-covariance spectrum approaches this edge in the large-dimensional limit (Marchenko & Pastur, 1967). For $d_l = 128$ and $B = 200$, the edge is $c_l = 3.24$. Under the idealized

conditions $A_l = G_l^*$, negligible ridge, and Gaussian whitened activities, the batch reference rate is therefore approximately $1/c_l$, even though the full-data unit rate is one.

Panel (c) compares the observed steps with the Gaussian batch reference using

$$\widetilde{s}_{l,i} = c_l s_{l,i} = \eta_l \lambda_i (A_{l,\rho}^{-1} G_l^*) \left(1 + \sqrt{d_l/B}\right)^2, \tag{39}$$

where d_l is the number of presynaptic neurons and B is the batch size. A normalized value of one corresponds to an actual effective step of approximately 0.31, not exact likelihood maximization. The normalization is applied equally to Natural PC and standard PC and never changes the training update. The Gaussian edge is a reference scale, not a finite-sample safety bound for the nonlinear, dependent activities generated during training. Consequently, deviations from one in the normalized plot need not indicate a failure to learn the Gram. The ridge, non-Gaussian activity statistics, the smoothed eigenvalue estimate, and changes in the weights can all affect the normalized steps.

The eigenvalues are evaluated every ten updates, with medians and percentile bands as described in Figure 2c. The plotted statistics are smoothed over five measurements in log space.

H Benchmark Protocol

Model and inference. Every method trains the generative model of Eq. 1 with $L = 3$ hidden layers of 128 units, $\phi = \tanh$, and a fixed isotropic precision $\Pi_l = \sigma^{-2} I$ with $\sigma^2 = 0.1$ at every layer. The weights are initialized as Gaussian matrices whose columns are normalized to unit length. Inference starts from $z_{1:L} = 0$ and runs $1{,}000$ fixed-step gradient descent iterations on the energy of Eq. 2 at rate 0.002, with flat priors on all hidden layers unless stated otherwise. The same inference is used for training and for evaluation by every method, including the backprop model.

Data. Each dataset provides a training pool and a held-out set. The pool is drawn from the training split of MNIST, FashionMNIST, CIFAR-10 and Tiny ImageNet and from the unlabeled split of STL-10, in a fixed random order. The held-out set is the official test split of MNIST, FashionMNIST and CIFAR-10, the labeled training split of STL-10, and the validation split of Tiny ImageNet. Pixels are scaled to $[0, 1]$ and images are flattened. One epoch is one pass over the training pool, and held-out errors are evaluated on the first 512 images of the held-out set.

Training loops. Mini-batches contain 200 images. Standard PC and iPC on the three small datasets use an epoch loop that reshuffles the pool at every epoch, evaluates every 5 mini-batches, smooths the training error as $s_k = 0.6\, s_{k-1} + 0.4\, e_k$, and stops when the smoothed error improves by less than 0.5% over 500 consecutive evaluations. All other runs use a streaming loop that draws mini-batches from one random permutation of the pool, reshuffling when the pool is exhausted, evaluates every 20 mini-batches and additionally after each of the first 600, and stops when the minimum training error over the last 125 evaluations, is not below 99.5% of the minimum over all earlier evaluations. For Natural PC only the evaluations on the regular 20-batch grid enter the stopping rule. Streaming runs are capped at $20{,}000$ mini-batches for standard PC and iPC, $10{,}000$ for backpropagation, and $8{,}000$ for Natural PC. A run counts as diverged when the training error or any weight becomes non-finite. Each seed determines both the weight initialization and the mini-batch order, so within a seed all methods see the same initialization and the same data order.

Standard PC. After inference, the weights follow Eq. 3 with plain SGD at rate η and decoupled weight decay, $W_l \leftarrow (1 - \eta\,\lambda) W_l + \eta\, \Delta W_l$ with $\lambda = 0.03$, on the small datasets. On STL-10 and Tiny ImageNet the decay is a fixed factor of $1 - 3 \cdot 10^{-5}$ per weight update instead of $\eta\lambda$.

iPC. Following Salvatori et al. (2024), every mini-batch starts from $z_{1:L} = 0$ and runs $T = 20$ inner steps, each consisting of one inference step followed by one weight update with the standard PC rule, so one mini-batch costs 20 weight updates. On the small datasets the inference rate is 0.01 and the weight decay is $\eta\lambda$ with $\lambda = 0.03$. On STL-10 and Tiny ImageNet the inference rate is 0.002, training uses a unit Gaussian prior on every hidden layer, and the decay is a fixed factor of $1 - 1.5 \cdot 10^{-6}$ per weight update. Evaluation uses the flat priors and the $1{,}000$-step inference shared by all methods.

Natural PC. The lateral weights start at $A_l = I$ and are updated once per mini-batch as an exponential moving average of the batch Gram matrix of the equilibrium activities with rate $\alpha_A = 0.05$, corresponding to $\tau_A = 20$ mini-batches (Fig. 2). The interneurons relax for 2,000 steps from $a_l = 0$ on the ridge-regularized matrix $A_{l,\rho}$ of Eq. 33 with $\rho = 10^{-2}$, at a step size of $0.9/\lambda_{\max}(A_{l,\rho})$, and the update direction is the batch mean of $\varepsilon_l^* a_l^{*\top}$ following Eq. 9. The learning rate of each layer is set from the largest eigenvalue $\hat{\lambda}_l$ of $A_{l,\rho}^{-1}\hat{G}_l$, computed exactly from the lateral matrix before the current batch update and the Gram matrix of the current batch. Weight decay is decoupled with $\lambda = 0.03$ at the layerwise rate, $W_l \leftarrow (1 - \eta_l^{\text{code}}\lambda)W_l + \eta_l^{\text{code}}\Delta W_l$, on all datasets. For evaluation, Natural PC reports the error of an exponential average of the weights, $\bar{W}_l \leftarrow (1 - r_t)\bar{W}_l + r_t W_l$ with $r_t = \max(0.05, 1/t)$ after update t, which starts as a uniform average and becomes an exponential average with rate 0.05 after the first 20 updates.

Backprop. The baseline trains the same weights on the same inference. For every mini-batch the 1,000 inference steps are unrolled, and the MSE of the input layer, $\frac{1}{D}\|z_0 - W_1\phi(z_1)\|^2$ averaged over the batch, is differentiated through the whole unroll with gradient checkpointing. The weights are updated with AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and a fixed decay factor of $1 - 3 \cdot 10^{-5}$ per update. Non-finite gradient entries are set to zero. The backpropagation baseline, therefore, differs from standard PC only in the weight update, since inference, initialization, data, and evaluation are shared.

Learning-rate selection. Table 3 lists the selected learning rates of every method and dataset. We selected the learning rates whose mean plateau training error lies within 0.5% of the best mean over the swept grid, excluding any rate for which a seed diverged, and the selected rate is the candidate that reaches the plateau in the fewest mean weight updates. The selected rate is then run with two further seeds, giving five seeds per cell. Natural PC has no learning-rate grid because the rate is set by Eq. 12.

Table 3: Selected learning rates for each method and dataset from Table 1.

Method	MNIST	FMNIST	CIFAR	STL-10	Tiny ImageNet
Standard PC	$5 \cdot 10^{-2}$	$4 \cdot 10^{-2}$	$3 \cdot 10^{-2}$	10^{-2}	$3 \cdot 10^{-3}$
iPC	$3 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$
Backprop (AdamW)	$3 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	10^{-4}	$3 \cdot 10^{-4}$
Natural PC	Adaptive	Adaptive	Adaptive	Adaptive	Adaptive

Metrics. For every run, the plateau error is the minimum training error over all evaluations, and the plateau step is the first evaluation at which the training error is within 2% of that minimum. Table 1 reports, at the plateau step, the held-out reconstruction error $\frac{1}{D}\|z_0 - W_1\phi(z_1^*)\|^2$ scaled by 10^3, the number of epochs, and the number of weight updates, which equals the number of mini-batches for all methods except iPC, where each mini-batch contributes 20 updates. The plateau step is resolved to the evaluation grid, i.e. 5 mini-batches in the epoch loop and 20 in the streaming loop after the first 600.

I SCALING RESULTS

All cells follow the protocol of Table 1. Only the hidden width changes (bottleneck 128, depth 2). Scores are validation-selected plateau MSE ($\times 10^{-3}$), mean $\pm$ std over the five paired seeds. Width denotes the first hidden layer, the layer the reconstruction reads.

Compute. Floating-point operations count the dominant matrix multiplications per mini-batch of size B. With $P = \sum_l 2\, n_l n_{l+1}$ the cost of one pass through the weights, $T = 1{,}000$ inference steps, $R = 2{,}000$ relaxation steps, and $Q = \sum_l w_l^2$ the summed squared hidden widths,

$$\begin{aligned} \text{Standard PC:} &\quad (2T+2)\,P\,B, \\ \text{iPC:} &\quad 4\,P\,B \text{ per weight update}, \\ \text{Natural PC:} &\quad 2\,P\,B\,T + 2\,P\,B + 2\,B\,Q + 4\,R\,B\,Q, \\ \text{Backprop (AdamW):} &\quad 8\,P\,B\,T + P\,B, \end{aligned} \tag{40}$$

where the backpropagation count charges the reverse pass through the unrolled inference at three times the forward cost. The PFLOP values of Table 4 multiply the per-batch cost by the number of mini-batches to the plateau step.

Table 4: Performance and cost per method at two widths (layers $[D, w, w, 128]$), following the protocol of Appendix H. Every baseline uses the learning rate that reaches the plateau in the fewest weight updates among grid points within 0.5% of the best mean MSE with all seeds converged. MSE is the lowest training error of a run, and epochs, W-updates and PFLOP are counted to the first evaluation within 2% of that minimum. *Updates to Std. PC plateau* counts the weight updates until the training error first reaches $1.02\times$ the mean Standard PC MSE of the same cell. Mean $\pm$ standard deviation over 5 seeds. [a]Seed spread above 1%.

Metric	Method	STL-10		Tiny ImageNet	
		Width 128	**Width 256**	**Width 128**	**Width 256**
MSE ($\times 10^{-3}$)	Standard PC	$13.41^{\pm 0.24}$ [b]	$9.66^{\pm 0.05}$ [b]	$15.26^{\pm 0.03}$	$11.78^{\pm 0.03}$
	Natural PC	$\mathbf{13.13}^{\pm 0.03}$	$\mathbf{9.60}^{\pm 0.02}$	$\mathbf{15.17}^{\pm 0.02}$	$11.72^{\pm 0.01}$
	iPC	$14.90^{\pm 0.04}$	$9.83^{\pm 0.02}$	$15.49^{\pm 0.01}$	$11.72^{\pm 0.01}$
	Backprop (AdamW)	$27.68^{\pm 1.84}$ [a]	$10.62^{\pm 0.17}$ [a]	$15.75^{\pm 0.10}$ [a]	$\mathbf{11.71}^{\pm 0.01}$
Epochs	Standard PC	$2.05^{\pm 0.81}$	$2.91^{\pm 0.37}$	$3.85^{\pm 0.23}$	$1.67^{\pm 0.17}$
	Natural PC	$0.43^{\pm 0.05}$	$\mathbf{0.46}^{\pm 0.02}$	$\mathbf{0.29}^{\pm 0.00}$	$\mathbf{0.42}^{\pm 0.02}$
	iPC	$\mathbf{0.32}^{\pm 0.04}$	$4.13^{\pm 0.21}$	$2.88^{\pm 0.16}$	$1.36^{\pm 0.03}$
	Backprop (AdamW)	$3.93^{\pm 3.53}$	$11.19^{\pm 3.58}$	$4.69^{\pm 0.47}$	$7.36^{\pm 1.34}$
W-updates	Standard PC	$1{,}024^{\pm 403}$	$1{,}456^{\pm 184}$	$1{,}924^{\pm 117}$	$836^{\pm 87}$
	Natural PC	$\mathbf{214}^{\pm 27}$	$\mathbf{228}^{\pm 8}$	$\mathbf{143}^{\pm 2}$	$\mathbf{209}^{\pm 9}$
	iPC	$3{,}248^{\pm 396}$ [a]	$41.28\text{k}^{\pm 2.06\text{k}}$	$28.80\text{k}^{\pm 1.60\text{k}}$	$13.60\text{k}^{\pm 0.25\text{k}}$
	Backprop (AdamW)[b]	$1{,}965^{\pm 1{,}767}$	$5{,}596^{\pm 1{,}788}$	$2{,}344^{\pm 236}$	$3{,}680^{\pm 672}$
Updates to Std. PC plateau	Standard PC	$780^{\pm 51}$	$1{,}468^{\pm 131}$	$1{,}924^{\pm 79}$	$820^{\pm 22}$
	Natural PC	$\mathbf{133}^{\pm 8}$	$\mathbf{192}^{\pm 8}$	$\mathbf{130}^{\pm 4}$	$\mathbf{186}^{\pm 4}$
	iPC	never	$82\text{k}^{\pm 10.13\text{k}}$	$65\text{k}^{\pm 1.94\text{k}}$	$12\text{k}^{\pm 0.49\text{k}}$
	Backprop (AdamW)	never	never	never	$3{,}256^{\pm 432}$
PFLOP	Standard PC	$2.93^{\pm 1.15}$	$8.37^{\pm 1.06}$	$2.47^{\pm 0.15}$	$2.17^{\pm 0.23}$
	Natural PC	$0.63^{\pm 0.08}$	$1.36^{\pm 0.05}$	$0.20^{\pm 0.00}$	$0.59^{\pm 0.03}$
	iPC	$\mathbf{0.02}^{\pm 0.00}$	$\mathbf{0.47}^{\pm 0.02}$	$\mathbf{0.07}^{\pm 0.00}$	$\mathbf{0.07}^{\pm 0.00}$
	Backprop (AdamW)[b]	$22.46^{\pm 20.20}$	$128.52^{\pm 41.06}$	$12.05^{\pm 1.21}$	$38.21^{\pm 6.98}$

We also report depth and width scaling results on the exact same experimental setup like in the main text in Tab. 5, detailed in App. H. Diverged indicates that at least 2 of 5 seeds diverged during training. Mean $\pm$ std are reported over 5 seeds. The learning rate was individually tuned for each method.

Table 5: Reconstruction Error by Depth, Width ($W = 128$), Activation Function and Method

		tanh			ReLU		
Depth	**Method**	MNIST	FMNIST	CIFAR	MNIST	FMNIST	CIFAR
4	Standard PC	$5.33^{\pm 0.02}$	$7.83^{\pm 0.03}$	$5.82^{\pm 0.00}$	$5.36^{\pm 0.08}$	$7.93^{\pm 0.07}$	$11.33^{\pm 1.22}$
	iPC	$14.00^{\pm 0.04}$	$24.66^{\pm 0.16}$	$61.13^{\pm 0.01}$	$17.33^{\pm 0.16}$	$24.10^{\pm 0.10}$	$29.27^{\pm 0.30}$
	Natural PC	$4.09^{\pm 0.03}$	$6.74^{\pm 0.13}$	$5.55^{\pm 0.20}$	$4.84^{\pm 0.03}$	$\mathbf{6.64}^{\pm 0.19}$	$6.66^{\pm 0.15}$
	Implicit Natural PC	$4.98^{\pm 0.01}$	$7.59^{\pm 0.26}$	$5.94^{\pm 0.04}$	$8.26^{\pm 0.04}$	$11.02^{\pm 0.04}$	*diverged*
	Backprop (AdamW)	$4.08^{\pm 0.00}$	$6.12^{\pm 0.00}$	$5.31^{\pm 0.01}$	$\mathbf{4.59}^{\pm 0.01}$	$6.75^{\pm 0.03}$	$\mathbf{6.62}^{\pm 0.33}$
	Backprop (SGD)	$4.06^{\pm 0.01}$	*diverged*	*diverged*	$5.29^{\pm 0.08}$	$7.28^{\pm 0.09}$	$7.25^{\pm 0.30}$
	Backprop (L-BFGS)	$\mathbf{4.02}^{\pm 0.01}$	$\mathbf{6.07}^{\pm 0.00}$	$\mathbf{5.29}^{\pm 0.00}$	*diverged*	$17.81^{\pm 4.01}$	$23.04^{\pm 0.61}$
8	Standard PC	$5.33^{\pm 0.02}$	$7.93^{\pm 0.07}$	$6.17^{\pm 0.02}$	$5.24^{\pm 0.07}$	$7.86^{\pm 0.06}$	*diverged*
	iPC	$13.80^{\pm 0.03}$	$21.90^{\pm 0.09}$	$61.13^{\pm 0.01}$	$18.19^{\pm 0.16}$	$25.83^{\pm 0.09}$	$31.42^{\pm 0.22}$
	Natural PC	$4.12^{\pm 0.11}$	$6.76^{\pm 0.27}$	$\mathbf{5.27}^{\pm 0.09}$	$5.19^{\pm 0.08}$	$7.12^{\pm 0.23}$	$\mathbf{6.86}^{\pm 0.31}$
	Implicit Natural PC	$4.93^{\pm 0.01}$	$6.77^{\pm 0.02}$	$5.79^{\pm 0.06}$	$8.28^{\pm 0.03}$	*diverged*	*diverged*
	Backprop (AdamW)	$4.08^{\pm 0.00}$	$6.13^{\pm 0.01}$	$5.31^{\pm 0.01}$	$\mathbf{4.98}^{\pm 0.04}$	$\mathbf{7.03}^{\pm 0.04}$	$6.90^{\pm 0.40}$
	Backprop (SGD)	$4.08^{\pm 0.01}$	*diverged*	*diverged*	$5.78^{\pm 0.08}$	$8.18^{\pm 0.15}$	$7.51^{\pm 0.13}$
	Backprop (L-BFGS)	$\mathbf{4.03}^{\pm 0.00}$	$\mathbf{6.07}^{\pm 0.00}$	$5.28^{\pm 0.01}$	$22.94^{\pm 7.61}$	$16.09^{\pm 0.08}$	$20.51^{\pm 0.77}$
32	Standard PC	$5.12^{\pm 0.02}$	$7.61^{\pm 0.07}$	$6.04^{\pm 0.02}$	*diverged*	*diverged*	*diverged*
	iPC	$13.21^{\pm 0.02}$	$22.21^{\pm 0.08}$	$61.12^{\pm 0.00}$	$17.81^{\pm 0.14}$	$25.46^{\pm 0.10}$	$31.43^{\pm 0.21}$
	Natural PC	$\mathbf{4.02}^{\pm 0.18}$	$6.13^{\pm 0.14}$	$5.39^{\pm 0.22}$	$5.03^{\pm 0.09}$	$7.12^{\pm 0.12}$	$6.56^{\pm 0.47}$
	Implicit Natural PC	$4.74^{\pm 0.01}$	$7.11^{\pm 0.44}$	$5.80^{\pm 0.05}$	$7.94^{\pm 0.04}$	*diverged*	*diverged*
	Backprop (AdamW)	$4.08^{\pm 0.00}$	$6.12^{\pm 0.01}$	$5.33^{\pm 0.01}$	$\mathbf{4.92}^{\pm 0.05}$	$\mathbf{6.96}^{\pm 0.01}$	$\mathbf{6.37}^{\pm 0.32}$
	Backprop (SGD)	$4.08^{\pm 0.01}$	*diverged*	*diverged*	$5.75^{\pm 0.09}$	*diverged*	$7.47^{\pm 0.09}$
	Backprop (L-BFGS)	$4.04^{\pm 0.00}$	$\mathbf{6.09}^{\pm 0.00}$	$\mathbf{5.30}^{\pm 0.01}$	$19.06^{\pm 3.56}$	$16.28^{\pm 0.32}$	$20.29^{\pm 0.89}$
64	Standard PC	$5.12^{\pm 0.02}$	$7.61^{\pm 0.07}$	$6.04^{\pm 0.02}$	*diverged*	*diverged*	*diverged*
	iPC	$13.21^{\pm 0.02}$	$22.22^{\pm 0.07}$	$61.12^{\pm 0.00}$	$17.81^{\pm 0.14}$	$25.46^{\pm 0.10}$	$31.44^{\pm 0.21}$
	Natural PC	$\mathbf{3.94}^{\pm 0.08}$	$6.29^{\pm 0.24}$	$5.45^{\pm 0.21}$	$5.09^{\pm 0.10}$	$\mathbf{6.83}^{\pm 0.18}$	$\mathbf{6.51}^{\pm 0.19}$
	Implicit Natural PC	$4.74^{\pm 0.01}$	$7.08^{\pm 0.41}$	$5.81^{\pm 0.05}$	$7.93^{\pm 0.05}$	*diverged*	*diverged*
	Backprop (AdamW)	$4.08^{\pm 0.01}$	$6.33^{\pm 0.01}$	$\mathbf{5.39}^{\pm 0.02}$	$\mathbf{4.91}^{\pm 0.01}$	$6.94^{\pm 0.01}$	$6.54^{\pm 0.50}$
	Backprop (SGD)	$4.78^{\pm 0.11}$	$7.14^{\pm 0.13}$	$6.09^{\pm 0.08}$	$5.74^{\pm 0.05}$	$8.11^{\pm 0.08}$	$7.52^{\pm 0.06}$
	Backprop (L-BFGS)	$4.05^{\pm 0.00}$	$\mathbf{6.12}^{\pm 0.01}$	*diverged*	$17.58^{\pm 0.51}$	$16.08^{\pm 0.05}$	$19.90^{\pm 0.52}$

J IMAGE CLASSIFICATION

We evaluate all methods on supervised classification with the standard discriminative construction of predictive coding (Whittington & Bogacz, 2017; Pinchetti et al., 2025). The image is clamped at the deepest layer and the one-hot label is the observation. During learning, the hidden states are initialized by a forward sweep and relax for 50 fixed-step iterations with both ends clamped. At test time the prediction is the forward sweep, which coincides with the energy minimum when the output is free. Since the contribution of this work is the geometry of the weight update, every other choice is kept fixed and minimal.

Results are reported in Table 6. All PC methods update the weights with plain SGD, without momentum, learning rate schedules or output nudging, and use no weight decay, no biases, tanh activations and a learning rate matched per dataset. Standard PC is the plain Hebbian rule of equation 3. The BP baseline uses AdamW on the same architecture and squared-error loss. The L-BFGS baseline uses the two-loop recursion with a history of 10 pairs for minibatch and 20 for full-batch training, curvature pairs from the same batch, the cautious acceptance rule $s^\top y > \epsilon, |s|^2$, the initial scaling $H_0 = (s^\top y / y^\top y), I$, an Armijo line search starting from unit step length, a steepest-descent fallback when no pair is accepted, and no weight decay. Every run receives the same budget of 250 epochs without early stopping, because adaptive stopping criteria assign different budgets to different optimizers and can produce apparent stalls or collapses. Accuracies are reported at the end of the shared budget. Networks are three-layer perceptrons of width 256 trained on the full training sets.

Additionally, we report depth scaling results on the exact same experimental setup in Table 7. Diverged indicates that at least 2 of 5 seeds diverged during training. Mean $\pm$ std are reported over 5 seeds. The learning rate was individually tuned.

Table 6: Classification accuracy at a fixed budget of 250 epochs (width 256, depth 3, tanh. Mean $\pm$ standard deviation over three seeds). Accuracy is the test accuracy at the end of the budget. The speed columns give the epochs and weight updates needed to first reach a reference accuracy of 90, 80 and 30 percent on MNIST, FashionMNIST and CIFAR-10 respectively. [a]Full-batch L-BFGS runs 250 full-data iterations, about 2.7 times the compute of the other methods. At matched compute of about 90 iterations, the accuracies are 86.7, 76.4 and 15.4 percent.

Task	Method	Acc. (%) @250 ep	Epochs $\rightarrow$ ref.	W-updates $\rightarrow$ ref.
MNIST	Standard PC	$94.71^{\pm 0.26}$	$31.9^{\pm 4.2}$	$10\text{k}^{\pm 1\text{k}}$
	Natural PC	$\mathbf{97.72}^{\pm 0.12}$	$\mathbf{1.6}^{\pm 0.2}$	$473^{\pm 45}$
	iPC	$85.01^{\pm 0.10}$	never	never
	Backprop (AdamW)	$97.24^{\pm 0.12}$	$\mathbf{1.6}^{\pm 0.2}$	$\mathbf{467}^{\pm 47}$
	Backprop (L-BFGS)	$93.98^{\pm 0.18}$	$10.7^{\pm 1.9}$	$3\text{k}^{\pm 1\text{k}}$
	Backprop (L-BFGS, full)	$93.36^{\pm 0.16}$ [a]	$105.3^{\pm 4.0}$ [b]	$105^{\pm 4}$
FMNIST	Standard PC	$70.05^{\pm 17.08}$	$26.0^{\pm 6.6}$	$8\text{k}^{\pm 2\text{k}}$
	Natural PC	$88.62^{\pm 0.13}$	$2.2^{\pm 0.5}$	$680^{\pm 115}$
	iPC	$80.60^{\pm 0.09}$	$105.9^{\pm 3.5}$	$635\text{k}^{\pm 21\text{k}}$
	Backprop (AdamW)	$\mathbf{88.63}^{\pm 0.03}$	$\mathbf{1.7}^{\pm 0.3}$	$500^{\pm 82}$
	Backprop (L-BFGS)	$83.25^{\pm 0.05}$	$4.5^{\pm 0.4}$	1k
	Backprop (L-BFGS, full)	$81.81^{\pm 0.02}$ [a]	$59.7^{\pm 3.1}$ [b]	$\mathbf{60}^{\pm 3}$
CIFAR	Standard PC	$31.28^{\pm 1.32}$	$35.5^{\pm 7.3}$	$9\text{k}^{\pm 2\text{k}}$
	Natural PC	$\mathbf{48.03}^{\pm 0.45}$	$\mathbf{2.4}^{\pm 0.1}$	$\mathbf{353}^{\pm 21}$
	iPC	$29.11^{\pm 0.72}$	never	never
	Backprop (AdamW)	$45.15^{\pm 0.43}$	$6.5^{\pm 0.2}$	2k
	Backprop (L-BFGS)	$30.21^{\pm 0.94}$	$14.4^{\pm 2.1}$	$4\text{k}^{\pm 1\text{k}}$
	Backprop (L-BFGS, full)	$21.60^{\pm 7.71}$ [a]	never	never

Table 7: Classification Accuracy by Depth, Activation Function and Method

Method	d=4	d=8	d=32	d=64
tanh				
Standard PC	$87.1^{\pm 1.1}$	$82.1^{\pm 0.5}$	$34.7^{\pm 0.8}$	$25.2^{\pm 1.1}$
iPC	$33.3^{\pm 0.5}$	*diverged*	$10.9^{\pm 0.4}$	$10.7^{\pm 0.1}$
Implicit Natural PC	$70.1^{\pm 0.8}$	*diverged*	*diverged*	*diverged*
Natural PC (fixed lr)	$94.5^{\pm 0.2}$	$91.1^{\pm 0.8}$	$36.6^{\pm 0.7}$	$22.6^{\pm 2.2}$
Natural PC (Lipschitz step)	$97.1^{\pm 0.1}$	$96.7^{\pm 0.2}$	$\mathbf{77.7}^{\pm 9.5}$	$\mathbf{50.8}^{\pm 1.3}$
Backprop (AdamW)	$\mathbf{97.7}^{\pm 0.1}$	$\mathbf{97.5}^{\pm 0.1}$	*diverged*	*diverged*
Backprop (L-BFGS, full batch)	$89.3^{\pm 2.7}$	$90.4^{\pm 0.3}$	$77.2^{\pm 6.8}$	*diverged*
ReLU				
Standard PC	$87.7^{\pm 0.2}$	$67.3^{\pm 5.5}$	$15.5^{\pm 1.1}$	$12.7^{\pm 1.2}$
iPC	$19.0^{\pm 0.3}$	*diverged*	$9.9^{\pm 0.4}$	$10.2^{\pm 0.2}$
Implicit Natural PC	$20.2^{\pm 2.7}$	$11.1^{\pm 0.2}$	$10.6^{\pm 0.3}$	$10.1^{\pm 0.2}$
Natural PC (fixed lr)	$89.6^{\pm 0.4}$	$72.8^{\pm 1.2}$	$18.8^{\pm 1.5}$	$14.6^{\pm 1.3}$
Natural PC (Lipschitz step)	$93.1^{\pm 0.4}$	$81.0^{\pm 1.1}$	$23.3^{\pm 0.9}$	$\mathbf{15.9}^{\pm 1.8}$
Backprop (AdamW)	$\mathbf{97.2}^{\pm 0.1}$	$\mathbf{97.2}^{\pm 0.1}$	$\mathbf{94.9}^{\pm 1.6}$	*diverged*
Backprop (L-BFGS, full batch)	$92.9^{\pm 4.4}$	$96.8^{\pm 0.5}$	*diverged*	*diverged*

K Kronecker-Factored Lateral Weights for Convolutional Layers

In a convolutional layer, the presynaptic activity of every output position is a patch of k^2 spatial offsets across C input channels, so the lateral weights A_l of Eq. 7 form a $k^2C \times k^2C$ matrix over patch elements. The lateral circuit can be reduced to two smaller populations by factoring the patch statistics into a spatial and a channel part,

$$A_l \approx A_l^{\mathrm{s}} \otimes A_l^{\mathrm{c}}, \qquad \tau_A \dot{A}_l^{\mathrm{s}} = \sum_c \phi_c \phi_c^\top - A_l^{\mathrm{s}}, \qquad \tau_A \dot{A}_l^{\mathrm{c}} = \sum_o \phi_o \phi_o^\top - A_l^{\mathrm{c}}, \tag{41}$$

where $\phi_c \in \mathbb{R}^{k^2}$ collects the activities of channel c across the spatial offsets of a patch and $\phi_o \in \mathbb{R}^C$ collects the activities of offset o across channels. Each factor is the Gram matrix of a marginal of the same presynaptic activities, so both follow the lateral dynamics of the main text and converge to the corresponding marginal second moments. The approximation treats spatial and channel correlations as independent, which is the assumption underlying Kronecker-factored curvature for convolutional layers (Grosse & Martens, 2016).

The inverse of a Kronecker product is the Kronecker product of the inverses. Writing the presynaptic error signal of a patch as the matrix $E_l \in \mathbb{R}^{k^2 \times C}$, the whitened update of Eq. 9 becomes

$$(A_l^{\mathrm{s}} \otimes A_l^{\mathrm{c}})^{-1} \operatorname{vec}(E_l) = \operatorname{vec}\Big((A_l^{\mathrm{s}})^{-1} E_l \, (A_l^{\mathrm{c}})^{-1} \Big), \tag{42}$$

so the interneurons whiten the spatial and the channel dimension of the error separately, and the number of lateral weights drops from k^4C^2 to $k^4 + C^2$. The adaptive learning rate of Eq. 12 is unchanged, with the factored A_l in place of the full one, and the monotonicity argument of Appendix E applies verbatim since the factored matrix remains symmetric positive definite.

L IMAGENET EXPERIMENTS

Architecture. The generative model is a top-down VGG-10 decoder without biases. A $1{,}024$-dimensional code is mapped by a dense layer to a $7 \times 7 \times 512$ tensor and then through nine 3×3 convolutional layers with tanh nonlinearities, with resolutions and channels $14 \times 14 \times 512$, $14 \times 14 \times 512$, $28 \times 28 \times 256$, $28 \times 28 \times 256$, $56 \times 56 \times 128$, $56 \times 56 \times 128$, $112 \times 112 \times 64$, $112 \times 112 \times 64$ and $224 \times 224 \times 3$, for 32.73M parameters plus the 2M lateral weights make a total of 34.73M parameters. The autoencoder uses the same decoder and adds an untied bottom-up VGG-10 encoder with stride-2 convolutions from the image to the $1{,}024$-dimensional code, for 65.47M parameters in total.

Data and evaluation. Both models train on the $1{,}281{,}167$ training images of ImageNet-1k, stored at 256 pixels and served as random 224-pixel crops with horizontal flips, with pixels in $[0, 1]$. Mini-batches contain 800 images, distributed as 8×100 over eight GPUs, so one epoch is $1{,}601$ updates and the 10-epoch budget is $16{,}010$ updates. Every 400 updates, both models are evaluated on the first $5{,}000$ validation images with center 224-pixel crops. The reported error is the mean squared error between the image and the image decoded from the code. For Natural PC that is the $1{,}024$-dimensional code inferred with the image clamped and for the autoencoder that is the encoder output. For Natural PC we additionally record the pixel error of the clamped equilibrium, which lies at 10^{-5} throughout training, and the energy per image at equilibrium.

Natural PC. Inference clamps the image and runs 300 energy steps at rate 0.003 from a zero state, with residual variance $\sigma^2 = 0.1$ at every layer and a unit Gaussian prior on the code. The lateral weights of every convolutional layer follow the Kronecker factorization of App.K, with a 9×9 spatial factor and a $C_{in} \times C_{in}$ channel factor, and the dense top layer carries a full $1,024 \times 1,024$ lateral matrix. The lateral weights total 2.01M entries. They are updated once per batch as an exponential moving average with $\tau_A = 20$ and regularized with a ridge of 0.3 times their mean diagonal. The inverse is applied to the presynaptic patches by a direct linear solve. The learning rate of every layer follows Eq. 12, with the largest eigenvalue estimated by 30 Lanczos iterations, and weight decay is decoupled at 0.03 times the layerwise rate. The reported errors use an exponential weight average with rate 0.05. Convolutions run in bfloat16, and the Gram matrices, solves, states and weights are kept in float32. There is no learning rate to tune as the method takes unit steps.

Backprop autoencoder. The autoencoder minimizes the mean squared reconstruction error with AdamW at learning rate $3 \cdot 10^{-4}$, 10% linear warm-up followed by a cosine decay to a tenth of the peak rate, weight decay 0.03 and gradient clipping at norm 1, in float32. The run was resumed from a checkpoint at update 800 and the reported wall time adds the 151 seconds of the first segment.

Compute. Floating-point operations are counted analytically per update. A convolution forward pass costs $2\,B\,H\,W\,(9C_{\text{in}})\,C_{\text{out}}$ per layer. Natural PC charges 300 inference iterations at two forward-equivalents each, one forward pass and one backward pass with respect to the state, plus the Gram update $2nd^2$, the weight gradient $2ndC_{\text{out}}$ and the solve $d^3/3$ per layer, with $d = 9C_{\text{in}}$ and n the number of patches in the batch. The autoencoder charges three forward-equivalents of encoder and decoder. Per update this gives $5.14 \cdot 10^{15}$ operations for Natural PC and $4.99 \cdot 10^{13}$ for the autoencoder, over the 10-epoch budget, $82{,}291$ against 799 PFLOP. On $8\times$H200 GPUs the Natural PC run takes 12.7 hours of wall time, of which 12.2 hours are the update loop, at 2.74 seconds per update or 292 images per second, whereas the autoencoder takes 43.7 minutes, of which 32.5 minutes are the update loop, at 0.122 seconds per update or $6{,}575$ images per second.

Results. Table 8 lists both runs at every epoch and at the milestones quoted in the main text. Natural PC first evaluates below the autoencoder's final error of $8.62 \cdot 10^{-3}$ at update $2{,}001$, after 1.25 epochs, $10{,}285$ PFLOP and 1.61 hours, and below 10^{-2} at update 801. The autoencoder reaches 10^{-2} at update $12{,}401$. After the full budget, Natural PC ends at $7.95 \cdot 10^{-3}$ against $8.62 \cdot 10^{-3}$ for the autoencoder, whose best evaluation during training is $8.45 \cdot 10^{-3}$ at update $16{,}001$. The energy per image at equilibrium falls from $2 \cdot 10^5$ at initialization to 12.6 at the end of training, and the training energy over the last 50 updates is 10.85. Both runs are single seeds.

Table 8: ImageNet reconstruction error ($\times 10^{-3}$) on $5{,}000$ validation images at every epoch and at $2k$ and $8k$ updates. The last four columns give the cumulative compute in PFLOP and in GPU-hours on H200.

Update	Epoch	Natural PC		Autoencoder	PFLOP		GPU-hours	
		MSE	energy	MSE	Natural PC	AE	Natural PC	AE
1	0.00	265.7	199,968	268.9	4	0	0.1	0.05
801	0.50	9.00	22.9	33.6	3,016	40	3.8	0.39
1,601	1.00	8.65	19.4	27.4	6,029	80	7.6	0.68
2,001	1.25	8.61	17.5	23.9	7,535	100	9.4	0.81
3,201	2.00	8.43	18.4	17.5	12,053	160	15.0	1.25
4,801	3.00	8.24	14.3	18.9	18,079	240	22.4	1.83
6,401	4.00	8.01	13.2	14.8	24,104	320	29.8	2.39
8,001	5.00	8.59	26.1	12.8	30,129	399	37.3	2.97
9,601	6.00	8.45	15.3	12.5	36,154	479	44.7	3.56
11,201	7.00	8.19	15.1	10.3	42,179	559	52.2	4.12
12,801	8.00	8.09	15.7	9.42	48,204	639	59.6	4.68
14,401	9.00	8.07	17.2	8.85	54,229	719	67.0	5.25
16,010	10.00	**7.95**	12.6	8.62	60,286	799	74.5	5.83

Inference budget. Table 9 evaluates the checkpoint at update $6{,}400$ on 500 validation images with inference budgets from 100 to $3{,}000$ steps, each from a zero state. The reconstruction error reaches its plateau at 300 steps and does not decrease with longer inference.

Table 9: Inference budget check at update $6{,}400$ on 500 validation images. Reconstruction error ($\times 10^{-3}$) and equilibrium energy are reported per image against the number of inference steps.

Steps	100	200	300	500	1,000	2,000	3,000
MSE	45.9	8.83	8.24	8.27	8.30	8.32	8.36
Energy	1,252	45.6	13.6	10.1	9.12	8.76	8.82

Classification. For reference, a backprop VGG-10 classifier on the same data reaches 32.1% top-1 accuracy after 10 epochs at batch size 100 with a flat head, and 30.8% after 20 epochs at batch size 50 with a global-average-pooling head. A discriminative Natural PC classifier with the construction of Appendix J did not learn at this scale, so we report ImageNet results for the generative model only.

M Implicit Natural PC

Natural PC inverts the presynaptic correlations that defines the natural gradient and replaces the equilibrium activations that carry the correlations with this filtered learning signal. However, decorrelating the activity directly during inference offers a second route to natural gradient descent. When the equilibrium Gram G_l^* is diagonal, the preconditioner in equation 6 reduces to a per-coordinate rescaling, and the standard PC update of equation 3 already points along the natural gradient up to a scale. A lateral circuit that whitens the activity during inference should therefore make natural gradient learning implicit, without changing the weight update rule. This variant follows the view that lateral inhibition decorrelates the representation (King et al., 2013; Chettih & Harvey, 2019; Znamenskiy et al., 2024), and we refer to it as Implicit Natural PC.

Following the recurrent whitening network of Pehlevan & Chklovskii (2015), we add to each layer a symmetric lateral weight matrix B_l that contributes $\frac{1}{2}\phi(z_l)^\top B_l \phi(z_l)$ to the energy of equation 2, so that the activities settle under lateral competition. The lateral weights follow the anti-Hebbian rule

$$B_l \leftarrow \mathrm{clip}\big(B_l + \eta_B, (G_l - c, I)\big), \qquad c = \mathrm{mean}, \mathrm{diag}, G_l,$$

where G_l is the Gram of the equilibrium activity on the current batch. The rule increases competition between co-active units and decreases competition between anti-correlated units, and reaches a fixed

point when $G_l = cI$, that is, when the activity is white up to a scale. Because B_l enters the Hessian of the inference dynamics, we clip the eigenvalues of B_l to a bounded interval so that the fixed-step relaxation remains stable across layer widths. The weight update remains the standard PC rule of equation 3.

In Fig. 3, we plot the alignment of the applied update with the natural gradient against the off-diagonal mass of the equilibrium Gram, pooled over training checkpoints, layers and datasets. Standard PC aligns with the natural gradient only while the Gram is close to diagonal and falls to a cosine of about 0.2 as correlations increase. Implicit Natural PC lies above standard PC at matched off-diagonal mass, so the whitening improves alignment beyond what a more diagonal Gram alone would give. Natural PC holds a cosine above 0.99 at off-diagonal masses of 0.8 to 1.0, where every method without an explicit inverse has fallen to 0.2 to 0.5. Diagonality is a requirement only for updates that do not invert the Gram.

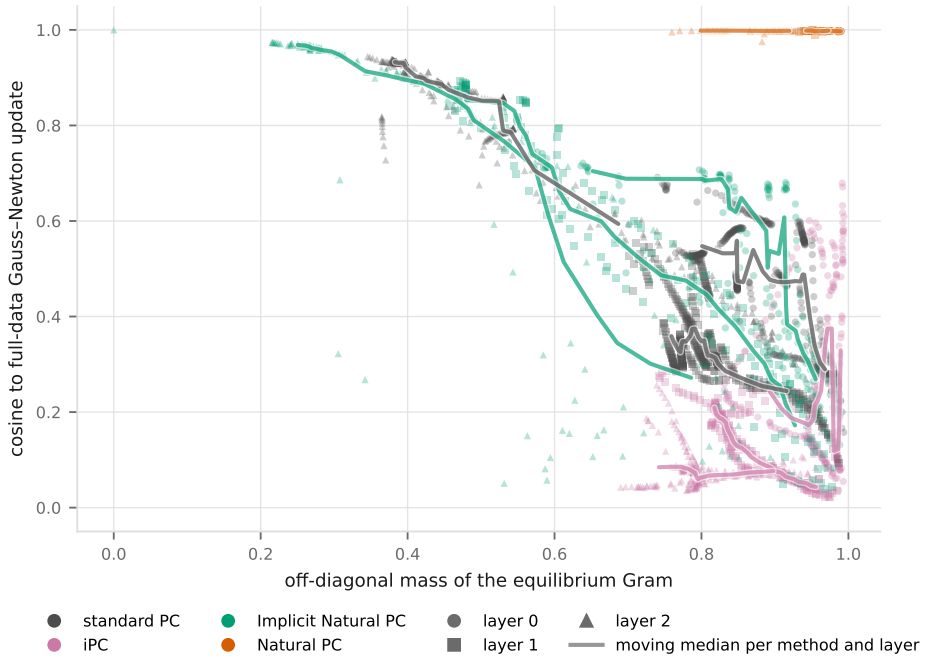


Figure 3: Alignment of each method's weight update with the natural gradient, plotted against the off-diagonal mass of the equilibrium Gram. Points are training checkpoints pooled over MNIST, FashionMNIST and CIFAR-10. Marker shape encodes the layer, colour the method, and heavy curves are moving medians per method and layer. The reference update uses the minibatch gradient with the full-data Gram.

N Analysis of the Lateral Circuit

The main text shows that Natural PC learns faster and reaches lower error than the other PC methods. This appendix asks what the lateral circuit computes and why the computation helps. We examine what the lateral weights store, how the learned metric evolves, how much temporal averaging the circuit needs and whether the speedup is caused by the circuit online. Throughout, Natural PC denotes the lateral-circuit model of the main text, and Implicit Natural PC the whitened-inference variant of App. M. All experiments use the width-64 reconstruction setting with shared initialization and batch order per seed. Curve panels report three seeds, with bands showing the range across seeds $\pm$ std, and landscape, heatmap and eigenmode panels show one representative seed. Misalignment is measured as $1 - \cos(\cdot, , U(g, G_N^*))$, where $U(g, G) = g, (G + \lambda I)^{-1}$ is the natural gradient update with the full-data Gram, evaluated at each run's own weights on a fixed probe set.

N.1 What do the lateral weights store?

The lateral weights of the first hidden layer, $A^{(0)}$, connect interneurons that are driven by principal neurons with decoder columns w_i and w_j. A hypothesis is that the weight between two interneurons follows how similar or how overlapping the two decoder columns are. We falsify these hypotheses (Fig. 4a). The correlation between $A_{ij}^{(0)}$ and the decoder similarity $\langle w_i, w_j \rangle$ is 0.01 on MNIST, 0.06 on FashionMNIST and 0.16 on CIFAR-10, and the correlation with the inverse-overlap prediction is 0.05. Moreover, digit classes also do not predict the structure of the lateral weights. When the matrix is sorted by digit class, weights between interneurons of the same class exceed weights between classes by only 5 percent (Fig. 4b).

Sorting by the leading eigenvector instead exposes the structure that is present (Fig. 4b'). Most of the variance in the lateral weights is account for by a few eigenvectors that span the whole population. The leading eigenvalue is six times the second, and the leading eigenvectors decode to image-scale patterns, a mean-digit blob followed by stroke-like components (Fig. 4c). These eigenvectors form within the first tens of updates. After removing the identity initialization, the off-diagonal fraction of A rises from near zero to about 0.9 within the first 50 updates in all three layers, while the effective rank, measured as the participation ratio of the eigenvalues, grows from about 2 to about 17 over training (Figure 4d). The lateral weights therefore store the main directions along which the principal neurons vary together, which is a statistical property of the activity and is unrelated to the features the principal neurons encode.

N.2 How is the Gram matrix learned?

The lateral weights are an estimate of a target that moves during training, since the equilibrium Gram $G_N^*(t)$ changes as the weights change. Fig. 5 compares the debiased lateral matrix $\hat{A}_t$, from which the decayed identity initialization $(1 - \alpha)^t I$ has been removed, of the top layer with the full-data Gram at the same time. The eigenvalues of the two matrices agree closely at every rank and at every time (Fig. 5a), even though the spectrum itself changes by more than an order of magnitude: the leading eigenvalue rises during the first 30 updates and then decays, and the eigenvalues of ranks two to sixteen rise sharply between 50 and 300 updates as the code develops structure. The eigenvectors of the two matrices are aligned as well, with the overlap matrix of the top sixteen eigenvectors close to diagonal at 54, 326 and $2{,}450$ updates (Fig. 5b). The subspace angles in Fig. 5c separate two processes. The left panel shows how closely the lateral matrix at time t matches the Gram at that same time point. The right panel shows how the metric settles toward the final Gram: the leading eigenvector stops rotating after about 500 updates and the full sixteen-dimensional subspace after about 10^3. In between, the angles for ranks three and higher are large, which is the phase in which many eigenvalues cross (Fig. 5a), so individual eigenvectors are poorly defined and the top-k subspaces of two nearly identical spectra can differ. The lateral weights therefore follow a moving metric rather than converging to a fixed one, and the eigenvalues are tracked throughout, while eigenvector tracking is limited only where the metric itself is degenerate.

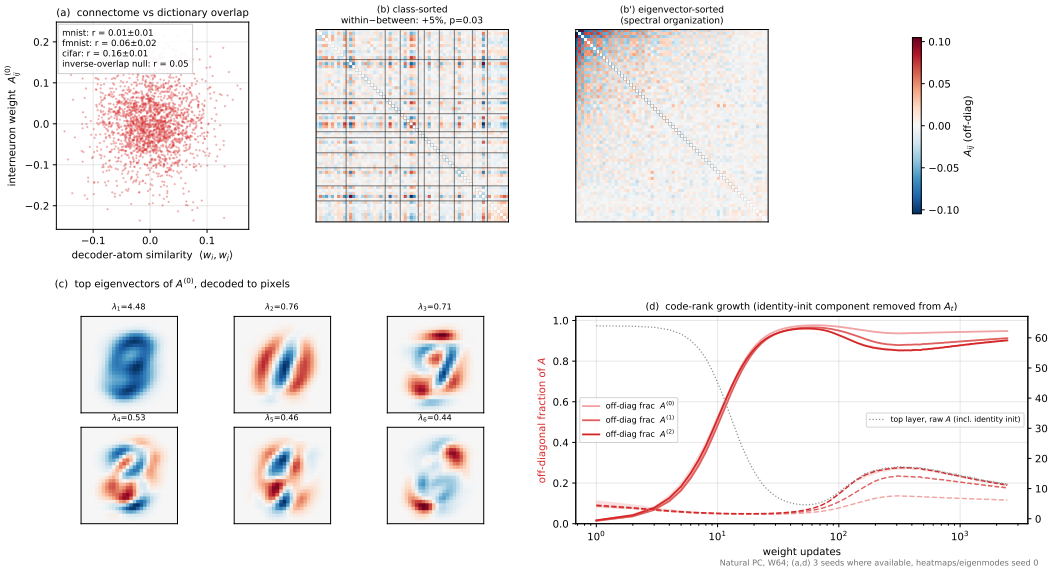


Figure 4: Structure of the learned lateral weights (Natural PC, width 64, MNIST unless stated). (a) Lateral weight $A_{ij}^{(0)}$ against the similarity of the corresponding decoder columns, with correlations for three datasets and the inverse-overlap prediction. (b) Final top-layer A sorted by class, with the within-class excess from a permutation test. (b') The same matrix sorted by loading on the leading eigenvector. (c) The six leading eigenvectors of $A^{(0)}$ decoded to pixel space, with eigenvalues. (d) Off-diagonal fraction of A over training for the three layers, after removing the identity initialization (solid, left axis), and participation ratio (dashed, right axis); the dotted line shows the raw top-layer matrix including the initialization.

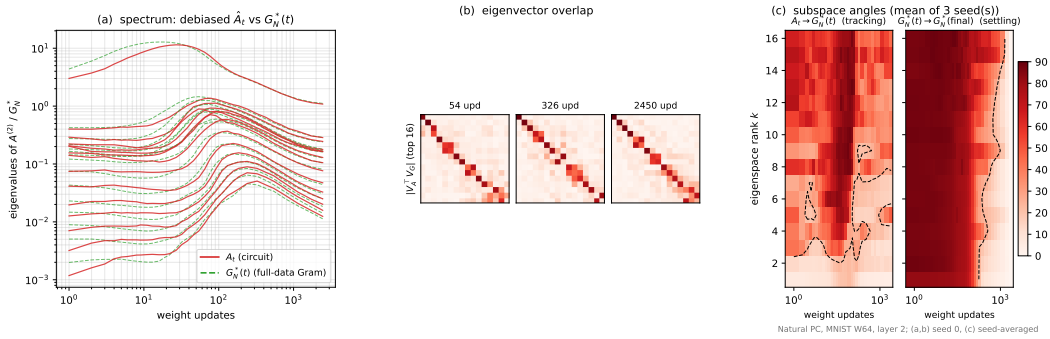


Figure 5: Learning of the equilibrium Gram (Natural PC, MNIST, width 64, top layer). (a) Eigenvalues of the debiased lateral matrix $\hat{A}_t$ (red) and of the full-data equilibrium Gram $G_N^*(t)$ (green, dashed) over training. (b) Absolute overlap between the top sixteen eigenvectors of the two matrices at three times. (c) Principal angles between the top-k eigenspaces, averaged over three seeds. Left: lateral matrix against the Gram at the same time. Right: Gram at time t against the final Gram. Dashed contours mark 45 degrees.

N.3 The speedup is caused by the lateral circuit acting online

Does Natural PC converge faster because the interneurons shape every weight update, or because their early influence puts the weights on a better path? To separate the two, we silence the interneurons between updates 100 and 600 while keeping the lateral weights frozen at their values from update 100, so that the weight update falls back to the standard PC rule (Fig. 6). The applied update changes geometry instantly. Misalignment with the natural gradient jumps from about 0.2 to about 0.65 at the

onset of the lesion and decays only slowly within the window (Fig. 6b). The loss responds in the same way. Inside the lesion window, the training error decreases at the rate of standard PC, which is visibly slower than the unlesioned run (Fig. 6c). When the interneurons are restored at update 600, the update realigns with the natural gradient within a few tens of updates, although the lateral weights are 500 updates out of date, and the loss drops to the unlesioned trajectory. In weight space the lesioned run leaves the unlesioned path during the window and returns to the same minimum afterwards (Fig. 6a). The acceleration is therefore produced by the interneurons at the time of each update, and the stored correlations remain useful over hundreds of updates.

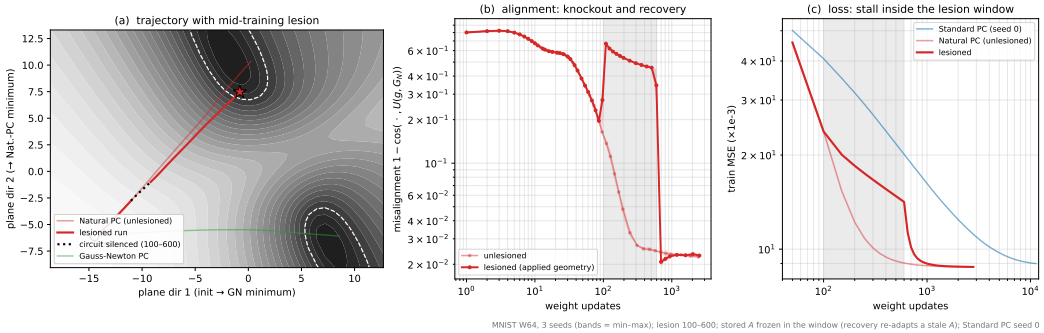


Figure 6: Mid-training lesion of the interneurons (Natural PC, MNIST, width 64, three seeds; bands show the range across seeds). The interneurons are silenced between updates 100 and 600 (shaded), with the lateral weights frozen. (a) Trajectories over the equilibrium loss in the plane spanned by the initialization, the Gauss-Newton PC minimum and the Natural PC minimum; the silenced segment is dotted. (b) Misalignment of the applied update with the full-data natural gradient for the lesioned and unlesioned runs. (c) Training error of the lesioned run compared with unlesioned Natural PC and with standard PC.