CLOSED-FORM PREDICTIVE CODING VIA HIERARCHICAL GAUSSIAN FILTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Predictive coding (PC) offers a local and biologically grounded alternative to backpropagation in the training of artificial neural networks, yet to date, it remains slower, and performance degrades sharply as network depth increases. We trace both problems to a single simplification: current PC networks fix the precision matrix to the identity, discarding precision-weighted prediction errors that the variational derivation requires to be fast, local, and Bayesian. We close this gap by expressing predictive coding networks as deep hierarchical Gaussian filters (HGFs) and restore precision-weighted message passing, yielding dynamic uncertainty estimates and Hebbian-compatible update rules at every layer. The resulting networks can simultaneously learn activations, weights, and precisions under a single free-energy objective, with no global error signal, and resolve inference without requiring iterations or automatic differentiation. On FashionMNIST, our solution approaches backpropagation in epoch-level wall-clock cost while converging in fewer epochs, and competes with it on online, data efficiency, and concept-drift tasks. Beyond simple fully connected networks, it also matches backpropagation's performance on convolutional architectures at various depths on CIFAR-10/100. We thus establish that closed-form variational inference with online precision learning provides a tractable foundation for deep, architecture-agnostic predictive coding networks without requiring iterative relaxation or global error signals.

1 INTRODUCTION

Backpropagation (BP) underwrites modern deep learning. Yet its core assumptions, like a global error signal and strict separation of inference and learning phases Rumelhart et al. (1986), are incompatible with both current knowledge of cortical computations (Whittington & Bogacz, 2019) and introduce interference (Song et al., 2024). Predictive coding (PC) (Rao & Ballard, 1999; Friston, 2005; Friston & Kiebel, 2009) offers a biologically grounded alternative in which each layer carries explicit prediction-error units, inference and learning both proceed by local energy minimisation, and weight updates are Hebbian (Salvatori et al., 2026; van Zwol et al., 2026). Under appropriate conditions, PC recovers the gradients computed by BP through purely local message passing (Whittington & Bogacz, 2017; Millidge et al., 2022; Song et al., 2020), making it a leading candidate for the next generation of energy-efficient, brain-inspired algorithms.

Scaling PC to modern deep architectures (Pinchetti et al., 2025; Goemaere et al., 2025) has nonetheless been obstructed by two practical limitations. For T iterations, PC is $O(T)$ times slower than BP on conventional sequential hardware, because activations must be relaxed toward equilibrium by iterative gradient descent, and performance degrades sharply with depth (Pinchetti et al., 2025). Solutions have been proposed to circumvent this problem (Qi et al., 2025; Innocenti et al., 2025; Goemaere et al., 2025), but they still require automatic differentiation or global signal propagation.

We trace both limitations to a single simplification: deep PC implementations almost universally fix the precision matrix to the identity and infer activations by iterative gradient descent. This discards the precision-weighted prediction errors that the variational derivation of PC requires (Friston & Kiebel, 2009), and with them the property that makes PC most distinctive as an algorithm, namely that in the linear case, learning under precision-weighted errors is formally equivalent to natural-gradient descent (see Millidge et al. (2021) Appendix B). Although learnable precisions have been

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.

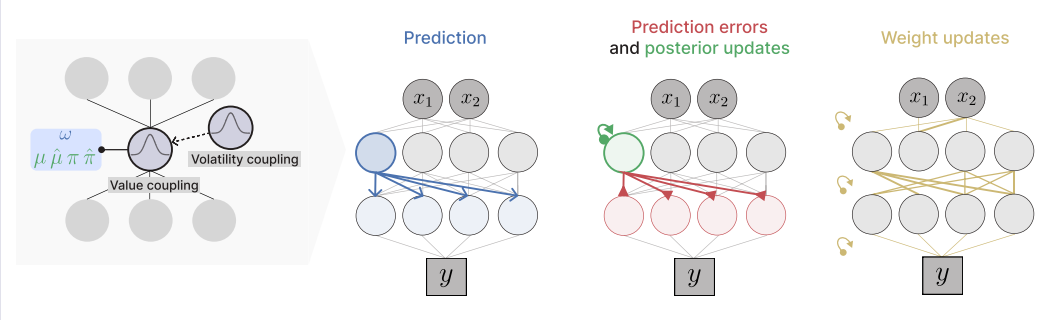

Figure 1: Predictive coding networks as deep hierarchical Gaussian filters. We express predictive coding networks as deep hierarchical Gaussian filters (HGF) where layers are connected through non-linear (here ReLU or leaky-ReLU) value coupling. An optional volatility parent can track downstream uncertainty for precision learning (left panel). While predictive coding networks rely on iterative gradient descent to minimise the local energy function, HGFs minimise the variational free energy in a single update sweep (right panel). (1) After clamping leaf nodes to predictor values, top-down predictions propagate expectations downstream, both for expected precision ($\hat{\pi}_\ell$) and expected mean ($\hat{\mu}_\ell$). (2) Bottom-up prediction errors and one-shot variational update of latent activations. (3) After the full update of latent activations, weights are adapted using a precision-weighted Hebbian update.

widely acknowledged as a promising direction (Salvatori et al., 2026), no existing deep PC network endogenously infers them as posterior beliefs coupled to the rest of the hierarchy.

We close this gap by expressing predictive coding networks as generalised hierarchical Gaussian filters (gHGF) (Mathys et al., 2011; 2014; 2026; Weber et al., 2026). The gHGF is a special case of generalised filtering, of which PC is the neurobiologically motivated instance (Friston & Kiebel, 2009), and its modular message-passing structure yields update equations that are derived analytically, are local by construction, require no automatic-differentiation scaffolding, and open the expressivity of network structures and dynamics that can be supported (Legrand et al., 2026). Inference therefore reduces to a one-shot, closed-form variational update, eliminating the iterative relaxation that obstructs both wall-clock efficiency and asynchronous deployment, and yields fast, Hebbian-compatible rules that simultaneously learn activations, weights, and precisions under a single free-energy objective.

Our contributions are:

1. A reformulation of PC networks as deep hierarchical Gaussian filters that replaces iterative state inference with a single closed-form variational update;
2. BP-level accuracy that is robust to depth and architectural scale, from deep fully connected networks to convolutional networks, at a cost per epoch close to BP's;
3. Two Hebbian-compatible weight updates that use posterior precision: a precision-weighted rule that follows the BP gradient, and a standard rule in which precision sets the plasticity of each layer;
4. Tonic volatility as a single interpretable parameter that trades plasticity, for learning from few samples, against stability, for adapting to concept drift.

Together, locality, single-step updates, and precision-weighted Hebbian plasticity provide a framework for building deep predictive coding networks without iterative relaxation or automatic differentiation.

2 Background and Related Works

2.1 Predictive Coding Networks

Predictive coding is a family of algorithms that minimise variational free energy using different approximations and optimisation dynamics (Salvatori et al., 2026; van Zwol et al., 2026). Friston (2005) recast Rao & Ballard's [1999] original predictive coding scheme as an approximate inversion

of a hierarchical Gaussian generative model under the Laplace approximation, exposing PC as a special case of variational inference in which the free energy reduces to a sum of layer-wise, precision-weighted squared prediction errors, where belief updates correspond to gradient descent on this energy (Bogacz, 2017). Both inference and learning therefore reduce to local, Hebbian-like updates emerging from the minimisation of a single objective (i.e., "inference learning" (van Zwol et al., 2026)). Whittington & Bogacz (2017) showed that under appropriate weight-balance conditions, inference learning approximates backpropagation in feedforward networks, a result subsequently extended to arbitrary computation graphs (Millidge et al., 2022) and to networks with arbitrary topology, including recurrent, lateral, and skip connections (Salvatori et al., 2022). When activations are instead permitted to relax fully before plasticity, PC follows the distinct principle of *prospective configuration* (Song et al., 2024), which empirically outperforms backpropagation on continual learning, online learning, and small-data regimes.

The framework has accordingly diversified into a family of variants targeting different applications and weaknesses: incremental schemes that reduce inference cost (Salvatori et al., 2024), prevent signal decay in deep networks (Goemaere et al., 2025; Innocenti et al., 2025), bidirectional and hybrid models unifying generative and discriminative pathways (Oliviers et al., 2025; Tscshantz et al., 2023), PC beyond the Gaussian assumption (Pinchetti et al., 2022), and associative memory architectures (Salvatori et al., 2021). Empirically, PC networks have been applied to image classification, generative modelling, and continual learning, with open-source libraries now providing standardised implementations (Innocenti et al., 2024; Pinchetti et al., 2025). Despite this breadth, these applications remain largely similar in that inference proceeds via continuous-time relaxation rather than closed-form one-step updates—therefore requiring automatic differentiation, multiplying the cost of every update by the number of inference steps, and degrading as networks deepen (Pinchetti et al., 2025)—and with precisions typically treated as fixed hyperparameters rather than as belief states updated online (Table 5).

2.2 Hierarchical Gaussian filters

The hierarchical Gaussian filter (HGF) is a Bayesian model of perception and learning (Mathys et al., 2011; 2014; 2026) that casts an agent as inverting a generative model of the world in which hidden states evolve as coupled random walks. It is especially suited to inferring volatility coupling from experimental paradigms (Sandhu et al., 2023), whereby the value at one level sets the log-variance (i.e., step size) of the random walk one level below. That way, the subjective estimates of environmental volatility directly modulate the precision afforded to lower-level beliefs and therefore the rate at which they are updated. Approximate variational inversion under a mean-field assumption and a quadratic expansion of the variational energy yields closed-form, one-step update equations in which posterior means and precisions are revised in proportion to precision-weighted prediction errors propagated from below. A recent generalization (Weber et al., 2026) and implementation (Legrand et al., 2026) extend this scheme to arbitrary networks of exponential-family distributions with hierarchical, state-dependent value and volatility coupling.

This combination of analytic tractability, biologically intuitive prediction-error semantics, and individually fittable parameters has made the framework a reference in cognitive neuroscience (Lawson et al., 2017; Powers et al., 2017; Mikus et al., 2025), with open-source implementations such as PyHGF (Legrand et al., 2026) distributed via TAPAS (Frässle et al., 2021). While HGF sits within the broader family of hierarchical Bayesian filtering schemes, of which predictive coding (PC) is the most prominent neurobiologically motivated instance (Rao & Ballard, 1999; Friston, 2005; Bogacz, 2017), it differs from classic PC approaches even without the volatility coupling it is best known for. Classic PC propagates point estimates through value coupling (Whittington & Bogacz, 2017; Millidge et al., 2022; Song et al., 2024), whereas the HGF tracks a posterior precision alongside every mean: in value coupling, a parent's precision is incremented by its child's predicted precision (Weber et al., 2026). While the generalised version allows for deep value-coupled networks, no previous work has implemented generalised HGF networks for complex discrimination tasks.

2.3 Precision and volatility learning in predictive coding

Precision (i.e., the inverse covariance of prediction errors) occupies a theoretically central position in predictive coding. The variational free energy decomposes into a sum of layer-wise, precision-

weighted squared prediction errors, and modulating precisions amounts to dynamically weighting the influence of each error signal on belief updates and learning (Friston, 2005; Bogacz, 2017). In the linear case, this weighting is formally equivalent to natural-gradient descent, with precisions playing the role of the Fisher information matrix and yielding adaptive, curvature-aware learning rates ((Millidge et al., 2021), Appendix B). Without it, layers cannot adaptively scale the influence of their errors, the cross-layer energy distribution becomes exponentially imbalanced Pinchetti et al. (2025); Qi et al. (2025), and the network forfeits any principled handle on uncertainty.

The practical role of precision in deep PC has nonetheless remained limited. The large majority of PCNs developed for machine-learning tasks (Whittington & Bogacz, 2017; Salvatori et al., 2021; 2024; Millidge et al., 2022; Pinchetti et al., 2025; Oliviers et al., 2025; Song et al., 2024) fix precisions to the identity throughout training, effectively assuming homoscedastic noise and reducing the free energy to a sum of unweighted squared errors. Where precision is treated as learnable, it is typically estimated as a slowly varying parameter in the optimiser Ofner & Stober (2021), at the layer level Ofner et al. (2021) or over weights Tschantz et al. (2025); Bonnet et al. (2025) rather than as a fast-changing belief state that is itself the target of hierarchical inference. It is acknowledged that precision-weighted Hebbian plasticity is a promising new research direction (Salvatori et al., 2026). To our knowledge, no existing deep PC network endogenously infers precisions online as posterior beliefs coupled to the rest of the hierarchy, and no systematic empirical comparison has been made between networks with fixed identity precisions and networks whose precisions are themselves the output of a hierarchical inference scheme.

3 METHOD

3.1 HIERARCHICAL GAUSSIAN FILTERS AS PREDICTIVE CODING NETWORKS

The computational models reported here are based on the generalised hierarchical Gaussian filter (gHGF) (Weber et al., 2026), an extension of the Hierarchical Gaussian Filter (HGF) (Mathys et al., 2011; 2014; 2026) to predictive coding networks that allows designing arbitrarily sized and connected networks. The gHGF tracks the posterior up to second order, updating both the mean μ and the precision π of every belief at each step, whereas standard PC schemes optimize only the first moment and treat precisions as fixed to a standard value like $\sigma = 1.0$ or slowly learned hyperparameters. This architectural difference sets HGFs apart, as precision is not only learned and inferred but also propagates dynamically in the hierarchy as a function of the energy gradient. This also creates the possibility of having activation precision as a weight on the weight updates themselves, where standard versions only use prediction errors.

In a value-coupled hierarchy, as used here, the predicted precision of each node is bounded by its tonic volatility ω_a,

$$\hat{\pi}_a = \frac{1}{\frac{1}{\pi_a} + \exp(\omega_a)},$$

and posterior precisions accumulate from bottom-up evidence towards a fixed point set by the network structure and ω. The gHGF also supports volatility coupling, in which a parent's mean adds to ω_a so that prediction errors can modulate precision (Weber et al., 2026); we do not use this feature in the current work.

Value couplings alone are sufficient to recast the gHGF as a deep network closely analogous to a PC network: stacking layers of continuous state nodes connected by nonlinear value couplings $g(\cdot)$ (e.g., ReLU or leaky ReLU as used here). The framework deviates from other PC setups in the three-step routine of predictions, prediction errors, and posterior updates (Fig. 1).

3.2 GENERATIVE MODEL

We treat a deep network as a hierarchy of belief nodes whose joint distribution forms a generative model, which is inverted online via approximate variational inference. Following the gHGF formulation of Weber et al. (2026), and dropping drift ($\rho_\ell = 0$) and autoconnection ($\lambda_\ell = 0$) since classification samples carry no temporal continuity in the mean, each continuous state node $\ell \in \{1, \ldots, L\}$ is generated as

$$x_\ell^{(k)} \sim \mathcal{N}\left(\sum_{b \in \mathrm{vapa}(\ell)} \alpha_{b,\ell}\, g\left(x_b^{(k)}\right),\ \exp\left(x_{\check{\ell}}^{(k)} + \omega_\ell\right)\right) \tag{1}$$

where $\mathrm{vapa}(\ell)$ denotes the set of value parents of node ℓ, $\alpha_{b,\ell}$ is the value-coupling strength from parent b to node ℓ, $\check{\ell}$ is an optional volatility parent, $g(\cdot)$ is a (leaky) ReLU nonlinearity, and ω_ℓ is a tonic log-volatility parameter. When the volatility parent is absent, the variance reduces to $\exp(\omega_\ell)$, and the node behaves as a static Gaussian variable centered on its predicted mean.

Continuous input nodes at the top of the hierarchy are clamped to the data point $\mathbf{x}$. At the bottom, outcomes are represented by binary state nodes, clamped to the one-hot class label $\mathbf{y}$ during training. A binary state node generates a prediction

$$\hat{\mu}_{\mathrm{bin}}^{(k)} = \sigma\left(\hat{\mu}_{pa}^{(k)}\right), \qquad \hat{\pi}_{\mathrm{bin}}^{(k)} = \frac{1}{\hat{\mu}_{\mathrm{bin}}^{(k)}\left(1 - \hat{\mu}_{\mathrm{bin}}^{(k)}\right)}, \tag{2}$$

where σ is the logistic sigmoid and pa indexes the binary node's continuous value parent. The corresponding surprise reduces to a binary cross-entropy and serves as the network's discriminative loss.

The architectural distinction of the gHGF is therefore that prediction precisions $\hat{\pi}_\ell$ are first-class belief states inferred online, rather than identity-clamped hyperparameters. Table 3 summarises the mapping. This opens new possibilities for the inference step that is central to most PC algorithms. In iterative inference learning, posterior means are obtained by gradient descent $\dot{\mu}_\ell = -\partial F/\partial \mu_\ell$ run for $T \approx 8$–128 steps until equilibrium (Song et al., 2024; Whittington & Bogacz, 2017; Salvatori et al., 2024; Innocenti et al., 2024). The quadratic expansion underlying Eqs. (8)–(9) in the gHGF instead admits an analytic stationary point in a single sweep. We provide the exact equations for prediction, prediction errors, and posterior updates in A.1 as described in Weber et al. (2026).

3.3 Hebbian weight updates

Once activations and precisions are updated, the value-coupling strengths $\alpha_{b,\ell}$ (which play the role of synaptic weights; Table 3) are revised by local Hebbian rules. We compare three rules of increasing dependence on inferred precision:

$$\alpha^* = \alpha + \eta\, \delta_\ell\, g\left(\mu_{\ell+1}\right), \tag{3}$$

$$\alpha^* = \alpha + \eta\, \delta_\ell\, g\left(\mu_{\ell+1}\right) \cdot \pi_\ell \tag{4}$$

$$\tag{5}$$

Eq. (3) is the standard rule used in predictive coding networks with fixed identity precision (Whittington & Bogacz, 2017; Song et al., 2024). Eq. (4) is the same Hebbian rule with a multiplicative precision factor that furnishes a learning rate per connection. We use Eq. (4) in §4.1, where the precision factor restores the full prediction error so that the update follows the BP gradient, and Eq. (3) in §5, where each hidden-layer update is divided by the posterior precision, so that the tonic volatility sets how plastic the hidden layers are.

Network structures. All the networks tested are sharing the prediction–update–PE routine described in §A.1. Samples are processed sequentially; μ_ℓ are given by the predictors, and π_ℓ is inherited via successive bottom-up posterior updates. Inherited precision competes with tonic volatility ω_ℓ at the prediction step (Eq. 7), but updates remain prediction-error-independent. The natural consequence is that precision increase as the network is exposed to new inputs carrying their intrinsic evidence. The tonic volatility express how much new variance should be nonetheless injected at each prediction step.

3.4 IMPLEMENTATION

All models compared in §4.1 were implemented in JAX (Bradbury et al., 2018) to ensure a fair comparison across frameworks. Predictive coding networks were implemented using JPC (Innocenti et al., 2024). Deep neural networks trained by backpropagation were implemented in Equinox v0.13.7 (Kidger & Garcia, 2021). Deep hierarchical Gaussian filters were implemented in PyHGF (Legrand et al., 2026) v0.3.2, using the vectorised JAX backend.

4 PERFORMANCE WITHOUT ITERATIVE INFERENCE

We evaluate deep hierarchical Gaussian filters (HGF) on FashionMNIST (Xiao et al., 2017). We restrict comparison to baselines that preserve PC's core commitments: local computation, a single free-energy objective, and no global error signal. We accordingly benchmark against three baselines: multilayer perceptrons trained with backpropagation (MLP); the canonical predictive coding network (PCN) (Whittington & Bogacz, 2017); and μPC (Innocenti et al., 2025), a state-of-the-art PC network designed to train deep networks. μPC is run both with $T = H$ inference steps, where H is the number of hidden layers, as in its original configuration, and with a single inference step ($T = 1$), which shows what iterative inference contributes to a PC network. The HGF performs a single, closed-form inference step per update.

Shared architecture. All models use a fully-connected architecture with leaky ReLU activations (negative slope 0.01) and bias terms at every layer. We sweep hidden depth $d \in \{2, 8\}$ and width $w \in \{32, 64, 128\}$ to assess how each method scales with model capacity. Weights are initialised with He initialisation, and performance is reported as the oracle-best learning rate (the LR maximising mean test accuracy across seeds), averaged over 3 random seeds. μPC is the exception to this shared architecture: it uses the architecture of its reference implementation in JPC (Innocenti et al., 2024), with skip connections, no biases, and Adam for the weights, and its learning rate is selected by the same oracle criterion.

Network configurations. All models minimise softmax cross-entropy with batch size 64. The MLP is optimised with Adam over learning rates $\{10^{-2}, 10^{-3}, 10^{-4}\}$. The PCN performs $T = 20$ inference steps per sample using SGD on activities (learning rate 0.1), followed by an Adam weight update with the same learning rate sweep as the MLP. The HGF has a categorical output layer and uses the precision-weighted rule (Eq. 4) with plain SGD. Tonic volatility is set to $\omega = -10$ and learning rates are swept over $\{0.01, 0.03, 0.1, 0.3\}$.

4.1 DIRECT COMPARISON

All methods are trained on the full FashionMNIST training set (60,000 samples) for 50 epochs and evaluated on the 10,000-sample test set. Figure 2 shows test accuracy learning curves across all depth-width configurations, and Table 4 reports oracle-best accuracy for each setting. At depth 2, μPC with $T = H$ and the HGF perform on par. At depth 8, the canonical PCN degrades sharply; μPC recovers much of this gap but remains below the HGF, which is 1.6–1.9 points below the MLP. Restricted to a single inference step ($T = 1$), μPC is 6–13 points below the HGF in every configuration. With a single inference step, an HGF epoch also costs about as much as a BP epoch, and an order of magnitude less than the PCN's (Appendix A.5).

4.2 CONVOLUTIONAL NETWORKS

We train VGG5, VGG7 and VGG9 (Simonyan & Zisserman, 2015) on CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) with HGF, BP and canonical PC. We do not include μPC here, as its implementation does not currently extend to convolutional networks; PC instead uses the PCX implementation (Pinchetti et al., 2025) (Appendix A.4), which extends PC networks to convolutional architectures. The HGF matches backprop at every depth, while PC falls further behind both as depth increases (Table 1).

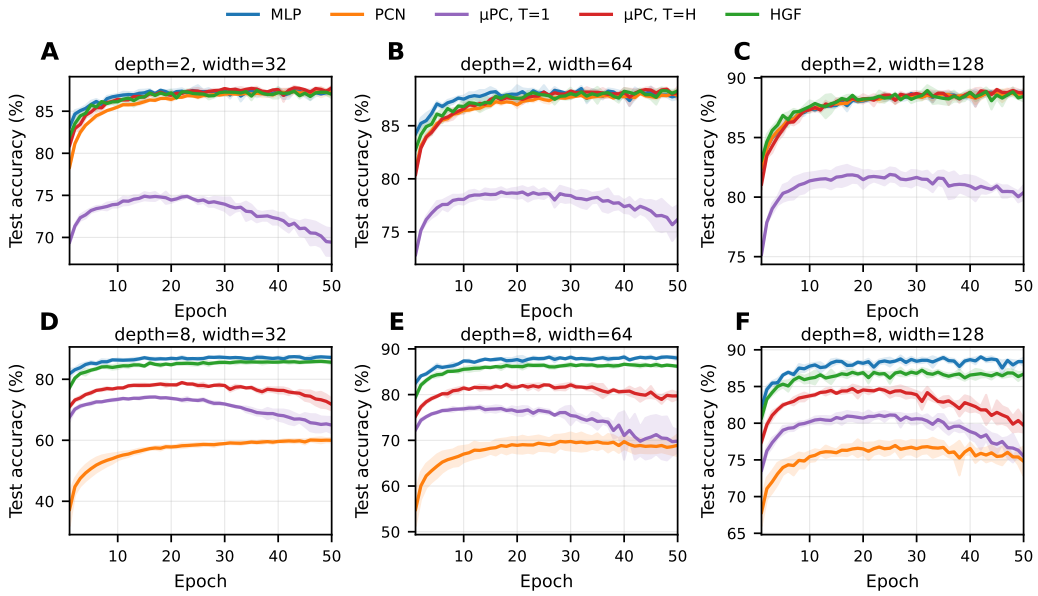

Figure 2: Test accuracy learning curves on FashionMNIST. Rows correspond to hidden depth ($d \in \{2, 8\}$), columns to hidden width ($w \in \{32, 64, 128\}$). μPC is run with $T = 1$ and $T = H$ inference steps, where H is the number of hidden layers. Shaded regions denote ± 1 standard deviation across 3 seeds. Learning rates selected by the oracle criterion per method and configuration.

Table 1: Test accuracy (%) of VGG networks on CIFAR-10 and CIFAR-100. Best test accuracy over 50 epochs; mean and standard deviation over 7 seeds (CIFAR-10) or 3 seeds (CIFAR-100).

% Accuracy	BP	PC	HGF
VGG5			
CIFAR-10	$88.60^{\pm 0.21}$	$86.60^{\pm 0.12}$	$88.54^{\pm 0.20}$
CIFAR-100 (Top-1)	$62.67^{\pm 0.20}$	$60.18^{\pm 0.19}$	$62.47^{\pm 0.19}$
CIFAR-100 (Top-5)	$86.31^{\pm 0.10}$	$85.02^{\pm 0.20}$	$86.33^{\pm 0.18}$
VGG7			
CIFAR-10	$91.78^{\pm 0.19}$	$83.18^{\pm 0.30}$	$91.87^{\pm 0.20}$
CIFAR-100 (Top-1)	$66.50^{\pm 0.34}$	$60.57^{\pm 0.26}$	$66.53^{\pm 0.40}$
CIFAR-100 (Top-5)	$88.48^{\pm 0.15}$	$85.60^{\pm 0.12}$	$88.61^{\pm 0.30}$
VGG9			
CIFAR-10	$92.17^{\pm 0.14}$	$82.98^{\pm 0.15}$	$92.14^{\pm 0.12}$
CIFAR-100 (Top-1)	$65.63^{\pm 0.63}$	$55.97^{\pm 0.42}$	$65.38^{\pm 0.53}$
CIFAR-100 (Top-5)	$88.06^{\pm 0.28}$	$82.01^{\pm 0.18}$	$87.87^{\pm 0.25}$

5 Tonic volatility trades plasticity for stability

We evaluate the HGF on the suite of online and continual learning benchmarks used by Song et al. (2024), against the same baselines. All models use a fixed architecture of two hidden layers of width 32, minimise the squared error on one-hot targets, and train their weights with plain SGD. The HGF uses the standard rule (Eq. 3), under which each weight moves by the prediction error of the node below divided by its posterior precision. The tonic volatility ω, which bounds how high that precision can grow, therefore sets how plastic the hidden layers are, and we sweep it over $\{-10, -8, -6, -4, -2\}$ (Table 2).

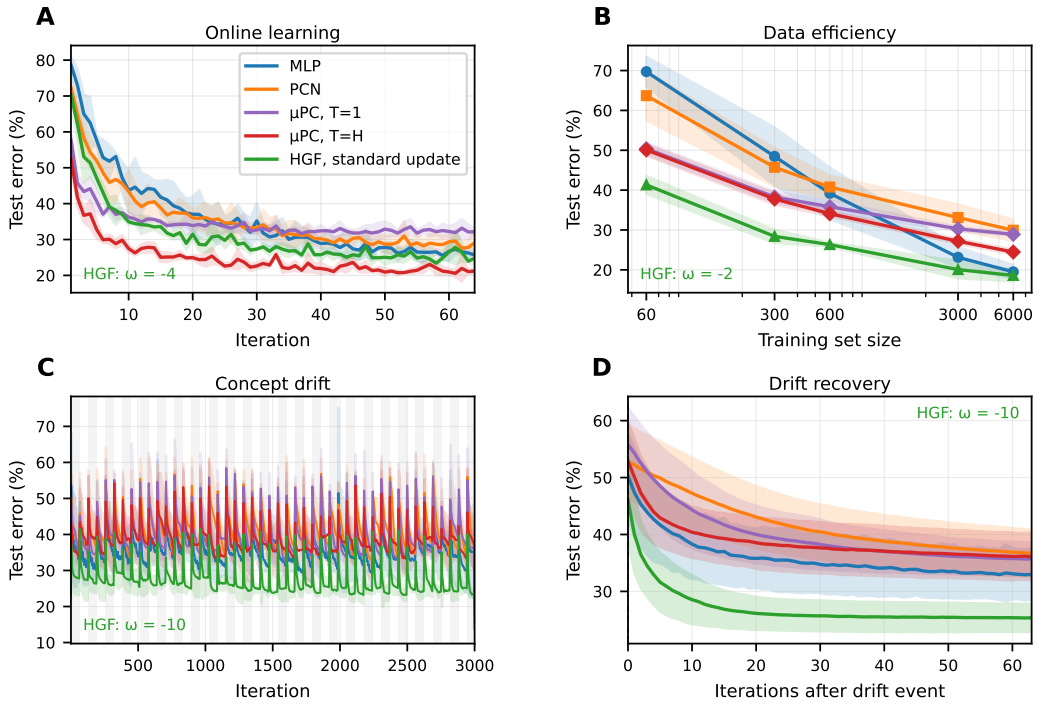

Figure 3: Adaptation and efficiency benchmarks. **(a)** Online learning: test error vs. iteration (batch size 1, 200 samples per iteration). **(b)** Data efficiency: test error vs. training set size (64 epochs per run). **(c)** Concept drift: test error over 3,000 iterations; label mapping for classes 5–9 permuted every 64 iterations (alternating grey bands). **(d)** Drift recovery: mean test error aligned to drift events, averaged over $\sim$47 events and 3 seeds. μPC is run with $T = 1$ and $T = H = 2$ inference steps, the PCN with $T = 10$. The HGF is shown at the tonic volatility with the lowest error in each panel, selected like the learning rates; it learns one sample at a time, while in (b) and (c) the other methods use batches of 60 and 32. All panels use oracle-best learning rate per method; shaded regions denote ± 1 std across seeds.

5.1 Online Learning

We evaluate all methods in a strictly online regime (batch size 1) over 64 iterations, each consisting of 200 randomly drawn FashionMNIST samples. Test error is recorded after every iteration, and learning rates are chosen by the oracle criterion.

Panel (a) of Figure 3 shows test error over iterations. The HGF converges faster and to a lower mean error than the MLP and PCN over the 64 iterations, but remains above μPC with $T = H$.

5.2 Data Efficiency

We assess data efficiency by training all models on subsets of FashionMNIST of sizes $n \in \{60, 300, 600, 3000, 6000\}$, evaluating on the full held-out test set. Each model is trained for 64 epochs per run; oracle-best accuracy across the learning rate sweep is reported.

Panel (b) of Figure 3 shows test error as a function of training set size. HGF achieves lower error than all baselines at small dataset sizes, with the gap narrowing as n increases.

5.3 Concept Drift

We evaluate adaptation to non-stationary environments using a concept drift protocol. Models are first pre-trained online on 6,000 samples for 64 epochs. During the drift phase, 3,000 iterations are

run; every 64 iterations the label mapping for classes 5–9 is randomly permuted, simulating abrupt concept drift, while classes 0–4 retain fixed labels. Each iteration uses 120 training and 1,000 test samples. Precision estimates in the HGF are *not* reset between drift events, so the model must adapt using its accumulated belief state.

Panels (c) and (d) of Figure 3 show the full test-error time series and the mean post-drift recovery curve (averaged over $\sim$47 drift events and 3 seeds), respectively. The HGF maintains a lower steady-state error throughout the drift sequence and recovers more rapidly following each drift event.

Table 2: Test error (%) of the HGF as a function of tonic volatility ω. Online: mean over 64 iterations; data efficiency: $n = 60$; drift: mean over 3,000 iterations. Mean over 3 seeds.

ω	-10	-8	-6	-4	-2
Online	45.39	41.11	34.60	**30.87**	30.98
Data efficiency	56.57	53.85	49.89	47.36	**41.32**
Concept drift	**27.13**	28.60	32.07	35.77	37.88

5.4 Effect of tonic volatility

The tasks favour opposite ends of the ω range (Table 2). Low precision (high ω) lets hidden beliefs move far before the weights change, as in prospective configuration (Song et al., 2024), and is best when much must be learnt from few samples. High precision (low ω) consolidates the hidden layers, so adaptation to drift is carried by the readout.

6 Conclusion

Expressing PC networks as deep HGFs replaces iterative inference with a single closed-form step. The resulting networks reach backprop-level accuracy without the degradation with depth seen in PC, on fully connected networks on FashionMNIST and VGG networks up to VGG9 on CIFAR-10 and CIFAR-100, at a cost per epoch close to BP's (Fig. 4). The tonic volatility provides a single interpretable parameter that moves the network between plastic and stable learning.

AI use statement

Large language models were used to assist with writing code and with drafting, editing and polishing the manuscript. All code and text were reviewed and verified by two authors, who take full responsibility for the content of the paper.

Reproducibility statement

Architectures, training protocols, learning-rate grids, seeds and model selection are described in §4.1–5 and the Appendix, which also details the closed-form update equations (§A.1). All experiments use public datasets (FashionMNIST, CIFAR-10, CIFAR-100). Anonymised code to reproduce all results is provided as supplementary material.

References

Rafal Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *Journal of Mathematical Psychology*, 76:198–211, February 2017. ISSN 0022-2496. doi: 10.1016/j.jmp.2015.11.003. URL http://dx.doi.org/10.1016/j.jmp.2015.11.003.

Djohan Bonnet, Kellian Cottart, Tifenn Hirtzlin, Tarcisius Januel, Thomas Dalgaty, Elisa Vianello, and Damien Querlioz. Bayesian continual learning and forgetting in neural networks. *Nature Communications*, 16(1), October 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-64601-w. URL http://dx.doi.org/10.1038/s41467-025-64601-w.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne,

and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Stefan Frässle, Eduardo A. Aponte, Saskia Bollmann, Kay H. Brodersen, Cao T. Do, Olivia K. Harrison, Samuel J. Harrison, Jakob Heinzle, Sandra Iglesias, Lars Kasper, Ekaterina I. Lomakina, Christoph Mathys, Matthias Müller-Schrader, Inês Pereira, Frederike H. Petzschner, Sudhir Raman, Dario Schöbi, Birte Toussaint, Lilian A. Weber, Yu Yao, and Klaas E. Stephan. Tapas: An open-source software package for translational neuromodeling and computational psychiatry. *Frontiers in Psychiatry*, 12, 2021. ISSN 1664-0640. doi: 10.3389/fpsyt.2021.680811. URL http://dx.doi.org/10.3389/fpsyt.2021.680811.

Karl Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1456):815–836, April 2005. ISSN 1471-2970. doi: 10.1098/rstb.2005.1622. URL http://dx.doi.org/10.1098/rstb.2005.1622.

Karl Friston and Stefan Kiebel. Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1211–1221, May 2009. ISSN 1471-2970. doi: 10.1098/rstb.2008.0300. URL http://dx.doi.org/10.1098/rstb.2008.0300.

Cédric Goemaere, Gaspard Oliviers, Rafal Bogacz, and Thomas Demeester. epc: Fast and deep predictive coding for digital hardware, 2025. URL https://arxiv.org/abs/2505.20137.

Francesco Innocenti, Paul Kinghorn, Will Yun-Farmbrough, Miguel De Llanza Varona, Ryan Singh, and Christopher L. Buckley. Jpc: Flexible inference for predictive coding networks in jax, 2024. URL https://arxiv.org/abs/2412.03676.

Francesco Innocenti, El Mehdi Achour, and Christopher L. Buckley. μpc: Scaling predictive coding to 100+ layer networks, 2025. URL https://arxiv.org/abs/2505.13124.

Patrick Kidger and Cristian Garcia. Equinox: neural networks in jax via callable pytrees and filtered transformations, 2021. URL https://arxiv.org/abs/2111.00254.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

Rebecca P Lawson, Christoph Mathys, and Geraint Rees. Adults with autism overestimate the volatility of the sensory environment. *Nature Neuroscience*, 20(9):1293–1299, 2017. ISSN 1546-1726. doi: 10.1038/nn.4615. URL http://dx.doi.org/10.1038/nn.4615.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. *Difference Target Propagation*, pp. 498–515. Springer International Publishing, 2015. ISBN 9783319235288. doi: 10.1007/978-3-319-23528-8_31. URL http://dx.doi.org/10.1007/978-3-319-23528-8_31.

Nicolas Legrand, Lilian Weber, Peter Thestrup Waade, Anna Hedvig Møller Daugaard, Mojtaba Khodadadi, Nace Mikuš, and Christoph Mathys. pyhgf: A neural network library for predictive coding. *PLOS Computational Biology*, 22(6):e1014340, 2026. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1014340. URL http://dx.doi.org/10.1371/journal.pcbi.1014340.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7(1), November 2016. ISSN 2041-1723. doi: 10.1038/ncomms13276. URL http://dx.doi.org/10.1038/ncomms13276.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

Christoph Mathys, Jean Daunizeau, Karl J Friston, and Klaas E Stephan. A bayesian foundation for individual learning under uncertainty. *Frontiers in Human Neuroscience*, 5, 2011. ISSN 1662-5161. doi: 10.3389/fnhum.2011.00039. URL http://dx.doi.org/10.3389/fnhum.2011.00039.

Christoph Mathys, Nicolas Legrand, Peter Thestrup Waade, Nace Mikus, and Lilian Aline Weber. Robust volatility updates for hierarchical gaussian filtering, 2026. URL https://arxiv.org/abs/2605.00966.

Christoph D. Mathys, Ekaterina I. Lomakina, Jean Daunizeau, Sandra Iglesias, Kay H. Brodersen, Karl J. Friston, and Klaas E. Stephan. Uncertainty in perception and the hierarchical gaussian filter. *Frontiers in Human Neuroscience*, 8, November 2014. ISSN 1662-5161. doi: 10.3389/fnhum.2014.00825. URL http://dx.doi.org/10.3389/fnhum.2014.00825.

Nace Mikus, Claus Lamm, and Christoph Mathys. Computational phenotyping of aberrant belief updating in individuals with schizotypal traits and schizophrenia. *Biological Psychiatry*, 97 (2):188–197, January 2025. ISSN 0006-3223. doi: 10.1016/j.biopsych.2024.08.021. URL http://dx.doi.org/10.1016/j.biopsych.2024.08.021.

Beren Millidge, Anil Seth, and Christopher L Buckley. Predictive coding: a theoretical and experimental review, 2021. URL https://arxiv.org/abs/2107.12979.

Beren Millidge, Alexander Tschantz, and Christopher L. Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, May 2022. ISSN 1530-888X. doi: 10.1162/neco_a_01497. URL http://dx.doi.org/10.1162/neco_a_01497.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks, 2016. URL https://arxiv.org/abs/1609.01596.

Andre Ofner, Raihan Kabir Ratul, Suhita Ghosh, and Sebastian Stober. Predictive coding, precision and natural gradients, 2021. URL https://arxiv.org/abs/2111.06942.

André Ofner and Sebastian Stober. Predprop: Bidirectional stochastic optimization with precision weighted predictive coding, 2021. URL https://arxiv.org/abs/2111.08792.

Gaspard Oliviers, Mufeng Tang, and Rafal Bogacz. Bidirectional predictive coding, 2025. URL https://arxiv.org/abs/2505.23415.

Luca Pinchetti, Tommaso Salvatori, Yordan Yordanov, Beren Millidge, Yuhang Song, and Thomas Lukasiewicz. Predictive coding beyond gaussian distributions, 2022. URL https://arxiv.org/abs/2211.03481.

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 Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 35701–35734, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/581df42e8ebbeeac39aeda03519b7c0e-Paper-Conference.pdf.

A. R. Powers, C. Mathys, and P. R. Corlett. Pavlovian conditioning–induced hallucinations result from overweighting of perceptual priors. *Science*, 357(6351):596–600, August 2017. ISSN 1095-9203. doi: 10.1126/science.aan3458. URL http://dx.doi.org/10.1126/science.aan3458.

Chang Qi, Matteo Forasassi, Thomas Lukasiewicz, and Tommaso Salvatori. Towards the training of deeper predictive coding neural networks, 2025. URL https://arxiv.org/abs/2506.23800.

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, January 1999. ISSN 1546-1726. doi: 10.1038/4580. URL http://dx.doi.org/10.1038/4580.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL http://dx.doi.org/10.1038/323533a0.

Tommaso Salvatori, Yuhang Song, Yujian Hong, Simon Frieder, Lei Sha, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding, 2021. URL https://arxiv.org/abs/2109.08063.

Tommaso Salvatori, Luca Pinchetti, Beren Millidge, Yuhang Song, Tianyi Bao, Rafal Bogacz, and Thomas Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 38232–38244. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/f9f54762cbb4fe4dbffdd4f792c31221-Paper-Conference.pdf.

Tommaso Salvatori, Yuhang Song, Yordan Yordanov, Beren Millidge, Zhenghua Xu, Lei Sha, Cornelius Emde, Rafal Bogacz, and Thomas Lukasiewicz. A stable, fast, and fully automatic learning algorithm for predictive coding networks, 2024. URL https://arxiv.org/abs/2212.00720.

Tommaso Salvatori, Ankur Mali, Christopher L. Buckley, Thomas Lukasiewicz, Rajesh P.N. Rao, Karl Friston, and Alexander Ororbia. A survey on neuro-mimetic deep learning via predictive coding. *Neural Networks*, 195:108161, March 2026. ISSN 0893-6080. doi: 10.1016/j.neunet.2025.108161. URL http://dx.doi.org/10.1016/j.neunet.2025.108161.

Timothy R. Sandhu, Bowen Xiao, and Rebecca P. Lawson. Transdiagnostic computations of uncertainty: towards a new lens on intolerance of uncertainty. *Neuroscience & Biobehavioral Reviews*, 148:105123, May 2023. ISSN 0149-7634. doi: 10.1016/j.neubiorev.2023.105123. URL http://dx.doi.org/10.1016/j.neubiorev.2023.105123.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation, 2016. URL https://arxiv.org/abs/1602.05179.

Jeffrey Seely and Julian Gould. Augmented lagrangian predictive coding, 2026. URL https://arxiv.org/abs/2605.31022.

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 Rafał Bogacz. Can the brain do backpropagation? —exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33:22566 – 22579, 2020. URL https://api.semanticscholar.org/CorpusID:227276519.

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, January 2024. ISSN 1546-1726. doi: 10.1038/s41593-023-01514-1. URL http://dx.doi.org/10.1038/s41593-023-01514-1.

Alexander Tschantz, Magnus Koudahl, Hampus Linander, Lancelot Da Costa, Conor Heins, Jeff Beck, and Christopher Buckley. Bayesian predictive coding, 2025. URL https://arxiv.org/abs/2503.24016.

Alexander Tscshantz, Beren Millidge, Anil K. Seth, and Christopher L. Buckley. Hybrid predictive coding: Inferring, fast and slow. *PLOS Computational Biology*, 19(8):e1011280, August 2023. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1011280. URL http://dx.doi.org/10.1371/journal.pcbi.1011280.

Björn van Zwol, Ro Jefferson, and Egon L. van den Broek. Predictive coding networks and inference learning: Tutorial and survey. *ACM Computing Surveys*, February 2026. ISSN 1557-7341. doi: 10.1145/3797870. URL http://dx.doi.org/10.1145/3797870.

Lilian Aline Weber, Peter Thestrup Waade, Nicolas Legrand, Anna Hedvig Møller, Klaas Enno Stephan, and Christoph Mathys. The generalized hierarchical gaussian filter. March 2026. doi: 10.7554/elife.110174.1. URL http://dx.doi.org/10.7554/eLife.110174.1.

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, May 2017. ISSN 1530-888X. doi: 10.1162/neco_a_00949. URL http://dx.doi.org/10.1162/neco_a_00949.

James C.R. Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23(3):235–250, March 2019. ISSN 1364-6613. doi: 10.1016/j.tics.2018.12.005. URL http://dx.doi.org/10.1016/j.tics.2018.12.005.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747.

A Appendix

A.1 Closed-form variational updates

Prediction step (top-down) Given posterior beliefs from the previous trial, each node generates a new mean and precision:

$$\hat{\mu}_\ell^{(k)} = \sum_{b \in \mathrm{vapa}(\ell)} \alpha_{b,\ell}\, g\left(\mu_b^{(k)}\right), \tag{6}$$

$$\hat{\pi}_\ell^{(k)} = \frac{1}{\frac{1}{\pi_\ell^{(k-1)}} + \exp\left(\mu_{\check{\ell}}^{(k)} + \omega_\ell\right)}. \tag{7}$$

For the batch-independent regime, $\pi_\ell^{(k-1)}$ is fixed between samples. For temporal regimes, $\pi_\ell^{(k-1)}$ is carried over, allowing posterior precision to accumulate across samples.

Posterior updates For value parent b of child a, we adapt Eqs. 50–51 of Weber et al. (2026) by evaluating the derivatives at the prediction $\hat{\mu}_b^{(k)}$ rather than the previous-trial posterior $\mu_b^{(k-1)}$. This reflects the fact that, with $\rho_\ell = \lambda_\ell = 0$, samples are not autoregressive in the mean, and the prediction is the appropriate expansion point:

$$\pi_b^{(k)} = \hat{\pi}_b^{(k)} + \hat{\pi}_a^{(k)} \left(\alpha_{b,a}^2\, g'\left(\hat{\mu}_b^{(k)}\right)^2 - \alpha_{b,a} g''\left(\hat{\mu}_b^{(k)}\right) \delta_a^{(k)} \right), \tag{8}$$

$$\mu_b^{(k)} = \hat{\mu}_b^{(k)} + \frac{\hat{\pi}_a^{(k)}\, \alpha_{b,a}\, g'\left(\hat{\mu}_b^{(k)}\right)}{\pi_b^{(k)}}\, \delta_a^{(k)}. \tag{9}$$

For piecewise-linear g (ReLU, leaky ReLU), $g''(\cdot) = 0$ everywhere, so the precision update (8) collapses to a non-negative term and the mean update (9) becomes a precision-weighted Kalman gain on the prediction error.

Prediction error step Each node sends to its parents from the layer above:

$$\delta_a^{(k)} = \mu_a^{(k)} - \hat{\mu}_a^{(k)}, \qquad \text{together with } \hat{\pi}_a^{(k)}. \tag{10}$$

A.2 Notation and mapping between gHGF and standard PC quantities.

Table 3: Notation and mapping between gHGF and standard PC quantities.

Symbol	gHGF role	PC equivalent
$\mu_\ell, \hat{\mu}_\ell$	posterior / predicted mean of node ℓ	μ_ℓ, $f(W\mu_{\ell+1})$
$\pi_\ell, \hat{\pi}_\ell$	posterior / predicted precision of node ℓ	Σ_ℓ^{-1} (typically fixed)
δ_ℓ	value prediction error (VAPE)	ϵ_ℓ
$\alpha_{b,\ell}$	value-coupling strength parent→child	$W_{\ell+1,\ell}$
ω_ℓ	tonic log-volatility	typically fixed to identity
$g(\cdot)$	nonlinear value coupling	activation function f

A.3 Algorithms

A.4 Convolutional networks: training protocol

All three methods follow the data augmentation and training protocol of PCX (Pinchetti et al., 2025): batch size 128, 50 epochs, AdamW (Loshchilov & Hutter, 2019) with a warmup-cosine schedule,

Algorithm 1 Deep Hierarchical Gaussian Filter — single training step

Input: input x, target y, network $\mathcal{N}$

$\mu_L \leftarrow x, \quad \mu_0 \leftarrow y$
Prediction
for $\ell = L-1$ **down to** 0 **do**
  $\hat{\mu}_\ell \leftarrow \alpha_\ell \, g(\mu_{\ell+1})$
  $\hat{\pi}_\ell \leftarrow \dfrac{1}{\frac{1}{\pi_\ell} + \exp(\omega_\ell)}$
end for
Prediction error
for $\ell = 0$ **to** $L-1$ **do**
  $\delta_\ell \leftarrow \mu_\ell - \hat{\mu}_\ell$
end for
Posterior update
for $\ell = 1$ **to** $L-1$ **do**
  $\pi_\ell \leftarrow \hat{\pi}_\ell + \hat{\pi}_{\ell-1} \, \alpha_{\ell-1}^2 \, g'(\hat{\mu}_\ell)^2$
  $\mu_\ell \leftarrow \hat{\mu}_\ell + \dfrac{\hat{\pi}_{\ell-1} \, \alpha_{\ell-1} \, g'(\hat{\mu}_\ell)}{\pi_\ell} \, \delta_{\ell-1}$
end for
Weight update
for $\ell = 0$ **to** $L-1$ **do**
  $\alpha_\ell \leftarrow \alpha_\ell + \eta \, \pi_\ell \delta_\ell \, g(\mu_{\ell+1})$
end for

and LeCun-uniform initialisation. As in PCX, we report the best test accuracy over training; learning rates are swept on one seed and the best rate is run over 7 (CIFAR-10) or 3 (CIFAR-100) seeds. PC's hyperparameters are PCX's published configurations, except for VGG7 and VGG9 on CIFAR-10, for which PCX provides none; there we ran a random search over PCX's PC search space (activation, inference learning rate and momentum, weight learning rate, weight decay and T) before seeding the best configuration. This search is wider than the learning-rate sweep given to BP and the HGF, and so can only favour PC.

A.5 Computational cost

Figure 4 reports the wall-clock time of one training epoch over the 60,000 FashionMNIST training samples at batch size 64, for the networks of Figure 2. Each method's whole epoch (on-device shuffling, batching and every training step) is compiled as a single JAX program (`lax.scan`), so the timings measure the algorithms rather than Python dispatch; each bar is the mean of 3 timed epochs after 2 warm-up epochs. The HGF is timed with its batched update, as trained in Figure 2. An HGF epoch takes 0.97–1.23 times as long as a BP epoch, and the PCN with $T = 20$ inference steps 8.7–12.1 times as long. μPC requires an older JAX version (0.5.2, against 0.6.2 for the other methods), in which BP itself runs faster, so we compare it only with BP timed in the same environment: μPC takes 1.0–1.2 times as long as BP with $T = 1$, and 1.3–1.5 (depth 2) and 3.3–3.8 (depth 8) times as long with $T = H$. All timings were run on one MIG slice (`1g.23gb`) of an NVIDIA B200 GPU, with the squared-error output layers of each method; at these network sizes each step is latency-bound, so the absolute times mostly reflect launch overhead and the ratios are the meaningful quantity.

A.6 Oracle-best test accuracy on FashionMNIST

Table 4 lists the accuracies behind Figure 2. Learning-rate grids: MLP and PCN $\{10^{-2}, 10^{-3}, 10^{-4}\}$; HGF $\{0.01, 0.03, 0.1, 0.3\}$; μPC weight learning rates $\{0.2, 0.1, 0.01\}$ with activity learning rates $\{1.0, 0.5, 0.1\}$. Where the best rate sat at the edge of a grid, one further step was run (MLP and PCN 10^{-5}, HGF 1.0, μPC 10^{-3}); none of these became the best rate.

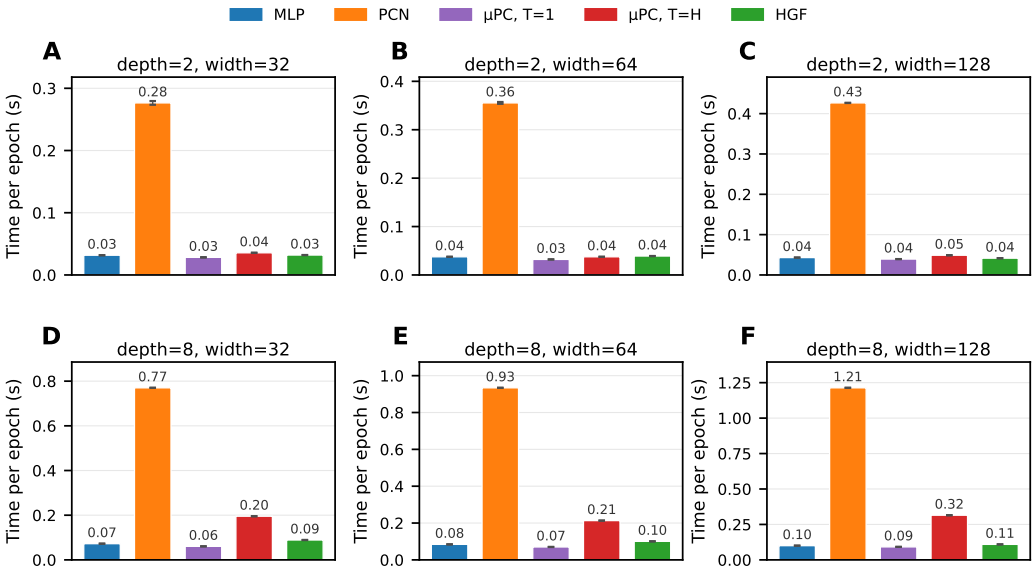

Figure 4: Wall-clock time per training epoch (60,000 FashionMNIST samples, batch size 64) across depth and width configurations. Each epoch is compiled as one program; mean over 3 timed epochs after 2 warm-up epochs, error bars denote ± 1 standard deviation. μPC is timed in its own JAX environment (see text).

Table 4: Oracle-best test accuracy (%) on FashionMNIST (softmax cross-entropy, batch size 64). Best test accuracy over 50 epochs, mean across 3 seeds; best learning rate per (method, depth, width). **Bold** marks the best method per row.

Depth	Width	MLP	PCN	μPC, $T{=}1$	μPC, $T{=}H$	HGF
2	32	87.77	87.46	75.19	**87.91**	87.68
	64	**88.65**	88.16	78.89	88.60	88.59
	128	88.94	88.96	82.10	**89.18**	89.14
8	32	**87.67**	60.29	74.27	78.91	86.04
	64	**88.52**	70.06	77.25	82.42	86.94
	128	**89.12**	77.16	81.36	84.94	87.25

Table 5: Features of deep HGFs and alternative learning rules. Each entry states whether the method, as defined in the cited paper, has the feature (✓) or not. Feature definitions: (1) hidden activity is inferred as a belief by minimising an energy or free energy, rather than set by a forward pass; (2) inference involves no iterative relaxation to equilibrium and no second (nudged) phase; (3) the precision of latent states is not fixed, but learned, scheduled or inferred; (4) precision is a per-node posterior belief updated online with each observation; (5) a distribution over weights is maintained; (6) trains networks with 8 or more layers or convolutional architectures.

Method	Inference		Uncertainty			Scale
	(1) Infers latent states	(2) No relaxation	(3) Adaptive precision	(4) Per-node posterior precision	(5) Weight uncertainty	(6) Deep or conv. nets
Backpropagation and feedback-based credit assignment						
Backpropagation (Rumelhart et al., 1986)		✓				✓
Feedback alignment (Lillicrap et al., 2016)		✓				
Direct feedback alignment (Nøkland, 2016)		✓				✓
Difference target propagation (Lee et al., 2015)		✓				
Energy-based learning						
Equilibrium propagation (Scellier & Bengio, 2016)	✓					
PCN (Whittington & Bogacz, 2017)	✓					
iPC (Salvatori et al., 2024)	✓					✓
μPC (Innocenti et al., 2025)	✓		[a]			✓
ePC (Goemaere et al., 2025)	✓[b]					✓
PC-ALM (Seely & Gould, 2026)	✓					✓
Predictive coding with adaptive precision						
Ofner et al. (2021)	✓		✓			
PredProp (Ofner & Stober, 2021)	✓		✓			
Qi et al. (2025)	✓		✓[c]			✓
Bayesian PC (Tschantz et al., 2025)	✓		✓	[d]	✓	
Bayesian filtering and Bayesian weight learning						
HGF (Mathys et al., 2014; 2011)	✓	✓	✓	✓	[e]	
gHGF (Weber et al., 2026)	✓	✓	✓	✓	[e]	
MESU (Bonnet et al., 2025)		✓			✓	✓
Deep HGF (this work)	✓	✓	✓	✓		✓

[a] Layer-wise scalar precisions are fixed.
[b] Inference runs on error variables, a reparameterisation of the latent states.
[c] Precision follows a hand-set schedule rather than being learned or inferred.
[d] A Wishart posterior over each layer's noise covariance, updated per batch.
[e] Coupling strengths are fixed parameters.