

Predictive Coding versus Backpropagation under Pixel-Noise and Geometric Distribution Shifts: A Controlled Comparison of Reconstruction and Linear-Probe Quality on Fashion-MNIST

Abhi Desai, Advait Chalke, Yung-Yi Tu
University of Southern California
{abhidesa, achalke, yungytu}@usc.edu

Abstract

We investigate predictive coding (PC) as a biologically plausible alternative to backpropagation (BP) for training deep generative models. After deriving the PC inference and learning rules from a Gaussian hierarchical generative model and stating their relationship to BP, we compare a PC-trained convolutional autoencoder against an architecture-matched BP baseline on Fashion-MNIST. We evaluate two qualitatively different forms of test-time distribution shift: *pixel-noise* corruptions (additive Gaussian, salt-and-pepper), which are i.i.d. across pixels and leave the image’s spatial structure intact, and a *geometric* corruption (random translation), which is a coordinated group action on the pixel grid and moves the data onto a different manifold. For each condition we train a separate *denoising-style* autoencoder (corrupted input, clean reconstruction target) and measure two complementary axes of quality: pixel-level reconstruction (PSNR, SSIM) on held-out corrupted inputs, and class discriminability of the frozen 64-dimensional bottleneck via a ridge linear probe. On pixel noise, BP retains a small but consistent reconstruction advantage (SSIM gap 0.020–0.038), while frozen-encoder probe accuracies are within 0.005 of each other. On the geometric shift the reconstruction gap is comparable ($\Delta\text{SSIM} = 0.028$), but the probe gap widens (up to 0.021 on translated test inputs). Both models lose substantially more probe accuracy when the test input is translated than when it is corrupted by pixel noise. We also report a practical finding: PC training is highly sensitive to the synaptic learning rate η_θ , and an order-of-magnitude reduction was necessary to escape a low-SSIM plateau caused by inference–learning timescale interference.

Our results support recent theoretical claims that PC can approximate BP [Whittington and Bogacz, 2017, Millidge et al., 2022b] while suggesting that the engineering cost of stable PC training is non-trivial in practice.

1 Introduction

Backpropagation (BP) is the algorithm for training deep neural networks, but it has long been criticized on grounds of biological implausibility: it requires symmetric forward and backward weights, a global error signal, and a strict separation of inference and learning phases [Lillicrap et al., 2020]. Predictive coding (PC), a framework originating in computational neuroscience [Rao and Ballard, 1999, Friston, 2005], offers an alternative in which learning is driven by local prediction errors propagated between hierarchically organized layers. Recent theoretical work has shown that, under suitable conditions, PC networks can approximate or even exactly implement the BP gradient using only local Hebbian-style updates [Whittington and Bogacz, 2017, Millidge et al., 2022b, Song et al., 2020].

This raises a natural empirical question: *how does PC compare to BP in practice, particularly on tasks where representation quality and in-distribution denoising matter, and how does the answer depend on the kind of corruption the model has to undo?* We address this in the setting of generative modeling, training a convolutional encoder–decoder on Fashion-MNIST [Xiao et al., 2017] as a denoising-style autoencoder [?]: a corrupted image is fed to the encoder, the decoder is asked to reconstruct the underlying clean image, and the same corruption is used at train and test time. Crucially, we evaluate two qualitatively different families of corruption

(pixel noise versus geometric shift, defined in Section 4) that stress different inductive biases. Our contributions are:

1. A self-contained derivation of the supervised PC inference and learning rules from a Gaussian generative model, together with a statement of the conditions under which they recover the BP gradient (Section 3).
2. An empirical comparison of PC and BP under four input conditions grouped by perturbation type (clean; pixel-noise: Gaussian, salt-and-pepper; geometric: random translation) using PSNR/SSIM as the primary reconstruction metrics and a frozen-encoder linear probe as a representation quality metric (Section 5).
3. A practical analysis of the optimization difficulties we encountered with PC, including a learning-rate sensitivity phenomenon we term the *PC training plateau* (Section 6).

2 Background and Related Work

Predictive coding. PC was introduced in computational neuroscience as a model of cortical processing in which each hierarchical level attempts to predict the activity of the level below, and only the unexplained component (the prediction error) is propagated upward [Rao and Ballard, 1999]. Friston later cast PC as a special case of variational inference under the free-energy principle [Friston, 2005, 2009], identifying the energy minimized by PC with a Gaussian-approximation evidence lower bound.

PC as approximate backpropagation. Whittington and Bogacz [2017] showed that in a supervised PC network with the output clamped to a target, the synaptic update at the inference fixed point approximates the BP gradient to first order in the output error. Millidge et al. [2022b] extended this result to arbitrary computation graphs, and Song et al. [2020] showed that an appropriately modified PC algorithm can implement BP *exactly*. These results form the theoretical basis for treating PC as a biologically plausible substitute for BP. See Millidge et al. [2022a] and Salvatori et al. [2023] for recent reviews.

PC and variational autoencoders. Marino [2022] establishes a direct correspondence between PC networks and variational autoencoders [Kingma and Welling, 2014]: both minimize a free-energy functional, both employ iterative inference over latent variables, and both can be trained end-to-end with gradient-based methods. This connection motivates our use of an autoencoder architecture as a natural testbed for PC.

Practical frameworks. We implement the PC model using the PCX library [Pinchetti et al., 2025], which provides JAX-based primitives for PC networks, including the *vode* (value node) abstraction used to track local activities, predictions, and prediction errors. PCX is built on top of JAX [Bradbury et al., 2018]. The BP baseline is implemented in the same JAX ecosystem using Equinox modules and `optax.adamw`, so that the encoder/decoder geometry is bit-for-bit identical between PC and BP and only the learning algorithm differs.

3 Theoretical Framework

We consider a feedforward network with L layers, activities $x_0, x_1, \dots, x_L$, and parameterized predictions $\hat{x}_l = f(x_{l-1}; \theta_l)$ for $l = 1, \dots, L$, where f is a differentiable function (e.g., affine map composed with a nonlinearity). The prediction error at layer l is

$$\epsilon_l := x_l - f(x_{l-1}; \theta_l). \quad (1)$$

3.1 Generative model and free energy

Following Rao and Ballard [1999] and Friston [2005], we posit a hierarchical Gaussian generative model in which each layer’s activity is drawn from a Gaussian centered on the prediction from the layer above, with identity covariance:

$$p(x_l | x_{l-1}; \theta_l) = \mathcal{N}(x_l | f(x_{l-1}; \theta_l), I). \quad (2)$$

Taking the negative log of the joint $p(x_{0:L}; \theta) = \prod_l p(x_l | x_{l-1}; \theta_l)$ and dropping additive constants yields the *free energy*

$$E(x_{0:L}, \theta) = \frac{1}{2} \sum_{l=1}^L \|\epsilon_l\|_2^2. \quad (3)$$

PC training proceeds by minimizing E via a two-timescale gradient descent: a fast inference phase

that updates the activities $\{x_l\}$, followed by a slow learning phase that updates the parameters $\{\theta_l\}$.

3.2 Inference dynamics: derivation of Δx_l

During inference, the input $x_0 = u$ and (in the supervised setting) the output $x_L = y$ are clamped, and the interior activities $x_1, \dots, x_{L-1}$ are updated to minimize E . We compute $\partial E / \partial x_l$ for $l \in \{1, \dots, L-1\}$. The activity x_l appears in exactly two terms of E :

(i) In $\frac{1}{2} \|\epsilon_l\|^2 = \frac{1}{2} \|x_l - f(x_{l-1}; \theta_l)\|^2$, where x_l enters linearly. Differentiating gives

$$\frac{\partial}{\partial x_l} \frac{1}{2} \|\epsilon_l\|^2 = \epsilon_l. \quad (4)$$

(ii) In $\frac{1}{2} \|\epsilon_{l+1}\|^2 = \frac{1}{2} \|x_{l+1} - f(x_l; \theta_{l+1})\|^2$, where x_l enters through the prediction $f(x_l; \theta_{l+1})$. Applying the chain rule,

$$\frac{\partial}{\partial x_l} \frac{1}{2} \|\epsilon_{l+1}\|^2 = - \left(\frac{\partial f(x_l; \theta_{l+1})}{\partial x_l} \right)^\top \epsilon_{l+1}. \quad (5)$$

The minus sign arises because ϵ_{l+1} depends on x_l via $-f(x_l; \theta_{l+1})$.

Summing the two contributions yields

$$\frac{\partial E}{\partial x_l} = \epsilon_l - \left(\frac{\partial f(x_l; \theta_{l+1})}{\partial x_l} \right)^\top \epsilon_{l+1}. \quad (6)$$

Gradient descent on x_l with step size η_x gives the inference update

$$\Delta x_l = -\eta_x \frac{\partial E}{\partial x_l} = \eta_x \left[-\epsilon_l + \left(\frac{\partial f(x_l; \theta_{l+1})}{\partial x_l} \right)^\top \epsilon_{l+1} \right], \quad (7)$$

which is the rule appearing in Algorithm 1. The first term pulls x_l toward the prediction from below; the second pulls x_l in a direction that reduces the error of the prediction it makes about x_{l+1} .

3.3 Synaptic learning: derivation of $\Delta \theta_l$

Once inference has converged to a fixed point $x_{0:L}^*$ where $\partial E / \partial x_l \approx 0$ for all interior l , the parameters are updated by gradient descent on E . Since θ_l appears only in ϵ_l ,

$$\frac{\partial E}{\partial \theta_l} = - \left(\frac{\partial f(x_{l-1}^*; \theta_l)}{\partial \theta_l} \right)^\top \epsilon_l^*, \quad (8)$$

giving the synaptic update

$$\Delta \theta_l = \eta_\theta \left(\frac{\partial f(x_{l-1}^*; \theta_l)}{\partial \theta_l} \right)^\top \epsilon_l^*. \quad (9)$$

This update is *local* in the neuroscientific sense: it depends only on the pre-synaptic activity x_{l-1}^* and the post-synaptic error ϵ_l^* , satisfying the Hebbian principle.

3.4 Relationship to backpropagation

Whittington and Bogacz [2017] prove that in a supervised PC network with $x_L = y$ clamped, the prediction errors at the inference fixed point satisfy a recursive relation: for an interior layer at the fixed point of (6),

$$\epsilon_l^* = \left(\frac{\partial f(x_l^*; \theta_{l+1})}{\partial x_l} \right)^\top \epsilon_{l+1}^*. \quad (10)$$

Equation (10) is structurally identical to the backpropagation recursion for the error signal $\delta_l = (\partial f / \partial x_l)^\top \delta_{l+1}$. Substituting (10) into (9) therefore recovers the BP gradient up to first order in the output error. Millidge et al. [2022b] generalize this equivalence to arbitrary computation graphs, and Song et al. [2020] show that with a small modification to the algorithm (delayed updating), the equivalence becomes exact. Conceptually, PC reframes the BP backward pass as the *equilibrium* of a local message-passing process, eliminating the need for a separate backward phase or a global error signal.

Algorithm 1 is the textbook two-phase formulation. In our implementation, inference uses SGD with Nesterov momentum on the activities $\{x_l\}$ and the synaptic step uses AdamW [?] on $\{\theta_l\}$; we run a fixed number T of inference steps per batch (no early stopping on δ), and perform one synaptic step at the end of each batch (“pure” PC; the iPC schedule of Salvatori et al. [2023] reduces to this with $k = 0$).

4 Methods

4.1 Dataset and input conditions

We use Fashion-MNIST [Xiao et al., 2017], consisting of 28×28 grayscale images across 10 garment categories. Pixel intensities are normalized to $[0, 1]$. We consider four input conditions:

- **Clean:** the original image with no perturbation.

Algorithm 1 Supervised Predictive Coding Optimization

Require: Dataset $\mathcal{D} = \{(u^{(i)}, y^{(i)})\}$, inference rate η_x , synaptic rate η_θ , max inference steps T , tolerance δ .

```

1: Initialize  $\Theta = \{\theta_1, \dots, \theta_L\}$ .
2: for each epoch do
3:   for each  $(u, y) \in \mathcal{D}$  do
4:      $\triangleright$  Boundary clamping & init.
5:      $x_0 \leftarrow u$ 
6:     for  $l = 1$  to  $L - 1$  do
7:        $x_l \leftarrow f(x_{l-1}, \theta_l)$   $\triangleright$  feedforward init
8:     end for
9:      $x_L \leftarrow y$ 
10:     $\triangleright$  Inference (fast dynamics)
11:     $t \leftarrow 0$ 
12:    while  $t < T$  and  $\max_l \|\Delta x_l\| > \delta$  do
13:      for  $l = 1$  to  $L$  do
14:         $\epsilon_l \leftarrow x_l - f(x_{l-1}, \theta_l)$ 
15:      end for
16:      for  $l = 1$  to  $L - 1$  in parallel do
17:         $\Delta x_l \leftarrow \eta_x \left[ -\epsilon_l + \left( \frac{\partial f(x_l, \theta_{l+1})}{\partial x_l} \right)^\top \epsilon_{l+1} \right]$ 
18:         $x_l \leftarrow x_l + \Delta x_l$ 
19:      end for
20:       $t \leftarrow t + 1$ 
21:    end while
22:     $\triangleright$  Synaptic update (slow dynamics)
23:    for  $l = 1$  to  $L$  do
24:       $\Delta \theta_l \leftarrow \eta_\theta \left( \frac{\partial f(x_{l-1}, \theta_l)}{\partial \theta_l} \right)^\top \epsilon_l$ 
25:       $\theta_l \leftarrow \theta_l + \Delta \theta_l$ 
26:    end for
27:  end for
28: end for

```

- **Gaussian noise:** additive i.i.d. $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.2$, followed by clipping to $[0, 1]$.
- **Salt-and-pepper noise:** each pixel independently replaced by 0 or 1 with total probability $p = 0.15$.
- **Random translation:** a per-image integer translation sampled uniformly from $[-4, 4]$ pixels along each axis, with bilinear interpolation and zero fill.

Two regimes of distribution shift. We evaluate two qualitatively different forms of test-time

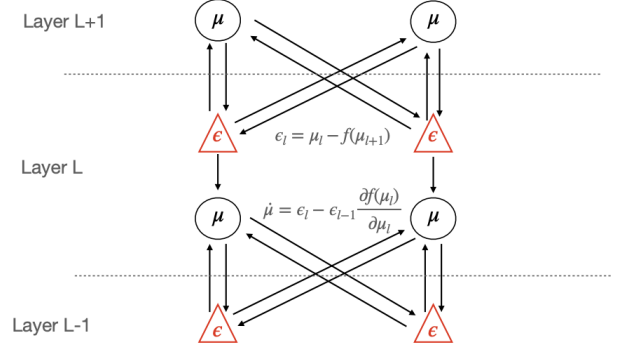


Figure 1: Hierarchical structure of a predictive coding network. Predictions $\hat{x}_l = f(x_{l-1}; \theta_l)$ flow top-down while prediction errors ϵ_l flow bottom-up. Adapted from Marino [2022].

distribution shift. *Pixel noise* (Gaussian, salt-and-pepper) is i.i.d. across pixels and leaves the image’s spatial structure intact; recovery is a denoising problem and the perturbation lies in a bounded ball around the clean manifold. *Geometric shift* (random translation) is a coordinated group action on the pixel grid; the corrupted image lies on a different manifold whose recovery requires the network to implement an (approximate) inverse transformation. The two regimes stress different inductive biases: pixel-noise robustness is a property of the layerwise denoising prior, while geometric robustness is a property of (approximate) equivariance. For this reason we group the conditions accordingly in Tables 1 and 2 and treat the corresponding rows as evaluating different scientific claims.

Training protocol. For each condition we train a *separate* denoising-style autoencoder [?]: the corrupted image is fed to the encoder, and the reconstruction is scored against the underlying clean image under the MSE training objective. Test inputs come from the *same* distribution shift as training; PSNR and SSIM are computed between the reconstruction and the clean ground truth. We therefore measure in-distribution recovery quality under each kind of shift (and, via the linear probe of Section 4.5, the discriminability of the bottleneck under that shift), not zero-shot robustness to unseen corruption.

4.2 Architecture

Both models share the same convolutional autoencoder topology, defined by a single specification shared between the JAX baseline and the PC network, so the channel ladder and spatial sizes are identical.

Encoder. Two convolutional blocks, each consisting of a 3×3 convolution with stride 1 and padding 1 ($1 \rightarrow 32$ and $32 \rightarrow 64$ channels respectively), a ReLU nonlinearity, and a 2×2 max-pool with stride 2. The resulting $7 \times 7 \times 64$ feature map is flattened and projected linearly to a 64-dimensional latent vector.

Decoder. Linear projection from $\mathbb{R}^{64}$ to $7 \times 7 \times 64$, followed by two 2×2 transposed convolutions with stride 2 returning to 28×28 . A sigmoid activation produces the output.

4.3 PC autoencoder as supervised PC with clean target

We frame the PC autoencoder as supervised PC in which the input layer x_0 is clamped to the (possibly corrupted) network input u and the output layer x_L is clamped to the corresponding clean ground-truth image y . The interior activities (encoder output, latent, decoder hidden) are inferred according to Section 3.2. This is analogous to the variational autoencoder formulation of Marino [2022], with the PC inference loop playing the role of amortized variational inference, and to a denoising autoencoder in the BP sense.

Following Pinchetti et al. [2025], the PC network is constructed by attaching a *vode* (value node) to each layer of the architecture; each vode stores its local activity x_l , prediction $\hat{x}_l$, and error ϵ_l . The PCX library handles the inference–learning split and computes the local updates via JAX autodiff.

4.4 Implementation details

The BP baseline is implemented in JAX/Equinox with MSE loss and `optax.adamw` ($\text{lr} = 10^{-3}$). The PC model uses PCX [Pinchetti et al., 2025] on JAX [Bradbury et al., 2018] over the same encoder/decoder modules; we report results with $\eta_\theta = 10^{-4}$ (see Section 6 for why this value matters), pure-PC weight updates ($k = 0$), and $T = 8$ inference steps per batch. Both models are trained for 10 epochs with batch size 128.

4.5 Evaluation metrics

We evaluate each trained model along two complementary axes.

Reconstruction quality. We report mean squared error (MSE), peak signal-to-noise ratio (PSNR), and the structural similarity index (SSIM) [Wang et al., 2004] between the model’s reconstruction and the *clean* target on the held-out test split, with the corrupted test input drawn from the same shift distribution as training. SSIM is the primary metric because it correlates better with perceptual reconstruction quality than pixel-level MSE.

Representation quality (frozen-encoder linear probe). PSNR/SSIM only measure end-to-end pixel fidelity; they say nothing about whether the bottleneck representation is semantically organized. To probe that, we additionally fit a 10-way linear classifier on top of the *frozen* encoder. Concretely, after training the denoising AE under a given condition: (i) we freeze all encoder weights; (ii) we pass the entire Fashion-MNIST training split through the encoder under the same validation-style forward used for PSNR/SSIM (single-pass for the BP baseline; a single `STATUS.INIT`-style forward with no extra PC inference steps for the PC model), collecting the 64-dimensional bottleneck activations as features; (iii) we estimate a multiclass ridge regression $W \in \mathbb{R}^{64 \times 10}$ on those training features with penalty $\lambda = 10^{-3}$; and (iv) we report classification accuracy of $\arg \max(z^\top W)$ on the test split. We report this accuracy under two test regimes: latents extracted from *clean* test inputs (“clean” regime) and latents extracted from *corrupted* test inputs in the same condition as training (“noisy” regime). The probe weights are identical across the two regimes; only the encoder input differs. The linear restriction is deliberate: a strong probe result with a linear head means the relevant class information is already organized linearly in the bottleneck, rather than being recoverable only after a nonlinear classification head; i.e., it is a measure of the encoder’s representation, not of the head.

5 Results

Clean baseline. On clean inputs, the BP baseline attains $\text{SSIM} = 0.869$ and $\text{PSNR} = 21.90$ dB; the PC

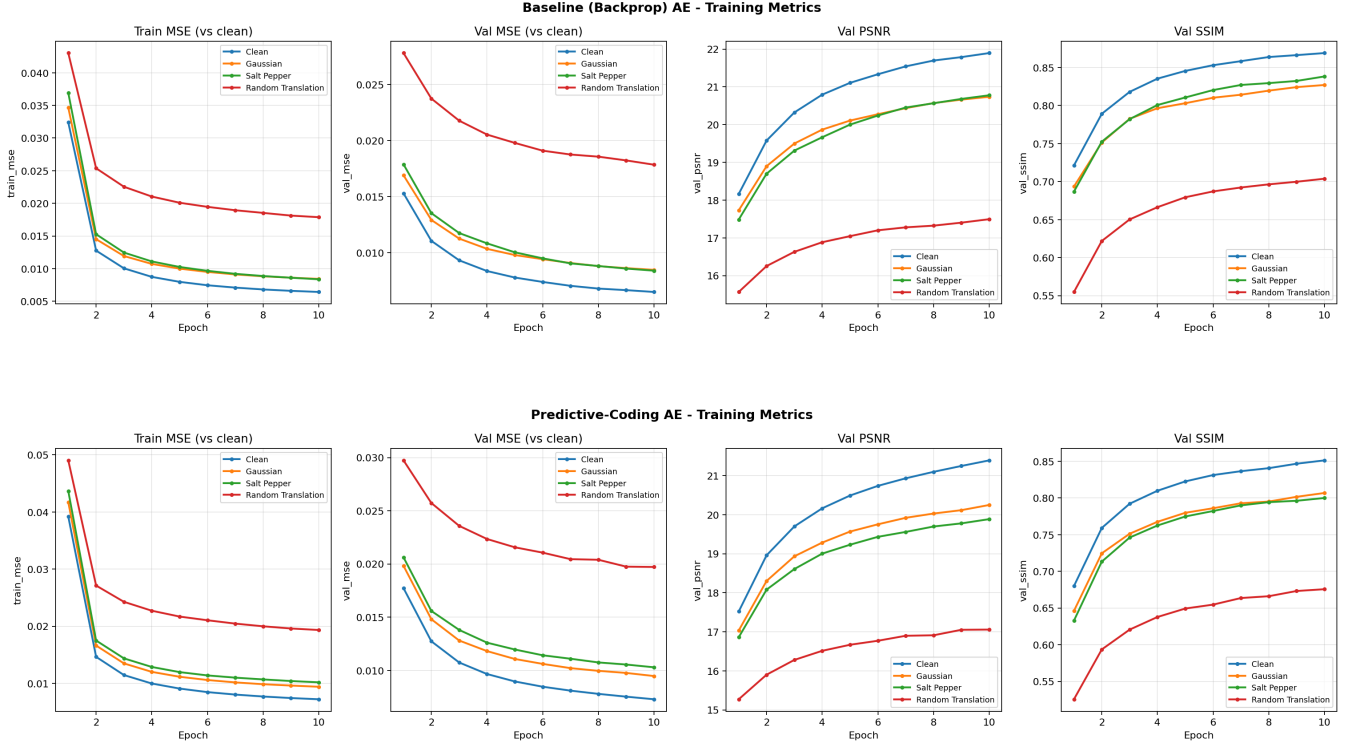


Figure 2: Training curves for the BP baseline (top row) and the PC model (bottom row) under each of the four input conditions (clean, Gaussian, salt-and-pepper, random translation). Columns, left to right: train MSE against the clean target, validation MSE against the clean target, validation PSNR, and validation SSIM. All curves are recorded once per epoch on the held-out test split with the model’s own corruption condition as input.

Table 1: Reconstruction performance on the Fashion-MNIST test split for each model, trained as a denoising-style autoencoder under the corresponding condition (end-of-training values; cf. Figure 2). Rows are grouped by the type of distribution shift.

Condition	Backpropagation		Predictive Coding	
	Val PSNR	Val SSIM	Val PSNR	Val SSIM
<i>No shift</i>				
Clean	21.90	0.869	21.39	0.851
<i>Pixel noise (within-manifold)</i>				
Gaussian	20.73	0.827	20.25	0.807
Salt & Pepper	20.78	0.838	19.88	0.800
<i>Geometric (between-manifold)</i>				
Random translation	17.51	0.704	17.06	0.676

model reaches $\text{SSIM} = 0.851$ and $\text{PSNR} = 21.39$ dB, a gap of 0.018 SSIM and 0.51 dB PSNR. This defines the architecture-and-training-budget ceiling for the comparison; the gap on the corrupted conditions can be read against this floor. Probe accuracy is essentially tied at this point (0.802 for both models).

Pixel noise (within-manifold). Under Gaussian noise ($\sigma = 0.2$), both models degrade by a comparable amount ($\Delta\text{SSIM} \approx 0.04$ relative to clean), and the BP–PC reconstruction gap widens marginally to 0.020 SSIM. Salt-and-pepper noise produces the largest BP advantage at the reconstruction level (0.038 SSIM, 0.90 dB PSNR), consistent with PC’s local error signals being more sensitive to high-magnitude pixel outliers than BP’s globally-optimized weights. In the probe regime the BP advantage is essentially absent: all four pixel-noise probe gaps are within 0.005, and on noisy Gaussian inputs the PC bottleneck is fractionally more class-linear than BP’s (0.791 vs. 0.789). This suggests that the extra pixel fidelity BP extracts on the pixel-noise rows does not correspond to a meaningfully better latent organisation.

Geometric (between-manifold). Random translation is by far the hardest condition for both models: SSIM drops to 0.704 for BP and 0.676 for

Table 2: Frozen-encoder linear probe accuracy (10-way Fashion-MNIST classification) on the test split. “Acc. clean” uses uncorrupted test inputs into the encoder; “Acc. noisy” uses test inputs drawn from the same shift as training. Probe weights are the same in both columns of each model and were fit on training-split latents with ridge penalty $\lambda = 10^{-3}$. Rows are grouped by shift type.

Condition	Backpropagation		Predictive Coding	
	Acc. clean	Acc. noisy	Acc. clean	Acc. noisy
<i>No shift</i>				
Clean	0.802	0.802	0.802	0.802
<i>Pixel noise (within-manifold)</i>				
Gaussian	0.793	0.789	0.789	0.791
Salt & Pepper	0.805	0.800	0.800	0.796
<i>Geometric (between-manifold)</i>				
Random translation	0.781	0.761	0.767	0.740

PC, and PSNR drops by ~ 4 dB compared to clean. This reflects the structural (rather than pixelwise) nature of the corruption: the decoder must invert a spatially varying transform rather than merely smooth out an i.i.d. perturbation, and neither convolutional decoder has built-in machinery to “translate the latent” in a content-aware way. The reconstruction gap between BP and PC (0.028 SSIM) is comparable to the pixel-noise gap, but the probe gap is the largest in our table: 0.014 on clean test inputs and 0.021 on translated test inputs, both favouring BP. Figure 2 shows the corresponding training trajectories; both models follow nearly identical curve shapes within each condition, with BP retaining a small lead at every epoch.

Latent structure. Across the four conditions, probe accuracies in Table 2 are in the 0.74–0.81 range, far above the 1/10 chance level, indicating that the encoder learns a class-discriminative representation despite being optimised purely for reconstruction. The clean-vs-noisy probe drop *within* a given model measures how much class information the encoder loses when the test input is corrupted while keeping the probe fixed: for pixel-noise corruptions the drop is small for both models (≤ 0.005 in absolute value), while for random translation it is much larger (-0.020 for BP, -0.026 for PC), suggesting that absolute spatial position is partially encoded by the bottleneck and is eroded by geometric shift.

6 Discussion

Two-regime reading of the gap. The 0.018–0.038 SSIM gap between BP and PC at the reconstruction level is consistent with the theoretical equivalence results of Whittington and Bogacz [2017] and Millidge et al. [2022b]: PC approximates the BP gradient at the inference fixed point, and the residual gap likely reflects (i) finite inference budget T , so x^* is only approximately reached, and (ii) the approximate nature of the equivalence outside the linearised regime. PC’s prediction errors are local in both layer and space, which in principle gives it a hierarchical denoising prior on i.i.d. pixel noise: high-frequency components of ϵ_l are absorbed at the lowest layer and damped before reaching the latent. On geometric shift, by contrast, the prediction error at every spatial location points in a globally coherent, low-frequency direction (the image is in the wrong place, in the right shape), and local PC inference has no built-in mechanism to translate the latent en bloc; the small inference budget we use cannot resolve this. Our data are partially consistent with this view: the BP–PC *probe* gap is essentially zero on the within-manifold pixel-noise rows (≤ 0.005) and visibly larger on the between-manifold geometric row (up to 0.021). We do not, however, observe a regime in which PC outperforms BP at the pixel level; the locality argument predicts a PC advantage on pixel-noise reconstruction that our matched train/test setup, single seed, and short inference budget cannot distinguish from noise.

The PC training plateau. Our most informative practical finding came from an early failure mode. With the synaptic learning rate set to $\eta_\theta = 10^{-3}$ (the value used for BP), the PC model’s SSIM stagnated at ~ 0.12 for the full training budget; reconstructions remained near-uniform grey. The cause is a violation of the two-timescale assumption underlying the derivation in Section 3: the inference dynamics must reach (approximate) equilibrium *before* the synaptic update is applied, otherwise the gradient (9) is evaluated at a non-stationary point and the update is biased. With η_θ comparable to η_x , the parameters drift on the same timescale as the activities and the system never settles.

Reducing η_θ to 10^{-4} restored the timescale separation and recovered SSIM ≈ 0.85 on clean inputs. This is consistent with the sensitivity analyses re-

ported by Pinchetti et al. [2025]: PC networks require substantially more conservative learning rates than their BP counterparts, and the practical cost of PC training is dominated by the inner inference loop rather than the outer parameter update.

Limitations. Our evaluation is restricted to one dataset and one (relatively shallow) architecture. We train and test under matched corruption (a denoising-style setting); we do not measure zero-shot robustness of a clean-trained model to unseen pixel noise or to held-out geometric group actions, which are complementary questions. The geometric column reports only translation (a group convnets already partially respect via weight sharing); a rotation row would be a sharper test of equivariance and would likely separate BP and PC differently. The PC–BP gap may also widen on deeper networks where inference convergence is slower, or on tasks where the BP gradient direction differs meaningfully from PC’s local approximation. The linear-probe results are from a single seed; with multiple seeds and a confidence interval the small differences in Table 2 might or might not survive. Wall-clock cost is significantly higher for PC because of the inner inference loop, though our setup was not optimized for speed.

7 Conclusion

We derived the supervised PC inference and learning rules from a Gaussian generative model, stated their first-order equivalence to BP, and empirically compared PC against an architecture-matched BP denoising-style autoencoder on Fashion-MNIST under four input conditions, organized into two regimes of test-time distribution shift: within-manifold pixel noise and between-manifold geometric shift. BP retains a small but consistent reconstruction advantage in both regimes (0.020–0.038 SSIM on pixel noise, 0.028 on geometric); the frozen-encoder probe gap is negligible on pixel noise (≤ 0.005) and visibly larger under random translation (0.014–0.021), with both models losing more probe accuracy from clean to noisy test inputs under geometric shift than under pixel noise. We also documented a learning-rate sensitivity that, while not novel to the PC literature, is in our experience under-discussed in introductory treatments. PC remains a theoretically appealing alternative to BP, but its practical deployment requires careful attention to the inference–

learning timescale separation that makes the derivation work, and an honest discussion of which kinds of distribution shift its locality assumption helps or hurts.

Contributions

The project was a collaborative effort between all team members. Yung-Yi Tu primarily led the writing of the report, organizing and articulating the theoretical framework, methods, and results. Abhi Desai and Advait Chalke focused on the implementation and coding of the models, including both the predictive coding and backpropagation approaches. In addition, all team members contributed to the research of the background literature and to developing the overall direction of the study.

AI Usage

We used Cursor (with Claude Opus 4.7 and GPT-5) for code scaffolding, debugging, and as an iterative draft reviewer. All experimental design, hyperparameter choices, mathematical derivations, results, and final wording are the authors’ own; all AI-generated content was reviewed and verified before inclusion.

Reproducibility

Instructions for re-creating our results can be found in the README of our public repository at <https://github.com/chalky25/Predictive-Coding>.

References

- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Neca, 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>.
- Karl Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1456):815–836, 2005.

- Karl Friston. The free-energy principle: a rough guide to the brain? *Trends in Cognitive Sciences*, 13(7):293–301, 2009.
- Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.
- Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(4):259–272, 2020.
- Joseph Marino. Predictive coding, variational autoencoders, and biological connections. *Neural Computation*, 34(1):1–44, 2022.
- Beren Millidge, Anil Seth, and Christopher L Buckley. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2022a.
- Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, 2022b.
- Luca Pinchetti, Chang Qi, Oleh Lokshyn, Gaspard Olivers, Cornelius Emde, Mufeng Tang, Amine M’Charrak, Simon Frieder, Bayar Menzat, Rafal Bogacz, et al. Benchmarking predictive coding networks — made simple. In *International Conference on Learning Representations*, 2025. arXiv:2407.01163.
- Rajesh PN 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.
- Tommaso Salvatori, Ankur Mali, Christopher L Buckley, Thomas Lukasiewicz, Rajesh PN Rao, Karl Friston, and Alexander Ororbia. Brain-inspired computational intelligence via predictive coding. *arXiv preprint arXiv:2308.07870*, 2023.
- Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation? exact implementation of backpropagation in predictive coding networks. *Advances in Neural Information Processing Systems*, 33:22566–22579, 2020.
- Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- 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.
- Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.