A Control-Theoretic Approximation to Predictive Coding Dynamics

Ryan Fayyazi[*]
New York University
ryan.fayyazi@nyu.edu

Kyle Daruwalla[*]
Cold Spring Harbor Laboratory
daruwal@cshl.edu

Mitra Javadzadeh
Cold Spring Harbor Laboratory
javadzadeh@cshl.edu

Abstract

Backpropagation remains the dominant method for training deep neural networks, but its reliance on non-local computations has motivated the search for biologically plausible alternatives. Two prominent frameworks, predictive coding (PC) and deep feedback control (DFC), offer distinct approaches to local learning, yet their relationship remains unclear. Here, we show that DFC arises as a local approximation to predictive coding dynamics near fixed points, with exact equivalence in shallow linear networks and systematic deviations arising with depth, nonlinearity, and distance from equilibrium. We characterize how this approximation, which we call Approx PC, deviates from PC in its activity dynamics, and show that these deviations can be mitigated by relinearizing the feedback controller and approaching the output target through a sequence of intermediate nudges. In networks trained on MNIST and FashionMNIST, Approx PC reaches task performance comparable to PC trained with the same nudges, and weights learned with it transfer directly to PC inference. Our results unify energy-based and control-theoretic perspectives on learning and open the door to applying tools from control theory to the training of predictive coding networks.

1 Introduction

Backpropagation has become the dominant algorithm for training deep neural networks, yet it relies on computational primitives that are difficult to reconcile with biological neural circuits, including weight transport across layers [1], global error signals [17, 22], and phased learning stages [7]. The brain, on the other hand, relies on synaptic plasticity, which is widely believed to depend on local information available at each synapse, such as pre- and post-synaptic activity and modulatory feedback, and does not require distinct forward and backward passes [18]. This contrast has motivated a large body of work seeking biologically plausible alternatives for credit assignment in deep neural networks [2, 3, 10, 23, 20, 4, 6, 11, 9, 12]. Among these, two influential approaches have emerged from distinct perspectives: (1) predictive coding (PC) networks [21, 5, 24, 16], inspired by sensory processing in cortical circuits and typically formulated as energy-based models that minimize a global objective by reducing local mismatch between bottom-up and top-down signals, and (2) deep feedback control (DFC) [13], motivated by principles of motor control and rooted in control-theoretic frameworks, where credit assignment is achieved via dynamical error correction through a feedback controller.

While both frameworks implement local learning rules, their underlying neural dynamics appear markedly different. Although both rely on error signals, their origin and role differ: in PC, errors are computed locally at each layer, whereas in DFC they arise from a global target and are distributed through the network via the controller (see Figure 1). This apparent difference raises the question

[*]Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

of whether these two classes of models implement fundamentally distinct mechanisms for credit assignment, or whether they can be understood within a common framework.

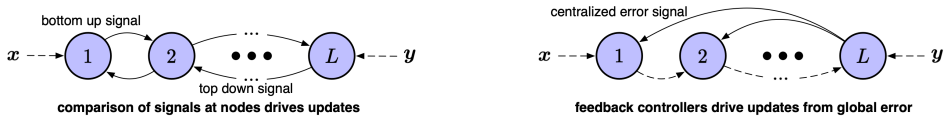

Figure 1: Illustrations of error signal sources in predictive coding (left) and deep feedback control (right). PC errors are distributed while DFC is centralized. Pathways involved in calculating errors are denoted by solid lines, while general information pathways are denoted by dashed lines.

From a biological perspective, the connectivity motifs required by both frameworks are present in the cortex: hierarchical organization with local feedforward–feedback loops supports predictive coding, while long-range feedback from higher-order to early sensory areas enables global control signals. This suggests that the brain may support both local error correction mechanisms and more global control-driven dynamics, raising the possibility that the two frameworks reflect different regimes of a shared underlying computation.

Here, we show that DFC can be interpreted as a local approximation to PC dynamics in the vicinity of fixed points. This equivalence is exact for shallow linear networks and progressively breaks down with increasing network depth, with the introduction of nonlinearities, and as the system moves further from equilibrium. Building on this connection, we propose a practical scheme to approximate PC using DFC by decomposing the trajectory toward a target into a sequence of intermediate subgoals. Within each subgoal, DFC dynamics drive the network toward a local equilibrium before transitioning to the next stage. This perspective suggests that distributed, layer-wise error signals can be effectively replaced by a centralized feedback controller that sequentially guides the network through intermediate targets. **Our contributions are organized in three parts:**

Theory: We establish a formal connection between predictive coding and deep feedback control by showing that PC activity dynamics can be written as a distributed feedback controller, which reduces near fixed points to a DFC-style controller driven by the output error. We call the resulting approximate dynamics Approx PC. In linear networks, we show that the two are exactly equivalent with a single hidden layer, and that with two hidden layers they share the same fixed point and differ only by an additional damping term in PC.

Characterizing the approximation: Through controlled probes of activity dynamics on single inputs, we show how Approx PC deviates from PC as perturbation size, depth, and nonlinearity increase. We demonstrate that these deviations are mitigated by relinearizing the feedback controller or by approaching the output target through a sequence of intermediate nudges.

Validation under learning: We test whether the approximation error compounds during training and find that it does not substantially affect what the network learns. On MNIST [8] and FashionMNIST [25], networks trained with Approx PC reach task performance comparable to that of PC trained with the same nudges, their learned weights transfer directly to PC inference, and their hyperparameter sensitivity is similar to that of PC.[1]

2 Related work

Although not the focus of this work, many alternative biologically plausible learning rules use local signals and objectives to learn [23, 20, 4] or various centralized feedback pathways to drive learning [2, 3, 10, 20, 4, 6]. Many of these have theoretical relationships to both PC and DFC, but our objective is to understand PC through a specific family of control-based rules. In particular, while recent hierarchical variants of PC incorporate long-range feedback pathways, here we focus on the standard PC formulation. The work most closely related to ours is [15], where the authors briefly note that PC can be understood as a solution to a least-control problem. Yet the controller they propose is distributed rather than a DFC-style feedback controller driven by the final layer's error. We discuss these distinctions in more detail in Section 4.

[1] Code is available at `https://github.com/darsnack/pc-dfc`.

3 Background

Consider a network comprising L layers of neurons, with input $\mathbf{x}(t) \in \mathbb{R}^d$. Let the membrane potential of neurons in layer ℓ be $\mathbf{v}^\ell(t)$, and let their activity be $\mathbf{r}^\ell(t) = \phi(\mathbf{v}^\ell(t))$, where ϕ is an activation function. Define $\mathbf{v}^0(t) := \mathbf{x}(t)$ as the input. The parameters connecting layer $\ell - 1$ and ℓ of the network are denoted $\mathbf{W}^\ell(t)$. In general, we will seek to optimize both the potentials $\mathbf{v}^\ell$ and the parameters $\mathbf{W}^\ell$ concurrently. For concision, variables with t in the subscript denote time-varying functions, i.e., $\mathbf{x}_t := \mathbf{x}(t)$.

3.1 Predictive coding

Given a network as defined above, predictive coding seeks to minimize the network's total energy, defined as the sum of layer-wise energies. The energy of a given layer is defined as the difference between its potential and the input it receives from the previous layer [2]:

$$E_t = \frac{1}{2} \sum_\ell \| \underbrace{\mathbf{v}_t^\ell - \mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1}}_{\mathbf{e}_t^\ell} \|_2^2 \tag{1}$$

To minimize the energy, the potential and weight dynamics at every layer follow the negative gradient,

$$\tau_v \frac{\mathrm{d}\mathbf{v}_t^\ell}{\mathrm{d}t} = -\nabla_{\mathbf{v}_t^\ell} E_t \quad = \begin{cases} -\mathbf{e}_t^\ell + \phi'\left(\mathbf{v}_t^\ell\right) \odot \left(\mathbf{W}_t^{\ell+1}\right)^\top \mathbf{e}_t^{\ell+1} & \text{if } \ell < L \\ -\mathbf{e}_t^L & \text{if } \ell = L \end{cases} \tag{2}$$

$$\tau_w \frac{\mathrm{d}\mathbf{W}_t^\ell}{\mathrm{d}t} = -\nabla_{\mathbf{W}_t^\ell} E_t = \mathbf{e}_t^\ell \left(\mathbf{r}_t^{\ell-1}\right)^\top \tag{3}$$

where τ_v is the neural time constant and τ_w is the weight time constant. Thus, the network activity at each layer attempts to align top-down and bottom-up signals, yielding representations that minimize the system's overall energy.

The network operates in two modes: *inference* and *learning*. In both modes, the input layer (i.e., $\mathbf{v}_t^0$) is *clamped*, meaning it is fixed to an input $\mathbf{x}_t$. This, along with (2), drives changes in the network activity during inference, while the weights are fixed. In supervised learning, we additionally clamp the final layer's activity to the labels $\mathbf{y}_t$. This, along with (2) and (3), drives both network activity and parameter updates during learning.

Note that we adopt a forward prediction convention, in which each layer predicts the activity of the next layer. While much of the PC literature uses the opposite (top-down) convention [21, 24], we choose the forward convention because it is more familiar to a general deep learning audience and matches the convention used for DFC [13].

3.2 Deep feedback control

Deep feedback control (DFC) [13–15] is an alternative framework, motivated by control theory, for adjusting the network activity and parameter dynamics to produce optimal predictions. DFC consists of a dynamical multi-layer network, where each layer integrates feedforward inputs with a feedback control signal [3]:

$$\tau_v \frac{\mathrm{d}\mathbf{v}_t^\ell}{\mathrm{d}t} = -\mathbf{v}_t^\ell + \mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1} + \mathbf{u}_t^\ell \tag{4}$$

Following control theory, we define a target, $\mathbf{r}^*$, for the final layer activity. The feedback controller determines the control signal at each layer, $\mathbf{u}_t^\ell$, to minimize the error between final layer activity and the target, $\mathbf{e}_t^L = \mathbf{r}^* - \mathbf{r}_t^L$. For proportional control, which we consider for this study, the overall control signal is $\mathbf{u}_t = K_p \mathbf{e}_t^L$, which drives each layer through feedback weights, i.e., $\mathbf{u}_t^\ell = \mathbf{Q}^\ell \mathbf{u}_t$.

We choose $\mathbf{r}^*$ to minimize a loss function, $\mathcal{L}$,

$$\mathbf{r}^* := \mathbf{r}_{\text{ss}}^L - \lambda \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{r}^L}\bigg|_{\mathbf{r}^L = \mathbf{r}_{\text{ss}}^L} = \mathbf{r}_{\text{ss}}^L + \delta^L \tag{5}$$

[2] The term in Eq. (1) associated with the final layer's energy corresponds approximately to an MSE loss, but there is a more general formulation which includes a generic loss by adding a term $\mathcal{L}\left(\mathbf{v}_t^L, \mathbf{y}_t\right)$ to the total energy [19].

[3] Our notation differs slightly from [13] for consistency with the rest of the text.

where $\mathbf{r}_{\text{ss}}^L$ is the steady state of $\mathbf{r}_t^L$ in the absence of control feedback (i.e., $\mathbf{u}_t^L = 0$). The *nudge size*, $\lambda > 0$, is chosen to be small to ensure that $\mathbf{r}^*$ remains close to the steady-state activity.

While (4) ensures that the activity of each layer is modulated to bring the final layer close to the target, this might require a large control signal—$\|\mathbf{u}_t\| \gg 0$. DFC follows the principle of least control and seeks to adjust $\mathbf{W}^\ell$ to reduce $\|\mathbf{u}_t\|$. This is achieved through the local learning dynamics,

$$\tau_w \frac{\mathrm{d}\mathbf{W}_t^\ell}{\mathrm{d}t} = \left(\mathbf{r}_t^\ell - \phi(\mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1})\right)\left(\mathbf{r}_t^{\ell-1}\right)^\top \tag{6}$$

which seeks to minimize the mean squared error between a layer's activity $\mathbf{r}_t^\ell$ and the input from the previous layer, $\phi(\mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1})$. Note that when $\tau_w \gg \tau_v$, both (4) and (6) can be applied synchronously. Additionally, one could adjust the feedback weights, $\mathbf{Q}^\ell$, to reduce $\|\mathbf{u}_t\|$, but in general a wide range of fixed feedback weights is sufficient.

4 Theoretical analysis of predictive coding as deep feedback control

Taken at face value, the activity dynamics of predictive coding (2) and deep feedback control (4) appear distinct. Our aim is to show that these two frameworks are closely related—specifically, that PC can be approximated by a specific DFC controller. For the remainder of this section, we focus on the activity dynamics, which occur much faster than the weight dynamics ($\tau_v \ll \tau_w$). In general, both PC and DFC follow equivalent weight dynamics (see (3) and (6), which differ only in activity vs. potential space). We use PC weight dynamics in our final proposed scheme, with the local error $\mathbf{e}^\ell$ replaced by the control signal $\mathbf{u}^\ell$.

In PC, following (2), the potentials of layers $\ell < L$ evolve according to (see Appendix A for more):

$$\tau_v \frac{\mathrm{d}\mathbf{v}_t^\ell}{\mathrm{d}t} = -\mathbf{e}_t^\ell + \left(\mathbf{J}_t^{\ell+1}\right)^\top \mathbf{e}_t^{\ell+1} = -\mathbf{v}_t^\ell + \mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1} + \left(\mathbf{J}_t^{\ell+1}\right)^\top \mathbf{e}_t^{\ell+1} \tag{7}$$

Thus, we see that PC activity dynamics are very similar to DFC dynamics in (4), with the layer-wise control given by $\mathbf{u}_t^\ell = \left(\mathbf{J}_t^{\ell+1}\right)^\top \mathbf{e}_t^{\ell+1}$. Unlike DFC, this controller does not directly feed back the error signal from the output. Instead, it defines a recurrence where each layer's error controls the activity of the preceding layer. So, while the dynamics as expressed in (7) are similar, PC dynamics follow an entirely different control strategy.

For the last layer, PC drives activity by clamping the output, while in DFC, the controller computes an error signal, $\mathbf{r}^* - \mathbf{r}_t^L$. We reconcile these here by clamping the output to the target $\mathbf{r}^*$. Then for PC, we have that $\mathbf{e}_t^L = \mathbf{v}_t^L - \mathbf{W}_t^L \mathbf{r}_t^{L-1} = \phi^{-1}(\mathbf{r}^*) - \mathbf{W}_t^L \mathbf{r}_t^{L-1}$ and $\nabla_{\mathbf{v}_t^L} E_t = 0$ (since $\phi^{-1}(\mathbf{r}^*)$ is constant) [4]. Thus, the final layer control signal is $\mathbf{u}_t^L = 0$. During inference, $\mathbf{r}_t^L$ is not clamped and a non-zero error, $\mathbf{e}_t^L = \mathbf{u}_t^L$, drives the output dynamics.

This completes the control-theoretic perspective on PC, which captures its dynamics as a distributed, time-varying proportional feedback controller (see Figure 8). Indeed, [15] show that under a specific objective, PC is the optimal solution to a least-control problem. We, however, are concerned not with any control strategy but specifically with those that match DFC, in which the error computed at the output modulates hidden layers directly through feedback control. Next, we will make approximations to the distributed controller near fixed points to make PC and DFC completely equivalent.

4.1 Equivalent control strategies for PC and DFC at fixed points

Once converged, the PC network is at a fixed point, so $\frac{\mathrm{d}v_t^\ell}{\mathrm{d}t}$ = 0. In this regime, (2) can be rewritten as $\mathbf{e}_t^\ell = \left(\mathbf{J}_t^{\ell+1}\right)^\top \mathbf{e}_t^{\ell+1}$. This implies that for PC networks at a fixed point,

$$\mathbf{u}_t^\ell = \left(\mathbf{J}_t^{\ell+1}\right)^\top \mathbf{e}_t^{\ell+1} = \left(\prod_{k=\ell+1}^{L} \mathbf{J}_t^k\right)^\top \mathbf{e}_t^L, \tag{8}$$

[4] We assume for simplicity that ϕ is the identity at the final layer.

so the $\mathbf{u}_t^\ell$ inputs to each layer depend only on the error in the last layer. The control signal in (8) matches that of DFC, where the error is calculated at the final layer, and the feedback weights for each layer are given by:

$$\mathbf{Q}_t^\ell K_p := \left(\prod_{k=\ell+1}^{L} \mathbf{J}_t^k \right)^\top, \tag{9}$$

where $K_p \geq 0$ is the proportional control constant. Thus, PC at fixed points can be exactly expressed using the feedback control strategy employed by DFC (see Appendix A). However, the feedback weights are time-dependent. So, *we make two approximations: first, that* (8) *and* (9) *hold locally near fixed points; second, that the time-dependent Jacobian can be approximated by linearizing the network dynamics around the fixed point.* The latter is only required for nonlinear networks, as the Jacobians for linear networks are constant with respect to time. **Our final approach, which we refer to as "Approx PC", is summarized by**

$$\tau_v \frac{\mathrm{d}\mathbf{v}_t^\ell}{\mathrm{d}t} = -\mathbf{v}_t^\ell + \mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1} + \mathbf{u}_t^\ell \quad \text{where} \quad \mathbf{u}_t^\ell = \left(\prod_{k=\ell+1}^{L} \mathbf{J}_0^k \right)^\top \mathbf{e}_t^L \tag{10}$$

where we assume the network is initially at a fixed point at $t = 0$.

4.2 Approximation characterization in linear networks

Assume that the network is at steady state such that (9) holds exactly. We then clamp the target to $\mathbf{r}^*$, which is selected to be close to the current steady-state output. This perturbs the system away from the fixed point by a small amount. There are multiple sources of error: (8) may not hold, and for nonlinear networks, the Jacobians are only exact at the fixed point. To disentangle these sources of error, we begin by considering linear networks.

In a two-layer linear network (one hidden layer), we have activity dynamics for only one layer (the hidden layer), as the input and output layers are clamped. Since $\mathbf{Q}^{L-1} = (\mathbf{W}^L)^\top$, DFC and PC dynamics are exactly the same. However, in a three-layer linear network, the hidden layers' activity dynamics are not identical for the PC and DFC cases. PC dynamics follow:

$$\begin{aligned} \tau \dot{\mathbf{v}}_t^1 &= -\left(\mathbf{v}_t^1 - \mathbf{W}_t^1 \mathbf{x}\right) + \left(\mathbf{W}_t^2\right)^T \left[\mathbf{v}_t^2 - \mathbf{W}_t^2 \mathbf{v}_t^1\right] \\ \tau \dot{\mathbf{v}}_t^2 &= -\left(\mathbf{v}_t^2 - \mathbf{W}_t^2 \mathbf{v}_t^1\right) + \left(\mathbf{W}_t^3\right)^T \left[\mathbf{r}^* - \mathbf{W}_t^3 \mathbf{v}_t^2\right] \end{aligned} \tag{11}$$

In contrast, DFC dynamics evolve according to:

$$\begin{aligned} \tau \dot{\mathbf{v}}_t^1 &= -\left(\mathbf{v}_t^1 - \mathbf{W}_t^1 \mathbf{x}\right) + \left(\mathbf{W}_t^2\right)^T \left(\mathbf{W}_t^3\right)^T \left[\mathbf{r}^* - \mathbf{W}_t^3 \mathbf{v}_t^2\right] \\ \tau \dot{\mathbf{v}}_t^2 &= -\left(\mathbf{v}_t^2 - \mathbf{W}_t^2 \mathbf{v}_t^1\right) + \left(\mathbf{W}_t^3\right)^T \left[\mathbf{r}^* - \mathbf{W}_t^3 \mathbf{v}_t^2\right] \end{aligned} \tag{12}$$

We show in Appendix C that the last layer's dynamics ($\mathbf{v}_t^2$) can be described by second-order ordinary differential equations for both the DFC and PC cases, i.e.,

$$\tau^2 \ddot{\mathbf{v}}_t^{2^{\text{PC}}} + \tau \mathbf{A}_{\text{PC}} \dot{\mathbf{v}}_t^{2^{\text{PC}}} + \mathbf{B} \mathbf{v}_t^{2^{\text{PC}}} = \mathbf{C} \tag{13}$$

$$\tau^2 \ddot{\mathbf{v}}_t^{2^{\text{DFC}}} + \tau \mathbf{A}_{\text{DFC}} \dot{\mathbf{v}}_t^{2^{\text{DFC}}} + \mathbf{B} \mathbf{v}_t^{2^{\text{DFC}}} = \mathbf{C} \tag{14}$$

where (13) follows from (11) and (14) from (12). Notably, the only coefficients that differ between these two systems are the damping coefficients, where $\mathbf{A}_{\text{PC}} - \mathbf{A}_{\text{DFC}} = \mathbf{W}_t^2 \left(\mathbf{W}_t^2\right)^T$. This suggests that after being perturbed from their initial fixed point, both systems converge to the same steady state given by $\mathbf{B}\mathbf{v}_*^2 = \mathbf{C}$. However, the transient dynamics will differ, with PC dynamics being more damped (as $\mathbf{W}_t^2 \left(\mathbf{W}_t^2\right)^T \succeq 0$).

We confirm these results empirically in Figure 2. Given a network and input, we ensure it is at a fixed point by running PC dynamics without output clamping until convergence. Then, from the converged fixed point, we perturb the system by clamping the output to a target a known distance away (reported as relative "perturbation size"). For a three-layer network, as seen in Figure 2 (left), both PC and our approximation converge to the same point, though they have different dynamics. As we increase the number of layers in the network to eight, the dynamics become oscillatory and converge to different values (Figure 2).

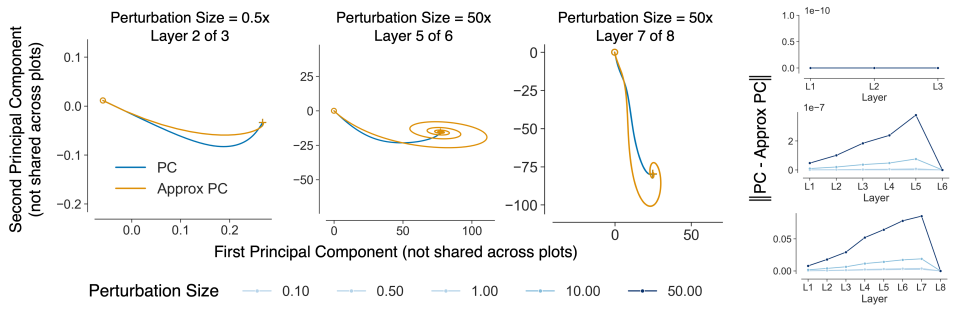

Figure 2: Differences in dynamics of PC and our approximation in their first two principal components. The clamped target for all plots is the same noise perturbation direction. See Appendix D for details. **(Left)** The potential trajectories for individual layers across networks of increasing depth. The three-layer network converges to the same endpoint as PC, while the six- and eight-layer networks converge to different values with L_2-error on the order of 10^{-7} and 10^{-2}, respectively. **(Right)** L_2-norm of the difference in final potentials across layers and perturbation sizes, for the three networks on the left.

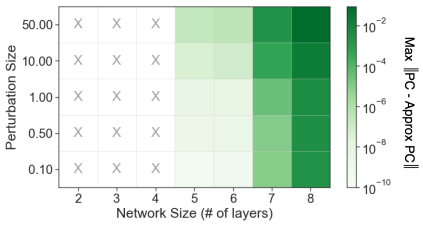

Figure 3: Maximum L_2-norm error across layers in networks of varying depth for different perturbation sizes. "X" indicates that the error is $\leq 10^{-10}$. See Appendix D for details.

Looking across networks of different depth and perturbations of varying size, we observe that both perturbation magnitude and depth increase the worst-case error (Figure 3). Notably, the error is numerically zero for networks with two or three layers, as predicted by our theory. Empirically, while this appears to extend to four-layer networks, deeper networks exhibit persistent residual error between PC and Approx PC, suggesting that the approximation may not be reliable away from fixed points. In the following section, we examine this residual error more generally for nonlinear networks and discuss techniques to mitigate it, allowing our framework to remain applicable beyond local regimes.

4.3 Approximation characterization in nonlinear networks

While linear networks allow us to characterize how violating the assumptions in (8) affects the approximate dynamics, we would also like to understand how the linearization of the network Jacobian produces errors away from the fixed point. To do this, we introduce the notion of relinearization time points—a set of time points, $\mathcal{T}$, that denote when to recompute the Jacobian of the network. Specifically, we modify (10) to

$$\tau_v \frac{\mathrm{d}\mathbf{v}_t^\ell}{\mathrm{d}t} = -\mathbf{v}_t^\ell + \mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1} + \mathbf{u}_t^\ell$$
$$\text{where} \quad \mathbf{u}_t^\ell = \left(\prod_{k=\ell+1}^{L} \mathbf{J}_{t'}^k \right)^\top \mathbf{e}_t^L \text{ and } t' = \arg\min_{\substack{t'' \in \mathcal{T} \\ t'' < t}} t - t'' \tag{15}$$

In practice, $\mathcal{T}$ is defined based on a relinearization frequency, i.e., the Jacobians are recomputed every N steps. We examine the effect of relinearization frequency on the network dynamics for a nonlinear network in Figure 4. Without relinearization, the network fails to converge to the same final potentials as PC. With infrequent relinearization, Approx PC approaches the same endpoint as PC, but the dynamics overshoot, requiring more total integration steps and relinearizations (on the order of 10^2) to converge. This persists until a relinearization frequency of 25 steps, at which point the dynamics are still underdamped compared to PC, but overshoot no longer occurs. As a result,

the system converges faster and requires an order of magnitude fewer relinearizations. Increasing the relinearization frequency beyond this point does not fundamentally change the dynamics, and only increases the total number of relinearizations without meaningfully reducing the final endpoint difference between PC and Approx PC.

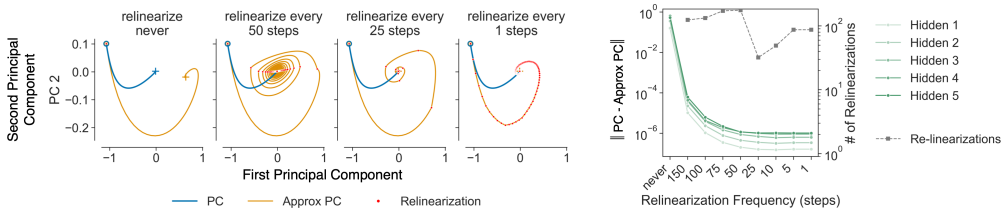

Figure 4: Differences in dynamics of PC and Approx PC in their first two principal components as the frequency of relinearization is adjusted. Never relinearizing corresponds to following the dynamics of (10). See Appendix D for details. **(Left)** Trajectories of the potentials of the fourth layer in a six-layer network with $\tanh$ activation functions. **(Right)** L_2-norm error in the endpoints of PC and Approx PC across layers decreases as relinearization frequency increases. The number of relinearizations (gray line) stays $\approx$ constant until an abrupt change around a frequency of 25 steps.

We note that there is nothing fundamental about 25 steps. Instead, given the initial fixed point of the system, its nonlinear dynamics, and a particular perturbation, 25 steps represents the distance the system is allowed to move away from the fixed point without incurring significant error due to linearization. We measure this in Figure 5 (left) as the difference in flow w.r.t. the initial linearized flow as we move along the Approx PC trajectory without relinearization. Essentially, this measures at each timestep how much the dynamics would change if we relinearize. The peak at 25 steps indicates the moment at which we would achieve the largest error correction. By relinearizing at this point, we move more efficiently toward the PC endpoint and quickly benefit from the slow dynamics of the nonlinear system around the PC endpoint (Figure 5, right). Thus, we conclude that relinearization can reduce the residual error of Approx PC *efficiently*, provided that we do not move "too far" before reorienting the dynamics.

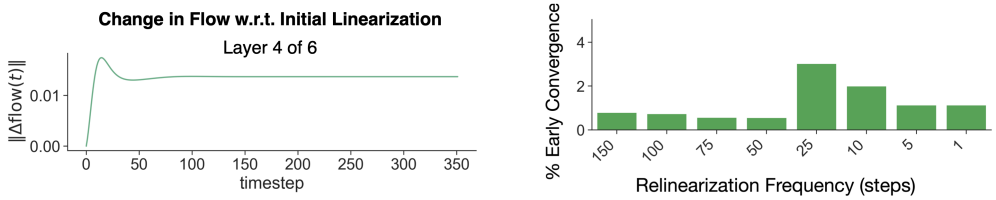

Figure 5: **(Left)** The L_2-norm of the change in flow w.r.t. the initial linearized flow moving along the Approx PC trajectory without relinearization in Figure 4. **(Right)** Percentage of segments between linearizations where the dynamics of Approx PC converge (i.e., the speed of dynamics $\leq 10^{-8}$) earlier than the relinearization frequency. See Appendix D for details.

5 Resolving residual errors with nudging

In the previous section, we showed that Approx PC without relinearization exhibits residual errors compared to PC for all but the simplest linear models. While relinearization addresses this issue, its timing greatly affects dynamical efficiency, and this depends on the network, the current fixed point, and the target. In a realistic training scenario, it is not clear how these three factors interact.

To address this, we introduce a multi-stage algorithm (Algorithm 1) in which the trajectory toward a target is segmented into multiple smaller subgoals. Each subgoal is a "nudge," i.e., a partial step toward the target, computed as a fraction of the linear path between the current output and the target (see step 2(a) of Algorithm 1). Importantly, after each nudge, the dynamics are allowed to relax to a new fixed point, after which a new subgoal is computed and the next nudge is initiated.

Figure 6 shows the resulting trajectories when we apply Algorithm 1. If the nudge scaling factor, γ, is small, the network follows a frequent relinearization-like trajectory (albeit with small oscillations to convergence between nudges). With large scaling factors, we observe a new behavior where the trajectory initially deviates from the PC endpoint significantly (similar to never relinearizing); however, since the network is allowed to relax to a fixed point, the next nudge reorients the trajectory toward the PC endpoint. This is in contrast to infrequent relinearization, which keeps overshooting the PC endpoint. As expected, the final difference between the endpoints of PC and Approx PC with nudges is small (of the same order as with relinearization). Notably, this applies to large scaling factors as well, where the total number of nudges is small. In this way, nudging preserves the low residual error of relinearization without consuming a significant number of additional integration steps.

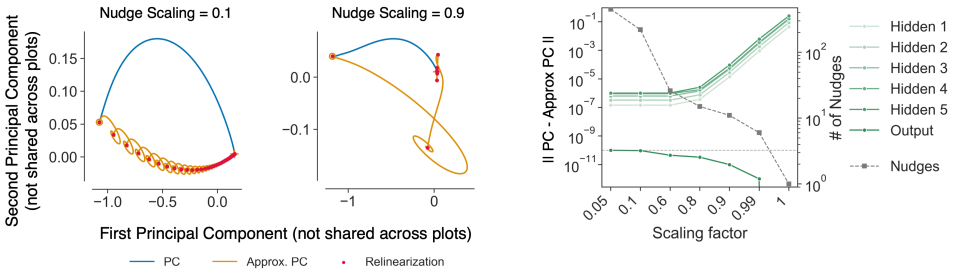

Figure 6: Differences in dynamics of PC and Approx PC in their first two principal components for different nudge scalings. A scaling factor of one corresponds to following the dynamics of (10). The perturbation size is one. See Appendix D for details. **(Left)** Trajectories of the potentials of the fourth layer in a six-layer network with $\tanh$ activation functions for two different nudge scaling factors. Nudges are recomputed at relinearization points. **(Right)** L_2-norm error in the endpoints of PC and Approx PC across layers decreases with smaller scaling factors. The number of nudges (gray line) consistently decreases with larger scaling factors. The horizontal dashed line indicates the convergence threshold for nudging.

Algorithm 1 Approx PC with nudges

1. **Requires:** a network of L layers at fixed point $\mathbf{v}_*$, input $\mathbf{x}$, target $\mathbf{y}$, nudge scaling factor $\gamma \in (0, 1]$, nudge convergence threshold $\epsilon > 0$, dynamics convergence threshold $\delta > 0$
2. **while** $\|\mathbf{v}_*^L - \mathbf{y}\| > \epsilon$ **do**
   (a) compute nudge: $\mathbf{r}_* \leftarrow \mathbf{v}_*^L + \gamma(\mathbf{y} - \mathbf{v}_*^L)$
   (b) clamp network: $\mathbf{r}^L \leftarrow \mathbf{r}_*$
   (c) compute Jacobian: $\mathbf{J}^\ell \leftarrow$ layer-wise Jacobian at current timestep for input $\mathbf{x}$ for all ℓ
   (d) run dynamics until convergence: **do**
      i. $\mathbf{e}^L \leftarrow \mathbf{v}^L - \mathbf{W}^L \mathbf{r}^{L-1}$
      ii. $\mathbf{u}^\ell \leftarrow (\prod_{k=\ell+1}^{L} \mathbf{J}^k)^\top \mathbf{e}^L$ for all ℓ
      iii. $\mathbf{v}_{\text{prev}}^\ell \leftarrow \mathbf{v}^\ell$
      iv. $\mathbf{v}^\ell \leftarrow \mathbf{v}^\ell + \tau_v^{-1}(-\mathbf{v}^\ell + \mathbf{W}^\ell \mathbf{r}^{\ell-1} + \mathbf{u}^\ell)$ for all ℓ
      v. $\mathbf{W}^\ell \leftarrow \mathbf{W}^\ell + \tau_w^{-1}[\mathbf{u}^\ell (\mathbf{r}^{\ell-1})^\top]$ for all ℓ
      while $\|\mathbf{v}^\ell - \mathbf{v}_{\text{prev}}^\ell\| > \delta$
   (e) update fixed point: $\mathbf{v}_* \leftarrow \mathbf{v}$

6 Approx PC with learning

To evaluate whether our Approx PC algorithm remains faithful to PC in practical settings with learning, we implemented and examined both approaches on standard supervised image classification tasks (see Appendix D for details). We first tested whether networks trained with Approx PC converge

to solutions that are compatible with predictive coding. To this end, we trained a network using Approx PC and then directly transplanted the learned weights into a PC model without further training for evaluation. **Approx PC networks achieved a final test accuracy of 97.32 ± 0.12% on MNIST across five seeds, and the corresponding PC networks achieved an evaluated test accuracy of 97.47 ± 0.05% across the same seeds.** This confirms that Approx PC learns solutions compatible with PC dynamics.

For both MNIST and FashionMNIST, Approx PC reaches accuracy similar to or higher than that of standard PC at their respective best learning rates (Figure 7). Interestingly, Approx PC also seems to learn faster per integration step (Figure 10). This could be due to the fact that Approx PC propagates a centralized error, allowing layers to converge concurrently, while PC uses distributed error signals that must propagate across the network one layer at a time. However, Approx PC also uses multiple nudges per sample, so the improvement may instead come from breaking each learning step into intermediate subgoals. To isolate this effect, we implemented PC with nudges, where the network is trained using intermediate nudges as clamped outputs but with standard PC dynamics. We found that adding nudges to PC recovers most of the learning performance and efficiency enjoyed by Approx PC (Figures 7 and 10). This suggests that most of the gain over standard PC comes from nudging rather than from the centralized controller. However, for the same number of samples seen, Approx PC and PC with nudges require more total integration steps than standard PC. Finally, hyperparameter sweeps on FashionMNIST (Table 1) show that both methods degrade similarly with network depth, but differ in their sensitivity to other hyperparameters. In particular, Approx PC is less sensitive to the neural time constant and dynamics convergence tolerance, but has a slightly narrower range of effective learning rates.

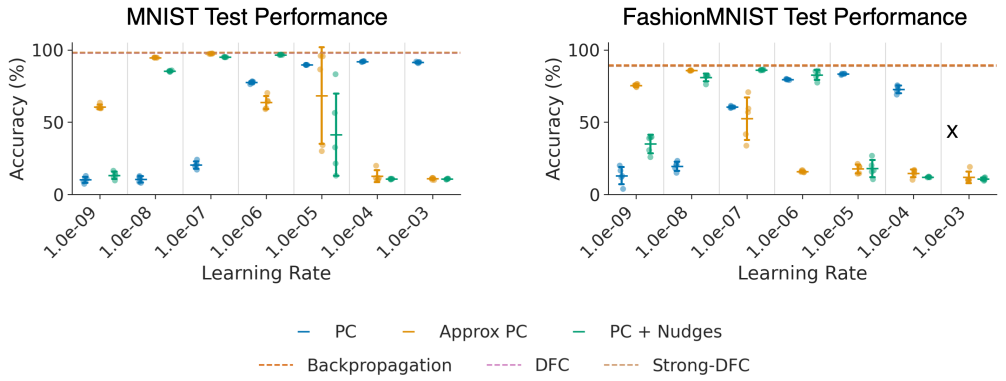

Figure 7: Final test accuracy on MNIST (left) and FashionMNIST (right) across learning rates for PC, Approx PC, and PC with nudges (mean and standard deviation across five seeds). Dashed lines show previously reported test accuracies for DFC [13], Strong-DFC [14], and standard backprop (via [14]). Some configurations did not learn and are excluded. "X" denotes configurations with no successful runs. Learning rate $:= \tau_w^{-1}$.

7 Discussion

We have shown that predictive coding and deep feedback control, two prominent frameworks for biologically plausible deep learning, can be understood within a common dynamical systems perspective. In particular, DFC provides a local approximation to PC dynamics near fixed points, revealing that these seemingly distinct approaches differ primarily in their transient dynamics rather than their steady-state behavior.

Implications for machine learning. We have shown that Approx PC reaches approximately the same fixed points as PC, while differing in its transient dynamics. Moreover, weights learned under Approx PC can be transplanted into PC networks for inference. These results suggest that part of the slow inference attributed to PC may stem from its particular dynamics rather than from energy-based learning itself. In PC, errors reach early layers only by propagating one layer at a time. Separately, nudging accounted for most of the accuracy gain over standard PC in our experiments. This may

partly reflect the extra weight updates that nudging entails. Another possibility is that clamping the network's output directly to the label may cause it to reach a different equilibrium than would be reached by moving toward the label more gradually. Equilibrium propagation makes a similar argument for weak clamping, where small nudges keep the network's new equilibrium close to its original one [23]. Whether moving the target gradually benefits energy-based learning more broadly remains to be tested.

Implications for biological interpretation. Control-theoretic and predictive coding frameworks have often been proposed as alternative accounts of credit assignment in the brain, distinguished by whether error signals are computed locally at each layer or generated centrally and broadcast through feedback pathways. Our results indicate that, near fixed points, PC dynamics can be expressed as an equivalent control strategy driven by output errors, making the two frameworks difficult to tell apart based on steady-state activity alone. Our theory instead suggests that distinguishing these mechanisms in the brain requires an analysis of transient activity. In particular, perturbations in one layer of a PC network immediately affect the preceding layer through local errors, whereas under DFC, preceding layers respond only after the perturbation has reached the output. We therefore suggest experiments that perturb one stage of a processing hierarchy and compare the timing of responses at earlier stages and at the output.

Limitations and future work. The primary purpose of our experiments was to validate predictions arising from the theoretical connection made between PC and DFC. Therefore, we focused on controlled experiments in fully connected networks, where making sense of activity and learning dynamics is more tractable. While we demonstrated the practical feasibility of Approx PC through MNIST and FashionMNIST experiments, future work should expand our approach to neural networks with more parameters and different architectures (e.g., convolutional networks), and assess whether Approx PC can be competitive with established training methods at scale. The appropriate nudge size and convergence thresholds also depend on the network and data, making them difficult to set in advance. In practice, we found the number of steps per nudge to be a useful indicator of convergence behavior, which future work could formalize into an adaptive nudging scheme. Beyond machine learning applications, an implementation of Approx PC in biology is unlikely to use the exact network Jacobian. Instead, one may consider slow-varying estimates of the Jacobians or arbitrary, structured feedback weights, whose exploration we leave to future work. Finally, we hope our work can inspire extensions that explore the use of control-theoretic improvements to PC training, such as PID control.

Acknowledgments and Disclosure of Funding

This work was supported by the Cold Spring Harbor Laboratory Fellows Program and Pershing Square Foundation. The authors declare no competing interests.

References

[1] Mohamed Akrout, Collin Wilson, Peter C. Humphreys, Timothy Lillicrap, and Douglas Tweed. Deep learning without weight transport. In *Advances in Neural Information Processing Systems*, 2019.

[2] David Balduzzi, Hastagiri Vanchinathan, and Joachim Buhmann. Kickback Cuts Backprop's Red-Tape: Biologically Plausible Credit Assignment in Neural Networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9467/9284`.

[3] Brian Crafton, Abhinav Parihar, Evan Gebhardt, and Arijit Raychowdhury. Direct Feedback Alignment With Sparse Connections for Local Learning. *Frontiers in Neuroscience*, 13:525, 2019. doi: 10.3389/fnins.2019.00525. URL `https://www.frontiersin.org/article/10.3389/fnins.2019.00525/full`.

[4] Kyle Daruwalla and Mikko Lipasti. Information bottleneck-based Hebbian learning rule naturally ties working memory and synaptic updates. *Frontiers in Computational Neuroscience*, 18:1240348, 2024. doi: 10.3389/fncom.2024.1240348. URL `https://www.frontiersin.org/articles/10.3389/fncom.2024.1240348/full`.

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

[6] Nicolas Frémaux and Wulfram Gerstner. Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules. *Frontiers in Neural Circuits*, 9:85, 2016. doi: 10.3389/fncir.2015.00085. URL http://journal.frontiersin.org/Article/10.3389/fncir.2015.00085/abstract.

[7] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[9] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, Carlos Soares, João Gama, and Alípio Jorge, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 498–515, Cham, 2015. Springer International Publishing.

[10] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random feedback weights support learning in deep neural networks. arXiv:1411.0247, 2014. URL http://arxiv.org/abs/1411.0247.

[11] 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. doi: 10.1038/s41583-020-0277-3. URL http://www.nature.com/articles/s41583-020-0277-3.

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

[13] Alexander Meulemans, Matilde Tristany Farinha, Javier Garcia Ordonez, Pau Vilimelis Aceituno, João Sacramento, and Benjamin F. Grewe. Credit Assignment in Neural Networks through Deep Feedback Control. In *Advances in Neural Information Processing Systems*, volume 34, pages 4674–4687. Curran Associates, Inc., 2021.

[14] Alexander Meulemans, Matilde Tristany Farinha, Maria R. Cervera, João Sacramento, and Benjamin F. Grewe. Minimizing Control for Credit Assignment with Strong Feedback. In *Proceedings of the 39th International Conference on Machine Learning*, pages 15458–15483. PMLR, June 2022.

[15] Alexander Meulemans, Nicolas Zucchet, Seijin Kobayashi, Johannes von Oswald, and João Sacramento. The least-control principle for local learning at equilibrium. In *Advances in Neural Information Processing Systems*, volume 35, pages 33603–33617, December 2022.

[16] Beren Millidge, Anil Seth, and Christopher L. Buckley. Predictive coding: A theoretical and experimental review. *arXiv:2107.12979*, 2022.

[17] Randall C. O'Reilly. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, 8(5):895–938, 1996.

[18] Alexander G. Ororbia. Brain-Inspired Machine Intelligence: A Survey of Neurobiologically-Plausible Credit Assignment. arXiv:2312.09257, 2023. URL http://arxiv.org/abs/2312.09257.

[19] Luca Pinchetti, Tommaso Salvatori, Yordan Yordanov, Beren Millidge, Yuhang Song, and Thomas Lukasiewicz. Predictive coding beyond Gaussian distributions. In *Advances in Neural Information Processing Systems*, 2022.

[20] Roman Pogodin and Peter E. Latham. Kernelized information bottleneck leads to biologically plausible 3-factor Hebbian learning in deep networks. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/517f24c02e620d5a4dac1db388664a63-Paper.pdf.

[21] 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.

[22] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems*, 2018.

[23] Benjamin Scellier and Yoshua Bengio. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 11: 24, 2017. doi: 10.3389/fncom.2017.00024. URL `http://journal.frontiersin.org/article/10.3389/fncom.2017.00024/full`.

[24] 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.

[25] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747, 2017. URL `http://arxiv.org/abs/1708.07747`.

A Rewriting PC activity dynamics

Starting from PC potential dynamics given by (2), we rewrite the dynamics in a form compatible with DFC. To begin, we will focus on layers $\ell < L$, where the inference and learning activity is governed by the same set of equations. Let us briefly review the derivation of (2). The gradient of the total energy is

$$\nabla_{\mathbf{v}_t^\ell} E_t = \frac{1}{2}\nabla_{\mathbf{v}_t^\ell}\|\mathbf{e}_t^\ell\|_2^2 + \frac{1}{2}\nabla_{\mathbf{v}_t^\ell}\|\mathbf{e}_t^{\ell+1}\|_2^2 \tag{16}$$

The other terms in (1) do not depend on $\mathbf{v}^\ell$, so their derivatives are zero. Next, we can substitute in the layer-wise errors and proceed as follows

$$\begin{aligned}\nabla_{\mathbf{v}_t^\ell} E_t &= \frac{1}{2}\nabla_{\mathbf{v}_t^\ell}\|\mathbf{v}_t^\ell - \mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1}\|_2^2 + \frac{1}{2}\nabla_{\mathbf{v}_t^\ell}\|\mathbf{v}_t^{\ell+1} - \mathbf{W}_t^{\ell+1}\mathbf{r}_t^\ell\|_2^2 \\ &= \underbrace{\mathbf{v}_t^\ell - \mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1}}_{\mathbf{e}_t^\ell} - \underbrace{\phi'\left(\mathbf{v}_t^\ell\right) \odot \left(\mathbf{W}_t^{\ell+1}\right)^\top}_{\left(\mathbf{J}_t^{\ell+1}\right)^\top} \underbrace{\left(\mathbf{v}_t^{\ell+1} - \mathbf{W}_t^{\ell+1}\mathbf{r}_t^\ell\right)}_{\mathbf{e}_t^{\ell+1}}\end{aligned} \tag{17}$$

Finally, the potential evolves according to the negative gradient of the energy, resulting in the following dynamics

$$\tau_v \frac{\mathrm{d}\mathbf{v}_t^\ell}{\mathrm{d}t} = -\mathbf{v}_t^\ell + \mathbf{W}_t^\ell \mathbf{r}_t^{\ell-1} + \left(\mathbf{J}_t^{\ell+1}\right)^\top \mathbf{e}_t^{\ell+1} \tag{18}$$

This matches the form of DFC dynamics in (4).

B Schematics for PC and DFC style controllers

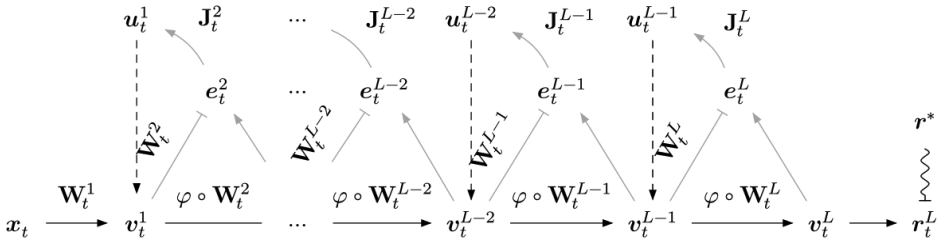

Figure 8: A schematic of the "control system" view of predictive coding. The output is clamped to an arbitrary set point, $\mathbf{r}^*$, and each layer's activity is modulated via a feedback controller driven by the subsequent layer's error. Edges meet to indicate addition; edges with flat ends denote negative (inhibitory) signals, gray edges highlight control signals, and wavy edges denote clamping.

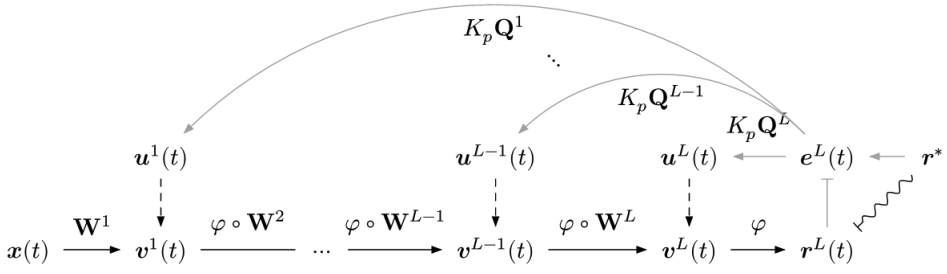

Figure 9: A schematic of DFC-style control. The output is clamped to an arbitrary set point, $\mathbf{r}^*$, and each layer's activity is modulated via a feedback controller driven by the output error.

C Derivation of PC and Approx PC exact dynamics in linear networks

In this appendix we derive the ordinary differential equations (ODEs) governing layer-wise representation for both PC and Approx PC. We consider a three-layer network $(x, \mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3)$, where the last layer $\mathbf{v}^3$ is clamped to the training output, $\mathbf{r}^*$. We also consider linear activations ($\phi(\mathbf{v}^\ell) = \mathbf{v}^\ell$, $\phi'\left(\mathbf{v}^\ell\right) = \mathbf{1}$). For both PCN and DFC approximation dynamics, the goal is to yield a closed-form ODE purely in $\mathbf{v}^\ell$. We then define the residual between the two systems and compare them.

Throughout, we consider $\mathbf{v}^1 \in \mathbb{R}^{d_1}$, $\mathbf{v}^2 \in \mathbb{R}^{d_2}$, input $\mathbf{x} \in \mathbb{R}^{d_0}$, and target $\mathbf{r}^* \in \mathbb{R}^{d_3}$. Weight matrices are $\mathbf{W}^1 \in \mathbb{R}^{d_1 \times d_0}$, $\mathbf{W}^2 \in \mathbb{R}^{d_2 \times d_1}$, $\mathbf{W}^3 \in \mathbb{R}^{d_3 \times d_2}$. We leave off the subscript t for clarity. The time constant τ_v is shared by all layers.

C.1 PC activity dynamics

Recall (2):

$$\tau \dot{\mathbf{v}}^1 = -\left(\mathbf{v}^1 - \mathbf{W}^1\mathbf{x}\right) + \left(\mathbf{W}^2\right)^T\left[\mathbf{v}^2 - \mathbf{W}^2\mathbf{v}^1\right] \tag{19}$$

$$\tau \dot{\mathbf{v}}_2 = -\left(\mathbf{v}^2 - \mathbf{W}^2\mathbf{v}^1\right) + \left(\mathbf{W}^3\right)^T\left[\mathbf{r}^* - \mathbf{W}^3\mathbf{v}^2\right] \tag{20}$$

Expanding (19) and (20) and collecting terms in v_1 and v_2:

$$\tau \dot{\mathbf{v}}^1 = -\left(\mathbf{I} + \left(\mathbf{W}^2\right)^T\mathbf{W}^2\right)\mathbf{v}^1 + \left(\mathbf{W}^2\right)^T\mathbf{v}^2 + \mathbf{W}^1\mathbf{x} \tag{21}$$

$$\tau \dot{\mathbf{v}}^2 = -\left(\mathbf{I} + \left(\mathbf{W}^3\right)^T\mathbf{W}^3\right)\mathbf{v}^2 + \mathbf{W}^2\mathbf{v}^1 + \left(\mathbf{W}^3\right)^T\mathbf{r}^* \tag{22}$$

From (22) we isolate the $\mathbf{v}^1$ -dependent term:

$$\mathbf{W}^2\mathbf{v}^1 = \tau \dot{\mathbf{v}}^2 + \left(\mathbf{I} + \left(\mathbf{W}^3\right)^T\mathbf{W}^3\right)\mathbf{v}^2 - \left(\mathbf{W}^3\right)^T\mathbf{r}^* \tag{23}$$

Differentiating (22) with respect to time gives:

$$\tau \ddot{\mathbf{v}}^2 = -\left(\mathbf{I} + \left(\mathbf{W}^3\right)^T\mathbf{W}^3\right)\dot{\mathbf{v}}^2 + \mathbf{W}^2\dot{\mathbf{v}}^1 \tag{24}$$

Substituting $\dot{\mathbf{v}}^1$ from (21) gives:

$$\tau^2 \ddot{\mathbf{v}}^2 = -\tau\left(\mathbf{I} + \left(\mathbf{W}^3\right)^T\mathbf{W}^3\right)\dot{\mathbf{v}}^2 + \mathbf{W}^2\left[-\left(\mathbf{I} + \left(\mathbf{W}^2\right)^T\mathbf{W}^2\right)\mathbf{v}^1 + \left(\mathbf{W}^2\right)^T\mathbf{v}^2 + \mathbf{W}^1\mathbf{x}\right] \tag{25}$$

Note that $\mathbf{v}^1$ coefficient can be written as:

$$-\mathbf{W}^2\left(\mathbf{I} + \left(\mathbf{W}^2\right)^T\mathbf{W}^2\right)\mathbf{v}^1 = -\left(\mathbf{I} + \mathbf{W}^2\left(\mathbf{W}^2\right)^T\right)\left(\mathbf{W}^2\mathbf{v}^1\right) \tag{26}$$

and we can replace $\mathbf{W}^2\mathbf{v}^1$ from (23), and write (25) as:

$$\begin{aligned}\tau^2 \ddot{\mathbf{v}}^2 = &-\tau\left(\mathbf{I} + \left(\mathbf{W}^3\right)^T\mathbf{W}^3\right)\dot{\mathbf{v}}^2 \\ &- \left(\mathbf{I} + \mathbf{W}^2\left(\mathbf{W}^2\right)^T\right)\left[\tau\dot{\mathbf{v}}^2 + \left(\mathbf{I} + \left(\mathbf{W}^3\right)^T\mathbf{W}^3\right)\mathbf{v}^2 - \left(\mathbf{W}^3\right)^T\mathbf{r}^*\right] \\ &+ \mathbf{W}^2\left(\mathbf{W}^2\right)^T\mathbf{v}^2 + \mathbf{W}^2\mathbf{W}^1\mathbf{x}\end{aligned} \tag{27}$$

Grouping terms, the final PCN second-order ODE for $\mathbf{v}_2$ can therefore be expressed as:

$$\begin{aligned}\tau^2 \ddot{\mathbf{v}}^2 + \tau \underbrace{\left(2\mathbf{I} + \mathbf{W}^2\left(\mathbf{W}^2\right)^T + \left(\mathbf{W}^3\right)^T\mathbf{W}^3\right)}_{A_{\text{PCN}}}\dot{\mathbf{v}}^2 + \underbrace{\left(\mathbf{I} + \left(\mathbf{W}^3\right)^T\mathbf{W}^3 + \mathbf{W}^2\left(\mathbf{W}^2\right)^T\left(\mathbf{W}^3\right)^T\mathbf{W}^3\right)}_{B_{\text{PCN}}}\mathbf{v}^2 \\ = \underbrace{\mathbf{W}^2\mathbf{W}^1\mathbf{x} + \left(\mathbf{I} + \mathbf{W}^2\left(\mathbf{W}^2\right)^T\right)\left(\mathbf{W}^3\right)^T\mathbf{r}^*}_{C_{\text{PCN}}}\end{aligned} \tag{28}$$

C.2 Approx PC activity dynamics

In Approx PC, $\mathbf{Q}^\ell$ is approximated by the product of the Jacobians of the following layers, that is:

$$\tau\dot{\mathbf{v}}^1 = -\left(\mathbf{v}^1 - \mathbf{W}^1\mathbf{x}\right) + (\mathbf{W}^2)^T(\mathbf{W}^3)^T\left[\mathbf{r}^* - \mathbf{W}^3\mathbf{v}^2\right] \tag{29}$$

$$\tau\dot{\mathbf{v}}^2 = -\left(\mathbf{v}^2 - \mathbf{W}^2\mathbf{v}^1\right) + (\mathbf{W}^3)^T\left[\mathbf{r}^* - \mathbf{W}^3\mathbf{v}^2\right] \tag{30}$$

Expanding (29) and (30):

$$\tau\dot{\mathbf{v}}^1 = -\mathbf{v}^1 + \mathbf{W}^1\mathbf{x} + (\mathbf{W}^2)^T(\mathbf{W}^3)^T\mathbf{r}^* - (\mathbf{W}^2)^T(\mathbf{W}^3)^T\mathbf{W}^3\mathbf{v}^2 \tag{31}$$

$$\tau\dot{\mathbf{v}}^2 = -\left(\mathbf{I} + (\mathbf{W}^3)^T\mathbf{W}^3\right)\mathbf{v}^2 + \mathbf{W}^2\mathbf{v}^1 + (\mathbf{W}^3)^T\mathbf{r}^* \tag{32}$$

From (32) we isolate the $\mathbf{v}^1$-dependent term:

$$\mathbf{W}^2\mathbf{v}^1 = \tau\dot{\mathbf{v}}^2 + \left(\mathbf{I} + (\mathbf{W}^3)^T\mathbf{W}^3\right)\mathbf{v}^2 - (\mathbf{W}^3)^T\mathbf{r}^* \tag{33}$$

Differentiating (32):

$$\tau^2\ddot{\mathbf{v}}^2 = -\tau\left(\mathbf{I} + (\mathbf{W}^3)^T\mathbf{W}^3\right)\dot{\mathbf{v}}^2 + \mathbf{W}^2\left(\tau\dot{\mathbf{v}}^1\right) \tag{34}$$

Replacing $\dot{\mathbf{v}}^1$ from (31) gives:

$$\begin{aligned}\tau^2\ddot{\mathbf{v}}^2 = &-\tau\left(\mathbf{I} + (\mathbf{W}^3)^T\mathbf{W}^3\right)\dot{\mathbf{v}}^2 \\ &+ \mathbf{W}^2\left[-\mathbf{v}^1 + \mathbf{W}^1\mathbf{x} + (\mathbf{W}^2)^T(\mathbf{W}^3)^T\mathbf{r}^* - (\mathbf{W}^2)^T(\mathbf{W}^3)^T\mathbf{W}^3\mathbf{v}^2\right]\end{aligned} \tag{35}$$

And finally replacing $\mathbf{W}^2\mathbf{v}^1$ from (33) and grouping terms gives us the final DFC second-order ODE:

$$\begin{aligned}\tau^2\ddot{\mathbf{v}}^2 + \tau\underbrace{\left(2\mathbf{I} + (\mathbf{W}^3)^T\mathbf{W}^3\right)}_{A_{\text{DFC}}}\dot{\mathbf{v}}^2 + \underbrace{\left(\mathbf{I} + (\mathbf{W}^3)^T\mathbf{W}^3 + \mathbf{W}^2(\mathbf{W}^2)^T(\mathbf{W}^3)^T\mathbf{W}^3\right)}_{B_{\text{DFC}}}\mathbf{v}^2 \\ = \underbrace{\mathbf{W}^2\mathbf{W}^1\mathbf{x} + \left(\mathbf{I} + \mathbf{W}^2(\mathbf{W}^2)^T\right)(\mathbf{W}^3)^T\mathbf{r}^*}_{C_{\text{DFC}}}\end{aligned} \tag{36}$$

C.3 Residual between PCN and DFC

Comparing (28) and (36), the two second-order ODEs have identical stiffness ($\mathbf{B}$) and forcing ($\mathbf{C}$) terms, and differ only in their damping matrices:

$$\mathbf{A}_{\text{PCN}} - \mathbf{A}_{\text{DFC}} = \mathbf{W}^2(\mathbf{W}^2)^\top. \tag{37}$$

Hence any fixed point of one system is also a fixed point of the other, and the two differ only in their transient dynamics, with PC carrying an additional damping term.

D Experimental details

D.1 Hyperparameters

In this section we specify the hyperparameters used to generate the results presented in this paper. For all experiments, the proportional control constant is $K_p = 1$. For Figures 2 to 6, the width of hidden layers is eight, the input and output dimensions are one, and the neural dynamics time constant is $\tau_v = 10$. The perturbation direction is a single unit vector sampled from $\mathcal{N}(0, \mathbf{I})$ and shared across comparisons; only its magnitude is varied where specified. For Figures 2 and 3, the dynamics convergence tolerance is 1×10^{-15}. For Figures 4 to 6, we used networks with five hidden layers and a dynamics convergence tolerance of 1×10^{-8}. For Figure 6, the nudge convergence tolerance is 1×10^{-10}. For the MNIST and FashionMNIST experiments, by default, two hidden layers of width 256 are used; the neural dynamics time constant is $\tau_v = 30$; the dynamics convergence tolerance is 1×10^{-2}; the nudge convergence tolerance is 1×10^{-2}; the nudge scaling factor is $\gamma = 0.1$; networks have a bias (all other experiments have no bias); and we use the Adam optimizer with weight decay of 1×10^{-4}. Figures 2 and 3 concern linear networks (identity activation functions), while all other experiments use $\tanh$ activation functions in the hidden layers.

D.2 Computational resources

For Figures 2 to 6, we used a MacBook Pro with M2 processor and 16GB of RAM. For Figures 7 and 10 and Table 1, we used a single H100 GPU with 96GB of VRAM per learning rate and seed configuration.

E Additional experimental results

Table 1: Hyperparameter sweep results for FashionMNIST. Unless a parameter is being swept, the default values are time constant = 30, dyn. conv. tol. = 1×10^{-2}, nudge conv. tol. = 1×10^{-2}, # of layers = 2, and learning rate = 1×10^{-5} (PC) or 1×10^{-8} (Approx PC). Values are mean and standard deviation over five random seeds.

LR	Time constant	Dynamics conv. tol.	Nudge conv. tol.	# of layers	PC Acc. (%)	Approx PC Acc. (%)
(time constant)						
1×10^{-8}	5	1×10^{-2}	1×10^{-2}	2	16.38 ± 2.22	85.87 ± 0.05
1×10^{-8}	10	1×10^{-2}	1×10^{-2}	2	15.71 ± 2.45	85.92 ± 0.03
1×10^{-8}	20	1×10^{-2}	1×10^{-2}	2	14.64 ± 2.35	84.75 ± 1.03
1×10^{-8}	30	1×10^{-2}	1×10^{-2}	2	13.67 ± 2.32	85.70 ± 0.06
1×10^{-8}	40	1×10^{-2}	1×10^{-2}	2	13.02 ± 2.24	85.63 ± 0.14
1×10^{-5}	5	1×10^{-2}	1×10^{-2}	2	84.81 ± 0.11	67.73 ± 2.34
1×10^{-5}	10	1×10^{-2}	1×10^{-2}	2	83.67 ± 0.12	39.99 ± 5.15
1×10^{-5}	20	1×10^{-2}	1×10^{-2}	2	82.50 ± 0.14	23.99 ± 3.20
1×10^{-5}	30	1×10^{-2}	1×10^{-2}	2	81.41 ± 0.17	18.18 ± 0.71
1×10^{-5}	40	1×10^{-2}	1×10^{-2}	2	80.16 ± 0.23	15.80 ± 0.76
(dyn. conv. tol.)						
1×10^{-5}	30	1×10^{-3}	1×10^{-2}	2	81.49 ± 0.17	—
1×10^{-8}	30	1×10^{-3}	1×10^{-2}	2	—	85.54 ± 0.32
1×10^{-5}	30	1×10^{-2}	1×10^{-2}	2	81.46 ± 0.21	—
1×10^{-8}	30	1×10^{-2}	1×10^{-2}	2	—	85.77 ± 0.12
1×10^{-5}	30	1×10^{-1}	1×10^{-2}	2	54.07 ± 1.77	—
1×10^{-8}	30	1×10^{-1}	1×10^{-2}	2	—	78.57 ± 0.65
(learning rate sweep)						
1×10^{-9}	30	1×10^{-2}	1×10^{-2}	2	13.03 ± 2.71	75.44 ± 0.45
1×10^{-8}	30	1×10^{-2}	1×10^{-2}	2	19.52 ± 1.40	85.73 ± 0.08
1×10^{-7}	30	1×10^{-2}	1×10^{-2}	2	60.45 ± 0.26	52.47 ± 6.59
1×10^{-6}	30	1×10^{-2}	1×10^{-2}	2	79.54 ± 0.14	15.94 ± 0.32
1×10^{-5}	30	1×10^{-2}	1×10^{-2}	2	83.27 ± 0.18	17.67 ± 1.36
1×10^{-4}	30	1×10^{-2}	1×10^{-2}	2	72.71 ± 1.16	14.58 ± 1.22
1×10^{-3}	30	1×10^{-2}	1×10^{-2}	2	N/A	11.84 ± 1.81
(nudge conv. tol.)						
1×10^{-5}	30	1×10^{-2}	1×10^{-3}	2	41.45 ± 16.44	—
1×10^{-8}	30	1×10^{-2}	1×10^{-3}	2	—	85.35 ± 0.14
1×10^{-5}	30	1×10^{-2}	1×10^{-2}	2	52.05 ± 14.96	—
1×10^{-8}	30	1×10^{-2}	1×10^{-2}	2	—	81.25 ± 3.16
1×10^{-5}	30	1×10^{-2}	1×10^{-1}	2	79.48 ± 2.22	—
1×10^{-8}	30	1×10^{-2}	1×10^{-1}	2	—	84.19 ± 0.06
(# layers)						
1×10^{-5}	30	1×10^{-2}	1×10^{-2}	1	82.25 ± 0.13	—
1×10^{-8}	30	1×10^{-2}	1×10^{-2}	1	—	84.49 ± 0.47
1×10^{-5}	30	1×10^{-2}	1×10^{-2}	2	81.28 ± 0.22	—
1×10^{-8}	30	1×10^{-2}	1×10^{-2}	2	—	85.72 ± 0.13
1×10^{-5}	30	1×10^{-2}	1×10^{-2}	3	79.92 ± 1.06	—
1×10^{-8}	30	1×10^{-2}	1×10^{-2}	3	—	82.49 ± 0.92

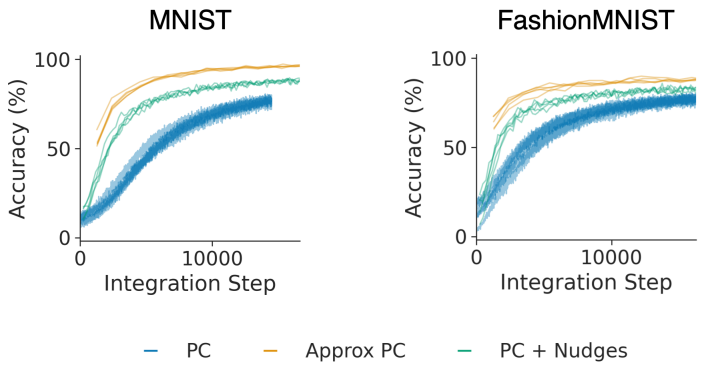

Figure 10: Results on MNIST (left) and FashionMNIST (right) for PC, Approx PC, and PC with nudges across five seeds. The total number of training samples is fixed, but the number of integration steps per sample varies. Per integration step, Approx PC and PC with nudges improve in test accuracy more than PC. The first data point is recorded after the first sample (including multiple nudges). The x-axis is cropped to zoom in on the initial learning curve ramp-up where the effect is most apparent. Note that PC takes fewer total integration steps to process the same total number of samples, which is reflected in the MNIST results where the PC learning curve stops earlier than those of the nudged variants. Learning rate $:= \tau_w^{-1} = 1 \times 10^{-6}$ (chosen so that the effect is easiest to visualize though this result holds across learning rates tested in the paper).

F Broader Impact

Our work is focused on making a theoretical link between two biologically plausible learning frameworks, and therefore has limited societal implications. However, biologically plausible learning rules, in general, contribute to work enabling energy-efficient training of artificial intelligence (AI) models. On one hand, this can enable training of state-of-the-art models, such as large language models, with limited hardware resources, allowing more individuals to access this technology and guide its development. On the other hand, more accessible training for AI models could expand misuse and documented harms caused by this technology.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We outline our claims at the end of Section 1. Our theoretical and algorithmic contributions are presented in Sections 4 and 5, respectively. We demonstrate our algorithm's performance compared to a baseline predictive coding approach in Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Section 7.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: In the main paper we specify the correspondence between activity dynamics in PC and DFC networks. We provide an extensive derivation for our main theoretical result in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide detailed pseudocode in Section 5 (Algorithm 1) and the hyperparameter values underlying our experimental results in Appendix D.1. Our code is publicly available at `https://github.com/darsnack/pc-dfc`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our code is publicly available at `https://github.com/darsnack/pc-dfc`. Both datasets used (MNIST and FashionMNIST) are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide a detailed list of hyperparameter values for each of the results presented in the paper in Appendix D.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For MNIST and FashionMNIST experiments, we report statistics over five repeats with different random seeds. Our other results are largely illustrative, and we therefore depict single runs. Qualitatively, these results are highly consistent across seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We outline our computational resources in Appendix D.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impact of our work in Appendix F.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper does not include data or models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the papers in which the MNIST and FashionMNIST datasets were introduced. Aside from standard open-source libraries (e.g., JAX, PyTorch), these are the only pre-existing assets we use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release our code at `https://github.com/darsnack/pc-dfc`, with documentation for running the experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.