ARTICLE **Communicated by Robert Rosenbaum**

Predictive Coding as a Neuromorphic Alternative to Backpropagation: A Critical Evaluation

Umais Zahid
umais.zahid@huawei.com
Huawei Technologies R&D, London N19 3HT, U.K.

Qinghai Guo
guoqinghai@huawei.com
Huawei Technologies R&D, Shenzhen 518129, China

Zafeirios Fountas
zafeirios.fountas@huawei.com
Huawei Technologies R&D, London N19 3HT, U.K.

Backpropagation has rapidly become the workhorse credit assignment algorithm for modern deep learning methods. Recently, modified forms of predictive coding (PC), an algorithm with origins in computational neuroscience, have been shown to result in approximately or exactly equal parameter updates to those under backpropagation. Due to this connection, it has been suggested that PC can act as an alternative to backpropagation with desirable properties that may facilitate implementation in neuromorphic systems. Here, we explore these claims using the different contemporary PC variants proposed in the literature. We obtain time complexity bounds for these PC variants, which we show are lower bounded by backpropagation. We also present key properties of these variants that have implications for neurobiological plausibility and their interpretations, particularly from the perspective of standard PC as a variational Bayes algorithm for latent probabilistic models. Our findings shed new light on the connection between the two learning frameworks and suggest that in its current forms, PC may have more limited potential as a direct replacement of backpropagation than previously envisioned.

1 Introduction

Predictive coding (PC) is a prominent neuroscientific theory that has emerged in the past two decades as a highly compelling computational model of perception and action in the brain (Friston, 2005; Rao & Ballard, 1999). From a theoretical standpoint, PC, in its standard formulation, has benefited from its interpretation as a variational Bayes algorithm for learning and inference in latent hierarchical models (Bogacz, 2017; Friston, 2003,

Neural Computation **35, 1881–1909** (2023)
https://doi.org/10.1162/neco_a_01620

2005; Friston & Kiebel, 2009), providing it credence as a plausible instantiation of normative theories such as the Bayesian brain hypothesis and the free-energy principle. More empirically, its candidate neurobiological implementation has demonstrated impressively close correspondences to the canonical circuitry of the cortex (Bastos et al., 2012; Shipp, 2016), while also successfully explaining or reproducing a number of neurophysiological and cognitive phenomena, such as end stopping (Rao & Ballard, 1999), binocular rivalry (Hohwy et al., 2008), attention (Feldman & Friston, 2010; Kanai et al., 2015), and biases in the perception of event duration (Fountas et al., 2022).

More recent work has suggested a connection between modified forms of PC, and backpropagation, suggesting the former as an alternative to the latter with the various benefits one may naturally expect to associate for a theory formulated with neurobiological constraints in mind (Millidge et al., 2022, 2020b; Song et al., 2020). These include locality of computation, and parallelizability that may lend it improved performance and greater amenability to implementation on neuromorphic hardware.

The purpose of this work is to investigate these suggestions and clarify more precisely the relationship between backpropagation and PC, particularly with regard to the kinds of assumptions that are necessary to enable such a comparison and, furthermore, define exactly what advantage PC-based formulations may have with respect to parallelizability, locality, and ultimately practical performance. In what follows, we re-review some of the ground covered by Rosenbaum (2022), while extending it to discuss the implications for PC in general with respect to computational and memory complexity, as well as compute and memory locality. We also present proofs for lower bounds on the time complexity of current PC variants relative to backpropagation and validate these proofs empirically. Our work finds that while the examined algorithms present interesting, biologically inspired, alternative implementations of backpropagation, they are provably guaranteed to be slower under identical problem settings and that it is unclear to what degree they improve on locality, particularly in comparison to modern candidate proposals for backpropagation. Finally, we note that when adopting the modifications required to enable such comparisons, PC models diverge from the generative variational Bayes framework, frequently used as the motivational basis of this theory.

2 Backpropagation

Backpropagation, introduced by Linnainmaa (1976) in the context of accumulating rounding errors and later popularized by Rumelhart et al. (1986) in the context of neural networks, has quickly become the workhorse credit assignment algorithm for modern deep learning tasks. Described most simply, backpropagation can be seen as an algorithmically efficient implementation of the chain rule from calculus that allows the computation of high-dimensional gradients for deeply nested functions with respect to

scalar or low-dimensional outputs. It accomplishes this by defining gradients of values deep inside a nested function in terms of the gradients of their children (their functional dependents). The resultant recursive relationship is then evaluated for values closest to the output, sequentially backward. This recursive relationship is what enables backpropagation to efficiently compute gradients with respect to all parameters in the function chain with a single forward and backward computational pass.

To illustrate how this operates, we present the following simple but highly generic problem setting. We consider an arbitrary function F, defined by the composition of many constituent functions f_i, each optionally parameterized by some set of parameters $\boldsymbol{\theta_i}$. We may then write the following:

$$\begin{aligned} f_0(x_0, \theta_0) &= \mu_1 \\ f_1(\mu_1, \theta_1) &= \mu_2 \\ &\cdots \\ f_{(L-1)}(\mu_{L-1}, \theta_{L-1}) &= \mu_L \\ f_L(\mu_L, x_L) &= E, \end{aligned} \tag{2.1}$$

where we use μ_i to denote the intermediary outputs of our composite function and x_0, x_L to denote known values that are provided as inputs and outputs of our function, respectively. x_0 and x_L may, for example, be inputs and classification labels, respectively; μ_ℓ may be intermediate activation vectors for a multilayer perceptron and f_L a classification loss.

If we wish to compute the gradient of any intermediate parameters with respect to our function output E, we may then do so by first defining the following simple recursive relationship:

$$e_\ell = \left\{ \begin{array}{ll} \frac{\partial f_\ell}{\partial \mu_\ell}^T e_{\ell+1} & \text{for } \ell = 1, \ldots, L-1 \\ \frac{\partial f_\ell}{\partial \mu_\ell}^T & \text{for } \ell = L \end{array} \right\}, \tag{2.2}$$

For multivariate elementary functions, the $\frac{\partial f_\ell}{\partial \mu_\ell}$ terms would correspond to the Jacobian matrices for the elementary functions f_i with respect to μ_ℓ. And by virtue of the chain rule, the error terms e_ℓ would thus equal the (transposed) gradient of E with respect to μ_ℓ: $\frac{\partial E}{\partial \mu_\ell}^T$.

Having established error vectors e_ℓ that correspond to the gradient with respect to μ_ℓ, one may then compute the gradient of output E with respect to an arbitrary intermediate parameter by matrix-multiplying the error vector from the next layer by the Jacobian of the function with respect to the parameters:

$$\frac{\partial E}{\partial \theta_\ell} = -e_{\ell+1}^T \frac{\partial \mu_{\ell+1}}{\partial \theta_\ell}. \tag{2.3}$$

Though obvious in this scenario, it will be important for our later comparisons to mention here that all Jacobians mentioned above are evaluated at the values of the feedforward outputs of our function, which, as we will discuss, may not necessarily be the case for Bayesian PC. We also note that in practice, the Jacobians specified in equations 2.3 and 2.2 are almost never instantiated explicitly. Rather, an automatic differentiation program that implements backpropagation would associate each elementary or constituent function with an associated vector-Jacobian product (VJP) function, which is generally far more computationally and memory efficient (Bradbury et al., 2018; Paszke et al., 2017).

2.1 Computational and Time Complexity. The cost of computing the forward pass (see equation 2.1 and subsequent backward recursions (see equations 2.2 and 2.3) can be shown to have computational and time cost bounded by a small constant multiple of the cost for a single forward pass through the corresponding elementary function. This well-known result from the automatic differentiation literature is sometimes called the cheap-gradient principle and is one reason why backpropagation has been so successful as a credit assignment algorithm in modern large data settings. This constant was shown to be 3 for rational functions in the seminal work of Baur and Strassen (1983) and 5 more generally for any function composed of elementary arithmetic and trigonometric functions (Griewank, 1997, 2009; Griewank & Walther, 2008). The precise value of this constant generally also depends on details regarding the relative costs of various basic mathematical operations (such as addition, multiplication, memory access, and nonlinear operations) on the specific hardware being considered. For modern deep neural network operations and associated hardware, this constant is generally taken to be 3 (Hoffmann et al., 2022; Kaplan et al., 2020), and we adopt this for our comparison.

As we will see, the basic unit of computation for both the current implementations of PC and backpropagation algorithms is the same: the VJP. The cost of computing this VJP is bounded in terms of a forward evaluation via the cheap gradient principle. To levy a comparison between algorithms, we may therefore report computational complexity with respect to the cost of computing a single forward pass, which we denote with $\mathcal{C}_i$ and $\mathcal{C}_F$, for elementary constituent functions f_i, and the overall composite function F respectively.

Following the notation in Griewank and Walther (2008), we denote a computational task associated with a function F as $task(F)$, with the corresponding computational and time complexity referred to as $\text{WORK}\{task(F)\}$ and $\text{TIME}\{task(F)\}$. For simplicity, we assume scalar complexity measures for work and time and relate the time complexity to work complexity for a single elementary function f_i using some positive valued constant w, such that $\text{TIME}\{task(f_i)\} = w\text{WORK}\{task(f_i)\}$. Note that this relationship may not be true for other high-level tasks on F such as PC inference due to

parallel computation, which we will discuss and accommodate for in subsequent sections. Using the cheap gradient principle, we can therefore write the following factorization and bound for the time complexity of backpropagation:

$$\begin{aligned}\text{TIME}\{backprop(F)\} &= \text{TIME}\{forward(F)\} + \text{TIME}\{backward(F)\} && (2.4)\\ &= \sum_i \text{TIME}\{forward(f_i)\} + \sum_i \text{TIME}\{vjp(f_i)\} && (2.5)\\ &\leq w \sum_i \mathcal{C}_i + 2\mathcal{C}_i = 3w \sum_i \mathcal{C}_i. && (2.6)\end{aligned}$$

3 Predictive Coding

3.1 Variational PC. We begin by describing the standard formulation of PC as a variational Bayes algorithm (Bogacz, 2017; Buckley et al., 2017; Friston, 2003, 2005). Within this context, the PC algorithm can be interpreted as a method for performing inference (over latent states) and learning (over parameters) for a latent hierarchical generative model. In keeping with much of recent work that renders a comparison with backpropagation we will restrict our focus to PC for static (i.e., nondynamical/time-series-based) observations and states. We will distinguish this formulation from recent modified formulations such as FPA-PC (Millidge et al., 2020) and Z-IL (Salvatori et al., 2021; Song et al., 2020) by calling it variational PC (VPC) to emphasize that modified formulations may not necessarily correspond to a variational Bayesian inference and learning algorithm.

For the sake of enabling our subsequent comparisons of PC with backpropagation, in this section we will reuse the elementary functions f_i, first defined in section 2, for defining the conditional relationships within our hierarchical model.

In particular, we will take the outputs of our constituent functions f_i to correspond to means of gaussian latent random variables, with each gaussian latent random variable conditioning on its parents. Then, when f_L is a loss function corresponding to the log likelihood of a particular choice of output distribution P_L (a common problem setting), this results in the following log-joint probability for our probabilistic graphical model:

$$\begin{aligned}&\log p(x_0, \ldots, x_{L-1}, x_L | \theta_0, \ldots \theta_L)\\ &= \log p(x_L | x_{L-1}, \theta_{L-1}) + \ldots + \log p(x_1 | x_0, \theta_0) \qquad (3.1)\end{aligned}$$

with

$$x_\ell | x_{\ell-1} \sim \begin{cases} \mathcal{N}(f_{\ell-1}(x_{\ell-1}, \theta_{\ell-1}), \Sigma_\ell)) & \text{for } \ell = 1, \ldots, L-1 \\ P_L(f_{\ell-1}(x_{\ell-1}, \theta_\ell)) & \text{for } \ell = L \end{cases}. \qquad (3.2)$$

Given this model, PC answers the question of how one can learn the parameters $\boldsymbol{\theta}$ that maximize their model evidence, $\log p(\mathbf{x}_{\text{OBS}}^{(1)}, \ldots \mathbf{x}_{\text{OBS}}^{(D)}|\boldsymbol{\theta})$, where for notational simplicity, we have combined all observations (x_0 and x_L), latent states ($x_1, \ldots, x_{L-1}$), and parameters ($\theta_0, \ldots, \theta_L$) into the vectors $\mathbf{x}_{\text{OBS}}$, $\mathbf{x}_{\text{LAT}}$, $\boldsymbol{\theta}$, respectively, and superscript indices denote samples from a data-generating (observation) distribution $\mathcal{X}$.

The standard difficulty with this optimization of the model parameters to maximize the model evidence rests on the intractable marginalization over latent states ($\mathbf{x}_{\text{LAT}}$). The variational Bayes solution to this issue rests on the optimization of an evidence lower bound (ELBO), equivalently often called the (negative) free energy. This bound relies on a tractable form for an approximate posterior density over latent states:

$$\text{ELBO} = E_{q(\mathbf{x}_{\text{LAT}})}\left[\log p(\mathbf{x}_{\text{OBS}}, \mathbf{x}_{\text{LAT}}|\boldsymbol{\theta})\right] - E_{q(\mathbf{x}_{\text{LAT}})}\left[\log q(\mathbf{x}_{\text{LAT}})\right]. \tag{3.3}$$

For the approximate posterior density $q(\mathbf{x}_{\text{LAT}})$, PC adopts a point-mass (Dirac δ) distribution, either implicitly (Bogacz, 2017; Buckley et al., 2017; Millidge et al., 2020) or explicitly (Friston, 2003, 2005). By denoting the center of this Dirac δ delta distribution with $\boldsymbol{\phi}$, the above bound simplifies to the objective,

$$\text{ELBO}_{\mathbf{PC}}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \log p(\mathbf{x}_{\text{OBS}}, \boldsymbol{\phi}|\boldsymbol{\theta}). \tag{3.4}$$

PC optimizes this function by first enacting an ascent with respect to the Dirac δ parameters ϕ corresponding to the modes of the approximate posterior over latent states. Once the maxima for the log joint probability with respect to ϕ is obtained, we then update our parameters θ by computing the gradient with respect to the log joint evaluated at these values of ϕ.

Depending on one's perspective, this initial step can be seen variously as obtaining an MAP (maximum a posteriori) estimate over latent states (due to the maximization of the log-joint) or as an expectation step within an expectation-maximization scheme (Friston, 2005), or as a variational bound tightening step from a variational Bayes perspective. The subsequent maximization step (optimization of θ) can then be implemented via a mini-batch or stochastic gradient descent (SGD) procedure allowing one to tractably optimize against a large data set of observations. The final algorithm can be summarized succinctly as follows:

1. Define a (possibly hierarchical) graphical model over latent states ($\mathbf{x}_{\text{LAT}}$) and observations ($\mathbf{x}_{\text{OBS}}$) with parameters $\boldsymbol{\theta}$ (i.e., $\log p(\mathbf{x}_{\text{OBS}}, \mathbf{x}_{\text{LAT}}|\boldsymbol{\theta})$)
2. For minibatch $\mathbf{x}_{\text{OBS}} \sim \mathcal{D}$, where $\mathcal{D}$ is the data-generating distribution:

   Inference: Obtain MAP estimates ($\mathbf{x}_{\text{MAP}}$) for the latent states by enacting a gradient descent on $-\log p(\mathbf{x}_{\text{OBS}}, \mathbf{x}_{\text{LAT}}|\boldsymbol{\theta})$.

Learning: Update the parameters $\boldsymbol{\theta}$ using SGD with respect to the negative log joint evaluated at the MAP estimates found at the end of inference: $-\log p(\mathbf{x}_{\text{OBS}}, \mathbf{x}_{\text{MAP}}|\boldsymbol{\theta})$.

For the sake of subsequent discussion, it will be useful to look at the exact functional form and computations occurring in the gradient ascent (inference) procedure outlined in the above algorithm. First, since the conditional log likelihoods of latent states in the log joint described by equation 3.1 are gaussian, these therefore take the form of a series of squared precision-weighted prediction errors, each corresponding to a constituent function f_i, plus the output log likelihood,

$$\begin{aligned}
\text{ELBO}_{\text{PC}} &= \log p(x_0, x_1, \ldots, x_{L-1}, x_L) && (3.5)\\
&= f_L(x_L, x_{L-1}) && (3.6)\\
&\quad + \ldots\\
&\quad + (x_\ell - f_{\ell-1}(x_{\ell-1}))^T \Sigma_\ell^{-1}(x_\ell - f_{\ell-1}(x_{\ell-1}))\\
&\quad + \ldots\\
&\quad + (x_1 - f_0(x_0))^T \Sigma_1^{-1}(x_1 - f_0(x_0)), && (3.7)
\end{aligned}$$

where we have temporarily removed the dependence on parameters θ_ℓ for brevity.

The gradient of this objective with respect to a particular latent state x_ℓ is then simply the precision-weighted prediction errors corresponding to predictions of x_ℓ from its parents as well as precision-weighted prediction errors corresponding to predictions that x_ℓ makes of its child nodes $x_{\ell+1}$. The gradient descent (inference) procedure can therefore be described by the following discretized gradient flow, assuming a gaussian output log likelihood for f_L (i.e., Euclidean output loss):

$$\begin{aligned}
x_{\ell,t+1} &= x_{\ell,t} - \gamma \left.\frac{\partial F}{\partial x_\ell}\right|_{x_{\ell,t}} && (3.8)\\
&= x_{\ell,t} - \gamma \left[\Sigma_\ell^{-1}(x_{\ell,t} - f_{\ell-1}(x_{\ell-1,t})) - \left.\frac{\partial f_\ell}{\partial x_\ell}\right|_{x_{\ell,t}}^T \Sigma_{\ell+1}^{-1}(x_{\ell+1,t} - f(x_{\ell,t}))\right], && (3.9)
\end{aligned}$$

where γ is some inference step size. Under the assumption of identity variances, this equation simplifies to

$$= x_{\ell,t} - \gamma \left[(x_{\ell,t} - f_{\ell-1}(x_{\ell-1,t})) - \left.\frac{\partial f_\ell}{\partial x_\ell}\right|_{x_{\ell,t}}^T (x_{\ell+1,t} - f(x_{\ell,t}))\right]. \qquad (3.10)$$

Written more generically in terms of errors, we can write this to include arbitrary definitions of output loss log likelihood

$$= x_{\ell,t} - \gamma \left[e_{\ell,t} - \frac{\partial f_\ell}{\partial x_\ell}\bigg|_{x_{\ell,t}}^{T} e_{\ell+1,t} \right] \tag{3.11}$$

with

$$e_{\ell,t} = \begin{Bmatrix} (x_{\ell,t} - f_{\ell-1}(x_{\ell-1,t})) & \text{for } \ell = 1, \ldots, L-1 \\ \frac{\partial f_\ell}{\partial f_{\ell-1}}^{T} & \text{for } \ell = L \end{Bmatrix}, \tag{3.12}$$

where we abuse notation and use f_{L-1} to denote both the function and its output evaluated at the current value of x_{L-1}, that is, $f_{L-1}(x_{L-1,t})$. We do this to distinguish it from μ_L, and the associated Jacobian in equation 2.2, which denotes the pre-loss output prediction computed using the feedforward value μ_{L-1}, that is, $f_{L-1}(\mu_{L-1})$. That is, at every inference step, the latent states are updated such that they act to minimize the error corresponding to the current prediction of their latent states and the error corresponding to the current prediction of their children's latent states.

The model parameters are then updated (via SGD) with the derivative of the log joint evaluated at these converged MAP values. This gradient is the product of the error associated with the conditional distribution it parameterizes and the Jacobian of that function with respect to θ. Let t_c be the time at which the variational modes have converged. Then the gradient of the log joint with respect to the model parameters is given by

$$\frac{\partial F}{\partial \theta_\ell} = e_{\ell+1,t_c}^{T} \frac{\partial f_\ell}{\partial \theta_\ell}. \tag{3.13}$$

Equations 3.9 and 3.10 describe the dynamics of the standard PC formulation. Under these dynamics, inference over latent states can be considered MAP inference on a corresponding probabilistic model, and learning can be considered as the maximization of an ELBO with respect to its model parameters. In particular, this bound corresponds to a point-mass variational distribution assumed over our latent states.

3.1.1 Inverted Configurations. This particular formulation of PC is inverted relative to the standard PC formulation (Buckley et al., 2017; Friston, 2003, 2008; Rao & Ballard, 1999) present in cognitive science and, indeed, most generative modeling schemes, in the sense that observations parameterize, and are thus hierarchically higher than, latents such as classification labels. Inverting PC in this way, to render a comparison with backpropagation, has nontrivial implications on neurobiological plausibility. One of

the strengths of standard PC formulations has historically been in the notion that latents hierarchically higher in the cortex have a nonlinearly mixing and modulatory effect ($f(x_\ell, \theta_\ell)$)) that is congruent with what we know about cortical—that top-down backward connections are generally more bifurcating and modulatory, while bottom-up influences, or forward connections, are more driving ($-e_\ell$) (Bastos et al., 2012; Friston, 2005, 2008; Markov et al., 2014; Shipp, 2016).

It is interesting to note that supervised generative (i.e., noninverted) formulations of standard PC have thus far struggled to produce competitive or close to competitive results for complex classification tasks. For example, traditional generative PC models trained to classify and generate MNIST digits are consistently unable to surpass a test set accuracy of 80%, both from our experience and observed by Kinghorn et al. (2022). We are not aware of any example of a traditional (generative) PC succeeding in obtaining competitive classification performance on standard machine learning tasks. Though the possible reasons for this are outside the scope of this article, it is likely a consequence of either the gaussian-based supervision signal generally used in existing attempts or the limitations of the Dirac δ (point-mass) approximate posterior, for training generative models, as highlighted in Zahid et al. (2023).

3.2 Memory Locality, Weight Transport, and Weight Symmetry. Local computation within a neurobiological context generally refers to local plasticity, synaptic weight updates that are determined by the activity of the neurons they connect, that is, presynaptic and postsynaptic neuronal activity and synaptic locality. This notion of locality is often used in the context of assessing learning algorithms for their plausibility of implementation in the brain. Historically, a key criticism in the biological plausibility of backpropagation has been due to the presence of this type of nonlocality, where it has been called the weight transport problem (Crick, 1989; Grossberg, 1987; Zipser & Rumelhart, 1993).

Under the problem setting described above equation 2.1, the weight transport problem can be summarized as the issue of a particular weight or parameter matrix (e.g., θ_ℓ) involved in the computation of feedforward activation $\mu_{\ell+1}$, being transported for use in the computation of errors e_ℓ. Specifically, if one considers a constituent function f_ℓ consisting of an affine transformation given by the weight matrix θ_ℓ and a nonlinearity $g(\cdot)$, such that

$$\mu_{\ell+1} = f_\ell(\mu_\ell, \theta_\ell) = g(\theta_\ell \mu_\ell), \tag{3.14}$$

the computation of e_ℓ is then

$$e_\ell = \frac{\partial \mu_{\ell+1}}{\partial \mu_\ell}^T e_{\ell+1} = g'(\theta_\ell \mu_\ell)\theta_\ell^T e_{\ell+1}, \tag{3.15}$$

where we require reusing the forward weights due to the transpose weight term θ_ℓ^T.

With respect to backpropagation, there have been a number of well-known attempts at resolving the implausibility of the weight transport issue. These include the influential work of Lillicrap et al. (2016), which demonstrated that computing the error via a fixed random matrix B replacing θ_ℓ could nonetheless facilitate learning without the requirement that the forward parameters be reused. However, the use of random feedback weights remained performant only for shallow networks, failing for deeper models. Subsequent and more recent work has demonstrated that training these feedback weights separately (Akrout et al., 2019; Amit, 2019) or using meta-learned plasticity rules for the forward weights (Shervani-Tabar & Rosenbaum, 2023) may improve performance for deeper models.

While these various models resolve the weight transport issue in one way or another, they frequently require or induce additional assumptions about the neurobiological machinery required for their implementation. The work of Zipser and Rumelhart (1993), for example, required initializing feedforward and feedback matrices identically, which was criticized and resolved using weight decay by Kolen and Pollack (1994). Both Kolen and Pollack (1994) and Zipser and Rumelhart (1993), however, required transmitting the synaptic weight changes between separate networks, which has itself also been argued as implausible (Akrout et al., 2019). Similarly, both Lillicrap et al. (2016) and Akrout et al. (2019) require interactions between the activations within forward and backward pathways, usually implemented as three-factor-style learning rules, which induce a complexity beyond simple two-factor Hebbian learning rules. And while there exists a growing body of literature presenting empirical evidence or nascent mechanistic proposals for such learning rules (Pawlak et al., 2010; Sjöström & Häusser, 2006; Urbanczik & Senn, 2014), they nonetheless induce a complexity beyond simple two-factor Hebbian learning, while also frequently requiring that modulators do not affect postsynaptic activity in the feedforward path. Finally, for completeness, we note that Amit (2019) required strict and regimented sequencing of computation, with feedforward neurons subsequently also acting as error neurons, which can be seen diagrammatically in Figure 1.

To better understand the relationship of these various proposals, one can decompose the general issue of neurobiological plausibility and weight transport into two distinct sets of issues: weight nonlocality (synaptic modification based on distinct error and value activity) and that of weight symmetry (parity between reciprocating connections between the same or different sets of neurons) (roughly mapping to constraints 2.2 and 2.3, respectively, from Payeur et al., 2021). In the backpropagation literature, solutions to this first issue generally appeal to either some form of signal multiplexing (i.e., reuse of the same neuron for distinct activities in either time; Amit, 2019), or space (via, for example, apical, dendritic, and somatic compartments (see Sacramento et al., 2018; Whittington & Bogacz, 2019;

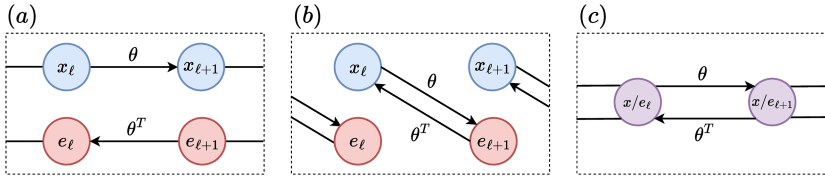

Figure 1: (a) Weight nonlocality problem in a naive proposal implementation for backpropagation. (b) Weight symmetry issue in standard PC formulations. Note that we exclude additional connectivity necessary for PC, but not relevant to the weight transport problem, for the sake of clarity. (c) An example alternative proposal aimed at solving the weight nonlocality issue for backpropagation (Amit, 2019).

and the discussion in Shervani-Tabar & Rosenbaum, 2023), or they appeal to third-factor learning rules (Lillicrap et al., 2016), while attempts at resolving the latter issue generally appeal to the use of fixed (Lillicrap et al., 2016) or updating (Amit, 2019) random feedback and/or decay (Kolen & Pollack, 1994).

Having distinguished these issues, we can now note that a similar issue to that of weight symmetry exists within PC due to the presence of the Jacobian term ($\frac{\partial f_\ell}{\partial x_\ell}$) in equations 3.9 and 3.10. For the same definition of f_ℓ (affine + nonlinearity), this results in the following inference equations for standard PC,

$$x_{\ell,t+1} = x_{\ell,t} - \gamma \left[e_{\ell,t} - \frac{\partial f_\ell}{\partial x_\ell}\Big|^T_{x_{\ell,t}} e_{\ell+1,t} \right] \tag{3.16}$$

$$= x_{\ell,t} - \gamma \left[e_{\ell,t} - g'(\theta_\ell \mu_\ell)\theta_\ell^T e_{\ell+1,t} \right], \tag{3.17}$$

with θ_ℓ also being used in the computation of $e_{\ell+1}$

$$e_{\ell+1,t} = x_{\ell+1,t} - g(\theta_\ell x_\ell). \tag{3.18}$$

Thus, the negative of the weights that mediate the influence of the subsequent error units ($e_{\ell+1}$) on the previous latent states (x_ℓ) is also required for the reciprocal connection from latent states to error units. (See Figure 1 for a depiction of this.) This property was identified in some of the earliest narratives on PC (Friston, 2005, 2008), albeit from the perspective of an advantage of THE PC's neurobiological plausibility due to the reported prevalence of reciprocal connections in the brain (Felleman & Van Essen, 1991). Prevalence of reciprocal connections is not sufficient, however, as we, at least naively, require exact equivalence in strengths also.

However, while the issue of weight symmetry remains, by mediating value neuron activity transmission through error neurons and their dynamics and relying on converged value and error neuron activity to update

weights, standard PC abrogates the need to update weights nonlocally using activities between separate populations of value and error neurons. This benefit is not obtained for free, and the converged error values that result from standard PC dynamics are generally distinct from those obtained by backpropagation. Thus, to obtain parity with backpropagation, modified forms of PC adopt changes to modifications that themselves may be neurobiologically implausible. These include rescaling the synaptic learning rate up to infinity (see section 3.3), requiring postsynaptic value neuron activity to remain fixed and stored somewhere throughout multiple steps of inference and error computation (see section 3.4), or requiring strictly regimented activity propagation and weight updates (see section 3.5).

Here we have distinguished two closely related issues of weight nonlocality and symmetry as they appear to have distinct considerations and solutions. In particular, weight symmetry is arguably a more addressable issue from the perspective of neurobiological plausibility over weight nonlocality. The reason is that plasticity is generally considered to be some simple function of presynaptic and postsynaptic activity. Given that both of these activities are accessible for forward and reciprocal synapses, it is not implausible that forward and backward weights could come into parity. In particular, given a simple two-factor Hebbian learning rule, one would expect, at least for simple Hebbian plasticity, that both forward and reciprocal connections could experience the same weight modifications (the product of presynaptic and postsynaptic activity). Further incorporating weight decay would result in weights that eventually synchronize, as demonstrated for the weight transport case by Kolen and Pollack (1994). This approach, without decay, has indeed been demonstrated with minimal performance degradation on standard PC networks (Millidge et al., 2020a).

We next discuss a number of the modified formulations of PC that have recently been presented as neurmorphic alternatives to backpropagation.

3.3 Backpropagation as PC in the Infinite Variance Limit. The link between PC and backpropagation was first presented in Whittington and Bogacz (2017), where it was shown that the resultant weight updates under the PC algorithm outlined above would equal that for the equivalent backpropagation graph given the following specific restrictive set of circumstances:

Requirement 1

- When the variational modes or latent states $(x_1, \ldots, x_{L-1})$ are initialized to the feedforward values of the corresponding backpropagation-based computational chain $(\mu_1, \ldots, \mu_{L-1})$ and:

Requirement 2

- Case 1. When the feedforward prediction results in zero output loss (i.e., zero negative log likelihood loss)

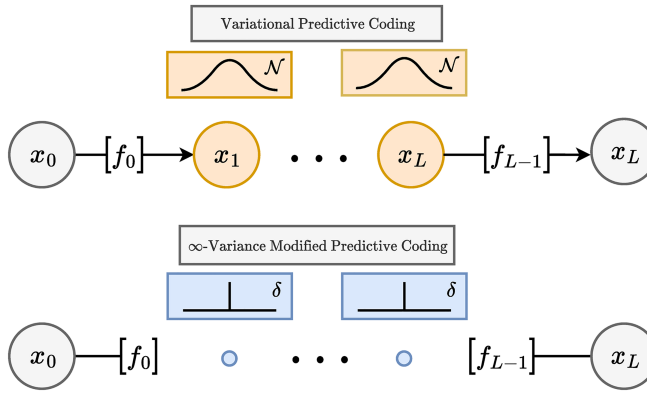

Figure 2: Underlying probabilistic model assumed by standard, that is, variational, PC (VPC) and recent modified forms of PC. Note that VPC incorporates probabilistic latent states, whereas FPA-PC assumes deterministic states and observations only. Gray circles correspond to observed variables, blue circles correspond to deterministic intermediate states, and orange circles correspond to probabilistic latent states.

- Case 2. When the output variance (Σ_L) is set sufficiently higher than the remaining variances ($\Sigma_{\ell<L}$), such that the ratio of $(\frac{\Sigma_L}{\Sigma_\ell}, \forall \ell < L)$ goes to infinity *and* the learning rate is scaled appropriately

Both of these cases correspond to the same requirement, namely, that the values of the latent states, $x_1, \ldots, x_{L-1}$ remain equal or close to their feedforward values at the end of the inference procedure. Said another way, this can be seen as the requirement that the MAP values for the latent states while observing both input observations (e.g., images) and output observations (e.g., classification labels) are equal to the MAP values when conditioning on input observations alone. Another more intuitive framework for understanding this requirement is to note that the inference and learning procedure that results under these sets of circumstances is equivalent to doing inference and learning for a probabilistic graphical model in which the intermediate latent states have zero conditional uncertainty, or variance, and are thus deterministic outputs of the input observations.

Given this framework, learning devolves into enacting maximimum likelihood learning for a probabilistic graphical model in which inputs x_0 parameterize an output likelihood via the deep "total" network function $F = f_L \circ f_{L-1} \ldots \circ f_1 \circ f_0$, with observations x_L. This is equivalent to enacting maximum likelihood training with backpropagation when the output loss function f_L (see equation 2.1) corresponds to a valid log likelihood. (See Figure 2 for a depiction of the probabilistic graphical model assumed by this approach versus standard VPC.)

To see this, observe that this is trivially true for case 1 where the weight updates from the deterministic latent maximum likelihood objective and the PC objective with probabilistic latent states are the same, namely zero, due to the zero error.

More interestingly, this can also be seen for case 2 if one considers that scaling up Σ_L and rescaling all learning rates by its inverse is equivalent to scaling down the intermediate variances $\Sigma_\ell, \forall \ell < L$, while keeping the output variances fixed and the learning rate the same. This is arguably a more natural perspective to view this procedure as it allows us to understand what is happening to the underlying probabilistic graphical model that we are training as we do this: that we are removing uncertainty over latent states and thus going from a variational Bayes (free energy minimizing) learning procedure to a maximum likelihood learning procedure. To demonstrate this, we can look at what happens to the inference and learning equations as we conduct the case 2 scaling procedure from Whittington and Bogacz (2017).

Consider that we have scaled the label variances Σ_L by a large constant factor k and subsequently scaled the parameter learning rate (α) by $\frac{1}{k}$ to compensate. This results in the following modified inference and learning equations (previously equations 3.9 and 3.13), again assuming a gaussian output log likelihood (squared Euclidean output loss),

$$x_{\ell,t+1} = x_{\ell,t} - \gamma \begin{Bmatrix} \Sigma_\ell^{-1}(x_{\ell,t} - f_{\ell-1}(x_{\ell-1,t})) & \\ \quad - \frac{\partial f_\ell}{\partial x_\ell}\Big|_{x_{\ell,t}}^T \Sigma_{\ell+1}^{-1}(x_{\ell+1} - f(x_{\ell,t})) & \text{for } \ell = 1, \ldots, L-2 \\ \Sigma_\ell^{-1}(x_{\ell,t} - f_{\ell-1}(x_{\ell-1,t})) & \\ \quad - \frac{\partial f_\ell}{\partial x_\ell}\Big|_{x_{\ell,t}}^T (\frac{1}{k} \cdot \Sigma_{\ell+1}^{-1})(x_{\ell+1} - f(x_{\ell,t})) & \text{for } \ell = L-1 \end{Bmatrix},$$

which may be rewritten as

$$x_{\ell,t+1} = x_{\ell,t} - \gamma' \begin{Bmatrix} (k \cdot \Sigma_\ell^{-1})(x_{\ell,t} - f_{\ell-1}(x_{\ell-1,t})) & \\ \quad - \frac{\partial f_\ell}{\partial x_\ell}\Big|_{x_{\ell,t}}^T (k \cdot \Sigma_{\ell+1}^{-1})(x_{\ell+1} - f(x_{\ell,t})) & \text{for } \ell = 1, \ldots, L-2 \\ (k \cdot \Sigma_\ell^{-1})(x_{\ell,t} - f_{\ell-1}(x_{\ell-1,t})) & \\ \quad - \frac{\partial f_\ell}{\partial x_\ell}\Big|_{x_{\ell,t}}^T (1 \cdot \Sigma_{\ell+1}^{-1})(x_{\ell+1} - f(x_{\ell,t})) & \text{for } \ell = L-1 \end{Bmatrix},$$

where, for inference, the $\frac{1}{k}$ factor has been absorbed by a new step size γ' and, thus, given Euler integration does not diverge, will result in the same converged errors at the end of inference. The updates to our parameters θ_ℓ are then

$$\Delta\theta_\ell = k \cdot \alpha \left\{ \begin{array}{ll} \frac{\partial f_\ell}{\partial \theta_\ell}^T \Sigma_{\ell+1}^{-1}(x_{\ell+1} - f(x_{\ell,t})) & \text{for } \ell = 1, \ldots, L-2 \\ \\ \frac{\partial f_\ell}{\partial \theta_\ell}^T (\frac{1}{k} \cdot \Sigma_{\ell+1}^{-1})(x_{\ell+1} - f(x_{\ell,t})) & \text{for } \ell = L-1 \end{array} \right\},$$

which may be rewritten as

$$\Delta\theta_\ell = \alpha \left\{ \begin{array}{ll} \frac{\partial f_\ell}{\partial \theta_\ell}^T (k \cdot \Sigma_{\ell+1}^{-1})(x_{\ell+1} - f(x_{\ell,t})) & \text{for } \ell = 1, \ldots, L-2 \\ \\ \frac{\partial f_\ell}{\partial \theta_\ell}^T (1 \cdot \Sigma_{\ell+1}^{-1})(x_{\ell+1} - f(x_{\ell,t})) & \text{for } \ell = L-1 \end{array} \right\}.$$

Therefore, the modification in case 2 corresponds to downscaling the variances of the intermediate latent states toward zero while keeping the output likelihood variance fixed. Obtaining exact equivalence to the equivalent backpropagation-based learning update for this nonlatent graph would require taking k to infinity, or more practically for getting approximate equivalence, some reasonably large value; Whittington and Bogacz (2017) use $k = 100$ for a small network. But this scaling clearly runs the risk of being numerically unstable and is also biologically unrealistic.

3.4 The Fixed-Prediction Assumption. An alternative to the scaling procedure mentioned above, dubbed the *fixed-prediction assumption* (FPA) was presented in Millidge et al. (2020), wherein the top-down predictions $f(x_\ell)$ and Jacobians $\left(\frac{\partial f_\ell}{\partial x_\ell}\right)$ corresponding to each latent were fixed to their feedforward predictions $\mu_{\ell+1}$ and $\left(\frac{\partial f_\ell}{\partial x_\ell}|_{\mu_\ell}\right)$, respectively, throughout the inference procedure.

The FPA modification of PC fixes the following terms in the discretized gradient flow to specific values corresponding to their feedforward initialization—that is,

$$f_{\ell-1}(x_{\ell-1,t}) = f_{\ell-1}(x_{\ell,0}) = f_{\ell-1}(\mu_{\ell-1}), \tag{3.19}$$

$$f_\ell(x_\ell, t) = f_\ell(x_{\ell,0}) = f_\ell(\mu_\ell), \tag{3.20}$$

$$\left.\frac{\partial f_\ell}{\partial x_\ell}\right|_{x_{\ell,t}} = \left.\frac{\partial f_\ell}{\partial x_\ell}\right|_{x_{\ell,0}} = \left.\frac{\partial f_\ell}{\partial x_\ell}\right|_{\mu_\ell}, \tag{3.21}$$

where we have assumed requirement 1—that the latent states are initialized at the feedforward values of the corresponding computational chain. We refer to the inference updating equation corresponding to these FPA

modifications as $u(x_{\ell,t}, x_{\ell+1,t})$, for a particular latent x_ℓ, to show that it is dependent on the instantaneous value of only the $x_{\ell,t}$ and its child variable $x_{\ell+1,t}$, and not the parent latent states $x_{\ell-1}$. This decoupling of the dynamics of each latent state from the values of its parents means that the trajectory of the latent states is no longer guaranteed to follow the gradient of the log-joint (or the free energy as defined for standard PC; see equation 3.5), and therefore these dynamics are also not guaranteed to find the MAP estimate of the latent states for the model defined in standard VPC,

$$x_{\ell,t+1} = x_{\ell,t} - \gamma\, u(x_{\ell,t}, x_{\ell+1,t}) \tag{3.22}$$

with

$$u(x_\ell, x_{\ell+1}) = \begin{Bmatrix} (x_\ell - \mu_\ell) - \frac{\partial f_\ell}{\partial x_\ell}^T (x_{\ell+1} - \mu_{\ell+1}) & \text{for } \ell = 1, \ldots, L-2 \\ (x_\ell - \mu_\ell) - \frac{\partial f_\ell}{\partial x_\ell}\Big|_{\mu_\ell}^T \frac{\partial f_{\ell+1}}{\partial \mu_{\ell+1}}^T & \text{for } \ell = L-1 \end{Bmatrix}. \tag{3.23}$$

Millidge et al. (2020) suggested that the fixed-prediction modifications (see equations 3.19 and 3.21) were equivalent to the infinite variance limit of case 2 from Whittington and Bogacz (2017). This is due to the fact that in the limit of infinite output variance, the predictions of the latent states change very little from their feedforward initializations by the end of inference. We argue, however, that this equivalence does not hold due to the fact that in the infinite or high-variance limit of case 2, the latent states x_ℓ also remain fixed, or arbitrarily close to, their feedforward values, and not just the predictions $f_\ell(x_\ell)$. This behavior is necessary for the inference to accurately correspond to inference over the latent intermediate states x_ℓ, which, in the limit of the zero-variance (deterministic) intermediate latent states assumed by the equivalent backpropagation model, become equal to the feedforward values μ_ℓ.

This is not the case for the FPA modification of PC, where the latent states are unconstrained and the prediction error is necessarily not zero except in the trivial case of a zero magnitude weight update.

This suggests that FPA-PC inference corresponds neither to neither the probabilistic graphical model assumed by standard VPC or that of the scaled variance (i.e., deterministic latent states) modification from case 2. This leaves us the question of how one can interpret FPA-PC inference if not in terms of inference over a probabilistic graphical model as with standard VPC. One particularly simple and elegant answer is that one can interpret the FPA-PC modified inference equations as directly implementing the recursive backpropagation equations via the steady state of a set of ordinarily differential equations. FPA-PC inference can then be interpreted as enacting an Euler integration scheme until one reaches this steady state. We derive this perspective in the following section.

3.4.1 Deriving FPA-PC as a Steady-State Implementation of Backpropagation. We can make the relationship between FPA-PC and backpropagation even clearer by showing that the FPA-PC equations can be interpreted as a direct implementation of backpropagation via differential equations. To illustrate this, we work our way backward from backpropagation, showing how a direct implementation of its recursion relationship results in the FPA-PC inference equations.

The key quantities that backpropagation requires computing are the errors e_i via the recursive expressions (see equations 2.2). We can trivially convert these expressions such that they correspond to the steady state of a set of simple differential equations,

$$\dot{e}_\ell = -\begin{cases} e_\ell - \frac{\partial \mu_{\ell+1}}{\partial \mu_\ell}^T e_{\ell+1} & \text{for } \ell = 1, \ldots, L-1 \\ e_\ell - \frac{\partial f_\ell}{\partial \mu_\ell}^T & \text{for } \ell = L \end{cases}, \tag{3.24}$$

such that when $\dot{e}_\ell = 0$, we obtain the required recursive relationships (see equation 2.2). Note that $\frac{\partial \mu_{\ell+1}}{\partial \mu_\ell}$ and $\frac{\partial E}{\partial \mu_L}$ are fixed Jacobians and require being evaluated at the feedforward values μ_ℓ. Also note that the outer negative sign has been added to make the relationship with FPA-PC clearer, and this does not affect the value of e_ℓ at the steady state.

We can then rewrite e_ℓ in terms of some variable x_ℓ minus the constant feedforward values μ_ℓ so that we have $e_\ell = x_\ell - \mu_\ell$, allowing us to write the above dynamic equations in terms of a variable x_ℓ as

$$\dot{x}_\ell = -\begin{cases} (x_\ell - \mu_\ell) - \frac{\partial \mu_{\ell+1}}{\partial \mu_\ell}^T (x_{\ell+1} - \mu_{\ell+1}) & \text{for } \ell = 1, \ldots, L-1 \\ (x_\ell - \mu_\ell) - \frac{\partial f_\ell}{\partial \mu_\ell}^T & \text{for } \ell = L \end{cases}. \tag{3.25}$$

If we instead assign e_L outright, as done by FPA-PC and keep the remaining errors dynamic, we obtain

$$\dot{x}_\ell = -\begin{cases} (x_\ell - \mu_\ell) - \frac{\partial \mu_{\ell+1}}{\partial \mu_\ell}^T (x_{\ell+1} - \mu_{\ell+1}) & \text{for } \ell = 1, \ldots, L-2 \\ (x_\ell - \mu_\ell) - \frac{\partial \mu_{\ell+1}}{\partial \mu_\ell}^T \frac{\partial f_{\ell+1}}{\partial \mu_{\ell+1}}^T & \text{for } \ell = L-1 \end{cases}, \tag{3.26}$$

which are precisely equal to the update equations 3.22 and 3.23 of FPA-PC.

The practical consequences of this modified version of PC were first questioned in Rosenbaum (2022), who showed that in the case of step size equal to 1, FPA-PC is not just functionally but algorithmically equivalent to backpropagation, suggesting at least in the specific case of inference step size equal to 1, there may be no benefit of FPA-PC over backpropagation. We extend this work and investigate the properties of Millidge et al. (2020),

as well as other recent modifications (Salvatori et al., 2021; Song et al., 2020) further, showing that these modifications have provable worse computational and time complexity for any step size.

3.4.2 Computational and Time Complexity. To investigate the computational and time complexity of the FPA-PC and other modified PC algorithms, we begin by proving that under the aforementioned dynamics, the number of inference steps taken for an error to propagate from an output node in a computational chain to an intermediate node is lower-bounded by its distance to the output node. See theorem 1 and corollary 1.

Theorem 1. *For a particular node (x_ℓ) in a computational chain that has nonzero gradient with regard to an output loss $\mathcal{L}$, initialized to feedforward values of the network, evolving under standard PC or FPA-PC dynamics, the error for that node (e_ℓ) will first become nonzero at time $t = L - \ell$, where L is the length of the chain and ℓ is the index of that node within the chain.*

Proof. Let the variational modes x be initialized to feedforward values of the network (i.e., requirement 1). That is, let $x_\ell = \mu_\ell$ for $\ell \in [1, \ldots, L-1]$. □

Let equations 3.22 and 3.23 describe the dynamics of our latent states under FPA-PC dynamics and equations 3.9 and 3.10 describe the dynamics under standard PC:

1. Because we are ignoring the trivial case of a zero gradient with respect to the output loss, we are guaranteed that the Jacobians $\frac{\partial f_\ell}{\partial x_\ell}$ are all nonzero as $\frac{\partial \mathcal{L}}{\partial x_\ell} = \frac{\partial f_\ell}{\partial x_\ell} \ldots \frac{\partial f_{L-1}}{\partial x_{L-1}} \frac{\partial \mathcal{L}}{\partial f_{\ell-1}} \neq 0$.
2. At time $t = 0$, the dynamics $u(x_\ell, x_{\ell+1})$ of all nodes are 0, except for x_{L-1}, by the definition of the update equations and requirement 1.
3. At time $t = 1$, the nodes x_{L-1} therefore update proportionally to $u(x_L)$
4. At any particular time step $t = t + 1$, if the dynamics $u(x_\ell, x_{\ell+1})$ associated with a node x_ℓ were zero in time step t, then they will become nonzero at $t + 1$ if and only if a change has occurred in $x_{\ell+1}$ from time step t to $t + 1$. (By the definition of the FPA-PC equations 3.22 and 3.23, or standard PC, equations 3.9 and 3.10, update equations, and statement 1.)
5. Thus, via induction, the first nonzero change to occur for an arbitrary variational mode x_ℓ at point ℓ in the computational chain will occur at a time $t = L - \ell$, its distance from the output with regard to the number of intermediary nodes.

Corollary 1. *The convergence time for errors associated with an arbitrary node in a computational chain is lower-bounded by its distance, in nodes, to the output node.*

Proof.

1. The fixed convergence points for the dynamical equations of FPA-PC are guaranteed to converge to an error equal to the gradient with regard to the output loss.
2. For a node, with nonzero gradient with regard to an output loss, to converge, the converged error for that node must therefore be nonzero.
3. Under theorem 1, the time taken for the error to first become nonzero is equal to its distance to the output node x_L, which is $t = L - \ell$ for a computational chain of length L.

□

This proof describes how, despite the ostensibly parallel computation occurring in FPA-PC, convergence nonetheless requires sufficient time steps for error information to propagate from the output node backward. This is necessary as information is still nonetheless only transmitted via adjacent nodes, as with backpropagation. Thus, inference must proceed for a minimum number of inference steps equal to this distance before convergence can occur.

We can now ask what the time complexity of a single inference step is under FPA-PC dynamics, and thus the time complexity of inference overall for FPA-PC, as well as how this compares to backpropagation. Note that for this comparison, we exclude the final VJP calculation for gradients associated with a particular θ, for both FPA-PC and backpropagation, as this has the same computational cost for both algorithms and only occurs once, at the end of inference and error propagation, respectively.

When considering the time complexity of FPA-PC inference, we will assume that the error computations ($e_\ell = x_\ell - \mu_\ell$) contribute negligible latency to the computation at every inference step. This is a generally reasonable assumption for the intermediate errors, given that these errors can be computed entirely in parallel and thus have the latency of a single subtraction operation, which is also generally among the lowest latency arithmetic instructions available on modern hardware (see Fog et al., 2011; Wong et al., 2010):

$$\text{TIME}\{\textit{FPA-PC}(F)\} = \text{TIME}\{\textit{forward}(F)\} + \text{TIME}\{\textit{FPA-Inf}(F)\} \tag{3.27}$$

We assume each inference step is maximally parallel, such that VJP computations across all nodes occur simultaneously. However, since each inference step must wait for the slowest (VJP) computation to complete, we can take the minimum time complexity for each inference step to be equal that of the slowest VJP computation: (Note that a similar result also follows if we assume the VJP computation for each function f_i takes roughly the same amount of time):

$$\text{TIME}\{\textit{FPA-PC}(F)\} = \sum_{i=0}^{L} \text{TIME}\{\textit{forward}(f_i)\} + \sum_{t=0}^{t_c} \text{TIME}\{\textit{vjp}(f_{\text{slow}})\} \quad (3.28)$$

$$\geq \sum_{i=0}^{L} \text{TIME}\{\textit{forward}(f_i)\} + \sum_{t=0}^{L} \text{TIME}\{\textit{vjp}(f_{\text{slow}})\} \quad (3.29)$$

$$\geq w \sum_{i=0}^{L} C_i + w \sum_{i=0}^{L} 2C_{\max} \quad (3.30)$$

$$\geq w \sum_{i=0}^{L} C_i + w \sum_{i=0}^{L} 2C_i \quad (3.31)$$

$$\geq \text{TIME}\{\textit{backprop}(F)\}, \quad (3.32)$$

where $C_{\max} = \max\{C_i : i = 1, \ldots, L\} = \text{WORK}\{\textit{vjp}(f_{\text{slow}})\}$. Thus, the time complexity for FPA-PC inference is provably greater than, or equal to, that of backpropagation for an equivalent computational chain. The bound we present here is a close to best-case scenario for FPA-PC for a number of reasons:

1. We have assumed that error computation ($e_\ell = x_\ell - \mu_\ell$) contributes negligible latency to each inference step.
2. Convergence will generally not occur within exactly $t_c = L$ steps except in particular circumstances (e.g., inference step size equal to 1), and so in practice, FPA-PC may have a significantly higher time complexity (scaling with inference length) than backpropagation.
3. Bottlenecks (slow VJP computations) will affect FPA inference significantly more as each inference step will be as slow as the slowest VJP computation (incurring a $t_c \cdot w \cdot C_{\max}$ latency cost), while a single slow VJP computation would only incur a single $C_{\max}$ latency cost for backpropagation.

Note that this is a separate and distinct proof to the result by Rosenbaum (2022), which showed that for a specific step size of 1, FPA-PC computes gradients in $t = L - \ell$ steps (i.e., in an equivalent number of steps to backpropagation). It was not clear, however, whether FPA-PC could converge in fewer time steps for arbitrary step sizes and whether FPA-PC inference had worse time complexity (or slower runtime), which was dependent on the time complexity of each inference step. Here we show that the FPA-PC convergence can provably never be faster than standard backpropagation for *any* step size.

Similar to Rosenbaum (2022), we test these results by enacting FPA-PC inference on multilayer perceptron networks of various sizes and compute the cosine similarity of gradient updates relative to those obtained via backpropagation. We focus on observing what happens as we vary the

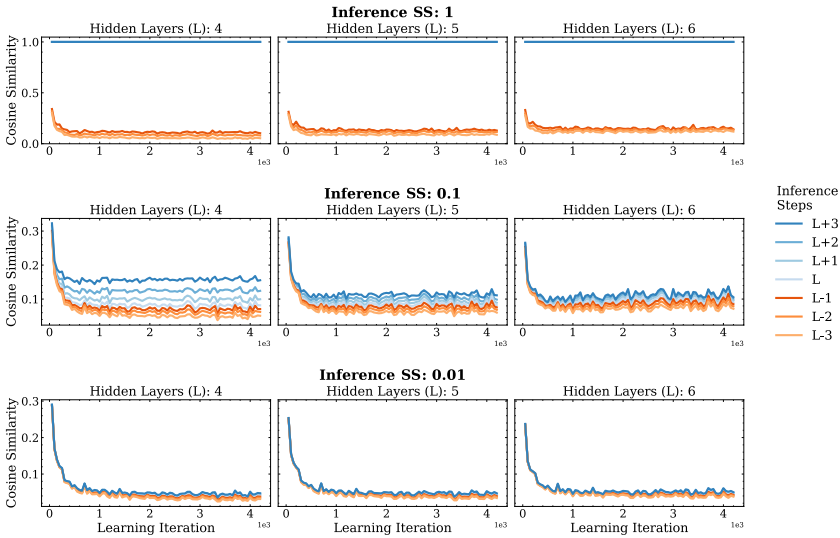

Figure 3: Cosine similarity between FPA-PC learning updates and backpropagation on MNIST, tested for varying inference step sizes (SS), and number of inference steps relative to the number of layers in the network (L).

number of inference steps (proportional to the computational cost) relative to the size of the network. We find, as expected, that cosine similarity drops rapidly as the number of inference steps falls below the number of layers in the network, demonstrating a failure to converge (see Figure 3). Inference step sizes lower than 1 show a failure to converge even for a number of inference steps higher than the number of layers in the network. We also report validation and test set accuracies, which experience significant drops as this occurs also (see Figures 4 and 5).

The results in Figures 3, 4, and 5 correspond to networks consisting of L hidden layers with 64 dimensionality, each followed by a tanh activation function and a 10-dimensional output layer followed by a log softmax. All networks were trained using SGD, with momentum equal to 0.9, initialized with default PyTorch linear initialization (Uniform$(-\sqrt{K}, \sqrt{K})$ where $K = \frac{1}{\text{Layer Input Size}}$), batch size 64, and standard categorical negative log-likelihood loss.

3.5 Zero Divergence Inference Learning. Subsequent to the FPA-modified formulation of PC, further modified formulations of PC were presented in Salvatori et al. (2021) and Song et al. (2020). The resultant algorithm, termed *zero divergence inference learning* (Z-IL), achieved a similar effect to FPA-PC without explicitly fixing the feedforward values and

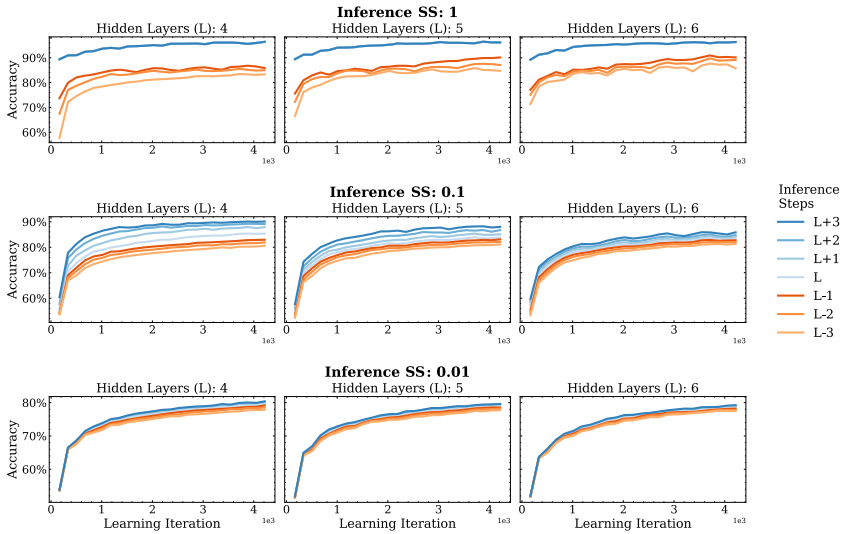

Figure 4: Validation set accuracy on MNIST with FPA-PC for varying inference step sizes (SS) and number of inference steps relative to the number of layers in the network (L).

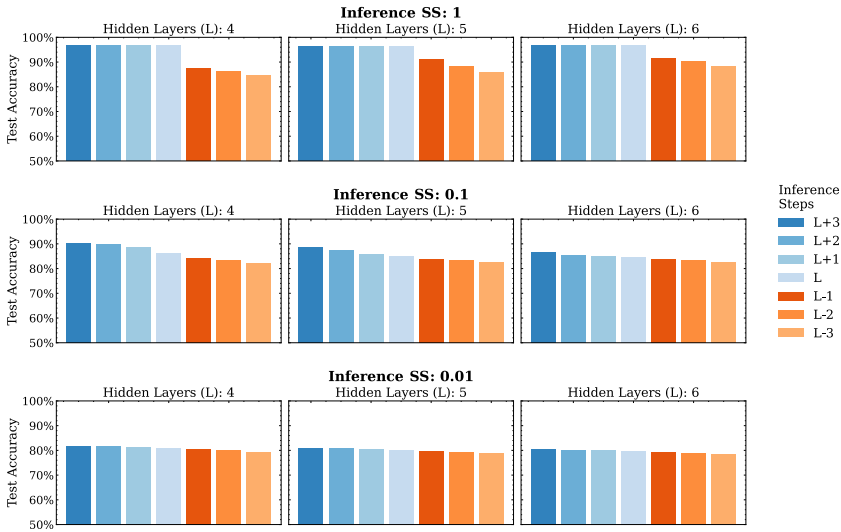

Figure 5: Final test set accuracy on MNIST with FPA-PC for varying inference step sizes (SS), and number of inference steps relative to the number of layers in the network (L).

Jacobians and did so by requiring a very specific set of changes. Specifically, the following changes to the standard PC algorithm were required:

- Requirement 1: (As before): The variational modes or latent states ($x_1, \ldots, x_{L-1}$) are initialized to the feedforward values of the corresponding backpropagation-based computational chain ($\mu_1, \ldots, \mu_{L-1}$).
- Requirement 2: The inference step size γ is set to 1.
- Requirement 3: A particular layer is updated specifically and exclusively at a particular inference time-step ($t = L - \ell$) corresponding to its distance from the output node in the computational chain.

Ablation of any one of these three requirements was shown to result in losing equivalence between Z-IL and backpropagation.[1]

This algorithm relies on the fact that at a particular time-step, the node corresponding to a particular layer x_ℓ is experiencing an update that is equivalent to that obtained under FPA-PC dynamics. This occurs because the current value of any particular node (x_ℓ) does not deviate from the feedforward values they were initialized with until after a particular ($t = L - \ell$) inference step (see lemma A.3 from supplementary material in Song et al., 2020). For the node corresponding to the weight being updated, this has the effect of mimicking the fixing of feedforward values seen under the FPA modified form (see equations 3.19 to 3.21), since both its predictions and the predictions of its parents ($\ell - 1$) remain fixed to their feed-forward values.

To see this explicitly, we describe the relevant computations occurring at each inference step for the example function F, in Table 1. We restrict ourselves at each inference step to only describing the changes that occur to x_ℓ, e_ℓ, and θ_ℓ for every time step ℓ. Therefore, we will not record in Table 1 any computations occurring at nodes $> \ell$, as they will have no future influence on any parameter learning, as well as nodes $< \ell$, as no change is occurring for that inference step. For notational simplicity, we drop time indices; rather, values on the right-hand side of the equations in Table 1 refer to those from the previous time-step.

We can see here that in each time step, for the computations relevant to parameter gradient updates, Z-IL engages in precisely the same computations as backpropagation (a VJP evaluated at the feedforward values and the subsequent error). These are then added to a constant (the feedforward activations μ_ℓ), before this constant is then subtracted. In addition to these computations, we note that Z-IL (and its variant, FA-Z-IL, by extension) engages in wasted computation for nodes x_i, $i > \ell$, which we do not depict in the walk-through of the computations above. These computations occur

[1] Note that for the purposes of this proof, we will ignore FA-Z-IL (fully autonomous Z-IL), which builds on Z-IL, as its primary purpose was to remove the neurobiologically implausible requirement that the weight update is triggered manually at a particular inference step and thus does not affect the results in this section.

Table 1: Unrolled Computations Occurring in Z-IL Alongside Calculations Occurring in Backpropagation.

Step (T)	Z-IL	Backpropagation
$T = 0$	$e_L = \frac{\partial f_L}{\partial \mu_L}$	$e_L = \frac{\partial f_L}{\partial \mu_L}$
$T = 1$	$x_{L-1} = \underbrace{x_{L-1}}_{=\mu_{L-1}} + \frac{\partial f_{L-1}}{\partial \mu_{L-1}}^T e_L$	$e_{L-1} = \frac{\partial f_{L-1}}{\partial \mu_{L-1}}^T e_L$
	$e_{L-1} = x_{L-1} - \mu_{L-1}$	
$\vdots$	$\vdots$	$\vdots$
$T = L - 1$	$x_1 = \underbrace{x_1}_{=\mu_1} + \frac{\partial f_1}{\partial \mu_1}^T e_2$	$e_1 = \frac{\partial f_1}{\partial \mu_1}^T e_2$
	$e_1 = x_1 - \mu_1$	

despite not resulting in, or contributing to, any parameter updates and are thus unnecessary.

Since the dynamics for Z-IL are a special case of the dynamics of standard PC, we may once again use theorem 1, corollary 1, and identical reasoning to that in section 3.4 to once again obtain complexity bounds for Z-IL which we find are similarly lower-bounded by that of backpropagation:

$$\text{TIME}\{\textit{Z-IL}(F)\} \geq \sum_{i=0}^{L} \text{TIME}\{\textit{forward}(f_i)\} + \sum_{t=0}^{L} \text{TIME}\{\textit{vjp}(f_{\text{slow}})\} \quad (3.33)$$

$$\geq w \sum_{i=0}^{L} \mathcal{C}_i + w \sum_{i=0}^{L} 2\mathcal{C}_{\max} \quad (3.34)$$

$$\geq w \sum_{i=0}^{L} \mathcal{C}_i + w \sum_{i=0}^{L} 2\mathcal{C}_i \quad (3.35)$$

$$\geq \text{TIME}\{\textit{backprop}(F)\} \quad (3.36)$$

3.6 Generalized-IL, Learning Rate Stability, and Online Learning. For completeness, we note that some recent work has suggested that predictive coding (PC) and related energy-based models, which do not yield backpropagation-equivalent updates, may nonetheless exhibit greater robustness to high learning rates and show less degraded performance in the online learning (batch size 1) settings (Alonso et al., 2022; Song et al., 2022). More specifically, Alonso et al. (2022) demonstrate that a variant of PC with additional regularization terms, called generalized-IL (G-IL), may approximate implicit SGD, with the approximation becoming exact under certain

limits. The authors of this work further present a modified algorithm (IL-prox) that guarantees equal updates to implicit SGD. The resultant schemes demonstrated greater robustness to learning rate and less degraded performance in online learning regimes. These properties are also demonstrated empirically, for energy-based models (EBMs) in general, as a consequence of a principle dubbed "prospective configuration" in Song et al. (2022).

These findings are intriguing, as robustness to learning rates aligns with expectations for a backward Euler integration scheme—the implicit counterpart to the forward Euler integration scheme from which SGD originates—an optimization approach that is demonstrably less prone to divergence in nonstochastic contexts.

It is unclear, however, whether the differences in learning rate stability would remain when testing against optimizers such as Adam or SGD with momentum, both of which are well understood in theory, and practice, to ameliorate the risk of divergence under high learning rates, by improving the conditioning of the loss landscape or dampening oscillations. Moreover, there is evidence to suggest that small-batch SGD itself has a greater robustness to learning rate (Masters & Luschi, 2018; Shallue et al., 2019), possibly due to the additional stochasticity in the small-batch setting preventing the accumulation of Euler integration errors. As such, the authors posit that it would be a compelling avenue for future research to explore whether the performance gap in online learning settings persists when online SGD is combined with high learning rates.

While these findings do not affect our presented results or the use of PC as a direct substitute for backpropagation, they raise thought-provoking questions on whether modified and unmodified PC forms may nonetheless possess attributes desirable for neuromorphic learning, even if they do not necessarily exhibit parameter updates identical to those under backpropagation.

4 Conclusion and Key Results

We now summarize the key results and points made in this article:

Result 1. The infinite-variance limit modification from Whittington and Bogacz (2017; see case 2) is equivalent to assuming a model with strictly deterministic (nonprobabilistic) latent states.

Result 2. Modified variants of PC—FPA-PC (Millidge et al., 2020), and Z-IL (Salvatori et al., 2021; Song et al., 2020)—do not follow a free energy gradient. Corollary: learning does not correspond to learning a latent probabilistic model via variational Bayes.

Result 3. Modified variants of PC (FPA-PC, and Z-IL) are guaranteed to have worse time complexity than backpropagation, even when accounting for fully parallel computation within each inference step. (See theorem 1, corresponding corollary 1, and equations 3.27 to 3.36.)

Point 1. Equivalence, or approximate equivalence, of PC with backpropagation mandates adopting an inverted scheme, unlike traditional formulations of generative PC, which has nontrivial implications for neurobiological plausibility.

Point 2. Implementations of standard PC do not suffer from the weight nonlocality issue as present in naive neurobiological proposals of backpropagation. They may, however, suffer from a, potentially less implausible, weight symmetry issue.

These results raise doubt with respect to the advantages that recent variants of PC may provide with regard to backpropagation and its neuromorphic implementation, given that PC variants that result in equivalent or close to equivalent gradient updates also engage in precisely the same computations as backpropagation, and do so in much the same, equally local/nonlocal, way. This is despite the ostensibly parallelized computation of PC networks, which from the perspective of backpropagating errors merely results in additional, or unused, computation, and not faster error propagation.

Furthermore, by introducing modifications to obtain, or approximate, equivalence, PC variants lose the generative variational Bayes interpretation of standard formulations of PC and its various strengths, such as maintaining uncertainty over latent hidden states or causes, or parity with our current understanding of the organizational and functional properties of pathways within the cortex.

Acknowledgments

We thank Christopher L. Buckley and his group for the valuable comments and feedback on this article.

References

Akrout, M., Wilson, C., Humphreys, P., Lillicrap, T., & Tweed, D. B. (2019). Deep learning without weight transport. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*, *32*. Curran.

Alonso, N., Millidge, B., Krichmar, J., & Neftci, E. (2022). *A theoretical framework for inference learning*. 10.48550/arXiv.2206.00164

Amit, Y. (2019). Deep learning with asymmetric connections and Hebbian updates. *Frontiers in Computational Neuroscience*, *13*. 10.3389/fncom.2019.00018

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., & Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, *76*(4), 695–711. 10.1016/j.neuron.2012.10.038

Baur, W., & Strassen, V. (1983). The complexity of partial derivatives. *Theoretical Computer Science*, *22*(3), 317–330. 10.1016/0304-3975(83)90110-X

Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of Mathematical Psychology*, *76*, 198–211. 10.1016/j.jmp.2015.11.003

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., . . . Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs.* http://github.com/google/jax

Buckley, C. L., Kim, C. S., McGregor, S., & Seth, A. K. (2017). The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, *81*, 55–79. 10.1016/j.jmp.2017.09.004

Crick, F. (1989). The recent excitement about neural networks. *Nature*, *337*(6203), 129–132. 10.1038/337129a0

Feldman, H., & Friston, K. (2010). Attention, uncertainty, and free-energy. *Frontiers in Human Neuroscience*, *4*, 215. 10.3389/fnhum.2010.00215

Felleman, D. J., & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, *1*(1), 1–47. 10.1093/cercor/1.1.1-a

Fog, A. (2011). *Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs*. Copenhagen University College of Engineering.

Fountas, Z., Sylaidi, A., Nikiforou, K., Seth, A. K., Shanahan, M., & Roseboom, W. (2022). A predictive processing model of episodic memory and time perception. *Neural Computation*, *34*(7), 1501–1544. 10.1162/neco_a_01514

Friston, K. (2003). Learning and inference in the brain. *Neural Networks*, *16*(9), 1325–1352. 10.1016/j.neunet.2003.06.005

Friston, K. (2005). A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *360*(1456), 815–836. 10.1098/rstb.2005.1622

Friston, K. (2008). Hierarchical models in the brain. *PLOS Computational Biology*, *4*(11), e1000211. 10.1371/journal.pcbi.1000211

Friston, K., & Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *364*(1521), 1211–1221. 10.1098/rstb.2008.0300

Griewank, A. (1997, February). On automatic differentiation.

Griewank, A. (2009). Complexity of gradients, Jacobians, and Hessians In C. A. Floudas & P. M. Pardalos (Eds.), *Encyclopedia of optimization* (pp. 425–435). Springer US. 10.1007/978-0-387-74759-0_78

Griewank, A., & Walther, A. (2008). *Evaluating derivatives*. SIAM.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, *11*(1), 23–63. 10.1016/S0364-0213(87)80025-3

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., . . . Sifre, L. (2022). *Training compute-optimal large language models.* 10.48550/arXiv.2203.15556

Hohwy, J., Roepstorff, A., & Friston, K. (2008). Predictive coding explains binocular rivalry: An epistemological review. *Cognition*, *108*(3), 687–701. 10.1016/j.cognition.2008.05.010

Kanai, R., Komura, Y., Shipp, S., Friston, K. (2015). Cerebral hierarchies: Predictive processing, precision and the pulvinar. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *370*(1668), 20140169. 10.1098/rstb.2014.0169

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., . . . Amodei, D. (2020). *Scaling laws for neural language models.* 10.48550/arXiv.2001.08361

Kinghorn, P. F., Millidge, B., & Buckley, C. L. (2022). *Preventing deterioration of classification accuracy in predictive coding networks.* 10.48550/arXiv.2208.07114

Kolen, J., & Pollack, J. (1994). Backpropagation without weight transport. In *Proceedings of 1994 IEEE International Conference on Neural Networks* (Vol. 3, pp. 1375–1380). 10.1109/ICNN.1994.374486

Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, *7*(1), 13276. 10.1038/ncomms13276

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, *16*(2), 146–160. 10.1007/BF01931367

Markov, N. T., Vezoli, J., Chameau, P., Falchier, A., Quilodran, R., Huissoud, C., . . . Kennedy, H. (2014). Anatomy of hierarchy: Feedforward and feedback pathways in macaque visual cortex. *Journal of Comparative Neurology*, *522*(1), 225–259. 10.1002/cne.23458

Masters, D., & Luschi, C. (2018, April). *Revisiting small batch training for deep neural networks.* 10.48550/arXiv.1804.07612

Millidge, B., Salvatori, T., Song, Y., Bogacz, R., & Lukasiewicz, T. (2022). *Predictive coding: Towards a future of deep learning beyond backpropagation?* 10.48550/arXiv.2202.09467

Millidge, B., Tschantz, A., & Buckley, C. L. (2020). *Predictive coding approximates backprop along arbitrary computation graphs*. arXiv:2006.04182

Millidge, B., Tschantz, A., Seth, A., & Buckley, C. L. (2020a). *Relaxing the constraints on predictive coding models.* 10.48550/arXiv.2010.01047

Millidge, B., Tschantz, A., Seth, A. K., & Buckley, C. L. (2020b). *Activation relaxation: A local dynamical approximation to backpropagation in the brain.* http://arxiv.org/abs/2009.05359.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., . . . Lerer, A. (2017). *Automatic differentiation in PyTorch.* NIPS Workshop on Automatic Differentiation.

Pawlak, V., Wickens, J. R., Kirkwood, A., & Kerr, J. N. D. (2010). Timing is not everything: Neuromodulation opens the STDP gate. *Frontiers in Synaptic Neuroscience*, *2*, 146. 10.3389/fnsyn.2010.00146

Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., & Naud, R. (2021). Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature Neuroscience*, *24*(7), 1010–1019. 10.1038/s41593-021-00857-x

Rao, R. P. N., & Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, *2*(1), 79–87. 10.1038/4580

Rosenbaum, R. (2022). On the relationship between predictive coding and backpropagation. *PLOS One*, *17*(3), e0266102. 10.1371/journal.pone.0266102

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. 10.1038/323533a0

Sacramento, J., Costa, R. P., Bengio, Y., & Senn, W. (2018, October). *Dendritic cortical microcircuits approximate the backpropagation algorithm.* arXiv:1810.11393 [cs, q-bio])

Salvatori, T., Song, Y., Lukasiewicz, T., Bogacz, R., & Xu, Z. (2021). *Predictive coding can do exact backpropagation on convolutional and recurrent neural networks.* arXiv:2103.03725

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., & Dahl, G. E. (2019). *Measuring the effects of data parallelism on neural network training.* arXiv:1811.03600

Shervani-Tabar, N., Rosenbaum, R. (2023). Meta-learning biologically plausible plasticity rules with random feedback pathways. *Nature Communications*, *14*(1), 1805. 10.1038/s41467-023-37562-1

Shipp, S. (2016). Neural elements for predictive coding. *Frontiers in Psychology*, *7*, 1792. 10.3389/fpsyg.2016.01792

Sjöström, P. J., & Häusser, M. (2006). A cooperative switch determines the sign of synaptic plasticity in distal dendrites of neocortical pyramidal neurons. *Neuron*, *51*(2), 227–238. 10.1016/j.neuron.2006.06.017

Song, Y., Lukasiewicz, T., Xu, Z., & Bogacz, R. (2020). Can the brain do backpropagation? Exact implementation of backpropagation in predictive coding networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems*, *33* (pp. 22566–22579). Curran.

Song, Y., Millidge, B., Salvatori, T., Lukasiewicz, T., Xu, Z., & Bogacz, R. (2022, May). *Inferring neural activity before plasticity: A foundation for learning beyond backpropagation*. 10.1101/2022.05.17.492325

Urbanczik, R., & Senn, W. (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, *81*(3), 521–528. 10.1016/j.neuron.2013.11.030

Whittington, J. C. R., & Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*, *29*(5), 1229–1262. 10.1162/NECO_a_00949

Whittington, J. C. R., & Bogacz, R. (2019). Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, *23*(3), 235–250. 10.1016/j.tics.2018.12.005

Wong, H., Papadopoulou, M.-M., Sadooghi-Alvandi, M., & Moshovos, A. (2010). Demystifying GPU microarchitecture through microbenchmarking. In *Proceedings of the 2010 IEEE International Symposium on Performance Analysis of Systems and Software* (pp. 235–246). 10.1109/ISPASS.2010.5452013

Zahid, U., Guo, Q., Friston, K., & Fountas, Z. (2023, March). *Curvature-sensitive predictive coding with approximate Laplace Monte Carlo.* 10.48550/arXiv.2303.04976

Zipser, D., & Rumelhart, D. E. (1993). The neurobiological significance of the new learning models. In E. L. Schwartz (Ed.). *Computational neuroscience* (pp. 192–2000). MIT Press.

Received April 5, 2023; accepted August 1, 2023.