Training a Predictive Coding Network on ImageNet using Equilibrium Propagation

Tugdual Kerjan*
tkerjan@outlook.com

Rasmus Høier*
rasmus@normalcomputing.com

Benjamin Scellier*
benjamin.scellier@polytechnique.org

Abstract

Equilibrium Propagation (EP) is a physics-based training framework that has primarily been employed in energy-based models, including continuous Hopfield networks, nonlinear resistive networks and coupled phase oscillators. However, EP's practical applications have so far remained limited to relatively small-scale problems. Predictive coding networks (PCNs), another class of energy-based models rooted in computational neuroscience, are typically trained with a specialized algorithm and have likewise not yet been demonstrated at large scale. In this work, we develop an EP-based training method for PCNs which combines the centered variant of EP with a novel equilibration scheme for PCNs. Using this approach, we train a 10-layer convolutional PCN (VGG10) on full-size ImageNet, achieving 13.27% test error rate on the top-5 classification task, close to the 12.2% backpropagation baseline. We also explore a single-phased variant using random-sign nudging which trails closely at 13.83% top-5 test error rate, while being twice as fast to simulate. To our knowledge, this is the first demonstration of both PCNs and EP-based training at ImageNet scale. These results significantly extend the scalability of both approaches and suggest that the primary challenges in scaling EP in other physical systems may come more from the computational properties of these systems than from inherent limitations of the EP framework.

1 Introduction

The growing computational demands of machine learning (ML) motivate the exploration of neuromorphic computing platforms based on analog physics, which promise substantial energy savings [Marković et al., 2020]. A prominent research direction within these efforts is the development of 'physical learning algorithms' that leverage the physics of the hardware for both inference and learning [Momeni et al., 2025]. Equilibrium Propagation (EP) [Scellier and Bengio, 2017] is a physics-based training framework that enables a class of systems to learn by adjusting parameters through comparisons of equilibrium states under varying boundary conditions.

EP belongs to a broader class of contrastive learning algorithms [Hinton and Sejnowski, 1983, Movellan, 1991, Baldi and Pineda, 1991, Hinton, 2002] and employs the concept of weak perturbations used in earlier works [Hinton and McClelland, 1987, O'Reilly, 1996]. EP distinguishes itself from these methods in that it allows optimization of arbitrary cost functions ; in the weak-perturbation regime it approximates gradient descent. EP has been studied in energy-based models such as continuous Hopfield networks [Scellier and Bengio, 2017], nonlinear resistor networks [Kendall et al., 2020], flow and elastic networks [Stern et al., 2021], and systems of Kuramoto oscillators [Zoppo et al., 2022,

*Equal contribution. Work performed while at Rain AI.

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

Wang et al., 2024, Rageau and Grollier, 2025]. Generally applicable to physical systems governed by variational principles, EP has been extended to stochastic and thermal systems [Scellier and Bengio, 2017, Massar and Mognetti, 2025], Lagrangian dynamical systems [Kendall, 2021, Scellier, 2021, Massar, 2025, Pourcel et al., 2025] and quantum systems [Massar and Mognetti, 2025, Wanjura and Marquardt, 2025, Scellier, 2024b]. Further developments include a version of EP which allows parameter updates in black-box systems using physical dynamics [Scellier et al., 2022] and a version of EP for nonlinear wave systems [Sajnok and Matuszewski, 2025].

Despite these theoretical developments, applications of EP have so far remained limited to small networks and datasets. Physical implementations of EP and variants have been demonstrated on elastic networks [Altman et al., 2024], resistor networks [Dillavou et al., 2022, 2024], memristor networks [Yi et al., 2023], D-wave's Ising machine [Laydevant et al., 2024], as well as spatial photonic and coherent Ising machines [Abeele et al., 2026, Fan et al., 2026], albeit only at small scales. On the simulation side, EP has been scaled to train convolutional Hopfield networks on a 32x32 downsampled version of ImageNet [Laborieux and Zenke, 2022, Nest and Ernoult, 2024] and to train a transformer-like model to reconstruct partially masked images [Høier et al., 2026]. Setting aside the challenges of hardware implementation, several factors contribute to the difficulty of scaling EP in numerical experiments. First, unlike backpropagation, EP computes only an approximate gradient of the cost function, with its accuracy depending on factors like perturbation strength, finite difference scheme and the proximity to equilibrium. Second, the expressivity and computational properties of EP-compatible physical networks remain poorly understood, as these systems lack a direct mapping to traditional neural networks [2]. Finally, simulations of such networks can be computationally expensive, since they require numerical optimization of the system's energy function to reach equilibrium[3]. As one attempts to scale EP to larger networks and more complex datasets, it has remained unclear whether the observed challenges should be attributed to the EP framework itself or the expressivity and trainability of the underlying networks (or a combination of both).

In this work, we investigate this question by applying EP to predictive coding networks (PCNs), a class of models studied in computational neuroscience that alleviates some of the challenges associated with simulations of physical networks. In their most common form, PCNs can be viewed as an energy-based counterpart to feedforward neural networks [Whittington and Bogacz, 2017, Millidge et al., 2021]. Although it is debated whether PCNs can offer advantages as a neuromorphic platform [Zahid et al., 2023], in numerical experiments they allow to isolate EP-related challenges from model-specific issues: at test time, PCNs behave like feedforward neural networks, enabling direct comparison with backpropagation in standard neural network models. PCNs also offer a computational advantage over simulations of physical systems such as resistor or oscillatory networks, as their 'free equilibrium' state (i.e. the inference equilibrium) can be obtained in a single forward pass.

To date, however, applications of PCNs have also been limited to relatively small datasets such as Tiny ImageNet [Pinchetti et al., 2024], and although some works on PCNs have explored the nudging-based perturbation mechanism of EP [Whittington and Bogacz, 2019, Millidge et al., 2023], PCNs are more typically trained using a different perturbation strategy based on output clamping. To identify the key ingredients for effectively training PCNs with EP, we conduct extensive experiments on a 5-layer convolutional PCN (VGG5) across four vision datasets. We perform a comprehensive sensitivity analysis of EP hyperparameters, including the perturbation method, finite difference scheme, perturbation strength and the number of iterations in the nudge-phase equilibration. Our main findings and contributions are as follows:

- While clamping-based perturbations perform comparably to EP's nudging-based approach on CIFAR-100, we find that, by allowing to optimize the categorical cross-entropy, the nudging-based method significantly outperforms clamping on more challenging datasets such as ImageNet 32x32 (Figure 1).
- By combining the nudging-based approach with the centered scheme, we train a 10-layer convolutional PCN (VGG10) on full-resolution ImageNet, achieving 13.27% top-5 test error rate, close to the 12.2% backpropagation baseline (Table 2). To the best of our knowledge, this is the first demonstration of both PCNs and EP-based training at ImageNet scale.

[2] Scellier and Mishra [2025] recently established a universal approximation result for nonlinear resistive networks.

[3] Algorithmic advances have accelerated this equilibration step in some systems [Scellier et al., 2023, Scellier, 2024a, Osana et al., 2026].

- Our results challenge common assumptions about EP's random and centered schemes, showing that the random scheme remains competitive even on full-size ImageNet (Figure 1 and Table 2). This scheme is particularly appealing for PCNs, as the learning rule involves only a single equilibrium state, rather than two as in the centered scheme.

2 Related Work on Predictive Coding Networks

The standard learning algorithm for PCNs as outlined in Whittington and Bogacz [2017], involves clamping the output units to desired outputs rather than nudging them as in EP. Millidge et al. [2023] proposed using the nudging method of EP in the context of PCNs, but they used specifically the forward scheme (which employs positive nudging) and found the standard learning algorithm for PCNs to be preferable. Conversely, Pinchetti et al. [2024] combined the weak clamping-based perturbation approach with the random scheme and achieved significantly better performance, using it to train convolutional PCNs on Tiny ImageNet. The centered scheme for EP [Laborieux et al., 2021], which involves comparing one positively-perturbed state with one negatively-perturbed state, has not been utilized in PCNs so far, presumably because it requires comparing two network states, making it harder to justify from a neuroscience perspective. Our paper is the first to explore EP's nudging approach combined with the backward, random and centered schemes in the context of PCNs.

Scaling PCNs to deeper architectures has been a key challenge [Pinchetti et al., 2024], with a number of recent papers exploring this issue. Innocenti et al. [2026] improved activation stability by introducing additional scaling factors in the network energy, enabling the training of densely connected ResNets with up to 128 layers on MNIST and Fashion-MNIST. Other works have focused on tackling the issue of signal attenuation due to limited machine precision. Goemaere et al. [2025] tackle this issue by reparametrizing PC to treat the local errors as the variable of optimization rather than the neural states, showing that the underlying optimization problems are equivalent, and the simulations are much faster (more than $100\times$). Qi et al. [2025] address the issue of signal attenuation by, among other things, introducing a schedule for the inference step size and using a surrogate objective when computing the weight updates. Complementing these depth-focused studies, recent work has also explored the scaling dynamics of PCNs with respect to network width [Ishikawa et al., 2025].

3 Background: Equilibrium Propagation

This section reviews EP and its variants depending on finite difference schemes (forward, backward, centered and random) and perturbation methods (nudging vs clamping).

In its original formulation, EP applies to *energy-based systems*, meaning systems with a state vector h whose dynamics evolve towards a minimum (or more generally critical point) of some function E, referred to as *energy function*. In supervised learning, $E(\theta, x, h)$ depends on trainable weights θ and a boundary input x. During inference, given x, the system settles into its equilibrium state $h(\theta, x)$, characterized by the first-order condition

$$\frac{\partial E}{\partial h}(\theta, x, h(\theta, x)) = 0. \tag{1}$$

The function $x \mapsto h(\theta, x)$ represents the system's forward computation. Training consists in adjusting θ so as to minimize the discrepancy between the prediction $h(\theta, x)$ and the desired result y, which is quantified using a cost function $C(h(\theta, x), y)$. The training objective we seek to minimize is

$$J(\theta) = \mathbb{E}_{(x,y)}\left[C(h(\theta, x), y)\right], \tag{2}$$

where the expectation is taken over the data distribution of input-target pairs (x, y). If, given an input-output pair (x, y) from the training data, one had direct access to $\nabla_\theta C(h(\theta, x), y)$, the usual gradient-descent update would be

$$\Delta\theta = -\eta \nabla_\theta C(h(\theta, x), y). \tag{3}$$

The key challenge in a physical system is to estimate these gradients using the system's dynamics itself. EP accomplishes this by introducing a small *nudging* parameter $\beta \in \mathbb{R}$, and interpreting

$\beta C(h, y)$ as an interaction energy between the state h and target output y, scaled by β. Incorporating this into the energy function leads to the *total energy function*

$$F(\theta, \beta, h) = E(\theta, x, h) + \beta C(h, y), \tag{4}$$

where β controls the strength of this interaction. With $\beta = 0$, the system relaxes to the *free* equilibrium, $h^0_\star = h(\theta, x)$. For nonzero β, the system settles to the *nudged* equilibrium, characterized by the first-order condition

$$\frac{\partial F}{\partial h}(\theta, \beta, h^\beta_\star) = 0. \tag{5}$$

The main insight of Scellier and Bengio [2017] is that the gradient of the cost function with respect to the trainable weights can be approximated as

$$\nabla_\theta C(h(\theta, x), y) = \frac{d}{d\beta}\frac{\partial F}{\partial \theta}(\theta, \beta, h^\beta_\star)\bigg|_{\beta=0} \approx \frac{1}{\beta}\left[\frac{\partial F}{\partial \theta}(\theta, \beta, h^\beta_\star) - \frac{\partial F}{\partial \theta}(\theta, 0, h^0_\star)\right]. \tag{6}$$

Substituting Eq. (6) into Eq. (3) yields a contrastive learning rule.

3.1 Finite Difference Schemes: Forward, Backward, Centered and Random Schemes

In contrast with standard ML methods based on automatic differentiation, EP does not provide the exact cost gradient but rather an approximation of it, due to the finite difference scheme employed for estimating the derivative with respect to β. We refer to the right-hand side of Eq. (6) with $\beta > 0$ as the *forward* scheme. Scellier and Bengio [2017] found that the *random* scheme, where the sign of β is chosen at random for each training pair (x, y), works better than the forward scheme. To further reduce the approximation error, Laborieux et al. [2021] introduced the *centered* scheme whose error term is of order $O(\beta^2)$,

$$\nabla_\theta C(h(\theta, x), y) \approx \frac{1}{2\beta}\left(\frac{\partial F}{\partial \theta}\left(\theta, \beta, h^\beta_\star\right) - \frac{\partial F}{\partial \theta}\left(\theta, -\beta, h^{-\beta}_\star\right)\right). \tag{7}$$

The purely *backward* scheme, employing Eq. (6) with $\beta < 0$, was studied only later by Scellier et al. [2022, 2023] and shown to optimize an upper bound of the objective function (2) provided the free state of Eq. (1) is a minimizer of E.

3.2 Perturbation Approaches: Nudging vs Clamping

An alternative to the nudging approach of Eq. (4) is to perturb the system by clamping its outputs rather than modifying its energy function. Let $h = (h_{\text{hid}}, h_{\text{out}})$ denote the system state, partitioned into hidden and output components. In the clamping-based approach, the output variables are set to

$$h^\beta_{\text{out}} = (1 - \beta)h^0_{\text{out}} + \beta y, \tag{8}$$

where h^0_{out} denotes the free-equilibrium output of the system. This strategy, which predates EP, was proposed by O'Reilly [1996] in the context of Hopfield-like networks, and has more recently been employed in physical energy-based networks [Stern et al., 2021] and in PCNs [Pinchetti et al., 2024].

Unlike nudging, which provides flexibility in defining the learning objective of Eq. (2) by allowing arbitrary choices of cost functions (C), the clamping-based approach does not define a cost function explicitly, and the resulting parameter updates do not generally correspond to gradient descent or to the minimization of an objective function, even in the limit $\beta \to 0$. A detailed comparison of the two approaches is discussed in Scellier et al. [2023].

4 Predictive Coding Networks Through the Lens of EP

While EP's main appeal lies in training physical systems such as those referenced in the introduction, here we apply EP to feedforward neural networks by defining an appropriate energy function. This leads to a collection of algorithms for Predictive Coding Networks (PCNs).

Consider a feedforward neural network with L layers:

$$h^\star_0 = x, \quad \text{and} \quad h^\star_k = f_k(\theta_k, h^\star_{k-1}), \quad 1 \le k \le L, \tag{9}$$

where $h_k^\star$ is the activation of layer k, f_k is the (learnable) transformation that maps $h_{k-1}^\star$ to $h_k^\star$, and θ_k are the corresponding trainable parameters (weights and biases). Later we will consider specifically convolutional (VGG) networks, but for now our discussion does not make such assumptions on the network architecture. One can view this network as an energy-based model whose energy function is

$$E_{\mathrm{PCN}}(\theta, x, h) = \frac{1}{2} \sum_{k=1}^{L} \|h_k - f_k(\theta_k, h_{k-1})\|^2 , \tag{10}$$

where $x = h_0$ and $h = (h_1, \ldots, h_L)$. Indeed, the state $h^\star = (h_1^\star, h_2^\star, \ldots, h_N^\star)$ satisfying the feedforward equations of Eq. (9) is a stationary point of E_{PCN}, specifically the unique global minimum of E_{PCN}, for which we have $E_{\mathrm{PCN}}(\theta, x, h^\star) = 0$.

Assuming we use the nudging-based perturbation technique, EP then operates on the total energy $F_{\mathrm{PCN}}(\theta, \beta, h) = E_{\mathrm{PCN}}(\theta, x, h) + \beta C(h_L, y)$, where $C(h_L, y)$ is the cost function. In the *free phase* ($\beta = 0$), finding the global minimum of F_{PCN} with respect to h reduces to a single forward pass, following the feedforward equations of Eq. (9). For $\beta \neq 0$ (nudge phase), however, finding a stationary point of F_{PCN} requires a specific algorithm. In this work, we use a modified form of gradient descent on F_{PCN} with respect to h, which we describe next (Section 4.1). Besides, it is easy to verify that the energy derivative at the free equilibrium state vanishes: $\frac{\partial F_{\mathrm{PCN}}}{\partial \theta}(\theta, 0, h_\star^0) = 0$. Therefore, when employing the forward, backward or random scheme, the EP gradient of Eq. (6) simplifies, requiring only one equilibrium state (the nudge state):

$$\nabla_\theta C(h(\theta, x), y) \approx \frac{1}{\beta} \frac{\partial F_{\mathrm{PCN}}}{\partial \theta}(\theta, \beta, h_\star^\beta). \tag{11}$$

Prior works discussing the relationship between EP and PCNs combined the nudging approach with the forward scheme [Millidge et al., 2023], or the forward, backward and random schemes with the clamping approach [Pinchetti et al., 2024]. Here, we perform a more comprehensive comparison of these perturbation techniques and finite difference schemes. We find that the best performing ones combine the nudging approach with the centered, random and backward schemes, which were not used in prior works on PCNs. Appendix C provides more details on the relationship between EP and the traditional learning algorithm for PCNs.

4.1 Nudged-Phase Equilibration via Projected Gradient Descent

To compute the nudge equilibria, we minimize the total energy function using projected gradient descent (PGD). The PCN's total energy function rewrites

$$F_{\mathrm{PCN}}(\theta, \beta, h) = \frac{1}{2} \sum_{k=1}^{L} \|\epsilon_k\|^2 + \beta C(h_L, y), \qquad \epsilon_k = h_k - f_k(\theta_k, h_{k-1}), \tag{12}$$

where ϵ_k, referred to as *error*, is the difference between the state h_k and the bottom-up prediction $f_k(\theta_k, h_{k-1})$. In the nudge-phase dynamics, units h_k adjusts so as to minimize the sum of squared errors. In what follows, we consider PCNs with ReLU units in the hidden layers and linear output units, more specifically:

$$f_k \in \{\text{ReLU} \circ \text{maxpool} \circ \text{conv2d},\ \text{ReLU} \circ \text{conv2d},\ \text{ReLU} \circ \text{matmul}\}, \qquad 1 \leq k \leq L-1, \tag{13}$$

$$f_L = \text{matmul}. \tag{14}$$

These are the operations we require for the VGG networks considered in our experiments.

PGD with step size $\alpha > 0$ on a ReLU layer reads

$$h_k \leftarrow \mathrm{Proj}_{[0,+\infty]} \left(h_k - \alpha \frac{\partial F_{\mathrm{PCN}}}{\partial h_k}(\theta, \beta, h) \right), \tag{15}$$

where $\mathrm{Proj}_{[0,+\infty]}$ is the projection onto the range of the ReLU, that is $[0, +\infty]$, which is equivalent to applying the ReLU itself. In our experiments, we use step size $\alpha = 1$ (see Appendix A for an empirical justification). For a hidden layer h_k, the gradient of F_{PCN} with respect to h_k is

$$\frac{\partial F_{\mathrm{PCN}}}{\partial h_k} = \frac{\partial (\frac{1}{2}\|\epsilon_k\|^2)}{\partial h_k} + \frac{\partial (\frac{1}{2}\|\epsilon_{k+1}\|^2)}{\partial h_k} = h_k - f_k(\theta_k, h_{k-1}) + \frac{\partial (\frac{1}{2}\|\epsilon_{k+1}\|^2)}{\partial h_k}. \tag{16}$$

Injecting Equation (16) into Equation (15) with $\alpha = 1$, the h_k terms cancel out and the PGD update rewrites more concisely as

$$h_k \leftarrow \text{ReLU}\left(f_k(\theta_k, h_{k-1}) - \frac{\partial(\frac{1}{2}\|\epsilon_{k+1}\|^2)}{\partial h_k}\right). \tag{17}$$

For the output layer whose activation function is the identity, the update rule simplifies as:

$$h_L \leftarrow f_L(\theta_L, h_{L-1}) - \beta\frac{\partial C}{\partial h_L}(h_L, y). \tag{18}$$

Intuitively, during the nudged dynamics ($\beta \neq 0$), output units are perturbed by bringing them to either lower cost values ($\beta > 0$) or higher cost values ($\beta < 0$), that is, closer to the targets or further away from the targets. These perturbations lead to non-zero error signal ϵ_L, which leads to perturbations in the layer $L - 1$, and so on.

In our experiments, we use a modified version of PGD with an asynchronous update scheme for the layers – see Appendix A for a full description and motivation. A detailed examination of this scheme (Appendix A) reveals that it is likely better interpreted as a fixed-point iteration converging to stationary points of the total energy F, rather than standard projected gradient descent. In any case, our equilibration approach is different from that of Pinchetti et al. [2024] who use standard GD with synchronous layer updates, tiny learning rates and momentum.

We emphasize that it remains unclear whether the nudge-phase dynamics (PGD) could be efficiently realized in neuromorphic hardware. The primary contribution of our work is to advance the state of EP-based and PCN numerical experiments (on GPUs) to full-size ImageNet, without hardware implementation considerations.

5 Experimental Results

We consider two convolutional (VGG) PCN models: a 5-layer network (VGG5) with 32x32 pixel input images, and a 10-layer network (VGG10) with 224x224 pixel input images. The VGG5 network is employed on MNIST, CIFAR10, CIFAR100 and ImageNet 32x32 (the 32x32 pixel downscaled version of ImageNet). To permit using VGG5 on MNIST, images from MNIST are padded to 32x32 input size. The VGG10 network is trained on the (full-size) ImageNet dataset. Network architectures and numerical simulation details are described in Appendix B.

The hyperparameter search space for EP-based training of these VGG networks includes:

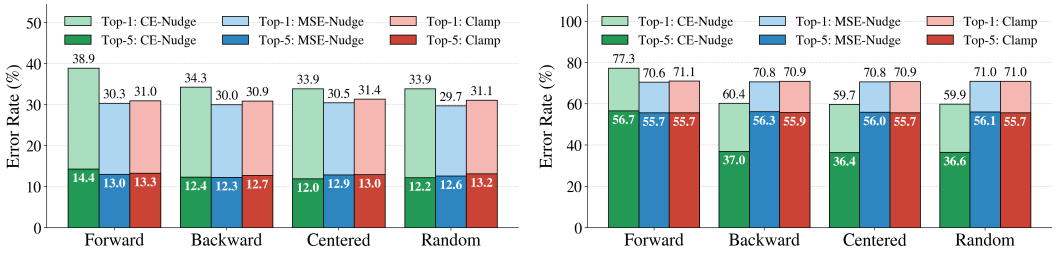
- perturbation method: nudging vs clamping,
- finite difference scheme: forward, backward, centered or random,
- nudging strength (β),
- number of nudge iterations (K),
- cost function (only applicable to the nudging-based perturbation approach): mean squared error (MSE) or cross-entropy (CE).

Backpropagation (BP) is also employed as a baseline for comparison. We use a two-stage hyperparameter tuning strategy, where we first tune the hyperparameters shared by BP-based and EP-based training (weight initialization schemes, learning rates, weight decay, batch-size, cost function, etc), and subsequently tune the EP-specific hyperparameters. This strategy is justified by the fact that, when employing the nudging-based perturbation method, in the limit when β tends to 0, the EP gradients are equal to the BP gradients (Equation (6)).

We find that the best performing version of EP involves the nudging-based perturbation approach and the centered scheme. Unless stated otherwise, and 'EP-trained network' refers to this configuration.

5.1 Comparison of Perturbation Approaches and Finite Difference Schemes

We compare the two perturbation approaches (nudging and clamping) and the four finite difference schemes (forward, backward, centered and random). For nudging-based perturbation, we use both the MSE and cross-entropy (CE) cost functions.



(a) CIFAR-100 (b) ImageNet 32x32

Figure 1: Comparison of perturbation approaches (CE-Nudge, MSE-Nudge and Clamp) and finite difference schemes (forward, backward, centered and random) on the VGG5 network trained on (a) CIFAR-100 and (b) ImageNet 32x32.

Figure 1 reports the results for the VGG5 trained on CIFAR100 and ImageNet 32x32. While all algorithms perform comparably on CIFAR100, important differences arise on the more challenging ImageNet 32x32 task. CE-Nudge performs significantly better than both MSE-Nudge and Clamp, except when employed in combination with the forward scheme. The best performing method is CE-Nudge with the centered scheme, in agreement with prior literature on EP [Laborieux et al., 2021, Scellier et al., 2023], but interestingly, both the random and backward schemes perform comparably with the centered scheme, which challenges the conclusions of these works, where larger gaps were observed.

These results highlight that, by allowing to choose the cost function to optimize, the nudging-based perturbation method of EP presents significant advantages over the clamping-based technique often used in the PC literature [4].

5.2 Sensitivity to Nudging Strength and Number of Nudge Iterations

Figure 2 displays the performance of a trained VGG5 network for different values of K (number of nudged iterations) and β (nudging strength). We observe that, as long as $K \geq 4$ and $0.0002 \leq \beta \leq 0.1$, the final error rate is mostly insensitive to the choice of K and β.

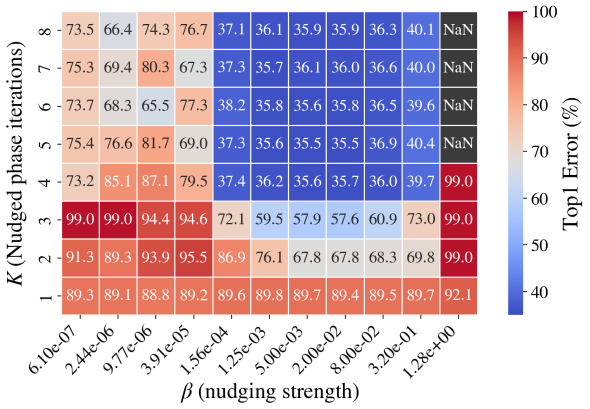


Figure 2: VGG5 network trained on CIFAR100 for 20 epochs. Sensitivity to the nudging strength (β) and number of nudge iterations (K).

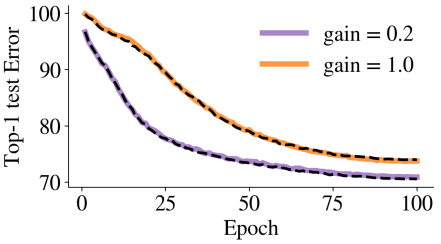


Figure 3: VGG5 trained on ImageNet 32x32 using the MSE cost function. Different weight initialization gains for the output layer lead to significantly different results ; however, EP consistently performs comparably to BP. Solid lines and dashed lines denote EP and BP, respectively.

Intuitively, K needs to be large enough that the network reaches equilibrium, but should be chosen as small as possible to avoid unnecessary computation. Empirically, we find that a good heuristic is to choose K to be equal to the number of layers in the network, allowing perturbations to fully propagate from outputs to inputs. Similarly, β needs to be small enough that EP gradients are

[4] We note that Pinchetti et al. [2022] proposed alternative clamping strategies for PCNs that also support cross-entropy.

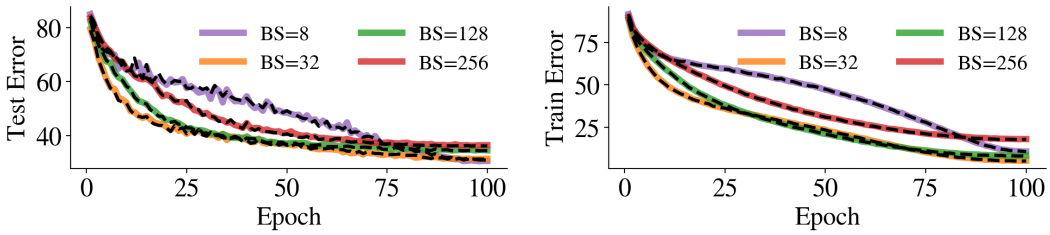
accurate (Equation (6)), but it needs to be large enough that perturbations can propagate throughout the network without vanishing (due to limited machine precision).

Interestingly, EP remains effective with nudging strengths as small as $\beta \approx 0.0002$, which is orders of magnitude smaller than values typically used in the EP literature for networks of similar depth.

Appendix A provides more insights into the nudge-phase equilibration dynamics, including sensitivity to the step size (α) and the traversal scheme, as well as equilibration curves (F as a function K).

5.3 Sensitivity to Batch Size, Cost Function and Weight Initialization

We now compare the performance of EP-trained PCNs against BP, and study their sensitivity to shared hyperparameters. Figure 4 compares EP and BP on CIFAR-100 for different batch sizes, while Table 1 reports the results of a more comprehensive comparison on four datasets (MNIST, CIFAR-10, CIFAR-100 and ImageNet 32x32) where we used two cost functions: the mean-squared error (MSE) and the cross-entropy (CE).



(a) Test error rate

(b) Train error rate

Figure 4: VGG5 trained on CIFAR100 using the MSE. Shapes of training curves depend on the batch size (BS), but not on the algorithm (EP vs BP). Solid lines denote EP and dashed lines denote BP.

Table 1: Sensitivity to the choice of the cost function: mean squared error (MSE) and cross-entropy (CE). Top1 (resp. Top5) refers to the test error rate in % for the classification task. In all cases, EP performs competitively with BP. Results were averaged over 3 random seeds.

Cost	Algorithm	MNIST	CIFAR-10	CIFAR-100		ImageNet 32x32	
				Top1	Top5	Top1	Top5
MSE	EP	0.32 ± 0.01	7.8 ± 0.1	30.1 ± 0.0	12.8 ± 0.1	70.8 ± 0.2	55.9 ± 0.3
	BP	0.33 ± 0.00	8.0 ± 0.0	30.1 ± 0.4	12.7 ± 0.5	70.7 ± 0.1	56.0 ± 0.2
CE	EP	0.45 ± 0.01	9.7 ± 0.2	34.7 ± 0.1	12.1 ± 0.2	60.0 ± 0.1	36.6 ± 0.4
	BP	0.48 ± 0.02	9.5 ± 0.0	34.2 ± 0.4	12.3 ± 0.2	59.8 ± 0.0	36.5 ± 0.0

In every situation, EP is competitive with BP. While the end performance depends prominently on the cost function and the batch size, it depends little on the learning algorithm used (BP or EP).

For ImageNet 32x32 experiments with the MSE, we find (Figure 3) that significantly better results are obtained using a different weight initialization scheme (scaling the output layer's weights by 0.2 instead of the default 1.0). Here again, however, EP performs comparably with BP.

These results position EP as an effective SGD-based (stochastic gradient descent) optimization method, while also highlighting that SGD is only one component of an effective learning system, with other factors - such as the cost function, initialization scheme and the batch size (and of course, the network architecture) - playing a crucial role in determining final performance.

5.4 Full-Size ImageNet Experiments

Table 2 reports the results (Top1 and Top5 error rates) obtained with the VGG10 network trained on full-size ImageNet. For these experiments, we used the cross-entropy (CE), and the VGG10 networks were trained for 50 epochs. We observe a small gap of 1.10% in top-5 accuracy between BP and centered EP, and another small gap of 0.56% between centered EP and random-sign EP.

Table 2: Performance of VGG10 on full-size ImageNet, compared to the results available in the related literature on PCNs and EP on smaller ImageNet variants (Tiny ImageNet and ImageNet 32x32). For each dataset, we write for comparison the number of images, size of images and number of classes (indicated in parentheses in this order). We note that, due to these differences, the performances on these three datasets are not directly comparable. Top1 (resp. Top5) refers to the test error rate in % for the top-1 (resp. top-5) classification task. Mean and standard deviation was computed across three random seeds. The EP-backward experiments are annotated with $*$ to indicate that the learning curves eventually collapsed to 99.9% top-1 error. Our results provide the first benchmark on full-size ImageNet for both EP-based training and PCNs.

Dataset	Model	Top1	Top5
Results available in the PCN and EP literature (smaller ImageNet variants)			
Tiny ImageNet (100K, 64x64, 200)	Pinchetti et al. [2024]	58.85	33.75
	Qi et al. [2025]	44.69	20.70
ImageNet 32x32 (1.28M, 32x32, 1000)	Laborieux and Zenke [2022]	63.5	39.2
	Høier et al. [2023]	58.52	35.10
	Nest and Ernoult [2024]	54.0	30.0
This work (full-size ImageNet)			
ImageNet (1.28M, 224x224, 1000)	VGG10 + EP + backward scheme	79.77 ± 3.95 *	57.65 ± 6.02*
	VGG10 + EP + random scheme	34.65 ± 0.32	13.83 ± 0.17
	VGG10 + EP + centered scheme	33.88 ± 0.19	13.27 ± 0.07
	VGG10 + BP (baseline)	32.20 ± 0.13	12.17 ± 0.03

We compare these results with existing literature on PCNs and EP. Crucially, no prior work on PCNs or EP has been carried out on full-size ImageNet classification. The best results available for PCNs were performed by Pinchetti et al. [2024], Qi et al. [2025] on Tiny Imagenet, a dataset with 13x less images (100,000 instead of 1.28 million), with a resolution 12x smaller (64x64 instead of 224x224) and with 5x fewer classes (200 instead of 1000) than ImageNet. Likewise, the best available results for EP were reported on ImageNet 32x32 [Nest and Ernoult, 2024], whose images are 50x smaller than the full-resolution version (32x32 instead of 224x224).

5.5 Integrating Skip Connections

To demonstrate that our algorithms are applicable to different network topologies, we perform experiments on a variant of the VGG10 model, with skip-layer connections. This network has a similar architecture as VGG10 but includes strided 1x1 convolutional skip-layer connections from layer 2 to layer 5, as well as from layer 5 to layer 8. We train this VGG10Skip model on full-size ImageNet and obtain similar performance as the VGG10 model. Specifically, centered EP achieves 33.92% Top1 and 13.31% Top5 test error rates, compared to 32.40% and 12.10% for the BP baseline.

We expect that better results could be obtained using deep ResNets with batch normalization, at the expense of significantly longer nudge-phase times. This is left for future work to investigate.

6 Conclusion

Prior numerical experiments of EP have primarily focused on Hopfield networks, with evaluations restricted to the 32x32-pixel version of ImageNet. Likewise, PCN experiments have remained limited to datasets of the size of Tiny ImageNet. By combining EP's nudging perturbation method with the centered finite-difference scheme and a novel equilibration dynamics for PCNs, we unlocked the training of a 10-layer convolutional PCN (VGG10) on full size ImageNet, providing the first benchmark on this dataset for both PCNs and EP-based training. Centered EP achieves a 13.27% top-5 test error rate, approaching the backpropagation baseline (12.20%) and outperforming AlexNet's performance (15.3%) [Krizhevsky et al., 2012]. Furthermore, random-sign EP closely trails centered EP with a 13.83% top-5 test error rate, while requiring a single nudge phase and roughly halving the duration of experiments.

Our work presents several limitations. First, our numerical evaluation focuses on VGG-style networks. Achieving higher performance with PCNs will likely require deeper architectures, such as ResNets. It remains to be seen if our methods, especially our novel nudged-phase equilibration algorithm, easily transfer to these architectures. Second, while our work focuses on a subclass of PCNs specifically aimed at approximating backpropagation in feedforward networks, a body of works highlights that PC weight updates capture richer optimization dynamics than standard gradient descent. For instance, Alonso et al. [2022, 2024] connect PC to Gauss-Newton updates and second-order optimization. Innocenti et al. [2024] showed that in linear networks, PC weight updates minimize a rescaled MSE loss and can incorporate arbitrarily high-order information depending on the depth of the network. Song et al. [2024] have also demonstrated empirical advantages of PC over first-order methods.

Beyond the PCNs studied in this work, EP has applications in a broad range of models and physical systems, including nonlinear resistive networks [Kendall et al., 2020] and coupled phase oscillators [Wang et al., 2024]. As one attempts to scale (the simulations of) EP in these systems, PCNs constitute a useful concept to decouple EP-related challenges from model-specific issues. Indeed, not only PCNs allow us to evaluate the performance of EP on standard feedforward neural network models and compare with the BP baseline, but they may also be used as a baseline for physical systems of the same size. For instance, the construction by Scellier and Mishra [2025] to convert an MLP into an approximately equivalent nonlinear resistor network provides a direct PCN benchmark for nonlinear resistor networks. In this endeavour, our experimental results strengthen EP's standing as an effective training method, and suggest that current challenges encountered in scaling simulations of EP in physical networks might come more from the trainability of these systems than from EP itself (or perhaps from the interplay between both).

Acknowledgements

This work was funded by Rain AI and the Advanced Research + Invention Agency's (ARIA) Scaling Compute programme.

References

Dimitri Vanden Abeele, Daniele Veraldi, Davide Pierangeli, Claudio Conti, and Serge Massar. Optical implementation of equilibrium propagation using spatial photonic ising machines. *arXiv preprint arXiv:2606.13454*, 2026.

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

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

Lauren E Altman, Menachem Stern, Andrea J Liu, and Douglas J Durian. Experimental demonstration of coupled learning in elastic networks. *Physical Review Applied*, 22(2):024053, 2024.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL `https://docs.pytorch.org/assets/pytorch2-2.pdf`.

Pierre Baldi and Fernando Pineda. Contrastive learning and neural oscillations. *Neural Computation*, 3(4):526–545, 1991.

Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In *Artificial Intelligence and Statistics*, pages 10–19. PMLR, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Sam Dillavou, Menachem Stern, Andrea J Liu, and Douglas J Durian. Demonstration of decentralized physics-driven learning. *Physical Review Applied*, 18(1):014040, 2022.

Sam Dillavou, Benjamin D Beyer, Menachem Stern, Andrea J Liu, Marc Z Miskin, and Douglas J Durian. Machine learning without a processor: Emergent learning in a nonlinear analog network. *Proceedings of the National Academy of Sciences*, 121(28):e2319718121, 2024.

Chen-Rui Fan, Bo Lu, Zhi-Hong Zhang, Run-Qing Zhang, Jing-Wei Wen, and Chuan Wang. Optimizing energy-based neural network training with coherent ising machine. *arXiv preprint arXiv:2606.09117*, 2026.

Cédric Goemaere, Johannes Deleu, and Thomas Demeester. Accelerating hopfield network dynamics: Beyond synchronous updates and forward euler. In *Proceedings of the 1st ECAI Workshop on "Machine Learning Meets Differential Equations: From Theory to Applications"*, volume 255 of *Proceedings of Machine Learning Research*, pages 1–21. PMLR, 2024.

Cédric Goemaere, Gaspard Oliviers, Rafal Bogacz, and Thomas Demeester. Error optimization: Overcoming exponential signal decay in deep predictive coding networks. *arXiv preprint arXiv:2505.20137*, 2025.

Akhilesh Gotmare, Valentin Thomas, Johanni Michael Brea, and Martin Jaggi. Decoupling backpropagation using constrained optimization methods. In *ICML's Workshop on Credit Assignment in Deep Learning and Deep Reinforcement Learning*, 2018.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton and James McClelland. Learning representations by recirculation. In *Neural information processing systems*, 1987.

Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, volume 448, pages 448–453. Washington, 1983.

Rasmus Høier, D Staudt, and Christopher Zach. Dual propagation: Accelerating contrastive hebbian learning with dyadic neurons. In *International Conference on Machine Learning*, pages 13141–13156. PMLR, 2023.

Rasmus Høier, Tugdual Kerjan, and Benjamin Scellier. Training a convergent energy transformer with equilibrium propagation. In *New Frontiers in Associative Memories-Workshop at ICLR 2026*, 2026.

Francesco Innocenti, El M Achour, Ryan Singh, and Christopher L Buckley. Only strict saddles in the energy landscape of predictive coding networks? *Advances in Neural Information Processing Systems*, 37:53649–53683, 2024.

Francesco Innocenti, El Mehdi Achour, and Christopher L Buckley. mu-pc: Scaling predictive coding to 100+ layer networks. *Advances in Neural Information Processing Systems*, 38:87414–87455, 2026.

Satoki Ishikawa, Rio Yokota, and Ryo Karakida. Local loss optimization in the infinite width: Stable parameterization of predictive coding networks and target propagation. In *International Conference on Learning Representations*, volume 2025, pages 49143–49182, 2025.

Jack Kendall. A gradient estimator for time-varying electrical networks with non-linear dissipation. *arXiv preprint arXiv:2103.05636*, 2021.

Jack Kendall, Ross Pantone, Kalpana Manickavasagam, Yoshua Bengio, and Benjamin Scellier. Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Axel Laborieux and Friedemann Zenke. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. *Advances in neural information processing systems*, 35: 12950–12963, 2022.

Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

Jérémie Laydevant, Danijela Marković, and Julie Grollier. Training an ising machine with equilibrium propagation. *Nature Communications*, 15(1):3671, 2024.

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.

TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

Danijela Marković, Alice Mizrahi, Damien Querlioz, and Julie Grollier. Physics for neuromorphic computing. *Nature Reviews Physics*, 2(9):499–510, 2020.

Serge Massar. Equilibrium propagation for learning in lagrangian dynamical systems. *Physical Review E*, 112(3):035304, 2025.

Serge Massar and Bortolo Matteo Mognetti. Equilibrium propagation: the quantum and the thermal cases. *Quantum Studies: Mathematics and Foundations*, 12(1):6, 2025.

Beren Millidge, Anil Seth, and Christopher L Buckley. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2021.

Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, 2022.

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

Ali Momeni, Babak Rahmani, Benjamin Scellier, Logan G Wright, Peter L McMahon, Clara C Wanjura, Yuhang Li, Anas Skalli, Natalia G Berloff, Tatsuhiro Onodera, et al. Training of physical neural networks. *Nature*, 645(8079):53–61, 2025.

Javier R Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist Models*, pages 10–17. Elsevier, 1991.

Timothy Nest and Maxence Ernoult. Towards training digitally-tied analog blocks via hybrid gradient computation. *Advances in Neural Information Processing Systems*, 37:83877–83914, 2024.

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

Filip Osana, Julie Grollier, and Damien Querlioz. Fast simulation of nonlinear deep resistive networks for energy-based computation. *arXiv preprint arXiv:2609.07356*, 2026.

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*, volume 35, pages 1280–1293. Curran Associates, Inc., 2022. doi: 10.52202/068431-0094.

Luca Pinchetti, Chang Qi, Oleh Lokshyn, Cornelius Emde, Amine M'Charrak, Mufeng Tang, Simon Frieder, Bayar Menzat, Gaspard Oliviers, Rafal Bogacz, Thomas Lukasiewicz, and Tommaso Salvatori. Benchmarking predictive coding networks – made simple. In *The Thirteenth International Conference on Learning Representations*, 2024.

Guillaume Pourcel, Debabrota Basu, Maxence Ernoult, and Aditya Gilra. Lagrangian-based equilibrium propagation: generalisation to arbitrary boundary conditions & equivalence with hamiltonian echo learning. *arXiv preprint arXiv:2506.06248*, 2025.

Chang Qi, Matteo Forasassi, Thomas Lukasiewicz, and Tommaso Salvatori. Towards the training of deeper predictive coding neural networks. *arXiv preprint arXiv:2506.23800*, 2025.

Théophile Rageau and Julie Grollier. Training and synchronizing oscillator networks with equilibrium propagation. *Neuromorphic Computing and Engineering*, 5(3):034008, 2025.

Karol Sajnok and Michał Matuszewski. Near-equilibrium propagation training in nonlinear wave systems. *arXiv preprint arXiv:2510.16084*, 2025.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR, 2009.

Benjamin Scellier. *A deep learning theory for neural networks grounded in physics*. PhD thesis, Université de Montréal, 2021.

Benjamin Scellier. A fast algorithm to simulate nonlinear resistive networks. In *International Conference on Machine Learning*, pages 43477–43503. PMLR, 2024a.

Benjamin Scellier. Quantum equilibrium propagation: gradient-descent training of quantum systems. In *NeurIPS 2024's Workshop on Machine Learning with New Compute Paradigms*, 2024b.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

Benjamin Scellier and Siddhartha Mishra. Universal approximation theorem for nonlinear resistive networks. *Physical Review Applied*, 23(4):044009, 2025.

Benjamin Scellier, Siddhartha Mishra, Yoshua Bengio, and Yann Ollivier. Agnostic physics-driven deep learning. *arXiv preprint arXiv:2205.15021*, 2022.

Benjamin Scellier, Maxence Ernoult, Jack Kendall, and Suhas Kumar. Energy-based learning algorithms for analog computing: a comparative study. *Advances in neural information processing systems*, 36:52705–52731, 2023.

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

Menachem Stern, Daniel Hexner, Jason W Rocks, and Andrea J Liu. Supervised learning in physical networks: From machine learning to learning machines. *Physical Review X*, 11(2):021045, 2021.

Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In *International conference on machine learning*, pages 2722–2731. PMLR, 2016.

Qingshan Wang, Clara C Wanjura, and Florian Marquardt. Training coupled phase oscillators as a neuromorphic platform using equilibrium propagation. *Neuromorphic Computing and Engineering*, 4(3):034014, 2024.

Clara C Wanjura and Florian Marquardt. Quantum equilibrium propagation for efficient training of quantum systems based on onsager reciprocity. *Nature Communications*, 16(1):6595, 2025.

James CR 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.

James CR Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250, 2019.

Su-in Yi, Jack D Kendall, R Stanley Williams, and Suhas Kumar. Activity-difference training of deep neural networks using memristor crossbars. *Nature Electronics*, 6(1):45–51, 2023.

Christopher Zach. Bilevel programs meet deep learning: A unifying view on inference learning methods. *arXiv preprint arXiv:2105.07231*, 2021.

Umais Zahid, Qinghai Guo, and Zafeirios Fountas. Predictive coding as a neuromorphic alternative to backpropagation: a critical evaluation. *Neural Computation*, 35(12):1881–1909, 2023.

Gianluca Zoppo, Francesco Marrone, Michele Bonnin, and Fernando Corinto. Equilibrium propagation and (memristor-based) oscillatory neural networks. In *2022 IEEE international symposium on circuits and systems (ISCAS)*, pages 639–643. IEEE, 2022.

A Nudge-Phase Dynamics: Traversal Scheme, Step Size, Modified PGD, and Equilibration Curves

In this appendix, we provide a complete description of the nudge-phase dynamics used in our experiments, based on a modified version of PGD. In particular, we empirically justify the use of our asynchronous traversal scheme, and the choice $\alpha = 1$ for the step size. We also take a close look at the nudge-phase equilibration curves, i.e. the total energy (F) as a function of the number of nudge iterations (K).

We note that our nudge-phase equilibration approach is different from that of Pinchetti et al. [2024], who employed standard GD with momentum, using tiny learning rates and synchronous layer updates.

A.1 Traversal Scheme and Step Size

Updating all the layers synchronously at each iteration of the PGD scheme does not work well in practice. Instead, we find that an 'asynchronous update' scheme works much better: at every iteration, we first update the layers of even indices (one half of the layers) and then we update the layers of odd indices (the other half of the layers). Relaxing all the layers once (first the even layers, then the odd layers) constitutes one *iteration*. We repeat as many iterations as is necessary until convergence to the nudged equilibrium. This asynchronous scheme is similar in spirit to the one used in Deep Boltzmann Machines [Salakhutdinov and Hinton, 2009], Deep Hopfield Networks [Scellier et al., 2023, Goemaere et al., 2024] and Deep Resistive Networks [Scellier, 2024a].

Figure 5 shows the performance (test error rates) of the VGG5 model trained on CIFAR-100 for different step sizes (α), different number of nudge-phase iterations (K) and different traversal schemes (synchronous vs asynchronous). The asynchronous update scheme with step size $\alpha = 1$ performs the best, and shows equivalent performance for 5, 10 and 20 nudged-phase iterations.

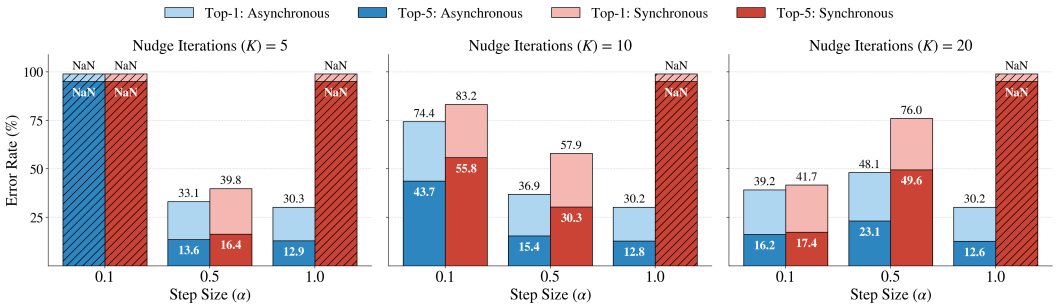


Figure 5: Test error rates obtained with the VGG5 trained on CIFAR100 using different traversal schedules (synchronous and asynchronous updates), number of nudged phase iterations K (5, 10 and 20) and step sizes α (0.1, 0.5, 1.0). NaN means that training failed and diverged, resulting in NaN values.

A.2 Modified Projected Gradient Descent (mod-PGD)

In our experiments, we used a modified version of PGD, which we found to work comparably or sometimes slightly better than standard PGD and GD. Recall that the GD update rule with step size $\alpha = 1$ writes:

$$h_k \leftarrow \text{ReLU}(a_k) - \frac{\partial(\frac{1}{2}\|\epsilon_{k+1}\|^2)}{\partial h_k}, \tag{19}$$

where a_k is a short hand for the pre-activations, defined such that $\text{ReLU}(a_k) = f_k(\theta_k, h_{k-1})$. Similarly, the PGD update rule (with step size $\alpha = 1$) writes:

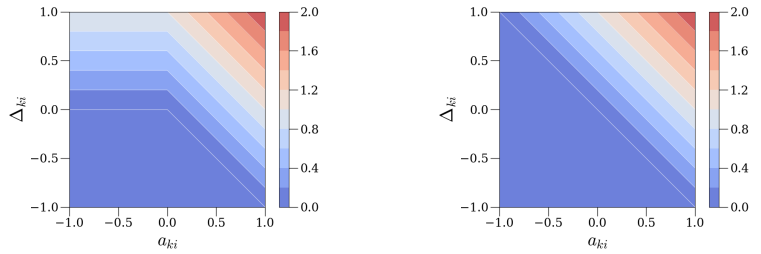
$$h_k \leftarrow \text{ReLU}\left(\text{ReLU}(a_k) - \frac{\partial(\frac{1}{2}\|\epsilon_{k+1}\|^2)}{\partial h_k}\right), \tag{20}$$

where the outer ReLU corresponds to the projection. In contrast, the 'mod-PGD' update rule reads

$$h_k \leftarrow \text{ReLU}\left(a_k - \frac{\partial(\frac{1}{2}\|\epsilon_{k+1}\|^2)}{\partial h_k}\right). \tag{21}$$

For the output layer, which does not use a nonlinearity, GD, PGD and mod-PGD results in the same update rule.

Figure 6 summarizes the behavior of the PGD (20) and mod-PGD (21) dynamics, as a function depending on the bottom-up signals to the k-th layer (a_k) and the top-down signals to the k-th layer ($\Delta_k := \frac{\partial(\frac{1}{2}\|\epsilon_{k+1}\|^2)}{\partial h_k}$). For a unit i in layer k, Equations (20) and (21) only differ in the case where $a_{ki} < 0$ and $\Delta_{ki} > 0$. This is reflected in Figure 6a and Figure 6b, which only differ in the top left quadrant. Notably for positive Δ_{ki}, the PGD dynamics ignores inhibitory bottom-up signals ($a_{ki} < 0$). The mod-PGD dynamics on the other hand allows negative bottom-up signals to act as an inhibitory buffer, meaning bottom-up inhibited units remain inactive unless the top-down signal is sufficiently strong. By tempering the top-down signals in this manner, the mod-PGD dynamics remain closer to the free phase equilibrium than the PGD dynamics.



(a) PGD (b) mod-PGD

Figure 6: Level curves of the update rules prescribed by PGD (RHS of Eq. (20)) and mod-PGD (RHS of Eq.(21)). We consider a single unit i in layer k.

Table 3 summarizes the experimental comparison between GD, PGD, and mod-PGD across datasets and number of nudge iterations (K).

When using $K = 5$ iterations on CIFAR-10, both GD and PGD exhibit training instability and eventually diverge. For these unstable configurations, the reported metrics represent the lowest error rates achieved prior to collapse. Increasing the number of nudge iterations to $K = 10$ eliminates this instability for both GD and PGD. On CIFAR-100 and ImageNet 32×32, mod-PGD achieves performance comparable to GD while slightly but consistently outperforming PGD across both $K = 5$ and $K = 10$. Notably, mod-PGD is the only algorithm that maintains training stability across all three datasets when constrained to $K = 5$ nudge iterations, motivating its adoption in our subsequent simulations. The corresponding training dynamics on CIFAR-10 are illustrated in Figure 7.

Table 3: Performance comparison of mod-PGD, PGD, and GD on the VGG-5 model trained with MSE-Nudge. K denotes the number of iterations in the nudge phase. Values indicate top-1 and top-5 error rates (%) averaged over 3 random seeds (± 1 std. dev.). Runs marked with † diverged during training (see Figure 7).

Algorithm (K iterations)	CIFAR-10 Top1	CIFAR-100 Top1	CIFAR-100 Top5	ImageNet 32x32 Top1	ImageNet 32x32 Top5
GD (K=5)	† 12.84 ± 0.48	29.91 ± 0.22	12.53 ± 0.33	70.87 ± 0.22	55.90 ± 0.16
GD (K=10)	7.73 ± 0.27	30.29 ± 0.16	12.62 ± 0.51	70.92 ± 0.03	56.06 ± 0.14
PGD (K=5)	† 18.84 ± 0.22	30.42 ± 0.02	12.95 ± 0.14	71.37 ± 0.19	56.27 ± 0.28
PGD (K=10)	7.78 ± 0.06	30.19 ± 0.29	12.86 ± 0.35	71.29 ± 0.09	56.47 ± 0.10
mod-PGD (K=5)	7.73 ± 0.14	30.04 ± 0.05	12.79 ± 0.26	70.80 ± 0.22	55.92 ± 0.28

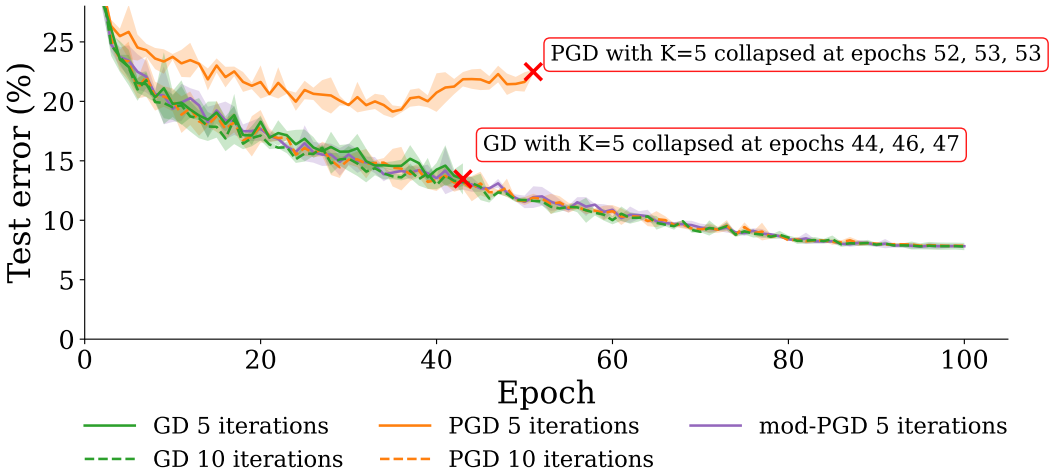


Figure 7: Training curves for GD, PGD, and mod-PGD using the VGG-5 model trained on CIFAR-10 with MSE-Nudge. Solid lines represent the mean over three random seeds; shaded regions indicate ± 1 standard deviation. Experiments with $K = 5$ for GD and PGD consistently collapsed. Red crosses mark the last epoch reached by all seeds before training loss became non-finite, with boxes indicating individual seed collapse epochs. Results correspond to those reported in Table 3.

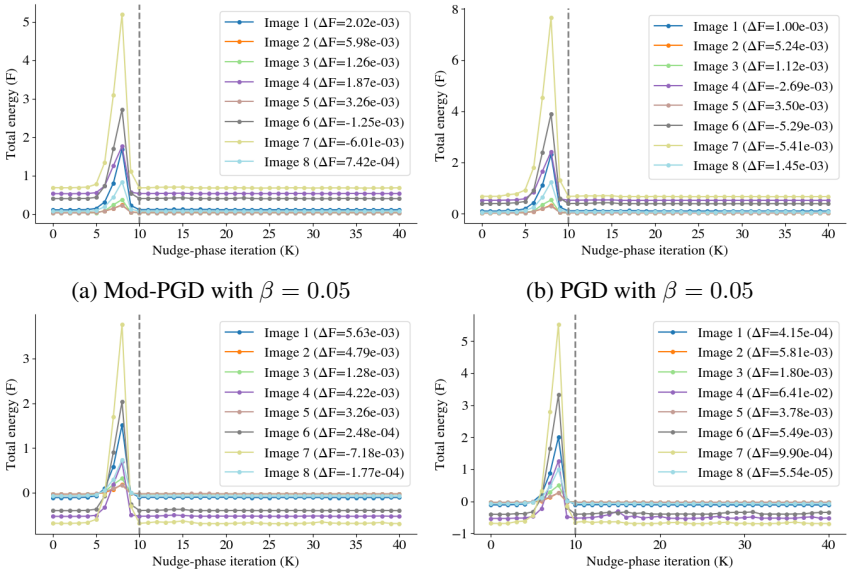
A.3 Nudge-Phase Equilibration Curves

Finally, we take a closer look at the nudge-phase equilibration dynamics. Figure 8 shows the total energy ($F = E + \beta C$) as a function of the number of nudge iterations (K) for the VGG10 model, using both PGD and mod-PGD. We make a few comments.

Perhaps the most important remark is that the total energy does not decrease as a function of K during the nudge-phase equilibration, directly contradicting the interpretation that the nudge-phase dynamics minimize the total energy. In fact, recalling that the initial state of the curve ($K = 0$) is the free equilibrium, and the final state ($K \geq 10$) is the nudge equilibrium, one sees that the nudge equilibrium often has higher energy than the free equilibrium, which translates as $\Delta F > 0$ in Figure 8. This shows that the nudge equilibrium is not, in general, the global minimum of F.

EP, however, only requires the first-order stationary condition of Eq. (5). Thus, the nudge equilibrium may as well be a local minimum or even a saddle point of F, and our GD/PGD/mod-PGD dynamics with step size $\alpha = 1$ might be better understood as fixed point iteration methods that find stationary points of F, rather than as minimizing F.

Finally, yhe equilibration curves show a significant spike in the total energy, starting around $K = 5$ and stabilizing at $K = 10$. These curves justify our choice of $K = 10$ in the VGG10 experiments. Using the asynchronous update scheme (where at each iteration one first updates the even layers and then the odd layers), the teaching signals introduced at the output layer at the beginning of the nudge phase propagate two layers per iteration. Hence, the beginning of the spike at $K = 5$ coincides with the minimal number of iterations required for the teaching signals to propagate from the output layer to the input layer. The spike is generally larger for PGD than for mod-PGD, which is consistent with our observation (Appendix A.2) that, in comparison with PGD, mod-PGD tempers top-down (error) signals.



(a) Mod-PGD with $\beta = 0.05$

(b) PGD with $\beta = 0.05$

(c) Mod-PGD with $\beta = -0.05$

(d) PGD with $\beta = -0.05$

Figure 8: Nudge-phase equilibration curves for the VGG10 model evaluated on 8 random images from the ImageNet validation set. The plot represents the total energy F as a function of the number of nudge iterations (K), for both mod-PGD and PGD as well as for both positive ($\beta = 0.05$) and negative ($\beta = -0.05$) values of the nudging parameter. The dashed gray line corresponds to $K = 10$, the value of K used in our training experiments. The energy gap between the free state and nudge state is defined as $\Delta F = F(K = 10) - F(K = 0)$.

B Experimental Details

In this appendix, we provide the implementation details for our experiments.

B.1 Datasets

We use five datasets: MNIST, CIFAR-10, CIFAR-100, ImageNet32 and (full-size) ImageNet.

The MNIST dataset is composed of images of handwritten digits [LeCun et al., 1998]. Each image x in the dataset is a 28×28 gray-scaled image and comes with a label $y \in \{0, 1, \ldots, 9\}$ indicating the digit that the image represents. The dataset contains 60,000 training images and 10,000 test images.

The CIFAR-10 dataset [Krizhevsky et al., 2009] consists of 60,000 colour images of 32×32 pixels. These images are split in 10 classes (each corresponding to an object or animal), with 6,000 images per class. The training set consists of 50,000 images and the test set of 10,000 images.

The CIFAR-100 dataset [Krizhevsky et al., 2009] also comprises 60,000 color images with a resolution of 32×32 pixels, featuring a diverse set of objects and animals. These images are categorized into 100 distinct classes, each containing 600 images. Like CIFAR-10, the dataset is divided into a training set with 50,000 images and a test set containing the remaining 10,000 images.

The ImageNet dataset [Deng et al., 2009] contains over a million high-resolution images, spanning thousands of object categories. Specifically, in the 2012 Large Scale Visual Recognition Challenge (ILSVRC2012), a popular subset of ImageNet, there are 1,000 distinct classes, each populated with a variable number of images. The ILSVRC2012 subset divides the dataset into a set for training, validation, and testing, facilitating comparisons across different approaches.

In addition to ImageNet (which here refers to the dataset with full-resolution images), we also consider the ImageNet32 variant, which consists of all images of the ImageNet dataset downsampled to 32x32 pixel images.

Data pre-processing and data augmentation. The data is normalized using the parameters provided in Table 4.

Table 4: Data normalization. We normalize the input images using the recommended mean (μ) and std (σ) values for each dataset. The MNIST images are gray-scale, i.e. they have a unique channel. The CIFAR-10, CIFAR-100 and ImageNet images are color images, i.e. they have three channels.

	mean (μ)	std (σ)
MNIST	0.1307	0.3081
CIFAR-10	(0.4914, 0.4822, 0.4465)	(0.2023, 0.1994, 0.2010)
CIFAR-100	(0.5071, 0.4867, 0.4408)	(0.2675, 0.2565, 0.2761)
ImageNet	(0.485, 0.456, 0.406)	(0.229, 0.224, 0.225)

To make the VGG5 model compatible with MNIST, each image of the MNIST dataset (of size 28x28) is augmented to a 32x32-pixel image by adding two pixels at the top, two pixels at the bottom, two pixels to the left and two pixels to the right.

Finally, during training, we use random horizontal flipping and random cropping on CIFAR-10, CIFAR-100 and ImageNet. These stochastic augmentations are not used at test time. These stochastic augmentations are not used on MNIST either.

B.2 VGG architectures

Table 5 and Table 6 contain the architectural details of the VGG5 and VGG10 models, respectively, as well as the hyperparameters used to obtain the results reported in Figure 4, Table 1 and Table 2. Both VGG models use a combination of stacked convolutions and pooling layers followed by dense layers. All convolutional operations use stride 1, while max pooling operations use stride 2.

Table 5: VGG5 architecture. num_inputs is 1 for MNIST, and 3 for other datasets (CIFAR10, CIFAR100 and ImageNet variants). num_outputs is 10 for MNIST and CIFAR10, 100 for CIFAR100, and 1000 for ImageNet and ImageNet32.

Operation	**Channels in**	**Channels out**	**Kernels**
Conv2d	num_inputs	128	3×3
Conv2d	128	256	3×3
MaxPool	–	–	2×2
Conv2d	256	512	3×3
MaxPool	–	–	2×2
Conv2d	512	512	3×3
MaxPool	–	–	2×2
Conv2d	512	512	3×3
MaxPool	–	–	2×2
Dense	num_outputs		

Table 6: VGG10 architecture

Operation	**Channels in**	**Channels out**	**Kernels**
Conv2d	3	64	3×3
MaxPool	–	–	2×2
Conv2d	64	128	3×3
MaxPool	–	–	2×2
Conv2d	128	256	3×3
Conv2d	256	256	3×3
MaxPool	–	–	2×2
Conv2d	256	512	3×3
Conv2d	512	512	3×3
MaxPool	–	–	2×2
Conv2d	512	512	3×3
Conv2d	512	512	3×3
MaxPool	–	–	2×2
Dense	2048		
Dense	1000		

Weight initialization. We initialize the weights of dense interactions and convolutional interactions according to

$$w_{ij} \sim \mathcal{U}(-c, +c), \qquad c = \gamma \sqrt{\frac{1}{\text{fan_mode}}} \tag{22}$$

which is the 'Kaiming uniform' scheme rescaled by a factor γ, that we call the 'gain' here (i.e. a scaling number). For both VGG5 and VGG10, dense weights of shape $(\text{size}_{\text{pre}}, \text{size}_{\text{post}})$, have $\text{fan}_{\text{in}} = \text{size}_{\text{pre}}$; for convolutional weights of shape $(\text{channel}_{\text{in}}, \text{channel}_{\text{out}}, \text{height}, \text{width})$, VGG5 uses $\text{fan}_{\text{in}} = \text{channel}_{\text{in}} \times \text{height} \times \text{width}$ and VGG10 uses $\text{fan}_{\text{out}} = \text{channel}_{\text{out}} \times \text{height} \times \text{width}$

The gain γ was consistently chosen to be 1, except in the setting of the VGG5 trained on ImageNet32 using the MSE where we found that using $\gamma = 0.2$ for the final (output) weight matrix led to significantly better results, as shown in Figure 3.

B.3 Hyperparameters related to training

We train our VGG networks with both equilibrium propagation (EP) and backpropagation (BP). Training using BP is done the usual way. As for EP, at each training step, we proceed as follows. First we pick a mini-batch of samples in the training set, x, and their corresponding labels, y. Then we perform a forward pass to compute the free equilibrium, which we store. Assuming we use the centered scheme, let $\beta > 0$ be the nudging value used for training. First, we set the nudging parameter to β and we perform K asynchronous iterations of mod-PGD (Eq. (21)). Next, we reset the state of the network to the free equilibrium, then we set the nudging parameter to $-\beta$, and we perform a new equilibration phase of K asynchronous iterations of mod-PGD. Finally, we update all the parameters simultaneously in a single 'parameter update' phase. See Table 7 and Table 8 for the hyperparameters used.

Optimizer and scheduler. We use mini-batch gradient descent (SGD) with momentum, weight decay and the Nesterov method. We also use a cosine-annealing scheduler for the learning rates with hyperparameters $T_{\max}$ and $\eta_{\min}$.

Hyperparameter Optimization. We used a two-stage hyperparameter tuning strategy, where we first tuned the hyperparameters shared by BP-based and EP-based training (weight initialization schemes, cost function, learning rates, weight decay, batch size, ...), and subsequently tuned the EP specific hyperparameters (perturbation approach, finite difference scheme, nudging strength β, number of nudge iterations K, and PGD variant). This two-stage strategy is justified by the fact that, when employing the nudging-based perturbation approach, in the limit when β tends to 0, the EP gradients are equal to the BP gradients. Tuning was done manually with hyperparameter sweeps, but without Optuna or other black-box tools.

Computational resources. The code for the simulations uses Pytorch 2.9 [Ansel et al., 2024] and TorchVision 0.24 [maintainers and contributors, 2016]. The simulations were carried on Nvidia A100 GPUs. The VGG10 experiments on ImageNet using EP took 18 days (resp. 10 days) for the centered scheme (resp. backward scheme and random scheme) to complete on a single GPU. The BP experiment took 36 hours to complete. Each VGG5 experiment on ImageNet32 using EP took 17 hours to complete.

Our implementation is based on PyTorch's eager execution mode, and could likely be accelerated by using *just in time* compilation (as is done in [Pinchetti et al., 2024]).

Back of the envelope FLOP estimate. Our EP experiments require one forward pass to find the free equilibrium, followed by K iterations of PGD for the nudge phase. For hidden layers, there is both bottom-up and top-down input to a layer so the cost of each nudged iteration is roughly 2x that of a forward pass. Thus the random, forward and backward EP schemes (which have a single nudged phase) require roughly $2K + 1$ times as many FLOPs as a single forward pass. The centered EP scheme (which has two nudged phases) requires roughly $4K + 1$ times as many FLOPs as a single forward pass. The BP experiments require one forward and one backward pass. We assume the forward and backward passes are approximately equally costly. Thus the random, forward and backward EP schemes require roughly $K + \frac{1}{2}$ times as many flops as BP and the centered EP scheme requires roughly $2K + \frac{1}{2}$ times as many FLOPs as BP. For the VGG10 network we have $K = 10$,

Table 7: Hyperparameter values used in the VGG5 experiments to obtain the results of Table 1.

	MNIST	CIFAR-10	CIFAR-100	ImageNet32
mini-batch size with MSE	16	16	16	256
mini-batch size cross-entropy (CE)	64	64	64	256
nudging (β)	0.02			
num. iterations of mod-PGD (K)	5			
learning rate (η) with MSE	0.04			
learning rate (η) with CE	0.01			
momentum	0.9			
weight decay	3 e-4			
number of epochs	100			
$T_{\max}$ (scheduler hyperparameter)	100			
$\eta_{\min}$ (scheduler hyperparameter)	1e-6			

Table 8: Hyperparameters used for the VGG10 training experiments on ImageNet.

	ImageNet
nudging strength (β)	0.05
number of nudge-phase iterations (K)	10
learning rates (for the trainable parameters)	0.02
momentum	0.9
weight decay	2 e-4
mini-batch size	256
number of epochs	50
cosine annealing scheduler parameter $T_{\max}$	50
cosine annealing scheduler parameter $\eta_{\min}$	2.0 e-6

so random, forward and backward requires 10.5 times the FLOPs of BP while centered requires 20.5 times the FLOPs of BP. If our simulations were purely compute bound then the run time ratios should closely match the FLOPs ratios, but this is not the case. We get $10\text{days}/1.5\text{days} = 6.67$ and $18\text{days}/1.5\text{days} = 12$, which suggest that our simulations are partially I/O bound.

C Traditional PCN Training, EP's Contrastive Function and the Penalty Method

In this appendix, we revisit several connections between PCNs, EP and the penalty method. In particular, we:

- derive the PCN energy function from its probabilistic interpretation,
- review the traditional learning algorithm for PCNs,
- relate the EP contrastive function to the standard PCN objective,
- discuss the connection between EP-trained PCNs and the penalty method.

C.1 Probabilistic Interpretation of PCNs

The PCN energy function is commonly motivated from a probabilistic perspective. For each layer k, the state h_k is assumed to be conditionally Gaussian given the previous layer:

$$h_k \sim \mathcal{N}\big(f_k(\theta_k, h_{k-1}), \sigma_k^2 I\big)\,, \tag{23}$$

where f_k is the feedforward transformation parameterized by θ_k, and σ_k is the standard deviation. Equivalently, the conditional density of h_k given h_{k-1} is

$$P_{\theta_k}(h_k \mid h_{k-1}) = \frac{1}{(2\pi\sigma_k^2)^{d_k/2}} \exp\!\left(-\frac{1}{2\sigma_k^2}\left\|h_k - f_k(\theta_k, h_{k-1})\right\|^2\right), \tag{24}$$

where d_k denotes the dimensionality of layer k. The full network therefore defines the conditional distribution

$$P_\theta(h_1, \ldots, h_L \mid h_0 = x) = \prod_{k=1}^{L} P_{\theta_k}(h_k \mid h_{k-1}), \tag{25}$$

where $h_0 = x$ is the network input. Given an input x, inference in a PCN consists of finding the most likely latent configuration:

$$\arg\max_h \log P_\theta(h \mid x), \tag{26}$$

where $h = (h_1, \ldots, h_L)$ denotes the network state. Taking the negative log-likelihood yields

$$-\log P_\theta(h \mid x) = \sum_{k=1}^{L}\left[\frac{d_k}{2}\log(2\pi\sigma_k^2) + \frac{1}{2\sigma_k^2}\left\|h_k - f_k(\theta_k, h_{k-1})\right\|^2\right]. \tag{27}$$

Choosing $\sigma_k = 1$ for all k, the negative log-likelihood reduces, up to an additive constant, to the PCN energy function of Eq. (10):

$$-\log P_\theta(h \mid x) = E_{\mathrm{PCN}}(\theta, x, h) + \text{const.} \tag{28}$$

Thus, maximizing the conditional likelihood is equivalent to minimizing the PCN energy function.

C.2 Traditional Learning Algorithm for PCNs

Under the probabilistic interpretation above, learning aims to maximize the conditional likelihood

$$\arg\max_\theta \log P_\theta(y \mid x), \tag{29}$$

where y is the desired target, associated with input x. However, evaluating this likelihood exactly requires marginalizing over all latent states:

$$\log P_\theta(y \mid x) = \log \int \exp(-E_{\mathrm{PCN}}(\theta, x, h_1, \ldots, h_{L-1}, y))\, dh_1 \cdots dh_{L-1}, \tag{30}$$

up to an additive constant. This integral is generally intractable. Traditional PCN learning therefore relies on a maximum a posteriori (MAP) approximation. Given a training pair (x, y), the output layer is clamped to the target value $h_L = y$, and the latent states are inferred by minimizing the energy:

$$h_{1:L-1}^{\mathrm{clamped}} = \arg\min_{h_1,\ldots,h_{L-1}} E_{\mathrm{PCN}}(\theta, x, h_1, \ldots, h_{L-1}, h_L = y). \tag{31}$$

The MAP approximation assumes that the integral is dominated by the lowest-energy configuration:

$$\int e^{-E(h)} dh \approx e^{-E(h^{\text{clamped}})}. \tag{32}$$

Under this approximation,

$$-\log P_\theta(y \mid x) \approx E_{\text{PCN}}(\theta, x, h_{1:L-1}^{\text{clamped}}, y) + \text{const}. \tag{33}$$

This leads to the objective

$$\mathcal{J}(\theta, x, y) = E_{\text{PCN}}(\theta, x, h_{1:L-1}^{\text{clamped}}, y), \tag{34}$$

which approximates the negative conditional log-likelihood up to an additive constant. The traditional PCN learning rule is obtained by differentiating this objective with respect to the parameters:

$$\nabla_\theta \mathcal{J}(\theta, x, y) = \frac{\partial E_{\text{PCN}}}{\partial \theta}(\theta, x, h_{1:L-1}^{\text{clamped}}, y) + \frac{\partial E_{\text{PCN}}}{\partial h}(\theta, x, h_{1:L-1}^{\text{clamped}}, y) \cdot \frac{\partial h_{1:L-1}^{\text{clamped}}}{\partial \theta} \tag{35}$$

$$= \frac{\partial E_{\text{PCN}}}{\partial \theta}(\theta, x, h_{1:L-1}^{\text{clamped}}, y), \tag{36}$$

where we use the first order equilibrium condition $\frac{\partial E_{\text{PCN}}}{\partial h}(\theta, x, h_{1:L-1}^{\text{clamped}}, y) = 0$. This recovers the standard predictive coding update rule: neural activities are first relaxed toward a minimum of the energy function, after which synaptic parameters are updated locally using prediction errors and presynaptic activities.

C.3 EP's Contrastive Function in PCNs

Unlike traditional PCN learning, EP does not require approximating an intractable posterior distribution and naturally accommodates arbitrary differentiable cost functions. EP introduces an explicit cost function C and defines the objective

$$\mathcal{L}(\theta, x, y) = C(h_\star^0, y), \tag{37}$$

where the free equilibrium state $h_\star^0$ satisfies

$$\frac{\partial E}{\partial h}\left(\theta, x, h_\star^0\right) = 0. \tag{38}$$

The original EP formula (Eq. (6)) states that

$$\nabla_\theta \mathcal{L}(\theta, x, y) = \frac{1}{\beta}\left[\frac{\partial F}{\partial \theta}(\theta, \beta, h_\star^\beta) - \frac{\partial F}{\partial \theta}(\theta, 0, h_\star^0)\right] + O(\beta), \tag{39}$$

where the total energy and nudged equilibrium are defined as:

$$F(\theta, \beta, h) = E(\theta, x, h) + \beta C(h, y), \qquad \frac{\partial F}{\partial h}\left(\theta, \beta, h_\star^\beta\right) = 0. \tag{40}$$

A more precise statement [Scellier et al., 2023] shows that EP performs exact gradient descent on a contrastive objective,

$$\mathcal{L}_\beta(\theta, x, y) = \frac{1}{\beta}\left[F(\theta, \beta, h_\star^\beta) - F(\theta, 0, h_\star^0)\right], \tag{41}$$

satisfying

$$\mathcal{L}_\beta(\theta, x, y) = \mathcal{L}(\theta, x, y) + O(\beta), \qquad \text{and} \tag{42}$$

$$\nabla_\theta \mathcal{L}_\beta(\theta, x, y) = \frac{1}{\beta}\left[\frac{\partial F}{\partial \theta}(\theta, \beta, h_\star^\beta) - \frac{\partial F}{\partial \theta}(\theta, 0, h_\star^0)\right]. \tag{43}$$

For PCNs, the free equilibrium satisfies $E_{\text{PCN}}(\theta, h_\star^0) = 0$, so that $F(\theta, 0, h_\star^0) = 0$. The EP contrastive objective therefore simplifies to

$$\mathcal{L}_\beta(\theta, x, y) = \frac{1}{\beta} F(\theta, \beta, h_\star^\beta) = \min_h \left[\frac{1}{\beta}\sum_{k=1}^{L} \|h_k - f_k(\theta_k, h_{k-1})\|^2 + C(h_L)\right]. \tag{44}$$

This objective is illustrated in Figure 9. The EP gradient also simplifies:

$$\nabla_\theta \mathcal{L}_\beta(\theta, x, y) = \frac{1}{\beta} \frac{\partial F}{\partial \theta}(\theta, \beta, h_\star^\beta). \tag{45}$$

To see this, we note that $\frac{\partial F}{\partial h}(\theta, 0, h_\star^0) = 0$ by definition, since $h_\star^0$ is a stationary point. Moreover, $\frac{d}{d\theta} F(\theta, 0, h_\star^0) = 0$ since $F(\theta, 0, h_\star^0)$ is identically zero as a function of θ. Therefore $\frac{\partial F}{\partial \theta}(\theta, 0, h_\star^0) = \frac{d}{d\theta} F(\theta, 0, h_\star^0) - \frac{\partial F}{\partial h}(\theta, 0, h_\star^0) \cdot \frac{\partial h_\star^0}{\partial \theta} = 0$.

As in Eq. (35), EP-based learning in PCNs proceeds in two stages: first, the network state is relaxed toward a stationary state of the total energy, and second, the parameters are updated using local derivatives evaluated at equilibrium.

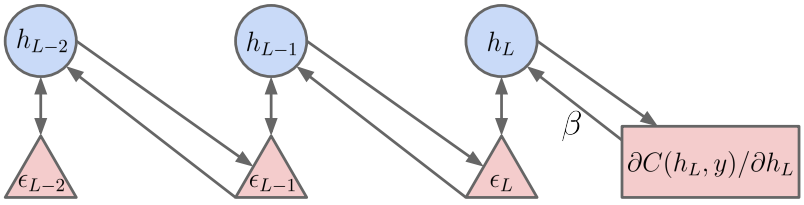


Figure 9: Illustration of the nudging-based perturbation approach of EP in a PCN. In contrast to the clamping-based approach typically employed in PCNs, it allows to choose the cost function C to optimize. Layers of units are represented by h_k, while layer-wise errors are denoted as $\epsilon_k = h_k - f_k(h_{k-1}, \theta_k)$. The total energy is the sum of squared errors and cost function, $F_{\text{PCN}} = \frac{1}{2} \sum_{k=1}^{L} \|\epsilon_k\|^2 + \beta C(h_L, y)$. During inference (free phase), the nudging parameter β is set to zero, so that no teaching signal enters the network. During training (nudge phase), β is set to a non-zero value, so that $-\beta \frac{\partial C}{\partial h_L}$ nudges the output units h_L as in Eq. (18). The centered scheme, which performs best in our experiments, requires two nudged equilibria: one with $\beta > 0$ and one with $\beta < 0$. The diagram is adapted from Millidge et al. [2022].

C.4 Relation to the Penalty Method

The function

$$\frac{1}{\beta} F(\theta, \beta, h) = C(h_L, y) + \frac{1}{\beta} \sum_{k=1}^{L} \|h_k - f_k(\theta_k, h_{k-1})\|^2 \tag{46}$$

appearing in the definition of $\mathcal{L}_\beta(\theta)$ can be viewed as a penalty function associated with the following constrained minimization problem:

$$\text{minimize } C(h_L), \qquad \text{subject to } h_k = f_k(\theta_k, h_{k-1}), \qquad 1 \le k \le L. \tag{47}$$

This essentially recovers the penalty method. As the coefficient $\lambda = 1/\beta$ tends to $+\infty$, the unconstrained problem is equivalent to the constrained problem.

Similar ideas to train feedforward networks using alternatives to backpropagation have been explored in multiple works [Carreira-Perpinan and Wang, 2014, Taylor et al., 2016, Gotmare et al., 2018, Zach, 2021].

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: Our work presents the first demonstration of both PCNs and EP-based training at ImageNet scale.

   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: The limitations of our work are addressed in the conclusion section. We also explain in section 4.1 that it remains unclear whether the nudge-phased dynamics could be efficiently realized in neuromorphic hardware.

   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: [N/A]

Justification: While there is no novel formal Theorem or proof in our manuscript, we clearly present EP and the conditions under which it applies (Section 3), and show that PCNs satisfy these conditions and can therefore be trained using EP (Section 4).

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: In Appendix B, we provide all the experimental details required to reproduce the results presented in Section 5.

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: We provide clearly structured and well documented code to reproduce the results presented in Section 5. The code is attached in the supplemental material.

   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: The full details of our experiments are provided in Appendix B.

   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: Tables 1, 2 and3 report the mean and standard deviations of experimental results averaged over three runs each.

   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: The full details of experiments are provided in Appendix B. Our most computationally demanding experiment (the VGG10 trained using the centered version of EP) required 18 days on a single A100 GPU.

   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 work does not involve human subjects and only uses publicly available datasets (MNIST, CIFAR-10, CIFAR-100 and ImageNet). Our work has no immediate application.

   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: [N/A]

    Justification: Our work has no immediate application. It is concerned with EP, an unconventional computing paradigm that might, one day, allow us to build new processors for machine learning that are more energy efficient than GPUs. This could be beneficial to society as AI systems become less energy demanding and more affordable, or detrimental

as AI systems can be deployed more pervasively in potentially harmful ways (e.g. mass surveillance, profiling and automated warfare).

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 poses no such risks.

    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: The datasets used are cited, and the code we use for the simulations is our own.

    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: The code to reproduce our simulations is made publicly available and is documented.

    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 nor 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: LLMs were not used for any important purpose in our work.

    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.