PCX: Benchmarking Predictive Coding Networks – Made Simple

Luca Pinchetti[1], **Chang Qi**[2], **Oleh Lokshyn**[2], **Gaspard Olivers**[3], **Cornelius Emde**[1], **Mufeng Tang**[3], **Amine M'Charrak**[1], **Simon Frieder**[1], **Bayar Menzat**[2], **Rafal Bogcaz**[3], **Thomas Lukasiewicz**[2,1], **Tommaso Salvatori**[4,2]
[1]Department of Computer Science, University of Oxford, Oxford, UK
[2]Institute of Logic and Computation, Vienna University of Technology, Vienna, Austria
[3]MRC Brain Network Dynamics Unit, University of Oxford, UK
[4]VERSES AI Research Lab, Los Angeles, US

Abstract

We propose a library for research in predictive coding (PC) called PCX. Its focus lies on performance and simplicity, and provides a user-friendly, deep-learning oriented interface. With it, we implement a large set of benchmarks for the community to use for their experiments. Our goal is to encourage collaboration and reproducibility in PC research: Most works propose their own tasks and architectures, fail to rigorously compare one against each other, and focus on small-scale tasks. Providing a simple and fast open-source library adopted by the whole community would address all of these concerns. We perform extensive benchmarks using multiple algorithms to unambiguously resolve what the state-of-the-art is, highlighting limitations inherent to PC that should be addressed. Thanks to the efficiency of PCX, we are able to analyze larger architectures than commonly used, providing baselines to galvanize community efforts towards one of the main open problems in the field: scalability. The code for PCX is available at `https://github.com/liukidar/pcax`.

1 Introduction

In 1999, Rao and Ballard [1999] proposed a hierarchical formulation of predictive coding (PC) as a model of visual processing. Recently, researchers realized that this framework could be used to train neural networks using a bio-plausible learning rule [Whittington and Bogacz, 2017]. This has led to different directions of research that either explored interesting properties of PC networks, such as their robustness [Song et al., 2024, Alonso et al., 2022] and flexibility [Salvatori et al., 2022], or proposed variations to improve the performance on specific tasks [Salvatori et al., 2024]. While interesting and important for the progress of the field, these lines of research have the tendency of not comparing their results against other papers or those of related fields and they only perform small-scale experiments. The field is hence avoiding what we believe to be the most important open problem: scaling such results to large scale tasks.

There are two reasons why such an important problem has been overlooked. First, it is a hard problem, and it is still unclear why PC is able to perform as well as classical gradient descent with backprop only up to a certain scale, which is of small convolutional models trained to classify the CIFAR10 dataset [Salvatori et al., 2024]. Understanding the reason for this limitation would allow us, for example, to develop regularization techniques to allow better performance on more complex tasks, similarly to dropout and batch normalization [Srivastava et al., 2014, Ioffe and Szegedy, 2015]. Second, the lack of specialized libraries makes PC models extremely slow: a full hyperparameter search on a small convolutional network can take days. At the same time, reproducibility and iterative contributions are made almost impossible by the absence of a common evaluation framework,

38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks.

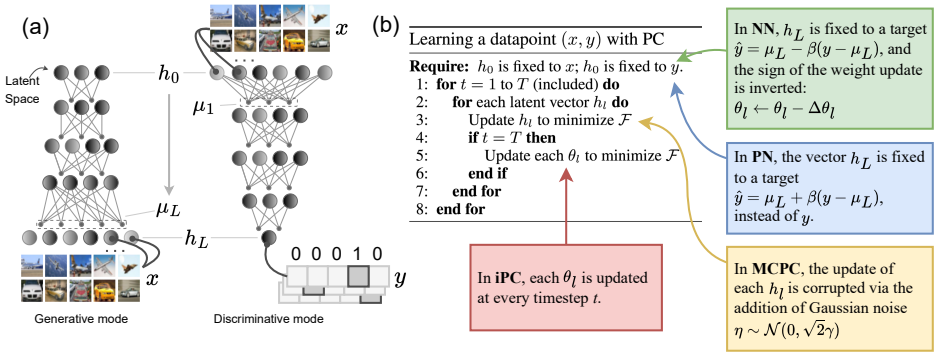


Figure 1: Left: Generative and discriminative modes, Right: Pseudocode of one parameter update of PC in supervised learning, as well as an informal description of the different algorithms considered in this work.

as implementation details or code are rarely provided. In this work, we make first steps toward addressing these problems with three contributions: *tool*, *benchmarking*, and *analysis*.

Tool. We release an open-source library for accelerated training for predictive coding called PCX. It uses JAX [Bradbury et al., 2018], but offers user-friendly syntax inspired from PyTorch as it implements an object-oriented interface (on top of the functional JAX approach) which ensures reliability, extendability, and compatibility with ongoing research developments. Furthermore, it supports JAX's Just-In-Time (JIT) compilation, making it extremely efficient, allowing easy development and execution of PC networks. Analogously to other deep learning libraries, PCX defines "core building blocks" that can be assembled together to build PC networks.

Benchmarking. We propose a uniform set of tasks, datasets, metrics, and architectures that should be used as a skeleton to test the performance of future variations of PC. The tasks that we propose are the standard ones in computer vision: image classification and generation. The models and datasets are picked according to two criteria: (i) to allow researchers to test their algorithm from the easiest MLP model to more complex ones, where we failed to get acceptable results, and should hence pave the way for future research; (ii) to favor the comparison against related fields in the literature, such as equilibrium propagation [Scellier and Bengio, 2017]. To this end, we have picked some of the models that are consistently used in their research papers.

Analysis. We evaluate such benchmarks and to provide the baselines for future research by performing an extensive comparative study between different hyperparameters and PC algorithms. We considered standard PC, incremental PC [Salvatori et al., 2024], PC with Langevin dynamics [Oliviers et al., 2024, Zahid et al., 2023], and nudged PC, as done in the Eqprop literature [Scellier and Bengio, 2017, Scellier et al., 2024]. This is also the first time nudging algorithms were applied in PC models. In terms of quantitative contributions, we get state-of-the-art results for PC on multiple benchmarks and show for the first time that it is able to perform well on more complex datasets, such as CIFAR100 and Tiny Imagenet, where we get results comparable to these of backdrop. In image generation tasks, we present experiments on datasets of colored images. We conclude with an analysis on credit assignment, which tries to shed light on current PC problems. To this end, we believe that future effort in the field should aim towards improving the numbers that we show in this work, as they represent the state of the art in the field.

2 Background and Notation

Predictive coding networks (PCNs) are hierarchical Gaussian generative models with L levels with parameters $\theta = \{\theta_0, \theta_1, \theta_2, ..., \theta_L\}$, in which each level models a multi-variate distribution parameterized by activation of the preceding level. Let h_l be the realization of the vector of random variables H_l of level l, then we have that

$$P_\theta(h_0, h_1, \ldots, h_L) = P_{\theta_0}(h_0) P_{\theta_1}(h_1|h_0) \cdots P_{\theta_L}(h_L|h_{L-1}).$$

For simplicity, we write $P_{\theta_l}(h_l)$ instead of the likelihood of H_l parameterized by θ_l evaluated at h_l, $P_{\theta_l}(H_l = h_l)$. PC assumes that the prior on h_0 and the relationships between levels are governed by a normal distribution parameterized as follows:

$$\begin{aligned} P_{\theta_0}(h_0) &= \mathcal{N}(h_0, \mu_0, \Sigma_0), & \mu_0 &= \theta_0, \\ P_{\theta_l}(h_l|h_{l-1}) &= \mathcal{N}(h_l; \mu_l, \Sigma_l), & \mu_l &= f_l(h_{l-1}, \theta_l), \end{aligned}$$

where θ_l are the learnable weights of the transformation f_l, and Σ_l is a covariance matrix. As it is standard practice, Σ_l is be fixed to the identity matrix [Whittington and Bogacz, 2017]. We refer to the scalar random variables of H_l as neurons. If, for example, $\theta_l = (W_l, b_l)$ and $f_l(h_{l-1}, \theta_l) = \sigma_l(W_l h_{l-1} + b_l)$, then the neurons in level $l-1$ are connected to level l via a linear operation, followed by a non linear map, that is the analogous to a fully connected layer in standard deep learning. Intuitively, θ are the learnable weights of the model, while $h = \{h_0, h_1, ..., h_L\}$ is data-point-dependent latent state, containing the abstract representations for the given observations.

Training. In supervised settings, training consists of learning the relationship between given pairs of input-output observations (x, y). In PC, this is performed by maximizing the joint likelihood of our generative model with the latent vectors h_0 and h_L respectively fixed to the input and label of the provided data-point: $P_\theta(h|_{h_0=x, h_L=y}) = P_\theta(h_L = y, \dots, h_1, h_0 = x)$. This is achieved by minimizing the so-called variational free energy $\mathcal{F}$ [Friston et al., 2007]:

$$\mathcal{F}(h, \theta) = -\ln P_\theta(h) = -\ln \left(\mathcal{N}(h_0|\mu_0) \prod_{l=1}^{L} \mathcal{N}(h_l; f_l(h_{l-1}, \theta_l)) \right) = \sum_{l=0}^{L} \frac{1}{2}(h_l - \mu_l)^2 + k. \quad (1)$$

The quantity $\epsilon_l = (h_l - \mu_l)$ is often referred to as *prediction error* of layer l, being the difference between the predicted activation μ_l and the current state h_l. Refer to the appendix, for a full derivation of Eq. (1). To minimize $\mathcal{F}$, the Expectation-Maximization (EM) [Dempster et al., 1977] algorithm is used by iteratively optimizing first the state h, and then the weights θ according to the equations

$$h^* = \arg\min_h \mathcal{F}(h, \theta), \quad \theta^* = \arg\min_\theta \mathcal{F}(h^*, \theta). \quad (2)$$

We refer to the first step described by Eq. (2) as *inference* and to the second as *learning* phase. In practice, we do not train on a single pair (x, y) but on a dataset split in mini-batches that are subsequently used to train the model parameters. Furthermore, both inference and learning are approximated via gradient descent on the variational free energy. In the inference phase, firstly h is *initialized* to an initial value $h^{(0)}$, and, then, it is optimized for T iterations. Then, during the learning phase we use the newly computed values to perform a single update on the weights θ. The gradients of the variational free energy with respect to both h and θ are as follows:

$$\nabla h_l = \frac{\partial \mathcal{F}}{\partial h_l} = \frac{1}{2}\left(\frac{\partial \epsilon_l^2}{\partial h_l} + \frac{\partial \epsilon_{l+1}^2}{\partial h_l} \right), \qquad \nabla \theta_l = \frac{\partial \mathcal{F}}{\partial \theta_l} = \frac{1}{2}\frac{\partial \epsilon_l^2}{\partial \theta_l}. \quad (3)$$

Then, a new batch of data points is provided to the model and the process is repeated until convergence. As highlighted by Eq. (3), each state and each parameter is updated using local information as the gradients depend exclusively on the pre and post-synaptic errors ϵ_l and ϵ_{l+1}. This is the main reason why, in contrast to BP, PC is a local algorithm and is considered more biologically plausible.

Evaluation. This phase is similar to the inference phase, with the difference that we perform it on a test point $\bar{x}$, used to infer the label $\bar{y}$. This is achieved by fixing $h_0 = \bar{x}$ and compute the most likely value of the latent states $h^*|_{h_0=\bar{x}}$, again using the state gradients of Eq. (3). We refer to this as *discriminative* mode. In practice, for discriminiative networks, the values of the latent states computed this way are equivalent to those obtained via a *forward* pass, that is setting $h_l^{(0)} = \mu_l^{(0)}$ for every $l \neq 0$, as it corresponds to the global minimum of $\mathcal{F}$ [Frieder and Lukasiewicz, 2022].

Generative Mode. So far, we have only described how to use PCNs to perform supervised training. However, as we will see in the experimental section, such models can also be used (and were initially developed to be used) to perform unsupervised learning tasks. Given a datapoint x, the goal is to use PCNs to compress the information of x into a latent representation, conceptually similar to how variational autoencoders work [Kingma and Welling, 2013]. Such a compression, that should contain all the information needed to generate x, is computed by fixing the state vector h_L to the data-point, and run inference – that is, we maximize $P_\theta(h|_{h_L=x})$ via gradient descent on h – until the process has converged. The compressed representation will then be the value of h_0 at convergence (or, in practice, after T steps). If we are training the model, we then perform a gradient update on the parameters to minimize the variational free energy of Eq.(1), as we do in supervised learning. A sketch of the discriminative and generative ways of training PCNs is represented in Fig. 1(a).

3 Implementation Details

PCX is developed on top of JAX, focusing on performance and versatility, and is built upon the following concepts: *compatibility*, *modularity*, and *efficiency*.

Epoch time (seconds)	BP	PC (ours)	iPC	PC (Song)
MLP - FashionMNIST	$1.01^{\pm 0.01}$	$0.94^{\pm 0.01}$	$1.82^{\pm 0.02}$	$4.47^{\pm 0.19}$
VGG-5 - CIFAR-100	$1.29^{\pm 0.00}$	$5.46^{\pm 0.01}$	$10.17^{\pm 0.00}$	$12.96^{\pm 0.06}$
VGG-7 - Tiny ImageNet	$7.52^{\pm 0.03}$	$57.52^{\pm 0.21}$	$96.29^{\pm 0.01}$	$137.08^{\pm 0.08}$

Table 1: Comparison of the training times of BP against PC on different architectures and datasets.

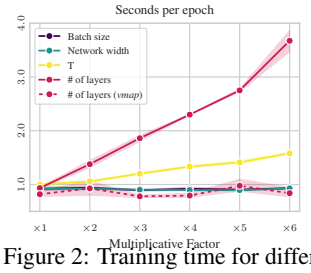


Figure 2: Training time for different network configurations.

Compatibility: PCX shares the same philosophy of equinox [Kidger and Garcia, 2021], according to which models are just PyTrees. Consequently, it is fully compatible, using a complete functional approach, with both libraries and many other tools developed for JAX, such as diffrax [Kidger, 2021] and optax [DeepMind et al., 2020]. To this end, it will be straightforward to implement novel development in deep learning into PCX. However, it also offers an imperative object-oriented interface, which allows to build PCNs following a PyTorch-like style.

Modularity: Thanks to the object-oriented abstraction, we built the modular primitives that can be combined to create a PCN, mainly: *EnergyModules* (i.e., the module class, representing an abstract energy-based model), *Vodes* (i.e., the vectorised nodes storing the state h), *Optims* (i.e., the optimizers), to perform the inference and learning process in a predictive coding network), and various standard *Layers*. Each benchmark we showcase in this work can be obtained by combining and configuring different "blocks" as needed.

Efficiency: PCX is built on JAX as the latter offers widespread support for Just-In-Time compilation. From our initial benchmarks, we observed a speed-up of up to 50x when compiling a PCN. We believe that this stark difference is due to the nature of PC, which relies on multiple smaller operations compared to backpropagation, i.e., the T inference step performed in each layer, and thus is more affected by the function calls overhead present in eager execution mode. Thus, we ensured that the full training loop can be jitted, and all PCX primitives are compatible with it.

PCX offers a unified interface to test multiple variations of PC on several tasks. Our modular code base can easily be expanded in the future to support new variations of PC, as we show complete compatibility with existing variations and training techniques. This is different from, for example, the monolithic or low-level approaches used in [Song, 2024] and [Ororbia and Kifer, 2022], respectively. In Appendix A, we provide an introduction of the library. Refer to the tutorial notebooks in the *examples* folder of the GitHub repository for a comprehensive overview of the available tools.

3.1 Computational resources and limitations.

We measured the wall-clock time of our PCNs implementation against another existing open-source library [Song, 2024] used in many PC works [Song et al., 2024, Salvatori et al., 2021, 2022, Tang et al., 2023], as well as comparing it with equivalent BP-trained networks (developed also with PCX for a fair comparison). Tab. 1 reports the measured time per epoch, averaged over 5 trials, using a A100 GPU. We also outperform alternative methods such as Eqprop: using the same architecture on CIFAR100, the authors report that one epochs takes ≈ 110 seconds, while we take ≈ 5.5 on the same hardware [Scellier et al., 2024]. However, this is not an apple-to-apple comparison, as the authors are more concerned with simulations on analog circuits, rather than achieving optimal GPU usage.

The efficiency of PCX could be further increased by fully parallelizing all the operations. In fact, in its current state, JIT is unable to parallelize the executions of the layers; a problem that can be addressed with the JAX primitive *vmap*, but only in the unpractical case where all the layers have the same dimension. To test how different hyperparameters of the model influence the training speed, we have taken a feedforward model, and trained it multiple times, each time increasing a specific hyperparameter by a multiplicative factor. The results, reported in Fig. 2, show that the two parameters that increase the training time are the number of layers L, and the number of steps T. Ideally, only T should affect the training time as inference is an inherently sequential process that cannot be parallelized, but this is not the case, as the time scales linearly with the amount of layers. Details are reported in Appendix F.

4 Experiments and Benchmarks

The benchmark that we propose is a standardized set of models, datasets, and training and testing procedure that have been consistently used to evaluate predictive coding, but in a non-uniform way.

Table 2: Test accuracies of the different algorithms on different datasets.

% Accuracy	PC-CE	PC-SE	PN	NN	CN	iPC	BP-CE	BP-SE
MLP								
MNIST	$98.11^{\pm 0.03}$	$98.26^{\pm 0.04}$	$98.36^{\pm 0.06}$	$98.26^{\pm 0.07}$	$98.23^{\pm 0.09}$	$\mathbf{98.45^{\pm 0.09}}$	$98.07^{\pm 0.06}$	$98.29^{\pm 0.08}$
FashionMNIST	$89.16^{\pm 0.08}$	$89.58^{\pm 0.13}$	$89.57^{\pm 0.08}$	$89.46^{\pm 0.08}$	$89.56^{\pm 0.05}$	$\mathbf{89.90^{\pm 0.06}}$	$89.04^{\pm 0.08}$	$89.48^{\pm 0.07}$
VGG-5								
CIFAR-10	$88.06^{\pm 0.13}$	$87.98^{\pm 0.11}$	$88.42^{\pm 0.66}$	$88.83^{\pm 0.04}$	$\mathbf{89.47^{\pm 0.13}}$	$85.51^{\pm 0.12}$	$88.11^{\pm 0.13}$	$89.43^{\pm 0.12}$
CIFAR-100 (Top-1)	$60.00^{\pm 0.19}$	$54.08^{\pm 1.66}$	$64.70^{\pm 0.25}$	$65.46^{\pm 0.05}$	$\mathbf{67.19^{\pm 0.24}}$	$56.07^{\pm 0.16}$	$60.82^{\pm 0.10}$	$66.28^{\pm 0.23}$
CIFAR-100 (Top-5)	$84.97^{\pm 0.19}$	$78.70^{\pm 1.00}$	$84.74^{\pm 0.38}$	$85.15^{\pm 0.16}$	$\mathbf{86.60^{\pm 0.18}}$	$78.91^{\pm 0.23}$	$85.84^{\pm 0.14}$	$85.85^{\pm 0.27}$
Tiny ImageNet (Top-1)	$41.29^{\pm 0.2}$	$30.28^{\pm 0.2}$	$34.61^{\pm 0.2}$	$\mathbf{46.40^{\pm 0.1}}$	$46.38^{\pm 0.11}$	$29.94^{\pm 0.47}$	$43.72^{\pm 0.1}$	$44.90^{\pm 0.2}$
Tiny ImageNet (Top-5)	$66.68^{\pm 0.09}$	$57.31^{\pm 0.21}$	$59.91^{\pm 0.24}$	$68.50^{\pm 0.18}$	$69.06^{\pm 0.10}$	$54.73^{\pm 0.52}$	$\mathbf{69.23^{\pm 0.23}}$	$65.26^{\pm 0.37}$
VGG-7								
CIFAR-100 (Top-1)	$56.80^{\pm 0.14}$	$37.52^{\pm 2.60}$	$56.56^{\pm 0.13}$	$59.97^{\pm 0.41}$	$64.76^{\pm 0.17}$	$43.99^{\pm 0.30}$	$59.96^{\pm 0.10}$	$\mathbf{65.36^{\pm 0.15}}$
CIFAR-100 (Top-5)	$83.00^{\pm 0.09}$	$66.73^{\pm 2.37}$	$81.52^{\pm 0.17}$	$81.50^{\pm 0.41}$	$84.65^{\pm 0.18}$	$73.23^{\pm 0.30}$	$\mathbf{85.61^{\pm 0.10}}$	$84.41^{\pm 0.26}$
Tiny ImageNet (Top-1)	$41.15^{\pm 0.14}$	$21.28^{\pm 0.46}$	$25.53^{\pm 0.77}$	$39.49^{\pm 2.69}$	$35.59^{\pm 7.69}$	$19.76^{\pm 0.15}$	$45.32^{\pm 0.11}$	$\mathbf{46.08^{\pm 0.15}}$
Tiny ImageNet (Top-5)	$66.25^{\pm 0.11}$	$44.92^{\pm 0.27}$	$50.06^{\pm 0.84}$	$64.66^{\pm 1.95}$	$59.63^{\pm 6.00}$	$40.36^{\pm 0.22}$	$\mathbf{69.64^{\pm 0.18}}$	$66.65^{\pm 0.20}$

By unifying all these benchmarks and training techniques within a single code base, we provide a fair and complete comparison between existing alternatives and ensure reproducibility and expandability for future iterations. To this end, other researchers can code their new algorithms easily with PCX, and simply test it on all of our proposed dataset-model-evaluation benchmark combinations that we provide as part of the library. Here, for a comprehensive evaluation, we test our models on multiple computer vision datasets, MNIST, FashionMNIST, CIFAR10/100, CelebA, and Tiny ImageNET; on models of increasing complexity, with both feedforward and convolutional/de-convolutional layers; and multiple learning algorithms present in the literature. This section is divided in two areas, that correspond to discriminative (supervised) and generative (unsupervised) inference tasks. A sketch illustrating the two modes is provided in Fig. 1.

Algorithms. We consider various different learning algorithms present in the literature: (1) Standard PC, already discussed in the background section; (2) Incremental PC (iPC), a simple and recently proposed modification where the weight parameters are updated alongside the latent variables at every time step; (3) Monte Carlo PC (MCPC), obtained by applying unadjusted Langevin dynamics to the inference process; (4) Positive nudging (PN), where the target used is obtained by a small perturbation of the output towards the original, 1-hot label; (5) Negative nudging (NN), where the target is obtained by a small perturbation *away* from the target, and updating the weights in the opposite direction. Among these, PC, iPC, and MCPC will be used for the generative mode, and PC, iPC, PN, and NN for the discriminative mode. See Fig. 1, and the supplementary material, for a more detailed description of such algorithms.

4.1 Discriminative Mode

Here, we test the performance of PCNs on standard image classification tasks. We compare PC against BP, using both *Squared Error* (SE) and *Cross Entropy* (CE) loss, by adapting the energy function as described in [Pinchetti et al., 2022]. For the experiments involving the MNIST and FashionMNIST datasets, we use feedforward models with 3 hidden layers of 128 hidden neurons each, while for CIFAR10/100 and Tiny ImageNET, we compare VGG-like models. We performed a large hyperparameter search over learning rates, optimizers, activation functions, and algorithm-specific parameters. All the details needed to reproduce the experiments, as well as a discussion about 'lessons learned' during such a large search, are provided in the Appendix B. The best results, averaged over 5 seeds are reported in Tab. 2.

Discussion. The results show that the best performing algorithms, at least on the most complex tasks, are the ones where the target is nudged towards the real label, that are PN and NN. This is in line with previous findings in the Eqprop literature [Scellier et al., 2024]. The recently proposed iPC, on the other hand, performs well on small architectures, as it is the best performing one on MNIST and FashionMNIST, but its performance worsten when it comes to train on large architectures. More broadly, the performance are comparable, albeit slightly lower, than those of backprop, especially on the largest model. An interesting observation, is that all the best results have been achieved using a VGG5, that has always outperformed the deeper VGG7. Future work should investigate the reason of such a phenomenon, as scaling up to more complex datasets will require the use of much deeper architectures, such as ResNets [He et al., 2016]. In Section 5, we analyze possible causes, as well as comparing the wall-clock time of the different algorithms.

Table 3: MSE loss for image reconstruction of BP, PC, and iPC on different datasets.

MSE ($\times 10^{-3}$)	PC	iPC	BP
MNIST	$9.25^{\pm 0.00}$	$9.09^{\pm 0.00}$	$\mathbf{9.08^{\pm 0.00}}$
FashionMNIST	$10.56^{\pm 0.01}$	$10.11^{\pm 0.01}$	$\mathbf{10.04^{\pm 0.00}}$

MSE ($\times 10^{-3}$)	PC	iPC	BP
CIFAR-10	$6.67^{\pm 0.10}$	$\mathbf{5.50^{\pm 0.01}}$	$6.17^{\pm 0.46}$
CELEB-A	$2.35^{\pm 0.12}$	$\mathbf{1.3^{\pm 0.12}}$	$3.34^{\pm 0.30}$

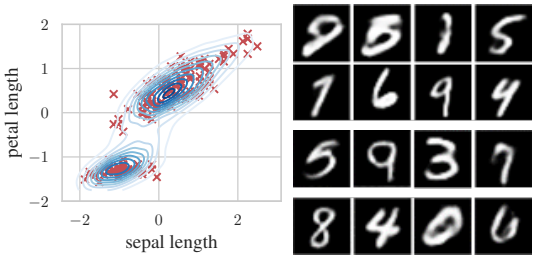


Figure 3: Generative samples obtained by MCPC. (left) Contour plot of learned generative distribution compared to Iris data samples (x). (right) Samples obtained for an MNIST trained PCN. In order: unconditional generation, conditional generation (odd), conditional generation (even).

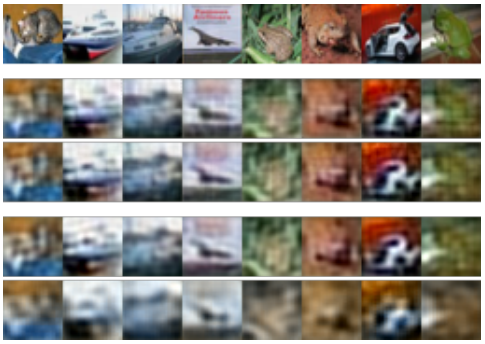


Figure 4: CIFAR10 image reconstruction via autoencoding convolutional networks. In order: original, PC, iPC, BP, BP (half of the parameters).

4.2 Generative Mode

We test the performance of PCNs on image generation tasks. We perform three different kinds of experiments: (1) generation from a posterior distribution; (2) generation via sampling from the learned joint distribution; and (3) associative memory retrieval. In the first case, we provide a test image y to a trained model, run inference to compute a compressed representation $\bar{x}$ (stored in the latent vector h_0 at convergence), and produce a reconstructed $\bar{y} = h_L$ by performing a forward pass with $h_0 = \bar{x}$). The model we consider are three layer networks. As this is an autoencoding task, we compare against autoencoders with three layer encoder/decoder structure (so, six layers in total). In the case of MNIST and FashionMNIST, we use feedforward layers, in the case of CIFAR10 and CelebA, deconvolutional (and convolutional for the encoder) ones. The results in Tab. 3 and Fig. 4 report comparable performance, with a small advantage for PC compared to BP on the more complex convolutional tasks. In this case, iPC is the best performing algorithm, probably due to the small size of the considered models. Furthermore, note that the BP architectures have double the amount of parameters, being the PC networks decoder only. If we halve the number of features in each layer of the autoencoder architecture (while keeping the bottleneck dimension unchanged), we get significantly reduced performance for BP (Fig. 4, bottom), achieving a loss of $10.66^{\pm 0.94} \times 10^{-3}$ on CIFAR10. Details about these and the following experiments are provided in Appendix C.

For the second category of experiments, we tested the capability of PCNs to learn, and sample from, a complex probability distribution. MCPC extends PC by incorporating Gaussian noise to the activity updates of each neuron. This change enables a PCN to learn and generate samples analogous to a variational autoencoders (VAE). This change shifts the inference of PCNs from a variational approximation to Monte Carlo sampling of the posterior using Langevin dynamics. Data samples can be generated from the learned joint $P_\theta(h)$ by leaving all states h_l free and performing noisy inference updates. Figure 3 illustrates MCPC's ability to learn non-linear multimodal distributions using the iris

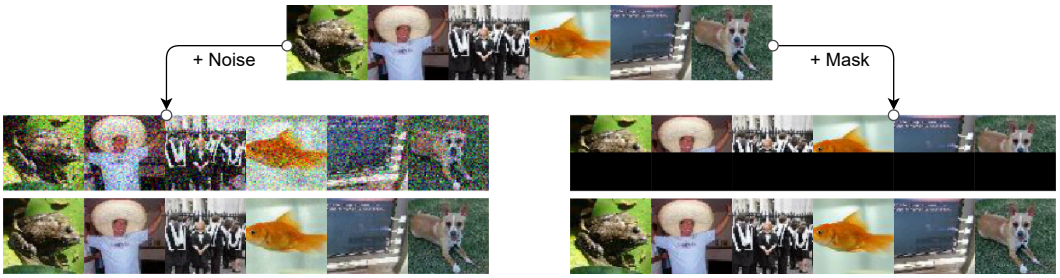
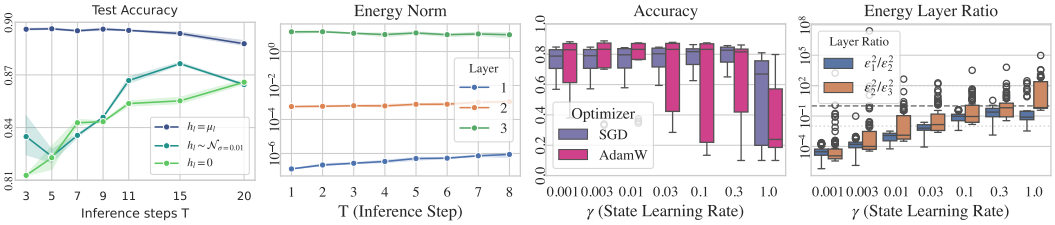


Figure 5: Memory recalled images. Top: Original images. Left: Noisy input (guassian noise, $\sigma = 0.2$) and reconstruction. Right: Masked input (bottom half removed) and reconstruction.

Table 4: MSE ($\times 10^{-4}$) of associative memory tasks. Columns indicate the number of hidden neurons while rows shows the training images to memorize. Results over 5 seeds.

Noise	512	1024	2048
50	$6.06^{\pm 0.11}$	$5.91^{\pm 0.14}$	$5.95^{\pm 0.06}$
100	$6.99^{\pm 0.19}$	$6.76^{\pm 0.23}$	$6.16^{\pm 0.07}$
250	$9.95^{\pm 0.05}$	$10.14^{\pm 0.06}$	$8.90^{\pm 0.06}$

Mask	512	1024	2048
50	$0.06^{\pm 0.02}$	$0.01^{\pm 0.00}$	$0.00^{\pm 0.00}$
100	$1.15^{\pm 0.78}$	$1.01^{\pm 0.79}$	$0.11^{\pm 0.03}$
250	$39.1^{\pm 10.8}$	$3.74^{\pm 0.73}$	$0.22^{\pm 0.06}$



(a) (b) (c) (d)

Figure 6: (a) Highest test accuracy reported for different initialization methods and iteration steps T used during training. (b) Energies per layer during inference of the best performing model. (c) Decay in accuracy when increasing the learning rate of the states γ, tested using both SGD and Adam. (d) Imbalance between energies in the layers. All figures are obtained using a three layer model trained on FashionMNIST.

dataset [Pedregosa et al., 2011] and shows generative samples for MNIST. When comparing MCPC to a VAE on MNIST, both models produced samples of similar quality despite the VAE having twice the number of parameters. MCPC achieved a lower FID score (MCPC: $2.53^{\pm 0.17}$ vs. VAE: $4.19^{\pm 0.38}$), whereas the VAE attained a higher inception score (VAE: $7.91^{\pm 0.03}$ vs. MCPC: $7.13^{\pm 0.10}$). In the associative memory (AM) experiments, we test how well the model is able to reconstruct an image already present in the training set, after it is provided with an incomplete or corrupted version of it, as done in a previous work [Salvatori et al., 2021]. Fig. 5 show the results obtained by a PCN with 2 hidden layers of 512 neurons given noise or mask corrupted images. In Tab. 4, we study the memory capacity as the number of hidden layers increases. No visual difference between the recall and original images can be observed for MSE up to 0.005.

Discussion. The results show that PC is able to perform generative tasks, as well as and associative memory ones using decoder-only architectures. Via inference, PCNs are able to encode complex probability distributions in their latent state which can be used to perform a variety of different tasks, as we have shown. Thus, compared to artificial neural networks, PCNs are more flexible and require only half the parameters to achieve similar performance. This comes at a higher computational cost due to the number of inference steps to perform. Future work should look into this issue, aiming at reducing the inference time by propagating the information more efficiently through the network.

5 Analysis and metrics

In this section, we report several metrics that we believe are important to understand the current state and challenges of training networks with PC and compare them with standard models trained with gradient-descent and backprop when suitable. More in detail, we discuss how regularly the energy flows into the model, and how stable training is when changing parameters, initializations, and optimizers. A better understanding of such phenomena would allow us to solve the current problems of PCNs and, hence, scale up to the training of larger models on more complex datasets.

5.1 Energy and stability

The first study we perform regards the initialization of the network states h, and how this influences the performance of the model. In the literature, they have been either initialized to be equal to *zero*, randomly initialized via a Gaussian prior [Whittington and Bogacz, 2017], and be initialized via a forward pass. This last technique has been the preferred option in machine learning papers as it sets the errors $\epsilon_{l \neq L} = 0$ at every internal layer of the model. This allows the prediction error to be concentrated in the output layer only, and hence be equivalent to the SE. To provide a comparison among the three methods, we have trained a 3-layer feedforward model on FashionMNIST. The results, plotted in Fig. 6a, show that forward initialization is indeed the better method, although the gap in performance shrinks the more iterations T are performed.

Energy propagation. Concentrating the total error of the model, and hence its energy, to the last layer as done when performing forward initialization, makes it hard for the model to then propagate such an energy back to the first layers. As reported in Fig. 6b, we observe that the energy in the last layer is orders of magnitude larger than the one in the input layer, even after performing several inference steps. However, this behavior raises the question whether better initialization or optimization techniques could result in a more balanced energy distribution and thus better weight updates, as learning in this unbalanced energy regime has been shown problematic for more complex models [Alonso et al., 2024]. An easy way of quickly propagating the energy through the network is to use learning rates equal to 1.0 for the updates of the states. However, both the results reported in Fig. 6c, as well as our large experimental analysis of Section 4 show that the best performance was consistently achieved for state learning rates γ significantly smaller than 1.0. We hypothesize that the current training setup for PCNs favors small state learning rates that are sub-optimal to scale to deeper architectures. Fig. 6d shows the energy ratios for different state learning rates: when $\gamma \ll 1.0$, the ratios of energies between layers are small, $\epsilon_{l+1}^2/\epsilon_l^2 \ll 1$. The energy in the the first hidden layer is on average 6 orders of magnitude below the last layer for $\gamma = 0.01$. While models trained with large γ values achieve better energy propagation, they achieve lower accuracy as shown in Fig 6c. Note that the decay in performance as function of increasing γ is stronger for Adam despite being the overall better optimizer in our experiments. This suggests limitations in the current training techniques and possible direction for future improvements aimed at reducing the energy imbalance between layers. We provide implementation details and results on other datasets in Appendix D.

Training stability. We observed a further link between the weight optimizer and the structure of a PCN that might hinder the scalability of PC, that is, the influence of the hidden dimension on the performance of the model. To better study this, we trained feedforward PCNs with different hidden dimensions, state learning rates γ and optimizers, and reported the results in Fig. 7. The results show that, when using Adam, the width strongly affects the values of the learning rate γ for which the training process is stable. Interestingly, this phenomenon does not appear when using both the SGD optimizer, nor on standard networks trained with backprop. This behavioral difference with BP is unexpected and suggests the need for better optimization strategies for PCNs, as Adam was still the best choice in our experiments, but could be a bottleneck for larger architectures.

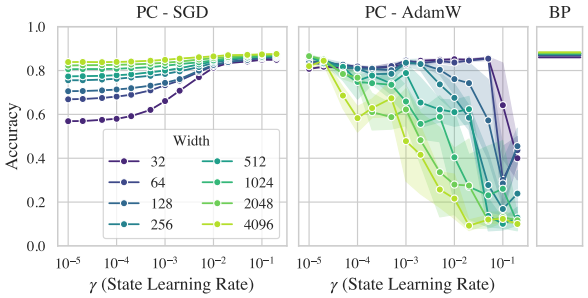


Figure 7: Updating weights with AdamW becomes unstable for wide layers. The optimal state learning rate depends on the width of the layers.

5.2 Properties of predictive coding networks

With PCX, it is straightforward to inspect and analyze several properties of PCNs. Here, we use $\mathcal{F}$ to differentiate between in-distribution (ID) and out-of-distribution (OOD) due to a semantic distribution shift Liu et al. [2020], as well as to compute the likelihood of a datasets Grathwohl et al. [2020]. This can occur when samples are drawn from different, unseen classes, such as FashionMNIST samples under an MNIST setup Hendrycks and Gimpel [2017]. To understand the confidence of the predictions of a PCN, we compare the distribution of probabilities $p(x_i, \hat{y}_i; \theta)$, for ID and OOD samples against these to the distribution of softmax values of the PCN classifier, and compute their negative log-likelihoods (NLL), according to the assumptions stated in Section 2, that is,

$$\mathcal{F} = -\ln p(x, y; \theta) \implies p(x, y; \theta) = e^{-\mathcal{F}}. \tag{4}$$

Our results in Fig. 8(left) demonstrate that a trained PCN classifier can effectively (1) assess OOD samples out-of-the-box, without requiring specific training for that purpose Yang et al. [2021], and (2) produce energy scores for ID and OOD samples that initially correlate with softmax values prior to the optimization of the states variables, h. However, after optimizing the states for T inference steps, the scores for ID and OOD samples become decorrelated, especially for samples with lower softmax values as shown in Fig. 8(center). To corroborate this observation, we also present ROC curves for the most challenging samples, including only the lowest 25% of the scores. As shown in Fig.8(right), the probability (i.e., energy-based) scores provide a more reliable assessment of whether samples are OOD. Experiment details and results on other datasets are provided in in Appendix E.

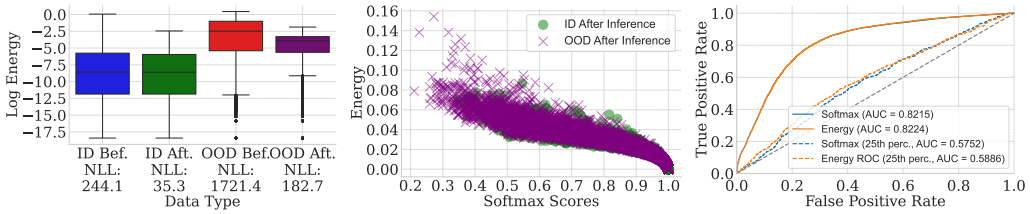


Figure 8: Left: Energy and NLL of ID/OOD data before/after state optmization. Centre: Nonlinearity between energy and softmax after convergence. Right: ROC curve of OOD detection of scores.

6 Related Works

Rao and Ballard's PC. The most related works are those that explore different properties or optimization algorithms of standard PC in the deep learning regime, using formulations inspired by Rao and Ballard's original work [Rao and Ballard, 1999]. Examples are works that study their associative memory capabilities [Salvatori et al., 2021, Yoo and Wood, 2022, Tang et al., 2023, 2024], their ability to train Bayesian networks [Salvatori et al., 2022, 2023b], and theoretical results that explain, or improve, their optimization process [Millidge et al., 2022a,b, Alonso et al., 2022]. Results in this field have allowed to either improve the performance of such models in different tasks, or to discover different tasks and properties that could benefit from the use of PCNs.

Variations of PC. In the literature, there are multiple variations of PC algorithms, which differ from Rao and Ballard's original formulation in the way they update the neural activities. Important examples of such variations are biased competition and divisive input modulation [Spratling, 2008], or the neural generative coding framework [Ororbia and Kifer, 2022]. The latter is already used in multiple reinforcement learning and control tasks [Ororbia and Mali, 2023, Ororbia et al., 2023]. For a review on how different PC algorithms evolved through time, from signal processing to neuroscience, we refer to [Spratling, 2017]; for a more recent review specific to machine learning applications, to [Salvatori et al., 2023a]. It is also worth mentioning the original literature on PC in the neurosciences, that does not intersect with ours as it is not related to machine learning, but has evolved from Rao and Ballard's work into a general theory that models information processing in the brain using variational inference, called the *free energy principle* [Friston, 2005, Friston and Kiebel, 2009, Friston, 2010].

Neuroscience-inspired approaches. Another line of related works is that of neuroscience methods applied to machine learning, like equilibrium propagation [Scellier and Bengio, 2017], which is the most similar to PC [Laborieux and Zenke, 2022, Millidge et al., 2022a]. Other methods able to train models of similar sizes are target propagation [Bengio, 2014, Ernoult et al., 2022, Millidge et al., 2022b] and SoftHebb [Moraitis et al., 2022, Journé et al., 2022]. The first two communities, that of targetprop and eqprop, consistently use similar architectures in different research papers to test the performance of their methods. In our benchmarking effort, some of the architectures proposed are the same ones they use, to favor a more direct comparison. There are also methods that differ more from PC, such as forward-only methods [Kohan et al., 2023, Nøkland, 2016, Hinton, 2022], methods that back-propagate the errors using a designated set of weights [Lillicrap et al., 2014, Launay et al., 2020], and other Hebbian methods [Moraitis et al., 2022, Journé et al., 2022].

7 Discussion

The main contribution of this work is the introduction and open-source release that can be used to perform deep learning tasks using PCNs. Its efficiency relies on JAX's Just-In-Time compilation and carefully structured primitives built to take advantage of it. A second advantage of our library is its intuitive setup, tailored to users already familiar with other deep learning frameworks such as PyTorch. Together with the large number of tutorials we release will make it easy for new users to train networks using PC.

We have also performed a large-scale comparison study on image classification and image generation tasks, unifying under the same computational framework multiple optimization algorithms for PCNs present in the literature. In addition, we have tried multiple, popular, optimizers and training techniques, as well as an extremely large choice of hyperparameters. For CIFAR100 only, for example, we have conducted thousands of individual experiments, that have been used to obtain new state of the art results, as well as provide insights on what works and what does not, that will be useful in the future to researchers tackling deep learning problem with PCNs. The code for PCX is available at *https://github.com/liukidar/pcax*.

8 Societal Impact

This work adheres to the established ethical standards prevalent in the field of AI and machine learning. In the short term, it does not introduce specific ethical concerns, as the models and technology we study are still in early-stage development, and do not perform as well as classic methods. However, we acknowledge the implications and responsibilities that accompany advancements in these technologies. We are committed to ongoing evaluation and responsible stewardship of our contributions to ensure they align with the ethical landscape of this dynamic field.

References

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.

Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv:1407.7906*, 2014.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL `http://github.com/google-deepmind`.

Arthur Dempster, Nan Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Maxence M Ernoult, Fabrice Normandin, Abhinav Moudgil, Sean Spinney, Eugene Belilovsky, Irina Rish, Blake Richards, and Yoshua Bengio. Towards scaling difference target propagation by learning backprop targets. In *International Conference on Machine Learning*, pages 5968–5987. PMLR, 2022.

Simon Frieder and Thomas Lukasiewicz. (non-) convergence results for predictive coding networks. In *International Conference on Machine Learning*, pages 6793–6810. PMLR, 2022.

K. Friston, J. Mattout, N. Trujillo-Barreto, J. Ashburner, and W. Penny. Variational free energy and the Laplace approximation. *Neuroimage*, 2007.

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

Karl Friston. The free-energy principle: A unified brain theory? *Nature Reviews Neuroscience*, 11 (2):127–138, 2010.

Karl Friston and Stefan Kiebel. Predictive coding under the free-energy principle. *Philosophical transactions of the Royal Society B: Biological sciences*, 364(1521):1211–1221, 2009.

Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

Adrien Journé, Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Hebbian deep learning without feedback. *arXiv preprint arXiv:2209.11883*, 2022.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Adam Kohan, Edward A Rietman, and Hava T Siegelmann. Signal propagation: The framework for learning and inference in a forward pass. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

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.

Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. *Advances in neural information processing systems*, 33:9346–9360, 2020.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020.

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. *arXiv preprint arXiv:2206.02629*, 2022a.

Beren Millidge, Yuhang Song, Tommaso Salvatori, Thomas Lukasiewicz, and Rafal Bogacz. A theoretical framework for inference and learning in predictive coding networks. *arXiv preprint arXiv:2207.12316*, 2022b.

Timoleon Moraitis, Dmitry Toichkin, Adrien Journé, Yansong Chua, and Qinghai Guo. Softhebb: Bayesian inference in unsupervised hebbian soft winner-take-all networks. *Neuromorphic Computing and Engineering*, 2(4):044017, 2022.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

Gaspard Oliviers, Rafal Bogacz, and Alexander Meulemans. Learning probability distributions of sensory inputs with monte carlo predictive coding. *bioRxiv*, pages 2024–02, 2024.

Alexander Ororbia and Daniel Kifer. The neural coding framework for learning generative models. *Nature communications*, 13(1):2064, 2022.

Alexander Ororbia and Ankur Mali. Active predictive coding: Brain-inspired reinforcement learning for sparse reward robotic control problems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3015–3021. IEEE, 2023.

Alexander G Ororbia, Ankur Mali, Daniel Kifer, and C Lee Giles. Backpropagation-free deep learning with recursive local representation alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9327–9335, 2023.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Luca Pinchetti, Tommaso Salvatori, Beren Millidge, Yuhang Song, Yordan Yordanov, and Thomas Lukasiewicz. Predictive coding beyond Gaussian distributions. *36th Conference on Neural Information Processing Systems*, 2022.

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

Tommaso Salvatori, Yuhang Song, Yujian Hong, Lei Sha, Simon Frieder, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Tommaso Salvatori, Luca Pinchetti, Beren Millidge, Yuhang Song, Tianyi Bao, Rafal Bogacz, and Thomas Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. *arXiv:2201.13180*, 2022.

Tommaso Salvatori, Ankur Mali, Christopher L Buckley, Thomas Lukasiewicz, Rajesh PN Rao, Karl Friston, and Alexander Ororbia. Brain-inspired computational intelligence via predictive coding. *arXiv preprint arXiv:2308.07870*, 2023a.

Tommaso Salvatori, Luca Pinchetti, Amine M'Charrak, Beren Millidge, and Thomas Lukasiewicz. Causal inference via predictive coding. *arXiv preprint arXiv:2306.15479*, 2023b.

Tommaso Salvatori, Yuhang Song, Beren Millidge, Zhenghua Xu, Lei Sha, Cornelius Emde, Rafal Bogacz, and Thomas Lukasiewicz. A stable, fast, and fully automatic learning algorithm for predictive coding networks. *International Conference on Learning Representations 2024*, 2024.

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, Maxence Ernoult, Jack Kendall, and Suhas Kumar. Energy-based learning algorithms for analog computing: a comparative study. *Advances in Neural Information Processing Systems*, 36, 2024.

Yuhang Song. Prospective-configuration. `https://github.com/YuhangSong/Prospective-Configuration`, 2024.

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*, pages 1–11, 2024.

Michael W Spratling. Reconciling predictive coding and biased competition models of cortical function. *Frontiers in Computational Neuroscience*, 2:4, 2008.

Michael W Spratling. A review of predictive coding algorithms. *Brain and Cognition*, 112:92–97, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 2014.

Mufeng Tang, Tommaso Salvatori, Beren Millidge, Yuhang Song, Thomas Lukasiewicz, and Rafal Bogacz. Recurrent predictive coding models for associative memory employing covariance learning. *PLOS Computational Biology*, 19(4):e1010719, 2023.

Mufeng Tang, Helen Barron, and Rafal Bogacz. Sequential memory with temporal predictive coding. *Advances in Neural Information Processing Systems*, 36, 2024.

James C. R. Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*, 29(5), 2017.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

Jinsoo Yoo and Frank Wood. Bayespcn: A continually learnable predictive coding associative memory. *Advances in Neural Information Processing Systems*, 35:29903–29914, 2022.

Umais Zahid, Qinghai Guo, and Zafeirios Fountas. Sample as you infer: Predictive coding with langevin dynamics. *arXiv preprint arXiv:2311.13664*, 2023.

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: [N/A]

   Guidelines:

   - The answer NA 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 NA 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: Section 6

   Guidelines:

   - The answer NA 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: [N/A]

Guidelines:

- The answer NA 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: [N/A]

   Guidelines:

   - The answer NA 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: (In case of acceptance)

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, we have dedicated sections in the supplementary material

   Guidelines:

   - The answer NA 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: [N/A]

   Guidelines:

   - The answer NA 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: Section 6

   Guidelines:

   - The answer NA 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: [N/A]

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: [N/A]

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer NA 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: [N/A]

    Justification: All the datasets we use are extremely popular in the literature

    Guidelines:

    - The answer NA 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: [N/A]

Justification: [N/A]

Guidelines:

- The answer NA 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: [N/A]

Guidelines:

- The answer NA 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: [N/A]

Guidelines:

- The answer NA 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.