Predictive Coding Graphs are a Superset of Feedforward Neural Networks

Björn van Zwol
Department of Information & Computing Sciences, Utrecht University
Utrecht, The Netherlands
bjornvanzwol@gmail.com

Abstract

Predictive coding graphs (PCGs) are a recently introduced generalization to predictive coding networks, a neuroscience-inspired probabilistic latent variable model. Here, we prove how PCGs define a mathematical superset of feedforward artificial neural networks (multilayer perceptrons). This positions PCNs more strongly within contemporary machine learning (ML), and reinforces earlier proposals to study the use of non-hierarchical neural networks for ML tasks, and more generally the notion of topology in neural networks.

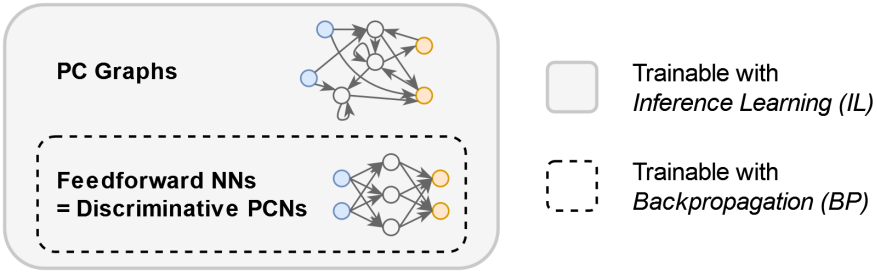

Figure 1: PCGs trained with IL generalize the structure of FNNs to arbitrary graphs, including loops and non-hierarchical structures, which are not trainable using BP.

1 Introduction

Predictive coding networks (PCNs), based on the neuroscientific framework predictive coding, have recently gained attention in machine learning (ML) [23, 15] for their increased biological plausibility compared to backpropagation (BP) [14, 4, 21], their parallelizability [18] and potential for probabilistic/generative modeling [8, 10, 26]. These networks can also be extended to arbitrary topologies, called predictive coding graphs (PCGs) [16].

When applied to supervised learning, standard (hierarchical) PCNs have the same outputs as traditional feedforward neural networks (FNNs, or multilayer perceptrons) during testing (i.e. inference) [24, 2]. This enables direct comparison of BP with PCN's training phase, also called inference learning (IL) – which has been the focus of most recent work on PCNs in ML [20, 13, 17, 6]. In this work we revisit the testing (inference) phase, and formally prove that PCGs are a superset of feedforward neural networks. This follows from the combination of two insights.

The first is that discriminative PCNs are *equivalent* to FNNs during testing, for which we provide a simple proof. This is a stronger variant of related statements in the literature which show how

1^{st} Workshop on NeuroAI @ 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

PCNs *converge to* an FNNs' computations [24, 2]. This subtle but important reframing enables the formal statement that the *universal approximation theorem* (UAT) [1] holds also for PCNs. The UAT is a foundational result which historically provided strong theoretical justification for using FNNs [12]. Its applicability to PCNs however, although generally believed in the PC community, lacked a principled formal proof. To the author's best knowledge, this is provided here for the first time, by virtue of the PCN-FNN testing equivalence.

Our second insight is that PCGs define a mathematical superset of PCNs, which we also prove formally. In ref. [16] it was well-illustrated how hierarchical structures could be obtained with PCGs by masking weights. However, precisely how such masked PCGs relate to PCNs and FNNs was unclear (i.e. their cost functions and dynamics), since a detailed formal analysis was lacking. Here, we prove that a PCG with a particular choice of weight matrix is exactly equivalent to a PCN, both in structure and dynamics.

Combining these two insights leads to the non-trivial conclusion that PCGs should be understood as a structural superset of FNNs that includes FNNs as a special case, as visualized in fig. 1. We believe this result substantially clarifies the relation between PC and traditional neural networks, as recently argued in [23]. Moreover, understanding PCGs in this way is potentially very interesting given the topological nature of important advances in ML in the past, such as residual/skip connections [5, 25, 9]. Thus, we underscore the point made by [16]: PCGs are a promising framework for studying the importance of network topology in ML tasks. Finally, our work also highlights the value of mathematical studies of PCNs, complementary to experimental approaches more commonly found in the literature.

2 Results

We first prove how PCNs are FNNs during testing, followed by how PCNs are subsets of PCGs. The problem setup is as follows: we are given a dataset of N labeled samples $\{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$ split into a training set and a test set, where $\boldsymbol{x}^{(n)} \in \mathbb{R}^{n_x}$ is a datapoint with dimension n_x and $\boldsymbol{y}^{(n)} \in \mathbb{R}^{n_y}$ its corresponding label with dimension n_y.

2.1 PCNs are FNNs during testing

We provide definitions and state the first result. We separate a neural network's *activity rule* (i.e. changing activity of nodes, operating during training and testing) and *learning rule* (changing weights, operating only during training) [7].

Definition 1. *A FNN is defined by a set of nodes $a_i^\ell \in \mathbb{R}$ in layers with $n_\ell \in \mathbb{N}^+$ nodes, with $0 \leq \ell \leq L$, $1 \leq i(\ell) \leq n_\ell$ and $n_0 = n_x$.*[2] *Its* activity rule *is:*

$$0 < \ell \leq L, \forall i: \quad a_i^\ell = f\Big(\sum_j w_{ij}^{\ell-1} a_j^{\ell-1}\Big), \tag{1}$$

with $\forall i$, $a_i^0 = x_i$, and where $w_{ij}^\ell \in \mathbb{R}$, $0 \leq \ell < L$ are the weights, and f is a non-linear, element-wise activation function.

Note that for FNNs, the learning rule is usually defined separate from the network definition, i.e. the canonical training algorithm backpropagation (BP) defines the learning rule: $\Delta w_{ij}^\ell \propto \partial \mathcal{L}/\partial w_{ij}^\ell$ where $\mathcal{L}$ is some loss function. For PCNs, however, the learning rule is typically understood to be part of the network definition itself.

Definition 2. *A PCN is defined by a set of nodes $a_i^\ell \in \mathbb{R}$ in layers with $n_\ell \in \mathbb{N}^+$ nodes, with $0 \leq \ell \leq L$, $1 \leq i \leq n_\ell$ and $n_0 = n_x$. Its* energy *is $E_N = \sum_{\ell=1}^{L} \sum_{i=1}^{n_\ell} (\epsilon_i^\ell)^2$, with $\epsilon_i^\ell = a_i^\ell - \mu_i^\ell$ and $\mu_i^\ell = f\Big(\sum_{j=1}^{n_{\ell-1}} w_{ij}^{\ell-1} a_j^{\ell-1}\Big)$, where $w_{ij}^\ell \in \mathbb{R}$, $0 \leq \ell < L$ are the weights, and f is a non-linear*

[2] The index i for FNNs/PCNs is always defined with respect to layer ℓ, but we leave out this dependence henceforth for simplicity.

element-wise activation function. Its activity rule *is (using hats for minimized values):*

$$0 < \ell < L, \forall i: \quad \hat{a}_i^\ell = \underset{a_i^\ell}{\operatorname{argmin}} E_N \qquad \text{(training)} \tag{2}$$

$$0 < \ell \leq L, \forall i: \quad \hat{a}_i^\ell = \underset{a_i^\ell}{\operatorname{argmin}} E_N \qquad \text{(testing)}, \tag{3}$$

with $\forall i,\ a_i^0 = x_i$*, and during training* $\forall i, a_i^L = y_i$*. The* learning rule *is:*

$$0 \leq \ell < L, \forall i, j: \quad \hat{w}_{ij}^\ell = \underset{w_{ij}^\ell}{\operatorname{argmin}} E_N \quad \text{(training)}, \tag{4}$$

with $\forall i : a_i^0 = x_i,\ a_i^L = y_i$.

Theorem 1. *During testing, a PCN is equivalent to an FNN.*

We prove this in appendix A.1, and provide an intuition here. Testing means that only the activity rule is relevant, and that $\forall i : a_i^0 = x_i$. Equivalence between an FNN and PCN then means that (1) is equivalent to (3):

$$\ell > 0, \forall i: \quad \hat{a}_i^\ell = \underset{a_i^\ell}{\operatorname{argmin}} E_N \iff a_i^\ell = f\Big(\sum_j w_{ij}^{\ell-1} a_j^{\ell-1}\Big). \tag{5}$$

To prove this, start from the left hand side, and find the minimum of E_N by taking derivatives w.r.t. a_i^ℓ, and setting to zero. This yields the following system of equations (cf. appendix B):

$$\frac{\partial E_N}{\partial a_i^\ell} = \begin{cases} \epsilon_i^\ell - \sum_{j=1}^{n_\ell} w_{ji}^\ell \epsilon_j^{\ell+1} f'\Big(\sum_m w_{jm}^\ell a_m^\ell\Big) = 0 & \text{if} \quad 1 \leq \ell < L \qquad (6) \\ \epsilon_i^L = 0 & \text{if} \quad \ell = L \qquad (7) \end{cases}$$

This may be solved by seeing that since $\epsilon_i^L = 0$ by (7), $\epsilon_i^{L-1} = 0$ by (6) – an argument which can be continued for each layer until $\epsilon_i^1 = 0$. Then, by definition of ϵ_i^ℓ, one has the right hand side of 5. This can be formalized by backwards induction.

Our proof is simpler than similar proofs in the literature that we are aware of [19, 17, 2], since we employ a more general definition of the activity rule. Instead of (2), (3), these works define the activity rule of PCNs as $\Delta a_i \propto \partial E / \partial a_i$, which is shown to converge to (1). This however assumes gradient-based dynamics, which we argue is unnecessary using the more principled definition of IL as Expectation Maximization (as discussed e.g. in [23]). In public PCN implementations [22], (1) was already used during testing, a practice we have now given a more principled justification.

A corollary of this proof is that since FNNs are universal function approximators [1, 12], so are PCNs. This fact, although probably widely believed in the PC community, did not yet have a rigorous justification to the author's best knowledge. We also remark that by changing the definition of a_i^ℓ in the FNN, and correspondingly changing μ_i^ℓ in the PCN, to e.g. a convolutional prediction [23] or skip connections, the above result may be trivially extended to any hierarchical structure, i.e. not just MLPs.

2.2 PCNs are subsets of PCGs

We now define a PCG, a PCN generalization introduced by [16]. This work nicely illustrated how different network topologies, such as hierarchical networks, could be obtained by masking the PCG weight matrix. However, it was not discussed how the resulting energies and dynamics (activity and learning rules) relate to PCNs. Here, we provide a rigorous proof that a PCG with a certain choice of weight matrix is equivalent to a PCN. This enables the statement that PCGs are supersets of PCNs (as was mentioned in [23], but not yet proven). For clarity, we use tildes and Greek indices for PCG nodes and weights, and Latin indices for the PCN.

Definition 3. *A PCG is defined by a set of nodes* $a_\alpha \in \mathbb{R}$*,* $\alpha = 1, ..., N$*, and an* energy $E_G = \sum_\alpha \epsilon_\alpha^2$*, with* $\epsilon_\alpha = a_\alpha - \mu_\alpha$ *and* $\mu_\alpha = f\Big(\sum_{\beta=1}^N w_{\alpha\beta} a_\beta\Big)$*, where* $w_{\alpha\beta}^\ell \in \mathbb{R}$ *are the weights, and* f *is a non-linear element-wise activation function. Its* activity rule *is:*

$$n_x < \alpha \leq N - n_y: \quad \hat{a}_\alpha = \underset{a_\alpha}{\operatorname{argmin}} E_G, \qquad \text{(training)} \tag{8}$$

$$n_x < \alpha \leq N: \quad \hat{a}_\alpha = \underset{a_\alpha}{\operatorname{argmin}} E_G, \qquad \text{(testing)} \tag{9}$$

with $a_\alpha = x_\alpha$ for $0 < \alpha \leq n_x$, and during training $a_\alpha = y_i$ for $N - n_y < \alpha \leq N$, $1 \leq i \leq n_y$. The learning rule *is:*

$$\forall \alpha, \beta : \quad \hat{w}_{\alpha\beta} = \underset{w_{\alpha\beta}}{\operatorname{argmin}} E_G \quad \text{(training)}, \tag{10}$$

with $a_\alpha = x_\alpha$ for $0 < \alpha \leq n_x$ and $a_\alpha = y_i$ for $N - n_y < \alpha \leq N$ with $1 \leq i \leq n_y$.

Theorem 2. *The PCG is a superset of the PCN through its weight matrix $\widetilde{\boldsymbol{w}}$. With layers defined by $\{n_\ell\}_{\ell=0}^{L}$, with $n_\ell \in \mathbb{N}_+$, $\sum_\ell n_\ell = N$, partitioning nodes $\widetilde{\boldsymbol{a}}$ into layers $\boldsymbol{a}^\ell \in \mathbb{R}^{n_\ell}$; and weights $\widetilde{\boldsymbol{w}}$ into block matrices $\widetilde{\boldsymbol{w}}^{\ell k} \in \mathbb{R}^{n_\ell \times n_k}$, the choice $\widetilde{\boldsymbol{w}}^{\ell k} = \widetilde{\boldsymbol{w}}^{\ell k}\delta_{k\ell-1} \equiv \boldsymbol{w}^{\ell-1}$ (with $\boldsymbol{w}^\ell$ the PCN weight matrix) implies that their objective functions are equivalent:*

$$E_G = E_N + C \tag{11}$$

where C is a constant. Moreover, their activity rules are equivalent, both during training and testing:

$$\begin{aligned} \operatorname{argmin}_{a_i^\ell} E_N &= \operatorname{argmin}_{a_\alpha} E_G \quad \text{(training)}, \\ \operatorname{argmin}_{a_i^\ell} E_N &= \operatorname{argmin}_{a_\alpha} E_G \quad \text{(testing)}, \end{aligned} \tag{12}$$

and their learning rules are equivalent:

$$\operatorname{argmin}_{w_{ij}^\ell} E_N = \operatorname{argmin}_{w_{\alpha\beta}} E_G \quad \text{(training)}. \tag{13}$$

Other choices of non-zero block matrices leads to skip connections, backward (skip) connections and lateral connections.

The proof is given in A.2. It involves defining the exact mapping of nodes and weights, for which we again provide a brief intuition. From $\{n_\ell\}_{\ell=0}^{L}$, one may define a partitioning of node indices as:

$$I = \{\underbrace{1, \ldots, n_0}_{\text{layer } 0}, \underbrace{n_0 + 1, \ldots, n_0 + n_1}_{\text{layer } 1}, \ldots, \underbrace{s_{L-1} + 1, \ldots, s_L}_{\text{layer } L}\}, \tag{14}$$

where $s_\ell = \sum_{k=0}^{\ell} n_k$. With this, the PCG weights $\widetilde{\boldsymbol{w}}$ may be partitioned into block matrices $\widetilde{\boldsymbol{w}}^{\ell k}$, containing weights from layer k to layer ℓ, which yields:

$$\widetilde{\boldsymbol{w}} = \begin{bmatrix} \widetilde{\boldsymbol{w}}^{00} & \widetilde{\boldsymbol{w}}^{01} & \ldots & \widetilde{\boldsymbol{w}}^{0L} \\ \widetilde{\boldsymbol{w}}^{10} & \widetilde{\boldsymbol{w}}^{11} & & \vdots \\ \vdots & & \ddots & \\ \widetilde{\boldsymbol{w}}^{L0} & \ldots & & \widetilde{\boldsymbol{w}}^{LL} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \ldots & & \mathbf{0} \\ \boldsymbol{w}^0 & & & & \vdots \\ & \boldsymbol{w}^1 & & & \\ \vdots & & \ddots & & \\ \mathbf{0} & \ldots & & \boldsymbol{w}^{L-1} & \mathbf{0} \end{bmatrix}. \tag{15}$$

Here, in the last equality we set the hierarchical structure using $\widetilde{\boldsymbol{w}}^{\ell k} = \widetilde{\boldsymbol{w}}^{\ell k}\delta_{k\ell-1} \equiv \boldsymbol{w}^{\ell-1}$, with δ_{ij} the Kronecker delta, identifying the PCN weight matrices $\boldsymbol{w}^\ell$. The proof then proceeds to show how this leads to the same energy (up to a constant), and identical dynamics (activity and learning rules), both during training and testing.

The main corollary of our theorem is that during testing, PCGs are also a superset of FNNs, by virtue of theorem 1. As a consequence, PCGs are also universal function approximators when a hierarchical structure (15) is chosen. Whether PCGs with additional connections can also approximate any function is left for future work.

The superset nature of PCGs is illustrated in fig. 2, complementing fig. 4 in [16]. A partitioned weight matrix is shown, where block matrices are color-mapped to visualizations of connections allowed by PCGs: forward (skip) connections like FNNs, and additionally backward (skip) connections and lateral connections.

3 Discussion

We have proven how PCGs form a mathematical superset of FNNs. By virtue of using IL, a more biologically plausible alternative to BP [4, 21], they generalize FNNs to arbitrary graphs. This clarifies the relation between traditional neural networks and PCNs, the latter not being widely known in the broader ML community (as also recently argued by [23]).

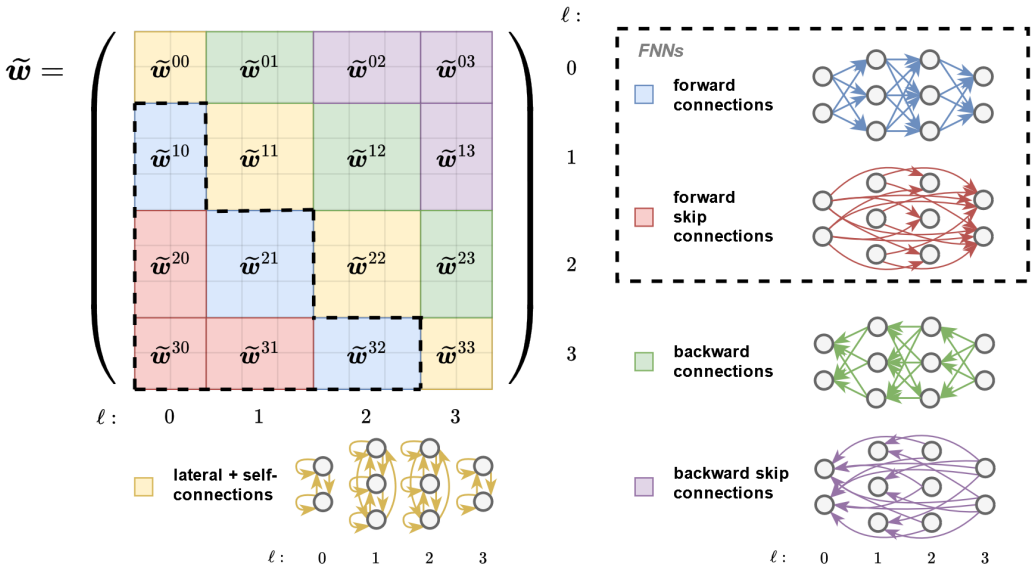

Figure 2: A PCG weight matrix partitioned into block matrices ($N = 10$ nodes, partitioned by 4 layers with 2, 3, 3, and 2 nodes/layer, respectively). In traditional ANNs trained with BP, only feedforward connections (blue and red blocks) are trainable, whereas the full matrix is trainable with IL. This figure extends fig. 4 in [16].

In the process, we showed that PCNs are *equivalent* to FNNs during testing – a subtle change from earlier work showing that PCNs *converge* to FNNs. As such, this provides a rigorous argument for why the universal approximation theorem is applicable to PCNs, a statement which does not appear yet in the PC literature to the author's knowledge.

PCGs allow a large new set of structures untrainable by BP, as was already argued by [16]. Our results rigorously show how these models includes FNNs as a special case. Moreover, fig. 2 clarifies how *skip connections* can be seen as a part of the PCG weight matrix $\widetilde{\boldsymbol{w}}$ (extending fig. 4 in [16]). Such connections are the main innovation in ResNets [5], and are well-known provide great benefits for many ML tasks compared to standard MLPs [25, 9]. Understanding these as a part of $\widetilde{\boldsymbol{w}}$, then, begs the question of whether the remainder of $\widetilde{\boldsymbol{w}}$ – that is, backward (skip) connections, lateral connections and self-connections – does, by analogy, also bring benefits. Future work should investigate this. So far, only all-to-all connected PCGs (the full matrix except self-connections) have been studied empirically [16]: these appear not to perform as well as hierarchical PCNs/FNNs, but they do outperform other all-to-all connected networks for classification on MNIST by a large margin (12-35% better than Boltzmann machines and Hopfield networks for three datasets), which is encouraging.

A practical limitation of current PCG implementations is that using non-feedforward connections comes at a computational cost, because gradient-based inference of nodes is relatively expensive. In FNNs, the time complexity of testing one batch is $\mathcal{O}(LM)$, where $M = \max(\{n^\ell n^{\ell+1}\}_\ell)$ is the number of weights in the largest block matrix in $\widetilde{\boldsymbol{w}}$ [23]. In contrast, in PCGs this is $\mathcal{O}(N^2T)$ where T is the number of inference steps. If sparsity of the PCG weight matrix is leveraged, this reduces to $\mathcal{O}(dNT)$ where d is the number of nonzero weights. For an FNN with a comparable number of weights, one has $d \approx LM$. I.e., testing one batch is a factor NT slower in a PCG. This is not necessarily undesirable, however, since increased testing time could potentially be compensated by other favorable properties of the training algorithm and topology used. For further discussion of practical issues in PCNs and PCGs, we refer to [23].

As also mentioned in [23], we emphasize that the non-feedforward connections in $\widetilde{\boldsymbol{w}}$ introduce a notion of *recurrence* distinct from that in recurrent neural networks (RNNs), cf. fig. 2. RNNs, made specifically to handle sequential data, have recurrence with respect to 'data time', characterized by

weight sharing. In PCGs, by contrast, recurrence is in 'inference time', like in Hopfield networks [7]. This distinction was not yet highlighted in [16].

Finally, our work underscores the value of theoretical work and rigorous mathematical arguments for studying PCNs, which is still limited in the literature – a point also recently made by [3, 23]. By providing theoretical justification, such work can usefully guide and constrain future experimental studies of PCNs.

Acknowledgments and Disclosure of Funding

The author is grateful to Egon L. van den Broek and Ro Jefferson for valuable discussions and feedback on the manuscript. He also thanks Lukas Arts for helpful discussions.

References

[1] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, December 1989.

[2] Simon Frieder and Thomas Lukasiewicz. (Non-)Convergence Results for Predictive Coding Networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6793–6810. PMLR, July 2022.

[3] Simon Frieder, Luca Pinchetti, and Thomas Lukasiewicz. Bad Minima of Predictive Coding Energy Functions. In *The Second Tiny Papers Track at ICLR 2024*, 2024.

[4] Siavash Golkar, Tiberiu Tesileanu, Yanis Bahroun, Anirvan Sengupta, and Dmitri Chklovskii. Constrained Predictive Coding as a Biologically Plausible Model of the Cortical Hierarchy. *Advances in Neural Information Processing Systems*, 35:14155–14169, December 2022.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. pages 770–778, June 2016.

[6] Francesco Innocenti, Ryan Singh, and Christopher Buckley. Understanding Predictive Coding as a Second-Order Trust-Region Method. In *ICML Workshop on Localized Learning (LLW)*, 2023.

[7] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003.

[8] Joseph Marino. Predictive Coding, Variational Autoencoders, and Biological Connections. *Neural Computation*, 34(1):1–44, January 2022.

[9] Emin Orhan and Xaq Pitkow. Skip Connections Eliminate Singularities. February 2018.

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

[11] Luca Pinchetti, Chang Qi, Oleh Lokshyn, Gaspard Olivers, Cornelius Emde, Mufeng Tang, Amine M'Charrak, Simon Frieder, Bayar Menzat, Rafal Bogacz, Thomas Lukasiewicz, and Tommaso Salvatori. Benchmarking Predictive Coding Networks – Made Simple, July 2024. arXiv:2407.01163.

[12] Simon J. D. Prince. *Understanding Deep Learning*. The MIT Press, 2023.

[13] Robert Rosenbaum. On the relationship between predictive coding and backpropagation. *PLOS ONE*, 17(3):e0266102, March 2022.

[14] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[15] Tommaso Salvatori, Ankur Mali, Christopher L. Buckley, Thomas Lukasiewicz, Rajesh P. N. Rao, Karl Friston, and Alexander Ororbia. Brain-Inspired Computational Intelligence via Predictive Coding, August 2023. arXiv:2308.07870. Retrieved from https://arxiv.org/abs/2308.07870.

[16] Tommaso Salvatori, Luca Pinchetti, Beren Millidge, Yuhang Song, Tianyi Bao, Rafal Bogacz, and Thomas Lukasiewicz. Learning on Arbitrary Graph Topologies via Predictive Coding. *Advances in Neural Information Processing Systems*, 35:38232–38244, December 2022.

[17] Tommaso Salvatori, Yuhang Song, Zhenghua Xu, Thomas Lukasiewicz, and Rafal Bogacz. Reverse Differentiation via Predictive Coding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):8150–8158, June 2022.

[18] Tommaso Salvatori, Yuhang Song, Yordan Yordanov, Beren Millidge, Lei Sha, Cornelius Emde, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. A Stable, Fast, and Fully Automatic Learning Algorithm for Predictive Coding Networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[19] Y. Song. *Predictive coding inspires effective alternatives to backpropagation*. Ph.D. Thesis, University of Oxford, 2021.

[20] Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the Brain Do Backpropagation? — Exact Implementation of Backpropagation in Predictive Coding Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 22566–22579, 2020.

[21] 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, January 2024.

[22] Alexander Tschantz and Beren Millidge. infer-actively/pypc, October 2023. Retrieved from https://github.com/infer-actively/pypc.

[23] Björn van Zwol, Ro Jefferson, and Egon L. van den Broek. Predictive Coding Networks and Inference Learning: Tutorial and Survey, July 2024. Preprint.

[24] James C. R. Whittington and Rafal Bogacz. An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. *Neural Computation*, 29(5):1229–1262, May 2017.

[25] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016 (BMVC 2016)*. BMVA Press, 2016.

[26] Umais Zahid, Qinghai Guo, and Zafeirios Fountas. Sample as you Infer: Predictive Coding with Langevin Dynamics. In *Forty-first International Conference on Machine Learning*, 2024.

A Proofs

This appendix provides proofs for the two theorems in the main text.

A.1 PCNs are FNNs during testing

This section proves theorem 1.

Proof. We have to prove (5), which we do using backwards induction. The induction hypothesis is, for some $k < L$:

$$\forall i: \quad a_i^k = f\Big(\sum_j w_{ij}^{k-1} a_j^{k-1}\Big) \iff \epsilon_i^k = 0\,. \tag{16}$$

The base case is $\ell = L$, by (7):

$$\forall i: \quad \epsilon_i^L = 0 \iff a_i^L = f\Big(\sum_j w_{ij}^{L-1} a_j^{L-1}\Big)\,. \tag{17}$$

Then, by (6), for $\ell = k$:

$$\forall i: \quad \epsilon_i^{k-1} - \sum_{j=1}^{n_{k-1}} w_{ji}^{k-1} \epsilon_j^k f'\Big(\sum_m w_{jm}^{k-1} a_m^{k-1}\Big) = 0 \tag{18}$$

Then, by the induction hypothesis (16) $\epsilon_j^k = 0$, so:

$$\forall i: \quad \epsilon_i^{k-1} = 0. \tag{19}$$

So, by induction the hypothesis (16) is proven for all $\ell \leq L$. ■

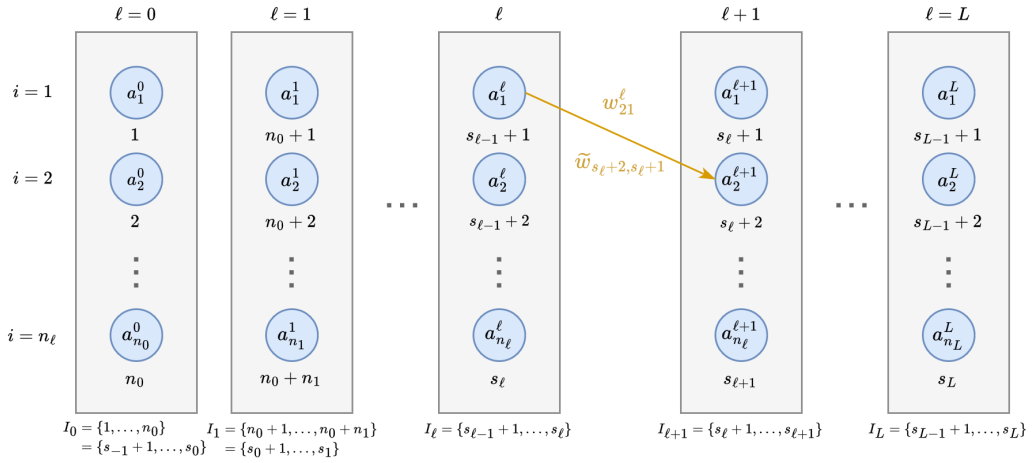

Figure 3: Mapping between PCN indices (blue nodes) and PCG indices (below nodes). Note that layers may have different widths n_ℓ. i denotes PCN index within a layer, and I_ℓ is defined by (22). One weight is shown in orange using PCN indices (top) and PCG indices (bottom), cf. (26).

A.2 PCNs are subsets of PCGs

This section proves theorem 2.

Proof. The PCN energy is:

$$E_N = \sum_{\ell=0}^{L} \sum_{i=1}^{n_\ell} \left[a_i^\ell - f\Big(\sum_{j=1}^{n_{\ell-1}} w_{ij}^{\ell-1} a_j^{\ell-1} \Big) \right]^2, \tag{20}$$

where $\boldsymbol{w}^\ell \in \mathbb{R}^{n_\ell \times n_{\ell+1}}$ are matrices, and $\boldsymbol{a}^0 = \boldsymbol{x}$, $\boldsymbol{a}^L = \boldsymbol{y}$. It is useful to define *the sum of node indices up to layer* ℓ as $s_\ell = \sum_{k=0}^{\ell} n_k$ where we additionally define $s_{-1} = 0$. Then, partition the PCG node indices i as follows (cf. fig. 3):

$$I = \{\underbrace{1,\dots,n_0}_{I_0}, \underbrace{n_0+1,\dots,n_0+n_1}_{I_1}, \dots, \underbrace{s_{L-1}+1,\dots,s_L}_{I_L}\}. \tag{21}$$

We defined I_ℓ as *the node indices in layer* ℓ, i.e.:

$$I_\ell = \{s_{\ell-1}+1, s_{\ell-1}+2, \dots, s_\ell\}. \tag{22}$$

Thus we have, if $\alpha \in I_\ell$, the corresponding PCN index $i(\ell) = \alpha - s_\ell$, meaning we can map PCG nodes to PCN nodes as follows:

$$\widetilde{a}_\alpha = a_i^\ell, \tag{23}$$

such that sums over these nodes may be written as

$$\sum_{\alpha \in I_\ell} = \sum_{i=1}^{n_\ell}. \tag{24}$$

The partition (21) also allows us to partition the weight matrix into block matrices. Defining $\widetilde{\boldsymbol{w}}^{\ell k} \in \mathbb{R}^{n_\ell \times n_k}$ the full weight matrix becomes:

$$\widetilde{\boldsymbol{w}} = \begin{bmatrix} \widetilde{\boldsymbol{w}}^{00} & \widetilde{\boldsymbol{w}}^{01} & \dots & \widetilde{\boldsymbol{w}}^{0L} \\ \widetilde{\boldsymbol{w}}^{10} & \widetilde{\boldsymbol{w}}^{11} & & \vdots \\ \vdots & & \ddots & \\ \widetilde{\boldsymbol{w}}^{L0} & \dots & & \widetilde{\boldsymbol{w}}^{LL} \end{bmatrix}. \tag{25}$$

Each matrix $\widetilde{\boldsymbol{w}}^{\ell k}$ contains weights from layer ℓ to layer k, cf. fig. 2. To obtain a hierarchical structure, one sets $\widetilde{\boldsymbol{w}}^{\ell k} = \widetilde{\boldsymbol{w}}^{\ell k}\delta_{k\ell-1} \equiv \boldsymbol{w}^{\ell-1}$, where δ_{ij} is the Kronecker delta, and we have found a mapping with the PCN weight matrix $\boldsymbol{w}^\ell$. Equivalently, this can be stated as follows: if $\alpha \in I_\ell$, then

$$\widetilde{w}_{\alpha\beta} = \begin{cases} w_{ij}^\ell & \text{if} \quad \beta \in I_{\ell-1} \\ 0 & \text{else} \end{cases} \tag{26}$$

where $i = \alpha - s_\ell$, $j = \beta - s_\ell$ (cf. fig. 3). This results in:

$$\widetilde{\boldsymbol{w}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & & \mathbf{0} \\ \widetilde{\boldsymbol{w}}^{10} & & & & \vdots \\ & \widetilde{\boldsymbol{w}}^{21} & & & \\ \vdots & & \ddots & & \\ \mathbf{0} & \dots & & \widetilde{\boldsymbol{w}}^{LL-1} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & & \mathbf{0} \\ \boldsymbol{w}^0 & & & & \vdots \\ & \boldsymbol{w}^1 & & & \\ \vdots & & \ddots & & \\ \mathbf{0} & \dots & & \boldsymbol{w}^{L-1} & \mathbf{0} \end{bmatrix} \tag{27}$$

Now, the PCG energy is:

$$E_G = \sum_{\alpha=1}^{N} \Big[\widetilde{a}_\alpha - f\Big(\sum_{\beta=1}^{N} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta\Big)\Big]^2 \tag{28}$$

We decompose the outer and inner sum according to (21), which we can write using (22):

$$\sum_{\alpha=1}^{N} = \sum_{\alpha\in I_0} + \sum_{\alpha\in I_1} + \cdots + \sum_{\alpha\in I_L}$$

which yields

$$\begin{aligned} E_G = &\sum_{\alpha\in I_0} \Big[\widetilde{a}_\alpha - f\Big(\sum_{\beta\in I_0} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta + \sum_{\beta\in I_1} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta + \cdots + \sum_{\beta\in I_L} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta\Big)\Big]^2 \\ &+ \sum_{\alpha\in I_1} \Big[\widetilde{a}_\alpha - f\Big(\sum_{\beta\in I_0} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta + \sum_{\beta\in I_1} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta + \cdots + \sum_{\beta\in I_L} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta\Big)\Big]^2 \\ &\vdots \\ &+ \sum_{\alpha\in I_L} \Big[\widetilde{a}_\alpha - f\Big(\sum_{\beta\in I_0} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta + \sum_{\beta\in I_1} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta + \cdots + \sum_{\beta\in I_L} \widetilde{w}_{\alpha\beta}\widetilde{a}_\beta\Big)\Big]^2, \end{aligned} \tag{29}$$

which we have written out to illustrate the decomposition into the block matrices in (25). If we then use the mappings (23), (24), and (26), this yields:

$$\begin{aligned} E_G &= \sum_{i=1}^{n_0} \Big[a_i^0 - f(0)\Big]^2 \\ &\quad + \sum_{i=1}^{n_1} \Big[a_i^1 - f\Big(\sum_{j=1}^{n_0} w_{ij}^0 a_j^0\Big)\Big]^2 + \cdots + \sum_{i=1}^{n_L} \Big[a_i^L - f\Big(\sum_{j=1}^{n_{L-1}} w_{ij}^{L-1} a_j^{L-1}\Big)\Big]^2 \\ &= \sum_{i=1}^{n_0} \Big[a_i^0 - f(0)\Big]^2 + \sum_{\ell=1}^{L}\sum_{i=1}^{n_\ell} \Big[a_i^\ell - f\Big(\sum_{j=1}^{n_{\ell-1}} w_{ij}^{\ell-1} a_j^{\ell-1}\Big)\Big]^2 \\ &= E_N + C \end{aligned} \tag{30}$$

Observe that during both training and testing, the lowest layer is clamped to the data, i.e. one always has $\forall i, a_i^0 = x^0$. Hence, the extra term $C = \sum_i [a_i^0 - f(0)]^2$ in E_G can be considered constant with respect to both the activity and learning rules, since one has (using the mappings (23), (24), and (26)):

$$\begin{aligned} &0 < \ell < L, \quad \operatorname{argmin}_{a_i^\ell} E_N = \operatorname{argmin}_{a_\alpha} E_G \\ &0 \le \ell < L, \quad \operatorname{argmin}_{w_{ij}^\ell} E_N = \operatorname{argmin}_{w_{\alpha\beta}} E_G. \end{aligned} \tag{31}$$

during training, and similarly during testing:

$$0 < \ell \leq L, \quad \text{argmin}_{a_i^\ell} E_N = \text{argmin}_{a_\alpha} E_G. \tag{32}$$

In sum, both during training and testing, the dynamics of a PCG *with weight matrix* (27), is exactly equal to the dynamics of the PCN, with equal energies up to a constant. [3] As a consequence, since other choices of weight matrix leads to other architectures (e.g. all-to-all connectivity) [16], PCGs define a superset of PCNs.[4] ■

B Update Rules

Here we derive the gradient-based updates for the activity rule and the learning rule. To clarify the relation to other works, we do this for the two main conventions found in the literature:

- 'Matrix-Activation', i.e. $\mu_i = f\left(\sum_j w_{ij} a_j\right)$. This is the convention most often considered for FNNs, used in the main text.
- 'Activation-Matrix', i.e. $\mu_i = \sum_j w_{ij} f\left(a_j\right)$. This is often considered in the PC literature, e.g. [20, 16].

We note that for the latter convention, the proofs above remains the same, with one slight exception in the second proof. In the final step, the constant term in E_G becomes $C = \sum_i (a_i^0)^2$, since the non-linearities f are multiplied by zero.

B.1 PCN, matrix-activation

Activations (eq. 6):

$$\begin{aligned}\frac{\partial E}{\partial a_i^\ell} &= \frac{1}{2}\sum_{k,j} 2\epsilon_j^k \Big[\delta^{k\ell}\delta_{ji} - f'\Big(\sum_m w_{jm}^{k-1} a_m^{k-1}\Big)\sum_m w_{jm}^{k-1}\delta^{k-1\ell}\delta_{mi}\Big] \\ &= \epsilon_i^\ell - \sum_j \epsilon_j^{\ell+1} f'\Big(\sum_m w_{jm}^\ell a_m^\ell\Big) w_{ji}^\ell\end{aligned}$$

where δ_{ij} is the Kronecker delta. For the weights:

$$\begin{aligned}\frac{\partial E}{\partial w_{ab}^\ell} &= -\frac{1}{2}\sum_{k,i} 2\epsilon_i^k f'\Big(\sum_j w_{ij}^{k-1} a_j^{k-1}\Big)' \sum_j x_j^{k-1}\delta_{ia}\delta_{jb}\delta^{k-1\ell} \\ &= -\epsilon_a^{\ell+1} f'\Big(\sum_j w_{aj}^\ell a_j^\ell\Big) a_b^\ell.\end{aligned}$$

B.2 PCN, activation-matrix

For completeness, we show the same derivation for the alternative convention.

$$\begin{aligned}\frac{\partial E}{\partial a_i^\ell} &= \frac{1}{2}\sum_{k,j} 2\epsilon_j^k \Big[\delta^{k\ell}\delta_{ji} - \sum_m w_{jm}^{k-1} f'(a_m^{k-1})\delta^{k-1\ell}\delta_{mi}\Big] \\ &= \epsilon_i^\ell - f'(a_i^\ell)\sum_j \epsilon_j^{\ell+1} w_{ji}^\ell\end{aligned}$$

And for the weights:

$$\begin{aligned}\frac{\partial E}{\partial w_{ab}^\ell} &= -\frac{1}{2}\sum_{k,i} 2\epsilon_i^k \sum_j \delta_{ia}\delta_{jb}\delta^{k-1\ell} f(a_j^{k-1}) \\ &= -\epsilon_a^{\ell+1} f(a_b^\ell).\end{aligned}$$

[3] The mapping in the last layer may be checked by observing that since $n_L = n_y$ by assumption, one has $y_i = a_i^L = a_\alpha$ where $1 \leq i < n_y$ and $N - n_y < \alpha \leq N \iff \sum_{k=0}^{L-1} n_k < \alpha \leq \sum_{k=0}^{L} n_k \iff \alpha \in I_L$.

[4] As a practical note, we mention that in implementations weight updates are typically calculated for all i, j automatically using matrix multiplication. This means one should ensure weights are appropriately set to zero according to (27) to keep the desired structure.

C Initialization

We mention one final point relevant to using PCGs in practice. As mentioned, PCNs perform iterative gradient-based minimization of E_N with respect to 'hidden' (non-clamped) activations in layers $0 < \ell < L$ during training. Unlike in FNNs, the result of inference (and hence, model performance) depends on *initialization* of these nodes. Typically in the PC literature, this is done using a 'feedforward pass' through the network, which typically appears to give better performance than random or zero initialization [11]. Writing activations at inference iteration t as $a_i^\ell = a_i^\ell(t)$, the feedforward initialization is defined by:

$$\begin{aligned}
\ell = 0, \forall i: \quad & a_i^0(0) = x_i \\
\ell = L, \forall i: \quad & a_i^L(0) = y_i \\
0 < \ell < L, \forall i: \quad & a_i^\ell(0) = \mu_i^\ell = f\Big(\sum_{j=1}^{n_{\ell-1}} w_{ij}^{\ell-1} a_i^{\ell-1}(0)\Big)
\end{aligned}$$

For our work, this is relevant insofar as the scheme crucially depends on having a feedforward structure: for a PCG with both forward and non-forward connections, an analogous scheme is not obvious. This means that if one desires to get the results quoted in the literature for PCNs but using a PCG, this scheme has to be separately implemented (cf. e.g. the library implemented in [23]).