Predictive Coding: Towards a Future of Deep Learning beyond Backpropagation?

Beren Millidge[*,1] , **Tommaso Salvatori**[*,2] , **Yuhang Song**[†,2,1]
Rafal Bogacz[1] , **Thomas Lukasiewicz**[3,2]
[1] MRC Brain Network Dynamics Unit, University of Oxford, UK
[2] Department of Computer Science, University of Oxford, UK
[3] Institute of Logic and Computation, TU Wien, Austria
{beren.millidge, yuhang.song, rafal.bogacz}@ndcn.ox.ac.uk
{tommaso.salvatori, thomas.lukasiewicz}@cs.ox.ac.uk

Abstract

The backpropagation of error algorithm used to train deep neural networks has been fundamental to the successes of deep learning. However, it requires sequential backward updates and non-local computations, which make it challenging to parallelize at scale and is unlike how learning works in the brain. Neuroscience-inspired learning algorithms, however, such as *predictive coding*, which utilize local learning, have the potential to overcome these limitations and advance beyond deep learning technologies in the future. While predictive coding originated in theoretical neuroscience as a model of information processing in the cortex, recent work has developed the idea into a general-purpose algorithm able to train deep neural networks using only local computations. In this survey, we review works that have contributed to this perspective and demonstrate the close connection between predictive coding and backpropagation in terms of generalization quality, as well as works that highlight the multiple advantages of using predictive coding over backpropagation-trained neural networks. Specifically, we show the substantially greater flexibility of predictive coding networks, which, unlike standard deep neural networks, can function as classifiers, generators, and associative memories simultaneously, and can be defined on arbitrary graph topologies. Finally, we review direct benchmarks of predictive coding networks on machine learning classification tasks, as well as its close connections to control theory and applications in robotics.

1 Introduction

Classical backpropagation (BP) [Rumelhart *et al.*, 1986] is the most successful algorithm in machine learning for training deep neural networks. Recently, however, limitations of BP have brought attention back to neuroscience-inspired learning. In particular, it is still unknown whether neural architectures trained via BP will be able to reach a level of intelligence, cognitive flexibility, and energy consumption comparable to the human brain. This could be solved using alternative learning methods that rely on locally available information, as learning in the brain does. Such an algorithm with extremely promising properties is *predictive coding (PC)*, an error-driven learning algorithm with local updates.

PC [Rao and Ballard, 1999; Friston, 2005; Srinivasan *et al.*, 1982] has emerged as an influential theory in computational neuroscience, which has a principled mathematical foundation as performing variational inference, linking it closely with normative theories of the Bayesian brain [Knill and Pouget, 2004], and which provides a single mechanism that can explain many varied perceptual and neurophysiological effects [Hohwy *et al.*, 2008; Auksztulewicz and Friston, 2016; Lotter *et al.*, 2016], while also postulating a biologically plausible neural dynamics and synaptic update rules [Friston, 2003; Lillicrap *et al.*, 2020; Millidge *et al.*, 2020c].

The fundamental idea of PC is to treat the cortex as performing inference and learning on a hierarchical probabilistic generative model, which is trained in an unsupervised setting to predict incoming sensory signals [Rao and Ballard, 1999; Friston, 2005; Clark, 2015]. In such an architecture, at each layer of the hierarchy, top-down predictions emanating from higher layers are matched with and cancel out incoming sensory data or prediction errors from lower layers. Unexplained aspects of the sensory data, in the form of prediction errors, are then transmitted upwards for higher layers of the hierarchy to explain. Transmitting only error information possesses a solid basis in information theory, where it is a way to maximize information transmission given a known model of an information source [Spratling, 2017; Bradbury, 2000], an important consideration for the brain, which is heavily optimized by evolution to satisfy tight constraints on energy usage and wire lengths, and thus must transmit and process as little information as possible [Barlow, 2001].

Historically, PC was first proposed for the retina [Srinivasan *et al.*, 1982], where neural circuits already subtract away much of the redundant information in the visual stimulus. The same principle was then applied as a general model for cortical processing by Rao and Ballard [1999], who showed that the model could replicate several well-known responses of neurons in the early visual cortex. The mathematical interpretation of the algorithm as performing variational inference was presented by Friston [2003; 2005; 2008].

[*]Equal contribution, listed in alphabetical order.
[†]Corresponding author: yuhang.song@ndcn.ox.ac.uk

PC is also closely related to the more general *free-energy principle* in theoretical neuroscience [Friston *et al.*, 2006], which states that the fundamental drive of the brain is to minimize the variational free energy via both perception (inference and learning) and action. PC networks (PCNs) can be derived as a special case (a "process theory") of the free-energy principle assuming a Gaussian generative model and performing inference and learning. The application of PC to robotics and its relationships to classical control theory depend on the third interpretation of the free-energy principle, where free energy is minimized via action, closely linked with the ideas of active inference [Friston *et al.*, 2017b; Friston *et al.*, 2017a]. For further reviews of PC, its mathematical foundation, and applications in neuroscience, see [Bogacz, 2017; Buckley *et al.*, 2017; Millidge *et al.*, 2021].

Although originating in neuroscience, a body of literature has investigated how PC can be related and applied to the existing deep learning literature (see Fig. 1). In this survey, we review this literature, which has developed in the last few years, focusing first on the recently uncovered relationships between the parameter updates in PCNs and in BP-trained artificial neural networks (ANNs), and second on the performance and superior flexibility of PCNs on large-scale deep learning tasks when applied to ANNs. This superior flexibility, combined with using only local computations ultimately enables a much greater parallelizability of PCNs compared to ANNs, especially on neuromorphic hardware. This greater scalability means that, as ANNs continue to scale [Kaplan *et al.*, 2020], the limited memory bandwidth afforded by current GPUs may become increasingly a limiting factor in training, and the greater parallelizability and memory bandwidth of neuromorphic hardware, where computations and memories are colocated, may lead to the adoption of PCN-like architectures, which can be efficiently trained on such hardware, leading ultimately to a future of PCNs trained without BP.

The rest of this survey is organized as follows. In Section 2, we present a general overview of PCNs, first describing their mathematical structure, their training and testing dynamics, and thirdly their interpretation as variational inference algorithms. In Section 3, we review the recently uncovered relationships between PC and BP. In Section 4, we discuss the capabilities of PCNs on classification, generation, and reconstruction tasks. In Section 5, we demonstrate that PCNs can function as associative memory models, and in Section 6, we generalize PCNs to arbitrary graph topologies. In Section 7, we review applications of PCNs to problems in control and robotics, and in Section 8, we summarize, and we discuss open research challenges for PC in machine learning.

2 Overview of Predictive Coding

We now recall the key concepts of PC. We give an overview of PCNs, which are the PC equivalent of ANNs, and we recall PC as variational inference, which gives additional insights into the computations and learning performed by PCNs.

2.1 Predictive Coding Networks

The mathematical formulation of PC can be interpreted as postulating two kinds of "neurons". The first encodes time-dependent neural predictions and is denoted by $\bar{x}_t$, while the second encodes prediction errors and is denoted by $\bar{\varepsilon}_t$. Mathematically, both quantities are N-dimensional vectors that change over time steps t during inference and learning. Hierarchical PCNs can be expressed as ANNs, so that PCNs can be directly compared to ANNs. As such, the value and error nodes are partitioned into $L+1$ layers. We denote by $\bar{x}_{L,t}$ the neurons of the input layer, and by $\bar{x}_{l,t}$ and $\bar{\varepsilon}_{l,t}$ the neurons of the other layers $l \in \{0, \dots, L-1\}$. In this case, a specific stimulus is propagated up the hierarchy, and the goal of each layer of neurons is to predict the value of the next layer via

$$\bar{\mu}_{l,t} = \bar{\theta}_{l+1} f(\bar{x}_{l+1,t}), \tag{1}$$

where f is a non-linearity, and $\bar{\theta}_{l+1}$ is the matrix of weights connecting layer $l+1$ to layer l. The error of this prediction is the difference between the neural activity of a layer and its prediction, i.e., $\bar{\varepsilon}_{l,t} = \bar{x}_{l,t} - \bar{\mu}_{l,t}$, which is then propagated down the hierarchy, and used in the learning process to update the weights of the network. Ultimately, the learning algorithm optimizes a global energy function, defined as the sum of squared prediction errors at each layer:

$$\mathcal{F}_t = \frac{1}{2} \sum\nolimits_l \|\bar{\varepsilon}_{l,t}\|^2. \tag{2}$$

Training: During training, the highest layer is fixed to an input data point $\bar{s}_{in}$, i.e., $\bar{x}_{L,t} = \bar{s}_{in}$ for every t, and the lowest layer is fixed to a label or target vector $\bar{s}_{out}$ in the same way. During a process called *inference*, the weight parameters are fixed, and the neural activities are continuously updated to minimize the energy function of Eq. (2) by running gradient descent until convergence, at which point, a single weight update is performed. Here, the value nodes are fixed, and the weight parameters are updated via gradient descent on the same energy function. When inference and weight updates are defined this way, every computation only needs local information. For a detailed derivation of these equations, refer to [Whittington and Bogacz, 2017].

Testing: Here, only the highest layer is fixed to the data, so the network infers the label given a test point. This process is equivalent to the inference phase described above: the weight parameters are fixed, and the neural activities are updated until convergence by running gradient descent on the energy function. Note that different works follow different paradigms for interleaving the updates of the neural activities $\bar{x}_t$ and weights $\bar{\theta}$. In most works, neural activities are all simultaneously updated for T iterations with the aim of reaching convergence of the inference, and the weights are updated once at convergence [Whittington and Bogacz, 2017; Salvatori *et al.*, 2021a; Millidge *et al.*, 2020a]. However, in certain cases, it has been noted that updating the weights and activities of different layers in different moments yields a better performance [Ororbia and Kifer, 2020; Han *et al.*, 2018].

2.2 Predictive Coding as Variational Inference

Mathematically, PCNs can also be expressed as variational inference on hierarchical Gaussian generative models. We define a hierarchy of layers indexed by l, where the distribution of activations at each layer is a Gaussian distribution

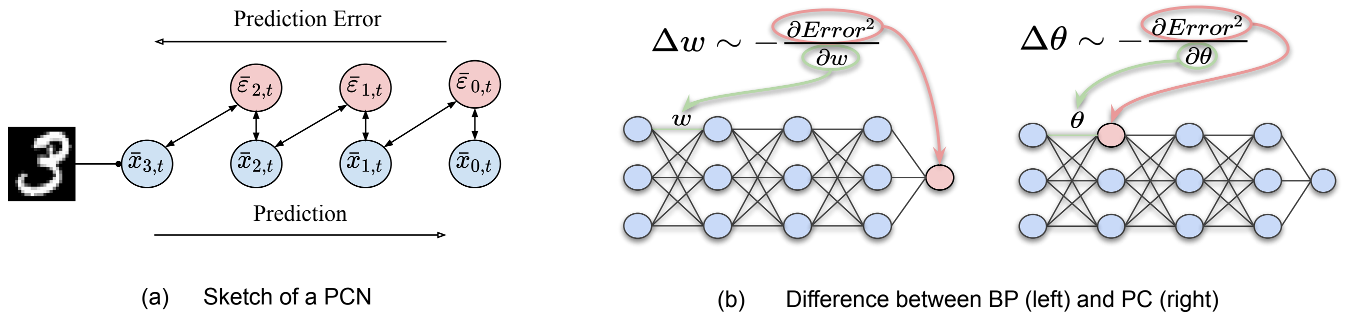

Prediction Error
$\bar{\varepsilon}_{2,t}$
$\bar{\varepsilon}_{1,t}$
$\bar{\varepsilon}_{0,t}$
$\bar{x}_{3,t}$
$\bar{x}_{2,t}$
$\bar{x}_{1,t}$
$\bar{x}_{0,t}$
Prediction
(a) Sketch of a PCN
$\Delta w \sim -\frac{\partial Error^2}{\partial w}$
w
$\Delta \theta \sim -\frac{\partial Error^2}{\partial \theta}$
θ
(b) Difference between BP (left) and PC (right)

Figure 1: (a): A multilayer PCN trained on a data point of MNIST. The neural activities of a specific layer predict the activities of the previous layer in a forward direction. The error in this prediction is then propagated back down the hierarchy. (b) Difference between the update rules of BP (left) and PC (right). Particularly, the loss function of BP defines an error only on the output layer, and this error is minimized via gradient descent. In very deep networks, this causes single weights to be updated to minimize an error that could be dozens of layers away. In contrast, PC minimizes a local energy function for each layer.

with a mean given by a nonlinear function of the layer below $f(\bar{x}_{l+1})$ with parameters $\bar{\theta}_{l+1}$ and an identity covariance I:

$$p(\bar{x}_{0,t} \dots \bar{x}_{L,t}) = p(\bar{x}_{L,t}) \prod_{l=0}^{L-1} p(\bar{x}_{l,t}|\bar{x}_{l+1,t})$$
$$p(\bar{x}_{l,t}|\bar{x}_{l+1,t}) = \mathcal{N}(\bar{x}_{l,t}; \bar{\theta}_{l+1} f(\bar{x}_{l+1,t}), I)\,.$$

The PCN can then be "queried" by conditioning on a data or label item. For instance, suppose that the input layer $l = L$ is fixed to some data item $\bar{s}_{in}$. We then wish to infer the state of the rest of the network given this conditioning $p(\bar{x}_{0,t} \dots \bar{x}_{L-1,t}|\bar{x}_L)$. This problem can be solved by variational inference [Beal, 2003; Wainwright and Jordan, 2008; Jordan *et al.*, 1998]. In general, variational inference approximates an intractable inference through an optimization problem, where an approximate *variational posterior* distribution q is optimized, so as to minimize its distance from the optimal posterior p. This optimization procedure is performed by minimizing an upper bound on this divergence known as the *variational free energy* $\mathcal{F}_t$.

In PCNs, we assume that the variational posterior is factorized into independent posteriors for each layer $q(\bar{x}_{0,t} \dots \bar{x}_{L-1,t}) = \prod_{l=0}^{L-1} q(\bar{x}_{l,t})$ and, combined with the Laplace approximation [Friston *et al.*, 2007], this allows us to considerably simplify the expression for the free energy into a sum of squared prediction errors (see [Buckley *et al.*, 2017]), equivalent to the one in Eq. 2 up to an additive constant:

$$\mathcal{F}_t \approx \sum_{l=0}^{L-1} \log p(\bar{x}_{l,t}|\bar{x}_{l+1,t}) \approx \sum_{l=0}^{L-1} \|\bar{\varepsilon}_{l,t}\|^2\,, \quad (3)$$

where $\bar{\varepsilon}_{l,t} = \bar{x}_{l,t} - \bar{\theta}_{l+1} f(\bar{x}_{l+1,t})$ is the "prediction error" for each layer. When applied to ANNs, we typically assume that the layerwise dependencies are parametrized by a parameter matrix $\bar{\theta}_{l+1}$, which corresponds to the weights in an ANN. Then, both the activations $\bar{x}_{l,t}$ and weights can be updated as a gradient descent on the free energy:

$$d\bar{x}_l/dt \propto -\partial \mathcal{F}_t / \partial \bar{x}_{l,t} \quad (4)$$
$$d\bar{\theta}_l/dt \propto -\partial \mathcal{F}_t / \partial \bar{\theta}_l \big|_{\bar{x}=\bar{x}^*_{l,t}}\,. \quad (5)$$

PCNs then operate in two phases, where first the activation means $\bar{x}_{l,t}$ are updated to minimize the free energy until they reach an equilibrium $\bar{x}^*_{l,t}$, and then the weights $\bar{\theta}_l$ are updated for a single step given the equilibrium values of the activations. These phases are known as *inference* and *learning*.

In the context of ANNs, PC recasts the feedforward pass in an ANN as an inference problem over the activations of the layers of an ANN given some conditioning on either input or output layers, or both, where the uncertainty about the "correct" activations is assumed to be Gaussian around a mean given by the top-down prediction from the layer above. Importantly, this inference problem is solved dynamically at each inference step, and the conditioning variables can be varied flexibly depending on the desired task. This enables the PCN to use its learned generative model (encoded in the weights $\bar{\theta}_l$) to be repurposed for different inference problems at run-time, and accounts for the superior flexibility of PCNs over ANNs, as demonstrated in recent work.

3 Predictive Coding and Backpropagation

Recently, multiple results have explored similarities and relationships between PC and BP, showing that PC can closely approximate or exactly perform BP under certain conditions on supervised learning tasks. Firstly, it has been shown that PC well approximates the parameter update of BP on multilayer perceptrons (MLPs), albeit under some strict conditions [Whittington and Bogacz, 2017]. This result has been recently extended in two orthogonal directions: it has been shown that PC converges to BP not only on MLPs, but also on any computational graph [Millidge *et al.*, 2020a]. For these results to hold, one of two conditions must be met: either the activity values remain very close to their feedforward pass values such that the prediction error is small, or else the layerwise derivatives must be held fixed to their feedforward pass values and the network run to equilibrium. Moreover, experimental results also empirically show that PC approximates BP updates under less restrictive conditions, i.e., a small output error is enough, and the energy does not have to be completely converged. An exactness result also holds, as a variation of PC, called Z-IL, performs exact BP on MLPs if the weights are updated after the first non-zero inference step at each layer when the network activations are initialized

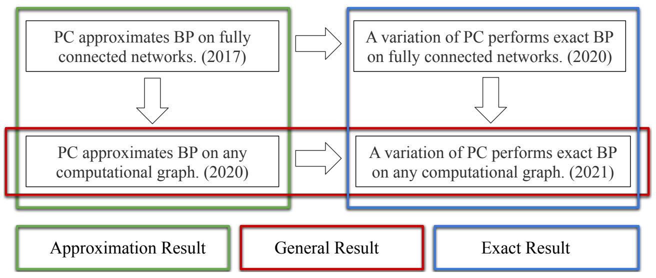

PC approximates BP on fully connected networks. (2017)
A variation of PC performs exact BP on fully connected networks. (2020)
PC approximates BP on any computational graph. (2020)
A variation of PC performs exact BP on any computational graph. (2021)
Approximation Result
General Result
Exact Result

Figure 2: Historical and conceptual sketch of the results unifying predictive coding (PC) and backpropagation (BP).

to their feedforward pass values [Song *et al.*, 2020]. These results also extend to Z-IL being able to exactly replicate the parameter update of BP on any computational graph [Salvatori *et al.*, 2021b]. The advantage of these exactness results are twofold: first, for Z-IL, they require only a small number of steps to perform a full update of the parameters, and empirical results show that Z-IL is almost as efficient as BP. Second, it provides a novel (but equivalent) implementation of BP, which is able to learn via local computations. All these results are experimentally validated on multiple architectures, such as LSTMs [Hochreiter and Schmidhuber, 1997], transformers [Vaswani *et al.*, 2017], and ResNets [He *et al.*, 2016]. A historical sketch of these results is given in Fig. 2.

4 Performance of Predictive Coding

In this section, we review recent results obtained by PCNs on classical computer vision benchmarks. Particularly, we briefly review results in image classification and generation.

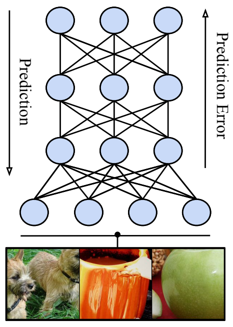
Classification: The connection to BP for supervised learning suggests that PCNs should perform equally well on image classification tasks. This is indeed the case: the first formulation of PC for supervised learning, equivalent to the one described in the preliminary section, shows that PC is able to obtain a performance comparable to BP on small multilayer networks trained on MNIST [Whittington and Bogacz, 2017]. A similar result was also obtained using a variation of PC not restricted to be trained using mean squared error as energy function. It is in fact possible to define a generalization of IL, which uses layer-specific loss functions [Ororbia and Mali, 2019]. This variation, called *local representation alignment*, is able to reach a competitive performance with BP on MNIST and FashionMNIST. On more challenging tasks, a deep convolutional PCN is able to achieve a performance similar to BP on complex datasets, such as CIFAR10, and acceptable results on ImageNet [Han *et al.*, 2018]. This model updates the value nodes of one layer at a time, multiple times via lateral recurrent connections, before performing a weight update. It is intuitively similar to a deep convolutional network trained via Z-IL, augmented with lateral connections.

Generation: Above, we have reviewed the connection between PC and BP for image classification, which implies that, with some effort, it should be possible to use PC to train classifiers on large-scale datasets. PC is, however, a generative model, as suggested by its formulation as variational inference [Rao and Ballard, 1999; Friston, 2005]. This implies that the PC models surveyed in the previous section can also be used for data generation from labels, as long as certain regularizations are applied due to the ill-posed nature of the inverse problem [Sun and Orchard, 2020]. Additionally, PCNs can also be used directly as generative models due to their interpretation as probabilistic graphical models by swapping the "direction" of the network, so that the label is treated as the "input", and the data are treated as the "output". The basic architecture used to perform generative tasks resembles the decoder part of autoencoders, and is sketched in Fig. 3(a). More complex models, which have a similar structure but are augmented with different kinds of connections, have been shown to generalize to unseen images [Ororbia and Kifer, 2020; Salvatori *et al.*, 2022]. Particularly, Ororbia and Kifer [2020] present three generative models: the first one is a novel model with recurrent connections, while the second and the third are implementations of Rao and Ballard's original PC [1999], and of a model designed by Friston [2008]. The extensive experiments show that generative PCNs are able to well generate novel black and white images of different datasets, as shown in Fig. 3(b). A qualitative evaluation against standard baselines in machine learning shows that this method obtains comparable results. Hence, an interesting future direction is to directly test the generation capabilities of large-scale PCN equivalents to deep convolutional networks on challenging image datasets such as ImageNet.

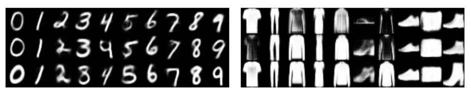
An important quality of generative PCNs is their ability to generalize well on novel tasks. This suggests that they learn an internal probabilistic representation of the dataset, and can apply it when tested on tasks that they were not trained for. This differs from ANNs, which exhibit a generalization across data from the same task but fail to generalize *across tasks*. The generalization capability of PCNs is more equivalent to meta learning and exhibits significantly more flexibiliy. This greater flexibility of PCNs originates from their inference phase. For instance, it is possible to provide a PCN with half a test image and let the network infer the missing pixels via running energy minimization, or to present a corrupted data point (e.g., with Gaussian noise), and ask the model to clear it, as shown in Fig. 3(c). Importantly, PCNs can accomplish these tasks even if not directly trained to do so, unlike ANNs, which must be trained to perform each specific task [Ororbia and Kifer, 2020; Salvatori *et al.*, 2021a; Salvatori *et al.*, 2022].

5 Associative Memories

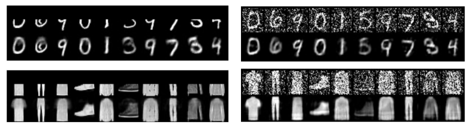
Generative PCNs have recently also been demonstrated to function as associative memories. In machine learning, the task of an associative memory model is to store and retrieve data points. When presented with a corrupted or incomplete variation of a stored data point, a good associative memory model has to detect the uncorrupted memory and return it as an output. Generative PCNs are able to store and retrieve complex memories, such as ImageNet pictures [Salvatori *et al.*, 2021a]. This is done by training the model on a subset of ImageNet, and retrieving the original data points via energy minimization when providing highly corrupted or in-



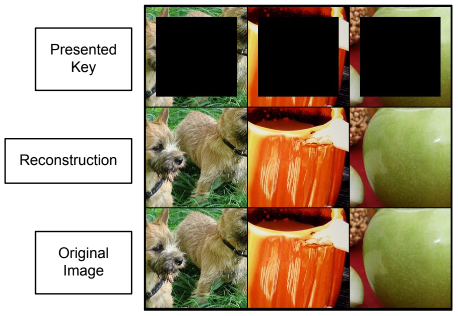
(a) Sketch of a Generative PCN



(b) Generation



(c) Reconstruction and Denoising



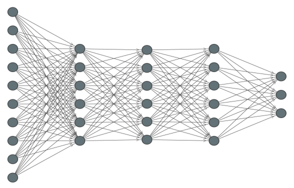
(d) Associative Memory Experiment

Figure 3: (a) Sketch of a generative PCN. In contrast to networks trained for classification, the input image is presented in the first layer of the network. The energy minimization updates the weights to get zero (or low) error on it. (b) Examples of generated MNIST and FashionMNIST using a generative PCN. (c) Examples of reconstructed (left) and denoised (right) MNIST and FashionMNIST images using a generative PCN. (d) Examples of retrieved ImageNet pictures when presenting a corrupted one, with a black patch covering the vast majority of the pixels. Particularly, 100 images were stored in this example. Figures (b), (c), and (d) are taken from the original papers, i.e., [Ororbia and Kifer, 2020], [Salvatori *et al.*, 2022], and [Salvatori *et al.*, 2021a], respectively.

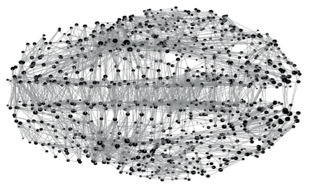
complete variants of them. While the fully connected PCN structure does not allow generalization to unseen data points on complex images such as ImageNet, their retrieval capacity demonstrates their ability to generate high-quality reconstructions, see Fig. 3(d) for an example. This model has been extensively compared with different associative memory models, such as continuous-state Hopfield networks [Ramsauer *et al.*, 2021] and autoencoders trained with BP [Radhakrishnan *et al.*, 2019]. In both cases, the PC-based memory model has outperformed its classic counterparts, showing a retrieval robustness superior to any other baseline. This model also possesses a high degree of biological plausibility, as it has been hypothesised that the brain stores and retrieves memories using a PC architecture where the hippocampus sends fictive prediction errors to the sensory neurons via hierarchical networks in the neocortex, which are minimized by memory retrieval [Barron *et al.*, 2020].

6 Learning on Arbitrary Graph Topologies

Learning on networks of any structure is not possible using BP, where information first flows in one direction via the feedforward pass, and then errors flow in the reverse direction during the backwards pass. Hence, a cycle in the computational graph of an ANN trained with BP would cause an infinite loop. While the problem of training on some specific cyclic structures has been partially addressed using BP through time [Hochreiter and Schmidhuber, 1997; Rumelhart *et al.*, 1986; Williams and Zipser, 1989] on sequential data, the restriction to hierarchical architectures may present a limitation to reaching brain-like intelligence, since the human brain has an extremely complex and entangled neural structure that is heterarchically organized with small-world connections [Avena-Koenigsberger *et al.*, 2018]—a topology that is likely highly optimized by evolution. Hence, a recent direction of research aimed to extend learning to arbitrary graph topologies. A popular example is the *assembly calculus* [Papadimitriou *et al.*, 2020], a Hebbian learning method that can perform different operations implicated in cognitive phenomena. However, Hebbian learning meth-



(a) Artificial Neural Network



(b) Biological Neural Network

Figure 4: Difference in topology between an ANN (left) and a sketch of a network of structural connections that link distinct neuronal elements in a brain (right). Figure taken from [Salvatori *et al.*, 2022].

ods cannot perform well compared to error-driven ones such as BP [Movellan, 1991]. PC, however, has both the desired properties that allow high-quality representation learning on arbitrary graph topologies: it is error-driven, and only learns via local computations. Moreover, it has been shown that it is possible to perform generation and classification tasks on extremely entangled networks, which closely resemble brain regions [Salvatori *et al.*, 2022], as schematically illustrated in Fig. 2. This enables a more general learning framework, which converges to a global solution via energy minimization that can perform multiple tasks simultaneously, such as classification and generation, but also to develop novel architectures, optimized for a single specific task. Tested on generation, reconstruction, and denoising tasks, this model has been shown to have a performance superior or comparable to standard autoencoders.

7 Predictive Coding for Control and Robotics

In line with the free-energy principle in considering both perception and action as emerging from an imperative to minimize free energy, PC methods can also be directly applied to control problems, possess close links to classical control theory, and have been applied productively to problems in robotics. Although the free energy does not explicitly include an action term, it includes one implicitly through the depen-

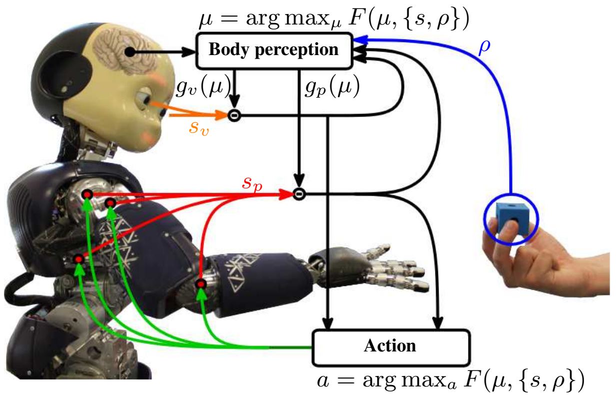

$\mu = \arg\max_\mu F(\mu, \{s, \rho\})$
Body perception
ρ
$g_v(\mu)$
$g_p(\mu)$
s_v
s_p
Action
$a = \arg\max_a F(\mu, \{s, \rho\})$

Figure 5: Graphical representation of the joint minimization of free energy by action and control to enable a simultaneous state estimation and action selection in a robotic grasping task with a humanoid robot. Figure taken from [Oliver *et al.*, 2021].

dence on data $\bar{x}_0$, which could depend on actions a. This dependency can be computed explicitly by using the chain rule [Friston *et al.*, 2009; Friston, 2011],

$$da/dt \propto -\partial\mathcal{F}/\partial a = -\partial\mathcal{F}/\partial\bar{x}_0 \cdot \partial\bar{x}_0/\partial a\,, \tag{6}$$

where $\partial\bar{x}_0/\partial a$ is known as a *forward model*, which explicitly quantifies how data depend on actions. In the case of linear generative models and using generalized coordinates of motion [Baltieri and Buckley, 2019], PC has been shown to be equivalent to PID control, a widely used method in classical control theory [Johnson and Moradi, 2005]. PC also provides a generalization of PID for both additional dynamical orders (i.e., using higher-order derivatives) as well as instant generalizations to non-identity and ultimately nonlinear dynamics.

PCNs have also been widely used in robotics. The generative model in PC can be set to model the dynamics of a system and then torques can be inferred to realize a desired motion [Lanillos and Cheng, 2018]. PC has thus been applied to a variety of robotics problems [Lanillos and Cheng, 2018] as well as drone and quadcopter control [Meera and Wisse, 2020; Meera and Wisse, 2021]. An additional advantage is that PC can also be used for state estimation [Pezzato *et al.*, 2020; Oliver *et al.*, 2021] (including with high dimensional image inputs [Sancaktar *et al.*, 2020], providing a joint solution of state estimation and control as depicted in Figure 5. For a recent full review of this literature, see [Lanillos *et al.*, 2021].

More generally, if we consider PCNs as embodying an implicit probabilistic graphical model, and interpret some nodes of the PCN as "action nodes", then given that other nodes can be fixed as to a set of desired outcomes of control, PC will infer the actions consistent with the conditioned outcomes. Explorations of this idea are presented in [Bogacz, 2020; Kinghorn *et al.*, 2021], and this mechanism is simply an example of the active inference [Friston *et al.*, 2017b; Millidge *et al.*, 2020b] and control as inference [Attias, 2003; Toussaint and Storkey, 2006; Levine, 2018] frameworks, which interpret control problems as probabilistic inference problems, which can be solved by simply inferring the correct action or action sequences conditioned on achieving a high reward or desired state trajectory.

8 Summary and Open Challenges

Both the theory and practice of PC have advanced substantially over the last few years, with an important set of theoretical results revealing a close connection with BP being developed, as well as significant empirical strides being made in the capacity of PCNs to perform successfully on large-scale machine learning benchmarks, often with a performance very close to equivalent ANNs. While current research has shown that PC is closely connected to BP and that PCNs can often match the performance of BP-trained models on a variety of machine learning benchmarks, there are only a few cases where PCNs perform demonstrably *better* than comparable ANNs. Thus, at present, PC is not yet used in industrial applications. However, the promising properties highlighted in this survey strongly suggest that this will change in the next years. Future research should be directed at finding applications where the unique properties of PC can be utilized to outperform existing methods, as well as theoretical research exploring these properties further.

One unique property of PC is its superior flexibility due both to its inference phase, and to the local computations that allow training on any graph structure. This means that PCNs can flexibly perform different inference tasks given the same network (while an ANN would have to be trained separately for each task), and allows PCNs to natively handle missing data, while ANNs need heuristic imputation schemes. The second property enables new progress in neural architecture search. Beyond this, it is worth noting that essentially all current benchmarks, layers, initializations, and other "tricks", have been developed specifically for ANNs, while no such effort has been performed for PCNs. BP also showed many flaws early on that did not allow it to scale to deep architectures. In the late 90's, kernel learning methods were more popular and effective than neural networks, which suffered from the vanishing gradient problem. Moreover, into the late 2000s, deep neural networks were thought to require unsupervised pretraining to be trained effectively [Vincent *et al.*, 2010; Hinton, 2007] Most of these limitations have now been addressed over thirty years of research by the development of multiple tricks, such as dropout [Hinton *et al.*, 2012; Srivastava *et al.*, 2014] and batch normalization [Ioffe and Szegedy, 2015], which make learning on huge networks effective. What are the "dropout" and "batch norm" equivalents for PC? Addressing these questions is of vital importance to scale these networks, and should hence be a topic of interest for future research.

A crucial property of PCNs is the locality of their weight updates and hence greater parallelizability compared to ANNs. If exploited properly, this may enable substantially more efficient training of extremely large-scale PCN models by reducing the communication requirements and wait time induced by the sequential backward step of BP, as well enable an efficient implementation on neuromorphic hardware. To conclude, while theoretical results indicate close connections with BP, these only apply under certain conditions. Future work may investigate the behavior of PCNs when these conditions are relaxed, and whether it has any advantages or different properties compared to standard BP.

Acknowledgements

Beren Millidge and Rafal Bogacz were supported by the BBSRC grant BB/S006338/1, and Rafal Bogacz was also supported by the MRC grant MC_UU_00003/1. Yuhang Song was supported by the China Scholarship Council under the State Scholarship Fund and by a J.P. Morgan AI Research Fellowship. Thomas Lukasiewicz was supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1, the AXA Research Fund, and the EPSRC grant EP/R013667/1.

References

[Attias, 2003] H. Attias. Planning by probabilistic inference. In *Proc. AISTATS*. PMLR, 2003.

[Auksztulewicz and Friston, 2016] R. Auksztulewicz and K. Friston. Repetition suppression and its contextual determinants in predictive coding. *Cortex*, 2016.

[Avena-Koenigsberger *et al.*, 2018] A. Avena-Koenigsberger, B. Misic, and O. Sporns. Communication dynamics in complex brain networks. *Nat. Rev. Neurosci.*, 19, 2018.

[Baltieri and Buckley, 2019] M. Baltieri and C. Buckley. PID control as a process of active inference with linear generative models. *Entropy*, 2019.

[Barlow, 2001] H. Barlow. Redundancy reduction revisited. *Network*, 2001.

[Barron *et al.*, 2020] H. Barron, R. Auksztulewicz, and K. Friston. Prediction and memory: A predictive coding account. *Prog. Neurobiol.*, 2020.

[Beal, 2003] M. Beal. *Variational algorithms for approximate Bayesian inference*. UCL, London, 2003.

[Bogacz, 2017] R. Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *J. Math. Psychol.*, 2017.

[Bogacz, 2020] R. Bogacz. Dopamine role in learning and action inference. *Elife*, 2020.

[Bradbury, 2000] J. Bradbury. Linear predictive coding. *McGraw Hill*, 2000.

[Buckley *et al.*, 2017] C. Buckley, C. S. Kim, S. McGregor, and A. Seth. The free energy principle for action and perception: A mathematical review. *J. Math. Psychol.*, 2017.

[Clark, 2015] A. Clark. *Surfing Uncertainty: Prediction, Action, and the Embodied Mind.* OUP, 2015.

[Friston *et al.*, 2006] K. Friston, J. Kilner, and L. Harrison. A free energy principle for the brain. *J. Physiol.*, 2006.

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

[Friston *et al.*, 2009] K. Friston, J. Daunizeau, S. Kiebel. Reinforcement learning or active inference? *PloS One*, 2009.

[Friston *et al.*, 2017a] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, and G. Pezzulo. Active inference: A process theory. *Neural Comput.*, 2017.

[Friston *et al.*, 2017b] K. J. Friston, T. Parr, and B. de Vries. The graphical brain: Belief propagation and active inference. *Netw. Neurosci.*, 2017.

[Friston, 2003] K. Friston. Learning and inference in the brain. *Neural Netw.*, 2003.

[Friston, 2005] K. Friston. A theory of cortical responses. *Philos. Trans. R. Soc. Lond., B, Biol. Sci.*, 2005.

[Friston, 2008] K. Friston. Hierarchical models in the brain. *PLoS Comput. Biol.*, 2008.

[Friston, 2011] K. Friston. What is optimal about motor control? *Neuron*, 2011.

[Han *et al.*, 2018] K. Han, H. Wen, Y. Zhang, D. Fu, E. Culurciello, and Z. Liu. Deep predictive coding network with local recurrent processing for object recognition. *Proc. NeurIPS*, 2018.

[He *et al.*, 2016] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[Hinton *et al.*, 2012] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[Hinton, 2007] G. E. Hinton. To recognize shapes, first learn to generate images. *Prog. Brain Res.*, 165:535–547, 2007.

[Hochreiter and Schmidhuber, 1997] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.

[Hohwy *et al.*, 2008] J. Hohwy, A. Roepstorff, and K. Friston. Predictive coding explains binocular rivalry: An epistemological review. *Cognition*, 2008.

[Ioffe and Szegedy, 2015] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, pages 448–456. PMLR, 2015.

[Johnson and Moradi, 2005] M. Johnson and M. Moradi. *PID Control*. Springer, 2005.

[Jordan *et al.*, 1998] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. In *Learning in Graphical Models*. 1998.

[Kaplan *et al.*, 2020] J. Kaplan, S. McCandlish, T Henighan, T. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

[Kinghorn *et al.*, 2021] P. Kinghorn, B. Millidge, and C. Buckley. Habitual and reflective control in hierarchical predictive coding. *arXiv:2109.00866*, 2021.

[Knill and Pouget, 2004] D. Knill and A. Pouget. The Bayesian brain: The role of uncertainty in neural coding and computation. *Tends Neurosci.*, 2004.

[Lanillos and Cheng, 2018] P. Lanillos and G. Cheng. Adaptive robot body learning and estimation through predictive coding. In *Proc. IROS*, 2018.

[Lanillos *et al.*, 2021] P. Lanillos, C. Meo, C. Pezzato, et al. Active inference in robotics and artificial agents: Survey and challenges. *arXiv:2112.01871*, 2021.

[Levine, 2018] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv:1805.00909*, 2018.

[Lillicrap *et al.*, 2020] T. Lillicrap, A. Santoro, L. Marris, C. Akerman, and G. Hinton. Backpropagation and the brain. *Nat. Rev. Neurosci.*, 2020.

[Lotter *et al.*, 2016] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv:1605.08104*, 2016.

[Meera and Wisse, 2020] A. Meera and M. Wisse. Free energy principle based state and input observer design for linear systems with colored noise. In *Proc. ACC*, 2020.

[Meera and Wisse, 2021] A. Meera and M. Wisse. A brain inspired learning algorithm for the perception of a quadrotor in wind. *arXiv:2109.11971*, 2021.

[Millidge *et al.*, 2020a] B. Millidge, A. Tschantz, C. Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv:2006.04182*, 2020.

[Millidge *et al.*, 2020b] B. Millidge, A. Tschantz, A. Seth, and C. Buckley. On the relationship between active inference and control as inference. In *Proc. IWAI*, 2020.

[Millidge *et al.*, 2020c] B. Millidge, A. Tschantz, A. Seth, and C. Buckley. Relaxing the constraints on predictive coding models. *arXiv:2010.01047*, 2020.

[Millidge *et al.*, 2021] B. Millidge, A. Seth, and C. Buckley. Predictive coding: A theoretical and experimental review. *arXiv:2107.12979*, 2021.

[Movellan, 1991] J. R. Movellan. Contrastive Hebbian learning in the continuous Hopfield model. In *Connectionist Models*, pages 10–17. Elsevier, 1991.

[Oliver *et al.*, 2021] G. Oliver, P. Lanillos, and G. Cheng. An empirical study of active inference on a humanoid robot. *IEEE Trans. Cogn. Dev. Syst.*, 2021.

[Ororbia and Kifer, 2020] A. Ororbia and D. Kifer. The neural coding framework for learning generative models. *arXiv:2012.03405*, 2020.

[Ororbia and Mali, 2019] A. Ororbia and A. Mali. Biologically motivated algorithms for propagating local target representations. In *Proc. AAAI*, 2019.

[Papadimitriou *et al.*, 2020] C.. Papadimitriou, S.. Vempala, D. Mitropolsky, M. Collins, and W. Maass. Brain computation by assemblies of neurons. *PNAS*, 2020.

[Pezzato *et al.*, 2020] C. Pezzato, R. Ferrari, and C. Corbato. A novel adaptive controller for robot manipulators based on active inference. *IEEE Robot. Autom. Lett.*, 2020.

[Radhakrishnan *et al.*, 2019] A. Radhakrishnan, M. Belkin, and C. Uhler. Overparameterized neural networks implement associative memory. *arXiv:1909.12362*, 2019.

[Ramsauer *et al.*, 2021] H. Ramsauer, B. Schäfl, J. Lehner, et al. Hopfield networks is all you need. *Proc. ICLR*, 2021.

[Rao and Ballard, 1999] R. Rao and D. Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.*, 1999.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 1986.

[Salvatori *et al.*, 2021a] T. Salvatori, Y. Song, Y. Hong, L. Sha, S. Frieder, Z. Xu, R. Bogacz, and T. Lukasiewicz. Associative memories via predictive coding. In *Proc. NeurIPS*, 2021.

[Salvatori *et al.*, 2021b] T. Salvatori, Y. Song, T. Lukasiewicz, R. Bogacz, and Z. Xu. Predictive coding can do exact backpropagation on any neural network. *arXiv:2103.04689*, 2021.

[Salvatori *et al.*, 2022] T. Salvatori, L. Pinchetti, B. Millidge, Y. Song, T. Bao, R. Bogacz, and T. Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. *arXiv:2201.13180*, 2022.

[Sancaktar *et al.*, 2020] C. Sancaktar, M. van Gerven, P. Lanillos. End-to-end pixel-based deep active inference for body perception and action. In *Proc. ICDL-EpiRob*, 2020.

[Song *et al.*, 2020] Y. Song, T. Lukasiewicz, Z. Xu, and R. Bogacz. Can the brain do backpropagation? — Exact implementation of backpropagation in predictive coding networks. In *Proc. NeurIPS*, 2020.

[Spratling, 2017] M. W. Spratling. A review of predictive coding algorithms. *Brain Cogn.*, 2017.

[Srinivasan *et al.*, 1982] M. Srinivasan, S. Laughlin, and A. Dubs. Predictive coding: A fresh view of inhibition in the retina. *Proc. R. Soc. B: Biol. Sci.*, 1982.

[Srivastava *et al.*, 2014] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[Sun and Orchard, 2020] W. Sun and J. Orchard. A predictive-coding network that is both discriminative and generative. *Neural Comput.*, 2020.

[Toussaint and Storkey, 2006] M. Toussaint and A. Storkey. Probabilistic inference for solving discrete and continuous state Markov decision processes. In *Proc. ICML*, 2006.

[Vaswani *et al.*, 2017] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In *Proc. NIPS*, 2017.

[Vincent *et al.*, 2010] P. Vincent, H. Larochelle, I. Lajoie, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11(12), 2010.

[Wainwright and Jordan, 2008] M. Wainwright and M. Irwin Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc, 2008.

[Whittington and Bogacz, 2017] J. C. R. Whittington and R. Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Comput.*, 2017.

[Williams and Zipser, 1989] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280, 1989.