# PREDICTIVE CODING: TOWARDS A FUTURE OF DEEP LEARNING BEYOND BACKPROPAGATION?

**Beren Millidge**[*,1], **Tommaso Salvatori**[*,2] , **Yuhang Song**[1,2,†], **Rafal Bogacz**[1], **Thomas Lukasiewicz**[2]
[1]MRC Brain Network Dynamics Unit, University of Oxford, UK
[2]Department of Computer Science, University of Oxford, UK
[1]firstname.lastname@ndcn.ox.ac.uk, [2]firstname.lastname@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 current deep learning technologies. 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 neural networks using only local computations. In this survey, we review works that have contributed to this perspective and demonstrate the close theoretical connections between predictive coding and backpropagation, as well as works that highlight the multiple advantages of using predictive coding models over backpropagation-trained neural networks. Specifically, we show the substantially greater flexibility of predictive coding networks against equivalent deep neural networks, which 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, Hinton, & Williams, 1986) is the most successful algorithm in AI and 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. An algorithm with extremely promising properties is *predictive coding (PC)*, an error driven learning algorithm with local updates.

PC (K. Friston, 2005; Rao & Ballard, 1999; Srinivasan, Laughlin, & Dubs, 1982) has emerged as an influential theory in computational neuroscience, which has a significant mathematical foundation as variational inference, linking it closely with normative theories of the Bayesian brain (Knill & Pouget, 2004), and which provides a single mechanism that can explain many varied perceptual and neurophysiological effects (Auksztulewicz & Friston, 2016; Hohwy, Roepstorff, & Friston, 2008; Lotter, Kreiman, & Cox, 2016), while also postulating a biologically plausible neural dynamics and synaptic update rules (K. Friston, 2003; Lillicrap, Santoro, Marris, Akerman, & Hinton, 2020; Millidge, Tschantz, Seth, & Buckley, 2020b).

The fundamental idea of PC is to treat the cortex as performing simultaneous inference and learning on a hierarchical probabilistic generative model, which is trained in an unsupervised setting to predict incoming sensory signals (Clark, 2015; K. Friston, 2005; Rao & Ballard, 1999). In such an architecture, at each layer of the hierarchy, top-down

[*]Equal contribution, listed in alphabetical order.
[†]Corresponding author

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. The transmission of only error information possesses a solid basis in information theory, where it is a way to maximize information transmission per bit given a known model of an information source (Bradbury, 2000; Spratling, 2017), an important consideration for the brain, heavily optimized by evolution to satisfy tight constraints on energy usage and wire lengths, which 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 K. Friston (2003) and K. Friston (2005).

PC is also closely related to the more general *free-energy principle* in theoretical neuroscience (K. Friston, Kilner, & Harrison, 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 (K. Friston, FitzGerald, Rigoli, Schwartenbeck, & Pezzulo, 2017; K. J. Friston, Parr, & de Vries, 2017). For further reviews of PC, its mathematical foundation, and applications in neuroscience, see (Bogacz, 2017; Buckley, Kim, McGregor, & Seth, 2017; Millidge, Seth, & Buckley, 2021).

Although originating in neuroscience, a body of literature has investigated how PC can be related and applied to the existing deep learning literature. 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. 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 and their training and testing dynamics, and secondly 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.

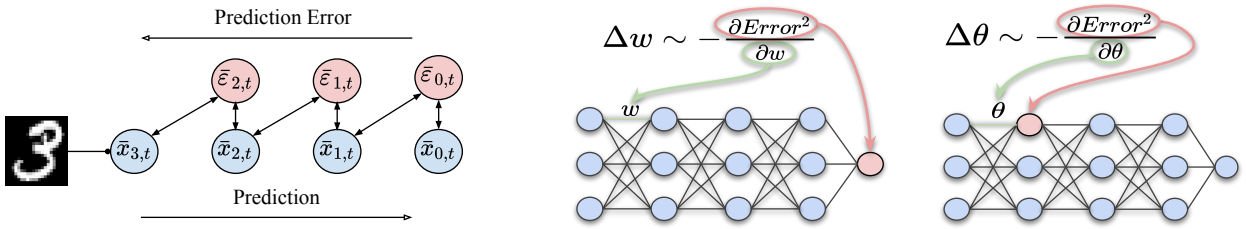
## 2 Overview of Predictive Coding

We now recall the key concepts of PC. We give an overview of PCNs, which are the PC equivalent of ANNs, and we review the interpretation of PC as variational inference, which provides additional insights into the computations and learning of 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 $\bar{\varepsilon}_t$. 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 every layer of neurons is to predict the value of the following layer according to

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




(a) Sketch of a PCN

(b) Difference between BP (left) and PC (right)

Figure 1: (a): A multilayer PCN trained on a data point of MNIST. Here, the neural activities of a specific layer predict the ones 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 weights to be updated to minimize an error that could be dozens of layers away. By contract PC minimizes a local energy function for each layer.

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:

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

**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. During the weight update, the value nodes are fixed, and the weight parameters are updated via gradient descent on the same energy function. When defining inference and weight update this way, every computation only needs local information to be updated. 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 process, and the weights are updated once upon convergence (Millidge, Tschantz, & Buckley, 2020; Salvatori, Song, Hong, et al., 2021; Whittington & Bogacz, 2017). However, in certain cases, it has been noted that updating the weights and activities of different layers in different moments yields a better performance (Han et al., 2018; Ororbia & Kifer, 2020).

### 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 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 inference problem can be solved by variational inference (Beal, 2003; Jordan, Ghahramani, Jaakkola, & Saul, 1998; Wainwright & Jordan, 2008). In general, variational inference approximates an intractable inference through an optimization problem, where the parameters of an approximate *variational posterior* distribution $q$

are 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 (K. Friston, Mattout, Trujillo-Barreto, Ashburner, & Penny, 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:

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

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,

$$d\bar{x}_l/dt \propto -\partial \mathcal{F}_t / \partial \bar{x}_{l,t} \tag{4}$$

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

PCNs 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, and then the weights $\bar{\theta}_l$ are updated for a single step given the equilibrium values of the activations $\bar{x}^*_{l,t}$. 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 during each inference phase, 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 demonstrated in recent work.

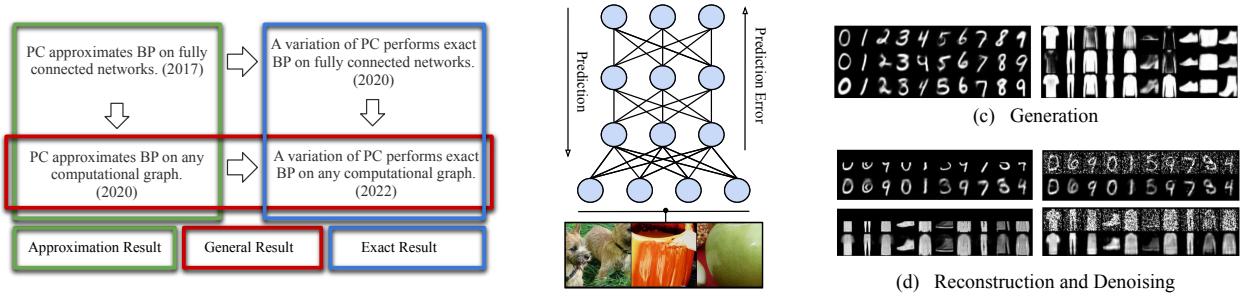
## 3 Predictive Coding and Backpropagation

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 multi-layer perceptrons (MLPs), albeit under some strict conditions (Whittington & 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, Tschantz, & Buckley, 2020). 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 to their feedforward pass values (Song, Lukasiewicz, Xu, & Bogacz, 2020). These results also extend to Z-IL being able to exactly replicate the parameter update of BP on any computational graph (Salvatori, Song, Lukasiewicz, Bogacz, & Xu, 2021). The advantage of these exactness results are twofold: first, for Z-IL, they require only a small number of time steps to perform a full update of the parameters, and empirical results show that Z-IL is almost as efficient as BP, at the cost of requiring complex control logic to synchronize parameter updates to occur at the correct times across layers. 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 & Schmidhuber, 1997), transformers (Vaswani et al., 2017), and ResNets (He, Zhang, Ren, & Sun, 2016). A historical sketch of these results is given in Fig. 2(a).

## 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 well on image classification tasks. This is indeed the case: the first formulation of PC for supervised learning, equivalent to the




(a) History of results connecting PC to BP

(b) Sketch of a Generative PCN

(c) Generation

(d) Reconstruction and Denoising

Figure 2: (a) Historical and conceptual sketch of the results unifying predictive coding (PC) and backpropagation (BP). (b) 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. (c) Examples of generated MNIST and FashionMNIST using a generative PCN. (d) Examples of reconstructed (left) and denoised (right) MNIST and FashionMNIST images using a generative PCN. Figures (c,d) are taken from the original papers, i.e., (Ororbia & Kifer, 2020) and (Salvatori et al., 2022), respectively.

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 & Bogacz, 2017). A similar result was also obtained using a variation of PC not restricted to be trained using mean squared error as an energy function. It is in fact possible to define a generalization of IL, which uses layer-specific loss functions (Ororbia & 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 using the Z-IL algorithm, 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 (K. Friston, 2005; Rao & Ballard, 1999). 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 & 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. 2(b). 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 & Kifer, 2020; Salvatori et al., 2022). Particularly, Ororbia and Kifer Ororbia and Kifer 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 (Rao & Ballard, 1999), and of a model designed by K. Friston (K. Friston, 2008). The extensive experiments show that generative PCNs are able to successfully generate novel black and white images of different datasets, as shown in Fig. 2(c). 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. The 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. 2(d). Importantly, PCNs can accomplish these task even if not directly trained to do so, unlike ANNs, which must be trained to perform each specific task (Ororbia & Kifer, 2020; Salvatori et al., 2022; Salvatori, Song, Hong, et al., 2021).

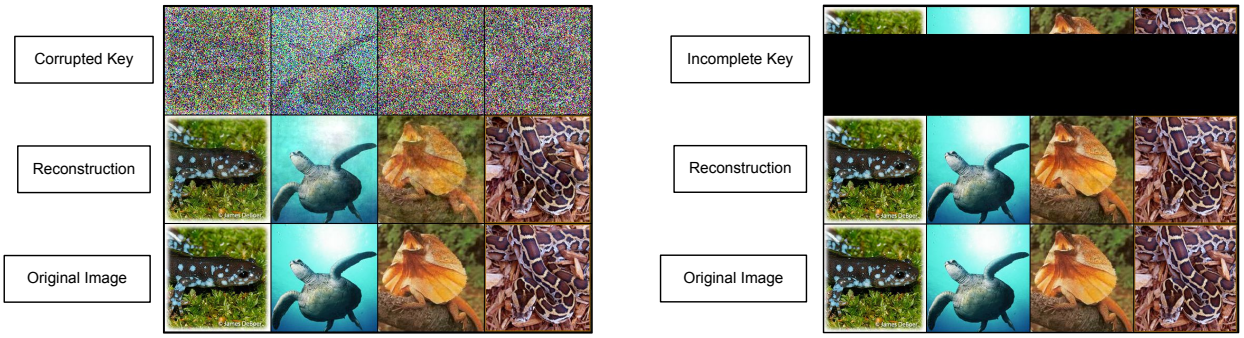



Figure 3: Examples of retrieved ImageNet pictures when presenting a corrupted key with gaussian noise of mean zero and variance $\eta = 2.0$ (left) and an incomplete key, where only $\frac{1}{4}$ of the original pixels were provided. Particularly, 100 images were stored in this example.

## 5 Associative Memories

It has also been recently demonstrated that generative PCNs also 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, Song, Hong, et al., 2021). 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 incomplete 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 for examples. 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, Belkin, & Uhler, 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, Auksztulewicz, & Friston, 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 the error 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 & Schmidhuber, 1997; Rumelhart et al., 1986; Williams & 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, Misic, & Sporns, 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, Vempala, Mitropolsky, Collins, & Maass, 2020), a Hebbian learning method that can perform different operations implicated in cognitive phenomena. However, Hebbian learning methods 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). 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.

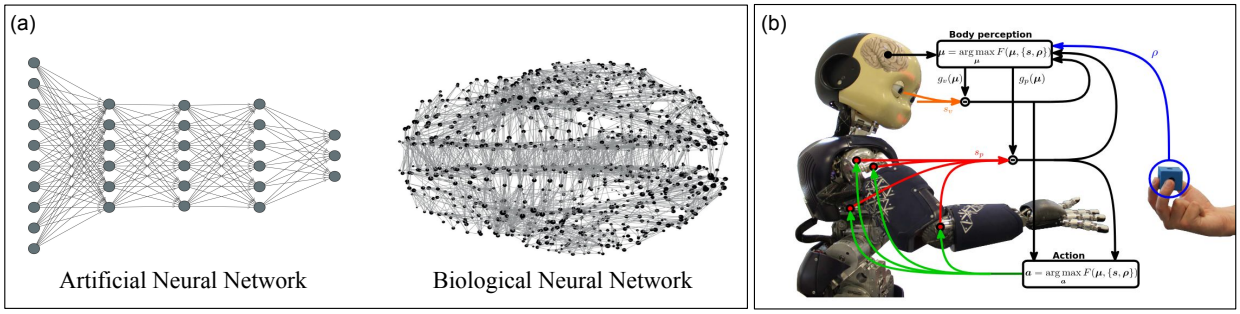



Figure 4: (a) Difference in topology between an ANN (left) and a sketch of a network of structural connections that link distinct neuronal elements in a brain (right). Figure taken from (Salvatori et al., 2022). (b) Graphical representation of the joint minimization of free energy by action and control 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).

## 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, with 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 dependence on data $\bar{x}_0$ which could depend on actions $a$. This dependency can be computed explicitly by using the chain rule (K. Friston, 2011; K. Friston, Daunizeau, & Kiebel, 2009),

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

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

PC methods 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 & Cheng, 2018). PC has thus been applied to a variety of robotics problems (Lanillos & Cheng, 2018) as well as drone and quadcopter control (Meera & Wisse, 2020, 2021). An additional advantage of PC is that it can also be used for state estimation (Oliver et al., 2021; Pezzato, Ferrari, & Corbato, 2020) (including with high dimensional image inputs (Sancaktar, van Gerven, & Lanillos, 2020), providing a joint solution of state estimation and control, as depicted in Figure 4(b). 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 to a set of desired outcomes of control, then the PC inference algorithm will infer the actions consistent with the conditioned outcomes. Explorations of this idea are presented in (Bogacz, 2020; Kinghorn, Millidge, & Buckley, 2021), and this mechanism is simply an example of the active inference (K. J. Friston et al., 2017; Millidge, Tschantz, Seth, & Buckley, 2020a) and control as inference (Attias, 2003; Levine, 2018; Toussaint & Storkey, 2006) frameworks, which interpret control problems as probabilistic inference problems, which simply require 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 both 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 comparable 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 (such as associative

memories) 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. Hence, 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, which is not shared by BP-trained ANNs, is its superior flexibility due both to its inference phase and mathematical interpretation as a generative model, and to the local computations that allow training on any graph structure. The first 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 also allows PCNs to natively handle missing data, while ANNs need heuristic imputation schemes. The second, could allow new progress in neural architecture search. Beyond this, it is worth noting that essentially all current benchmarks, layers, initializations, and other "tricks", have been heavily optimized specifically for ANNs, while no such optimization work has been performed for PCNs, thus indicating that an important line of future research will be to devise similar PCN-specific "tricks" to improve performance. This line of research is of vital importance to scale the applicability of PCNs.

An additional unique property of PCNs compared to ANNs is the locality of their weight updates and hence greater parallelizability compared to ANNs. If exploited properly, this may enable a substantially more efficient training of extremely large-scale PCN models by reducing the communication and wait time requirements induced by the sequential backward step of BP, as well as lend itself to an efficient implementation on neuromorphic hardware.

Finally, 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.

## Acknowledgments

This work was supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1, by the AXA Research Fund, the EPSRC grant EP/R013667/1, and by the EU TAILOR grant. We also acknowledge the use of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1) and GPU computing support by Scan Computers International Ltd.

## References

Attias, H. (2003). Planning by probabilistic inference. In *International workshop on artificial intelligence and statistics.*

Auksztulewicz, R., & Friston, K. (2016). Repetition suppression and its contextual determinants in predictive coding. *Cortex*.

Avena-Koenigsberger, A., Misic, B., & Sporns, O. (2018). Communication dynamics in complex brain networks. *Nature Reviews Neuroscience*, *19*(1).

Baltieri, M., & Buckley, C. (2019). PID control as a process of active inference with linear generative models. *Entropy*.

Barlow, H. (2001). Redundancy reduction revisited. *Network*.

Barron, H., Auksztulewicz, R., & Friston, K. (2020). Prediction and memory: A predictive coding account. *Progress in Neurobiology*.

Beal, M. (2003). *Variational algorithms for approximate bayesian inference*. University College London.

Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of Mathematical Psychology*.

Bogacz, R. (2020). Dopamine role in learning and action inference. *Elife*.

Bradbury, J. (2000). Linear predictive coding. *Mc G. Hill*.

Buckley, C., Kim, C. S., McGregor, S., & Seth, A. (2017). The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*.

Clark, A. (2015). *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford University Press.

Friston, K. (2003). Learning and inference in the brain. *Neural Networks*.

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

Friston, K. (2008). Hierarchical models in the brain. *PLoS Computational Biology*.

Friston, K. (2011). What is optimal about motor control? *Neuron*.

Friston, K., Daunizeau, J., & Kiebel, S. (2009). Reinforcement learning or active inference? *PloS One*.

Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., & Pezzulo, G. (2017). Active inference: A process theory. *Neural Computation*.

Friston, K., Kilner, J., & Harrison, L. (2006). A free energy principle for the brain. *Journal of Physiology*.

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

Friston, K. J., Parr, T., & de Vries, B. (2017). The graphical brain: Belief propagation and active inference. *Network Neuroscience*.

Han, K., Wen, H., Zhang, Y., Fu, D., Culurciello, E., & Liu, Z. (2018). Deep predictive coding network with local recurrent processing for object recognition. *Advances in Neural Information Processing Systems*, *31*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proc. of cvpr.*

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*.

Hohwy, J., Roepstorff, A., & Friston, K. (2008). Predictive coding explains binocular rivalry: An epistemological review. *Cognition*.

Johnson, M., & Moradi, M. (2005). *Pid control*. Springer.

Jordan, M., Ghahramani, Z., Jaakkola, T., & Saul, L. (1998). An introduction to variational methods for graphical models. In *Learning in graphical models.* Springer.

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

Kinghorn, P., Millidge, B., & Buckley, C. (2021). Habitual and reflective control in hierarchical predictive coding. *arXiv:2109.00866*.

Knill, D., & Pouget, A. (2004). The Bayesian brain: The role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*.

Lanillos, P., & Cheng, G. (2018). Adaptive robot body learning and estimation through predictive coding. In *Proc. of iros.*

Lanillos, P., Meo, C., Pezzato, C., Meera, A., Baioumy, M., Ohata, W., ... others (2021). Active inference in robotics and artificial agents: Survey and challenges. *arXiv:2112.01871*.

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

Lillicrap, T., Santoro, A., Marris, L., Akerman, C., & Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*.

Lotter, W., Kreiman, G., & Cox, D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. *arXiv:1605.08104*.

Meera, A., & Wisse, M. (2020). Free energy principle based state and input observer design for linear systems with colored noise. In *2020 american control conference (acc).*

Meera, A., & Wisse, M. (2021). A brain inspired learning algorithm for the perception of a quadrotor in wind. *arXiv:2109.11971*.

Millidge, B., Seth, A., & Buckley, C. (2021). Predictive coding: A theoretical and experimental review. *arXiv:2107.12979*.

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

Millidge, B., Tschantz, A., Seth, A., & Buckley, C. (2020a). On the relationship between active inference and control as inference. In *International workshop on active inference.*

Millidge, B., Tschantz, A., Seth, A., & Buckley, C. (2020b). Relaxing the constraints on predictive coding models. *arXiv:2010.01047*.

Movellan, J. R. (1991). Contrastive Hebbian learning in the continuous Hopfield model. In *Connectionist models* (pp. 10–17). Elsevier.

Oliver, G., Lanillos, P., & Cheng, G. (2021). An empirical study of active inference on a humanoid robot. *IEEE Transactions on Cognitive and Developmental Systems*.

Ororbia, A., & Kifer, D. (2020). The neural coding framework for learning generative models. *arXiv:2012.03405*.

Ororbia, A., & Mali, A. (2019). Biologically motivated algorithms for propagating local target representations. In *Proc. aaai.*

Papadimitriou, C., Vempala, S., Mitropolsky, D., Collins, M., & Maass, W. (2020). Brain computation by assemblies of neurons. *Proceedings of the National Academy of Sciences*.

Pezzato, C., Ferrari, R., & Corbato, C. (2020). A novel adaptive controller for robot manipulators based on active inference. *IEEE Robotics and Automation Letters*.

Radhakrishnan, A., Belkin, M., & Uhler, C. (2019). Overparameterized neural networks implement associative memory. *arXiv:1909.12362*.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., ... Hochreiter, S. (2021). Hopfield networks is all you need. In *International conference on learning representations.*

Rao, R., & Ballard, D. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*.

Salvatori, T., Pinchetti, L., Millidge, B., Song, Y., Bao, T., Bogacz, R., & Lukasiewicz, T. (2022). Learning on arbitrary

graph topologies via predictive coding. *arXiv:2201.13180*.
Salvatori, T., Song, Y., Hong, Y., Sha, L., Frieder, S., Xu, Z., ... Lukasiewicz, T. (2021). Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, *34*.
Salvatori, T., Song, Y., Lukasiewicz, T., Bogacz, R., & Xu, Z. (2021). Predictive coding can do exact backpropagation on any neural network. *arXiv:2103.04689*.
Sancaktar, C., van Gerven, M., & Lanillos, P. (2020). End-to-end pixel-based deep active inference for body perception and action. In *2020 joint ieee 10th international conference on development and learning and epigenetic robotics (icdl-epirob).*
Song, Y., Lukasiewicz, T., Xu, Z., & Bogacz, R. (2020). Can the brain do backpropagation? — Exact implementation of backpropagation in predictive coding networks. In *Advances in neural information processing systems.*
Spratling, M. W. (2017). A review of predictive co- ding algorithms. *Brain and Cognition*.
Srinivasan, M., Laughlin, S., & Dubs, A. (1982). Predictive coding: A fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences*.
Sun, W., & Orchard, J. (2020). A predicti- ve-coding network that is both discriminative and generative. *Neural Computation*.
Toussaint, M., & Storkey, A. (2006). Probabilistic inference for solving discrete and continuous state Markov decision processes. In *Proc. of icml.*
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems.*
Wainwright, M., & Jordan, M. I. (2008). *Graphical models, exponential families, and variational inference*. Now Publishers Inc.
Whittington, J. C. R., & Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*.
Williams, R. J., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, *1*(2), 270–280.