

An Analysis and Extension of Predictive Coding for Learning in the Brain



Gaspard Oliviers

Pembroke College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2026

Acknowledgments

I am deeply grateful to my supervisor, Prof. Rafal Bogacz, for his guidance throughout this PhD. His scientific approach, that simpler is better, has profoundly shaped how I think about research and problem-solving. His genuine excitement about the project has been consistently motivating. Our regular meetings and close supervision provided invaluable structure, while the freedom I was given to explore my own ideas allowed me to grow as a researcher.

My sincere thanks go to the Medical Research Council for funding my PhD. I am also grateful to Pembroke College for the additional funding and for providing accommodation throughout my studies.

I would like to thank both current and former members of the group, Mufeng, Charlotte, Yuhao, Nima, Benoit, Elene, Francesco, and Julian, for the stimulating research discussions and the many enjoyable lunch conversations. It has been encouraging to work alongside such dedicated colleagues, all striving towards ambitious goals.

To my friends, thank you for helping me maintain perspective. You have kept my mind healthy. Seeing each of you carve out your own path towards a fulfilling future has been a constant source of encouragement.

I owe a special debt of gratitude to my parents and siblings. Their unwavering support for my decision to pursue a PhD, their belief in my potential, and their drive to push me beyond my perceived limits have shaped who I am.

Finally, to Eleni: thank you for your love and steadfast support in every aspect of life. You have taught me balance and given me an appreciation of the present moment. Every moment of this PhD has been a pleasure by your side. Your careful proof-reading and your gentle but firm resistance to my procrastination have improved this thesis in countless ways. I could not have done this without you.

Statement of Intellectual Contribution

The research presented in this thesis is largely my work, with supervision from Prof. Rafał Bogacz. Co-authorship is detailed below:

- Chapter 2 draws on the publication Learning probability distributions of sensory inputs with Monte Carlo predictive coding [1], for which I am the lead author. Prof. Rafał Bogacz and Alexandre Meulemans supervised the project and contributed to the revision of the manuscript.
- Chapter 3 is based on the publication Bidirectional predictive coding [2], for which I am the lead author. Prof. Rafał Bogacz supervised the work and contributed to manuscript revisions. Mufeng Tang carried out some preliminary experiments reported in the Bidirectional Predictive Coding CCN abstract, which underpin the results presented in Section 3.4.1, and also contributed to editing the manuscript.
- Chapter 4 presents unpublished work on which I am the first author. Prof. Rafał Bogacz supervised the research and contributed to the revision of this work.

I have also contributed to other publications not included in this thesis:

- Benchmarking Predictive Coding Networks - Made Simple [3] for which I contributed to the experimentation and editing of the work.
- ePC: Fast and Deep Predictive Coding for Digital Hardware [4] for which I contributed to the experimentation and editing of the work.

Abstract

Understanding how the brain learns is a central objective of neuroscience, with important consequences for neurological disease, rehabilitation and education. Predictive coding has emerged as a compelling candidate for a unifying theory of brain function. It proposes that the brain learns by minimising the discrepancy between incoming sensory signals and its own predictions. In doing so, it offers a biologically plausible account of learning that respects three key constraints of neural systems: hierarchical organisation, local information processing and strong learning performance. Despite these strengths, predictive coding in its standard form does not fully capture the breadth of abilities exhibited by the brain within a single network. This thesis extends predictive coding along three complementary directions. First, we show how predictive coding can represent uncertainty, a capacity essential for robust perception and decision-making. We introduce Monte Carlo predictive coding, which augments predictive coding’s activity dynamics to estimate full posterior distributions while preserving local computations. Second, we demonstrate how predictive coding can support both supervised and unsupervised learning within a biologically plausible architecture that reflects computational motifs observed in visual perception. In this framework, each neural layer predicts the activity of both the layer above and the layer below in the hierarchy, enabling flexible learning across tasks. Third, we develop an analytical account explaining why predictive coding can learn with reduced interference, mirroring the brain’s ability to acquire new knowledge with minimal catastrophic forgetting. We identify the regimes in which these advantages are most pronounced and propose modifications that guarantee, or closely approximate, interference-free learning. Together, these contributions expand the functional scope of predictive coding and clarify the conditions under which its learning dynamics are both powerful and robust.

Contents

1	Introduction	1
1.1	Learning in the brain	3
1.1.1	Neural hardware	3
1.1.2	Constraints of learning in the brain	4
1.1.3	Artificial neural networks	6
1.1.4	Learning algorithms in the brain	9
1.2	Predictive coding	11
1.2.1	Perception and learning with an internal world model	12
1.2.2	Neural implementation of an internal world model	15
1.2.3	Constraints of learning in the brain	20
1.3	Thesis research and overview	22
1.3.1	Probabilistic inference and learning	22
1.3.2	Bidirectional learning	23
1.3.3	Interference-free learning	24
2	Probabilistic inference with Monte Carlo Predictive coding	25
2.1	Results	27
2.1.1	MCPC implementation with local computation and plasticity	28
2.1.2	MCPC infers posterior distributions	31
2.1.3	MCPC samples from its generative model in the absence of sensory inputs	34
2.1.4	MCPC learns efficient and generalisable generative models . .	35
2.1.5	MCPC captures the variability of cortical activity	40
2.1.6	MCPC robustly learns the data (co)variance across noise types and intensities	43
2.2	Discussion	46
2.2.1	Benefits from computational abilities of MCPC	47
2.2.2	Unified theory of cortical computation	47

2.2.3	Relationship to implementations of predictive coding	48
2.2.4	Relationship to other models introducing Langevin dynamics to brain-inspired generative models	49
2.2.5	Experimental prediction	51
2.2.6	Limitations and future work	52
2.2.6.1	Extending MCPC to learn precision weighted predic- tion errors	52
2.2.6.2	Improving the sampling speed of Langevin dynamics	52
2.2.6.3	Mapping MCPC's noise to sources of noise in the brain	53
2.3	Methods	55
2.3.1	Models	55
2.3.1.1	Monte Carlo predictive coding	55
2.3.1.2	Predictive coding	55
2.3.1.3	Deep latent Gaussian models	57
2.3.2	Learning tasks	58
2.3.2.1	Gaussian learning task	58
2.3.2.2	MNIST learning task	58
2.3.3	Measuring cortical-like properties of MCPC's neural activity .	62
2.3.3.1	Evaluating neural variability	62
2.3.3.2	Similarity of spontaneous and average evoked neural activity	62
3	Bidirectional predictive coding	65
3.1	Introduction	65
3.2	Background and related work	67
3.3	Bidirectional predictive coding	70
3.3.1	Neural implementation	71
3.3.2	Flexible learning	71
3.4	Experiments	72
3.4.1	bPC performs simultaneously well in classification and generation	72
3.4.2	bPC performs unsupervised representation and distribution learn- ing	74
3.4.3	bPC performs combined supervised and unsupervised learning	75
3.4.4	bPC's shared latent layers prevent biased or overconfident en- ergy landscape	77
3.4.5	Learning in biologically relevant tasks	79

3.4.6	Bidirectional processing in deep models	80
3.5	Discussion	82
3.5.1	Summary	82
3.5.2	Hybrid model of the visual system	82
3.5.3	Populations of error neurons	83
3.5.4	Future work	84
3.6	Methods	85
3.7	Details of experiments	85
3.7.1	Datasets	86
3.7.2	Model architectures	87
3.7.3	Training procedures	88
3.7.4	Training hyperparameters	93
3.7.5	Evaluation	94
3.8	Tabular results	102
3.9	Scaling bPC to deeper models	102
4	Optimal learning with target alignment in predictive coding	109
4.1	Introduction	109
4.2	Background	111
4.2.1	Deep linear networks	111
4.2.2	Learning with predictive coding	113
4.2.3	Target alignment of learning	114
4.3	Results	115
4.3.1	Origin of interference in BP	115
4.3.2	PC attenuates interference	117
4.3.3	Is PC's interference reduction mechanism effective?	118
4.3.4	Interference-free learning in PC with layer specific learning rates	121
4.3.5	Training on a batch of samples introduces inter-sample inter- ference	123
4.3.5.1	Change in prediction for batch training	124
4.3.5.2	Batch training causes additional interference	125
4.3.5.3	Mitigating batch inference in PC	126
4.4	Discussion	128
4.4.1	Summary	128
4.4.2	When does PC significantly outperform BP?	129

4.4.3	Benefits of natural gradients could extend to PC	129
4.4.4	Future work	130
4.4.4.1	Whole training trajectories	130
4.4.4.2	Biological plausibility	130
4.5	Methods	131
4.5.1	Training data	131
4.5.2	Training BP models	132
4.5.3	Training PC models	133
4.5.4	Mechanisms for improving alignment	134
4.5.4.1	Activity-dependent learning rates	135
4.5.4.2	Decorrelation matrix factors	136
4.5.5	Experimental model configurations	136
4.5.6	Metrics	137
4.5.6.1	Target alignment	137
4.5.6.2	Condition number	138
4.5.7	Derivation of PC's weight updates	139
4.5.8	Derivation of PC's weight updates	140
5	General Discussion	143
5.1	Summary of results	143
5.2	Does the research of the thesis meet the constraints of learning in the brain?	144
5.2.1	Monte Carlo predictive coding	144
5.2.2	Bidirectional predictive coding	145
5.2.3	Target alignment of predictive coding	146
5.3	Future work	147
5.3.1	Robust perception with predictive coding	147
5.3.2	Scaling predictive coding to brain-sized networks	148
5.4	Concluding remarks	149
A	Monte Carlo predictive coding	150
A.1	Supplementary figures	150
A.2	Supplementary videos	150
A.3	Learning in a linear model with one input neuron and one latent state using MCPC and PC	154
A.3.1	Learning with MCPC	154
A.3.2	Learning with PC	155

A.4	Predictive coding optimizes an infinitely loose bound on the marginal log-likelihood $\ln p(x; \theta)$	157
A.5	Proofs for MCPC capturing the variability of cortical activity	158
A.5.1	Neural variability decreases at stimulus onset	158
A.5.2	Natural stimuli specific increase in similarity between spontaneous and evoked neural activity	159
A.6	Hyperparameter search results	159
B	Bidirectional predictive coding	163
B.1	Other types of predictive coding models	163
B.2	Balancing discriminative and generative energies in bPC	165
B.3	Effect of max-pooling on generative artifacts	166
B.4	Sample generation of combined discBP + genBP models	167
B.5	Relationship between hybridPC and bPC	168
C	Target alignment with predictive coding	170
C.1	Experimental validation	170
	Bibliography	172

Notation Convention

Generic variables

x Sensory input, often a column vector.

y Target output, often a column vector.

z Collection of latent variables.

θ Collection of model parameters, e.g. weight matrices and biases.

Column vectors

x_l Neural activity at layer l ; $x_{l,i}$ denotes the i -th neuron in layer l .

ϵ_l Prediction error at layer l .

$\hat{y}$ Model feed forward output prediction.

r Feedforward output prediction error $y - \hat{y}$.

$\hat{x}_l$ Feed forward neural activity at layer l .

x_l^* Steady-state activity of x_l after predictive coding iterative inference.

μ Prior mean of the top latent layer x_L .

Matrices

W_l Top-down prediction weight matrix mapping from layer $l + 1$ to layer l .

V_l Bottom-up prediction weight matrix mapping from layer $l - 1$ to layer l .

I Identity matrix.

$V_{i:j}$ Products of layer matrices from layer j to i , $V_i \dots V_j$.

Constants

L Number of layers in a hierarchical model; index l runs over layers $1, \dots, L$.

α Learning rate (positive scalar).

Distributions

$p(x, z; \theta)$ Generative model with parameters θ .

$q(z)$ Approximate posterior used in free-energy expressions.

$N(\cdot; m, \Sigma)$ Gaussian distribution with mean m and covariance Σ .

$\delta(\cdot)$ Dirac delta function.

Functions

E Energy function of a predictive coding model.

$\mathcal{L}$ Energy function of model trained using backpropagation.

$F(q, \theta)$ Variational free energy.

$f(\cdot)$ Element-wise nonlinearity; $f'(\cdot)$ denotes its derivative.

Operators

Δ Difference between two timepoints of the same variable, e.g. ΔW_l weight update or $\Delta \hat{y}$ change in model prediction.

$\nabla_x f$ Derivative of function f with respect to x .

$\|\cdot\|_2^2$ Euclidean L^2 norm.

$E_p[\cdot]$ Expectation with respect to distribution p .

$\text{KL}(q||p)$ Kullback–Leibler divergence between distributions q and p .

$\text{sg}(\cdot)$ Stop-gradient.

$\cdot^\top$ Matrix transpose.

Chapter 1

Introduction

The mechanisms that enable the brain to acquire and adapt knowledge remain a fundamental question with clinical, societal and philosophical significance. Within neuroscience and medicine, mapping how the brain learns offers new avenues for therapeutic innovation. Such advances are particularly relevant for disorders such as anxiety [5] and depression [6, 7], which involve alterations to the neural mechanisms that support learning, and adaptation. They are also central to Parkinson’s disease [8], where disrupted signals lead to impairments in learning from feedback, alongside motor symptoms. Understanding how the brain learns is equally important for rehabilitation following injury or stroke [9], where functional improvements depend on the brain’s ability to reorganise through experience. At the level of everyday life, such insights can transform education by aligning teaching methods with the brain’s natural capacity for adaptation and change [10]. This knowledge can be used to cultivate healthier lives by explaining how beneficial habits are acquired and harmful ones are replaced [11]. At a broader societal and economic level, principles of biological learning can inform the development of intelligent and safe artificial agents capable of interacting effectively with the world [12]. Beyond these practical implications, learning raises fundamental philosophical questions about how knowledge shapes identity[13]. How does experience turn into knowledge? How does knowledge shape curiosity, creativity, skill and expertise? More broadly, how does the process of learning contribute to who we become over time?

Currently, the closest approximation of brain-like learning is found in artificial neural networks. Despite their remarkable successes [14], these networks still fall short of biological learning in several critical aspects. They struggle with continual learn-

ing, often failing to learn multiple tasks sequentially without overwriting previous knowledge [15]. Decision-making can lack robustness, particularly when faced with uncertainty or noisy inputs and feedback [16, 17]. Crucially, artificial neural networks offer limited insight into their own reasoning processes, making their decisions difficult to explain and interpret [18, 19]. Although there has been growing interest in unsupervised and self-supervised representation learning, artificial networks still depend heavily on vast labelled datasets [20, 21]. They learn slowly, generalise poorly from few examples [21], and consume far more energy than the brain [22]. Moreover, their architectures only loosely reflect known biological principles [23, 24, 25]. These shortcomings have motivated interest, within machine learning itself, in learning algorithms that depart from backpropagation. Beyond their value as models of the brain, biologically inspired algorithms are attractive for two practical reasons. First, by relaxing the assumptions of backpropagation, they may outperform it on specific, biologically relevant problems such as continual, online, and few-shot learning, where standard networks degrade. Second, because they rely on local computations rather than a global backward pass, they map naturally onto massively parallel and analog or neuromorphic hardware, offering a route to more energy-efficient learning at scale [26, 27].

At the same time, the fundamental principles that govern learning in the brain are not yet fully understood. The goal of this line of research is not to identify the exact mechanism the brain implements, which remains largely beyond experimental reach, but rather to understand how effective learning is at all possible within the known constraints of biology. In particular, brain computation is hierarchical and that both neural activity and synaptic plasticity rely only on locally available signals. Recent research, including the work presented in this thesis, pursues this question. In particular, predictive coding is proposed as a unifying framework that captures a broad range of learning capabilities while remaining consistent with established biological constraints.

1.1 Learning in the brain

1.1.1 Neural hardware

The most established knowledge in neuroscience concerns the brain’s basic architecture and constituent elements. A discussion of learning must first be grounded in this biological foundation.

Neurons: the basic processing unit

The human brain contains approximately 90 billion neurons [28], which form the fundamental computational elements for neural processing (Figure 1.1A). These neurons are organised into large-scale, hierarchically structured networks that coordinate information processing across different brain regions [29, 30]. Each neuron receives input signals through branching structures called dendrites. Input signals arrive as changes in membrane potential and may be excitatory or inhibitory [31]. These signals spread towards the cell body, or soma, where they are integrated [32]. Integration occurs both across space, combining inputs from different dendrites, and across time, summing inputs that arrive in close temporal proximity [31]. When the membrane potential at the axon initial segment reaches a critical threshold, the neuron generates an action potential [33]. This electrical impulse travels along the axon to communicate with other neurons.

Synapses: sites of neural communication and plasticity

Neural signals are transmitted across neurons via specialised junctions called synapses (Figure 1.1B). Two principal types of synapse exist: electrical and chemical. Electrical synapses, formed by gap junctions, allow direct bidirectional conduction of electrical current from one neuron to another [34]. Through these channels, ions pass directly between cells, enabling extremely rapid transmission and synchronisation of neural activity. Beyond signal propagation, gap junctions also support metabolic coupling and contribute to developmental processes such as cell migration, differentiation, and tissue homeostasis. Chemical synapses operate differently [35]. When an action potential reaches the presynaptic terminal, it triggers the release of neurotransmitters into the synaptic cleft. Neurotransmitters are molecules which bind to receptors on the postsynaptic neuron, producing either an increase or a decrease in membrane

potential. Unlike electrical synapses, chemical synapses are highly modifiable. By altering the strength of these connections, the brain regulates information flow between cells. This capacity for synaptic plasticity forms the basis of learning and memory.

Synaptic plasticity: the basis of learning

Chemical synapses are the principal sites of learning in the brain. They generate graded postsynaptic responses that depend on the quantity of neurotransmitter released, the type and density of receptors, and the recent activity of the synapse. This graded signalling allows fine control over the flow of information between neurons. The strength, structure, and receptor composition of synapses can be modified through processes such as long-term potentiation (LTP) and long-term depression (LTD) [36]. These mechanisms alter effective connectivity within neural circuits. Importantly, such changes depend on the activity of both the presynaptic and postsynaptic neurons (Figure 1.1C,D). In spike-timing-dependent plasticity [37], LTP is typically induced when presynaptic firing consistently precedes postsynaptic firing within a narrow temporal window of tens of milliseconds. Conversely, LTD often arises when presynaptic spikes follow postsynaptic activity. This temporal sensitivity links synaptic modification directly to patterns of neural activity.

1.1.2 Constraints of learning in the brain

Several key biological properties and constraints shape how learning takes place in the brain.

Hierarchical organisation of neural circuits

Neurons are arranged into large-scale networks with hierarchical structure. The neocortex is organised into six distinct layers, each characterised by specific cell types and connectivity patterns, suggesting structured and directional information flow [40]. Similar, hierarchical organisation is also evident in sensory systems [41]. In vision, neurons in the primary visual cortex respond to simple features such as edges and orientations, while neurons in higher areas, including V2 and V4 respond to more complex features such as shapes, objects, and faces. This progression reflects layered processing stages. Feedforward signals typically arise earlier than feedback signals, indicating temporally ordered processing across hierarchical levels [42]. Any model of

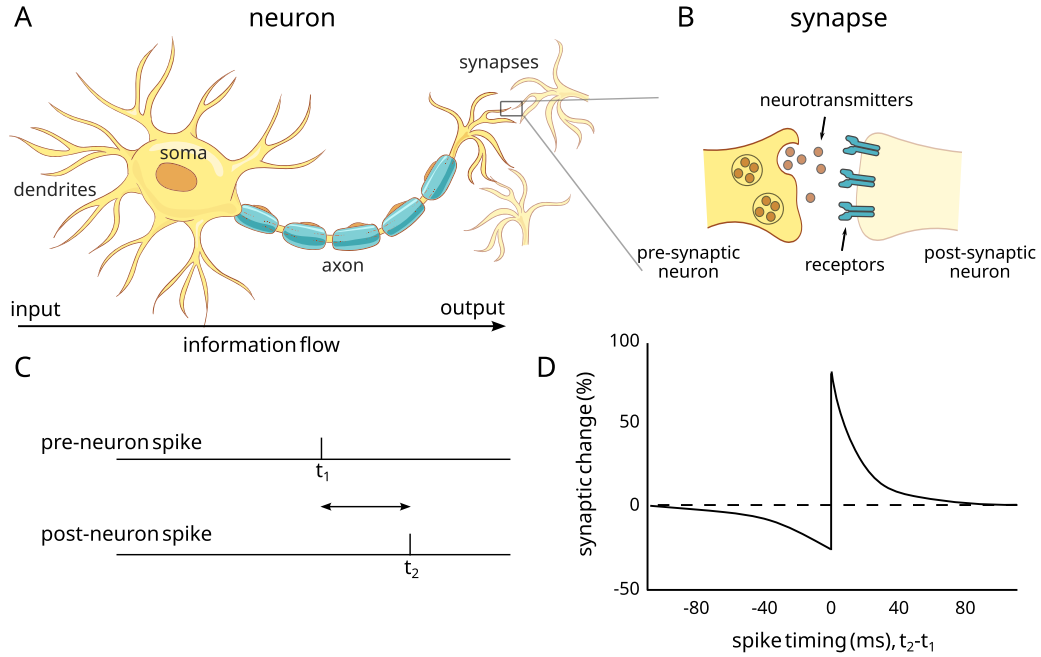


Figure 1.1: **Neurons learn through local information.** A: Anatomy of a neuron. B: Anatomy of a synapse. C and D: change in synaptic connectivity strength depending on the timing of spiking in pre- and postsynaptic neurons. D: is recreated from Bi and Poo [38]. Neuron illustration in A is adapted from NIAID BIOART [39].

brain-like learning should therefore incorporate hierarchical organisation and multi-level processing.

Local computation of activity and plasticity

Neural computation is fundamentally local. A neuron’s activity is determined by the inputs it receives through its synapses. Likewise, changes in synaptic strength depend on the activity of the presynaptic and postsynaptic neurons, as described earlier in the context of activity-dependent plasticity. Learning in the brain thus relies on local signals rather than global error broadcasts or non-local weight updates. A biologically grounded model of learning must therefore implement both activity and plasticity through local computations at the level of individual neurons and synapses.

Capacity for rapid and complex learning

The brain performs highly complex tasks, often with remarkable speed. Human learning spans perception, motor control, and abstract reasoning. An example of this property is language acquisition [43]. Infants learn to discriminate speech sounds, build vocabulary, and infer grammatical structure within a few years, largely without

explicit supervision. Another example is visual object recognition [41]. Humans can recognise faces and objects within fractions of a second, despite variation in lighting, viewpoint, or partial occlusion. Motor skills such as riding a bicycle, playing an instrument, or refining a golf swing can be acquired relatively quickly and improved through practice [44]. Humans also adapt efficiently to new environments [45], for example when adjusting to a different vehicle or navigating an unfamiliar city. A biologically plausible model of learning must therefore support the rapid acquisition of complex, structured behaviours across diverse domains.

1.1.3 Artificial neural networks

Backpropagation [46] has emerged as the most effective method for training artificial neural networks. Combined with large-scale models and substantial datasets, it achieves impressive results on demanding tasks. In vision, deep convolutional networks such as AlexNet demonstrated that neural networks trained with backpropagation could classify images into 1,000 categories with high accuracy, marking a major advance in large-scale image recognition [47]. In language, large language models trained using next-word prediction objectives generate coherent and contextually appropriate text [48]. By learning statistical structure from large text datasets, these systems can produce fluent speech, answer questions, and perform a wide variety of linguistic tasks. In planning and decision-making, systems such as AlphaGo and AlphaZero combine neural networks trained with backpropagation and reinforcement learning to achieve superhuman performance in games such as Go and chess [49]. These tasks require long-term planning and strategic evaluation of future outcomes and demonstrate the power of backpropagation-based learning in artificial systems. To understand the strengths and limitations of this approach, it is necessary to examine how artificial neural networks are designed and trained.

Artificial neural networks are loosely inspired by biological neurons. An artificial neuron receives a set of inputs, analogous to dendritic signals. These inputs are summed, similar to integration at the soma. The result is then passed through a non-linear activation function, which plays a role comparable to the thresholding process that determines whether a biological neuron emits an action potential. The neuron's output is transmitted to other units by multiplying its activity by adjustable weights, resembling the variable strength of chemical synapses. Artificial neurons are also arranged into layers, forming hierarchical networks. Each neuron in a given layer

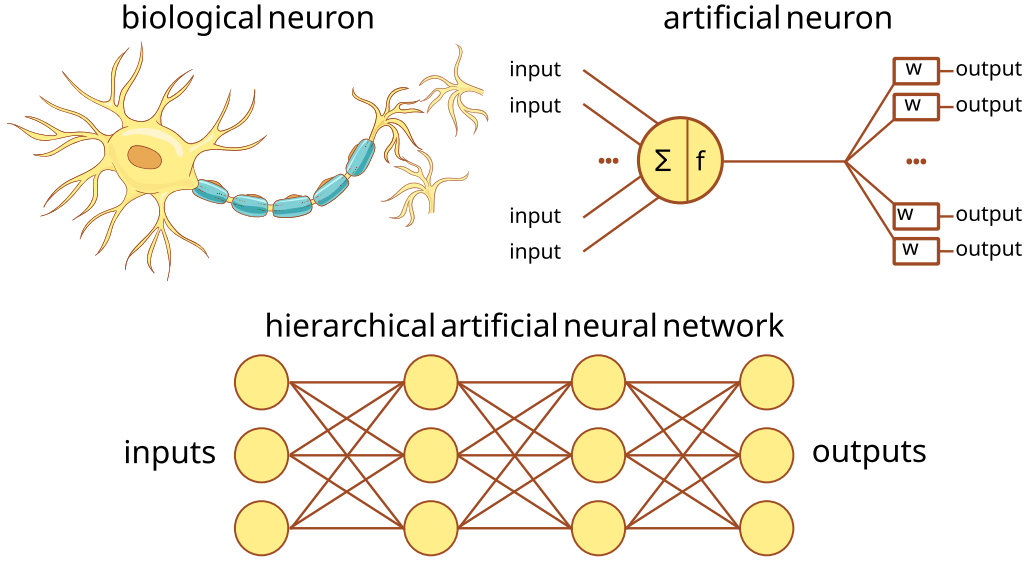


Figure 1.2: **From biological neurons to artificial neural networks.** Schematic comparison between a biological neuron and an artificial neuron. In artificial neurons, inputs are summed, analogous to integration at the soma, passed through a non-linear activation function, equivalent to thresholding to generate action potentials, and transmitted to other units via adjustable weights, resembling the variable strength of synapses. Artificial neurons are also combined in hierarchical networks. Neuron illustration is adapted from NIAID BIOART [39]

receives inputs from the previous layer and sends its output to the next. This layered structure enables progressively more abstract representations to be formed across the network.

Mathematically, the layered computation performed by an artificial neuron is given by:

$$x_{l,i} = f\left(\sum_j w_{l,ij}x_{l-1,j}\right)$$

where $x_{l,i}$ denotes the activity of the i -th neuron in the layer l , $f(\cdot)$ denotes an activation function, j indexes the neurons in layer $l-1$, and $w_{l,ij}$ denotes the synaptic weight between neuron j of layer $l-1$ and neuron i of layer l . For ease of notation, the expression can be rewritten by concatenating the activity of neurons in the same layer into a vector, x_{l-1} , and combining the weights between layers into a matrix, W_l .

$$x_{l-1} = \begin{pmatrix} x_{l-1,1} \\ x_{l-1,2} \\ \vdots \\ x_{l-1,n_{l-1}} \end{pmatrix}, \quad W_l = \begin{pmatrix} w_{l,11} & w_{l,12} & \cdots & w_{l,1n_{l-1}} \\ w_{l,21} & w_{l,22} & \cdots & w_{l,2n_{l-1}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{l,n_l1} & w_{l,n_l2} & \cdots & w_{l,n_ln_{l-1}} \end{pmatrix}$$

We can therefore rewrite the neural network's computation as:

$$x_l = f(W_l x_{l-1})$$

where x_l and x_{l-1} are vectors of the activity in layer l and $l-1$ and W_l is the weight matrix between layer l and $l-1$, $f(\cdot)$ is applied elementwise. The computation of each neuron's activity depends only on its incoming inputs and associated weights, and is therefore local.

When several layers are composed into a hierarchical network, the activity propagates sequentially from the input to the output layer. The prediction at the final layer L is given by:

$$\hat{y} = x_L = f(W_L f(W_{L-1} f(\dots W_2 f(W_1 x))))$$

where x is the input to the model, and $\hat{y}$ denotes the output of the model. This architecture is commonly referred to as a multi-layer perceptron (MLP). Although other architectures are used depending on the task [50], MLPs serves as the primary models in the experiments presented in this thesis.

To train a model to produce a desired target output, y , for a given input, x , the weights are adjusted using backpropagation. Learning proceeds by minimising an objective function, typically via gradient descent. A common choice is the mean squared error loss, given by:

$$\mathcal{L}_{BP}(x, y, W) = \frac{1}{2} \|y - \hat{y}\|_2^2 \quad (1.1)$$

where W denotes the weight matrices across layers from W_1 to W_L . The weights at each layer are updated according to:

$$\Delta W_l = -\alpha \nabla_{W_l} \mathcal{L}_{BP} \quad (1.2)$$

where the derivative can be calculated using the chain rule of derivatives, and α is a learning rate, controlling the step size of the update.

A key limitation of backpropagation is that it does not satisfy the locality constraints observed in biological learning. This becomes clear in the case of a linear network, where the weight update for layer l can be written as:

$$\Delta W_l = -\alpha \nabla_{W_l} \mathcal{L}_{BP} = W_{l+1}^\top \dots W_L^\top (y - \hat{y}) x_{l-1}^\top \quad (1.3)$$

Although the update depends on the presynaptic activity, x_{l-1} , it is also modulated by the output error $(y - \hat{y})$ and by the weights of all higher layers through the

product $W_{l+1}^\top \dots W_L^\top$. Thus, the change to a synapse at layer l requires information about the global output error and parameters elsewhere in the network. The update rule therefore relies on non-local information and cannot be implemented using only signals available at the synapse. The same conclusion holds for networks with non-linear activation functions, where gradients are still computed through the backward propagation of error signals across layers.

In summary, while backpropagation trains hierarchical models whose forward computations are local, the synaptic updates it prescribes are fundamentally non-local.

1.1.4 Learning algorithms in the brain

Local plasticity rules

Early theories of learning in the brain are rooted in Hebbian plasticity. Hebbian learning proposes that synaptic change depends on the joint activity of presynaptic and postsynaptic neurons [51]. When two neurons are active together, the connection between them is strengthened. Variants such as BCM, Oja’s rule, and spike-timing-dependent plasticity refine this principle. The BCM rule introduces a sliding modification threshold governed by historical averages of postsynaptic activity that homeostatically regulates whether activity leads to potentiation or depression [52]. Oja’s rule adds a normalisation term to stabilise synaptic growth [53]. Spike-timing-dependent plasticity incorporates temporal asymmetry, such that the precise timing between pre- and postsynaptic spikes determines whether synapses are strengthened or weakened [38]. These rules are biologically grounded and strictly local, relying only on signals available at the synapse. However, they do not provide an explicit mechanism for hierarchical credit assignment. In particular, they lack a way to propagate task-level error information through multi-layer circuits. Consequently, on their own, they cannot explain how deep cortical networks learn complex, goal-directed behaviours.

Global error signals carried by neuromodulators

An alternative line of work proposes that learning in the brain may rely on global error or reinforcement signals, potentially mediated by neuromodulatory systems such as dopamine. Synaptic plasticity remains local, but is modulated by a broadcast signal reflecting reward or performance. Early work by Mazzoni et al. [54] demonstrated that

neural networks could acquire biologically relevant properties using reinforcement-based learning rules, without requiring backpropagation. Williams [55] introduced policy gradient methods that adjust weights in the direction of the gradient of expected reinforcement. Seung [56] proposed that hedonistic synapses can approximate the gradient of average reward through stochastic synaptic modifications. Similarly, Unnikrishnan and Venugopal [57] developed rules based on local correlations between weight changes and variations in global error measure.

These strategies retain biological realism by combining local synaptic plasticity with a global modulatory drive. Nevertheless, reinforcement-driven approaches are known to converge slowly and to struggle as network size grows [58]. As a result, they do not adequately account for the speed and scalability observed in either biological systems or contemporary deep learning models. Augmenting the global reinforcement signal with an attentional feedback pathway from the output layer alleviates some of the scaling difficulties, yet this solution remains limited to one-hot output representations [59].

Can the brain approximate backpropagation?

A central question is whether cortical circuits can implement a biologically plausible approximation to backpropagation.

One proposal is contrastive learning, in which error signals are encoded in changes of neural activity across time [60]. Learning is divided into two phases: an initial prediction phase, during which synapses undergo an anti-Hebbian update, and a subsequent target phase, during which Hebbian plasticity occurs after the desired output is imposed. The difference between these two activity states approximates an error signal. A key challenge of this algorithm is that it requires a mechanism to signal which learning phase the network is in. While brain rhythms have been proposed as a coordinating mechanism [61], clear biological evidence for such structured phase control remains limited. Continuous-update variants [62] remove this strict separation into two phases, but still require a signal indicating when learning should occur.

An alternative method is equilibrium propagation [63]. In a first phase, the network settles to a fixed point corresponding to its prediction. In a second phase, a small perturbation nudges the output towards the target, and the resulting activity differences propagate naturally through the same recurrent dynamics. This method avoids a distinct error-computing pathway and is more biologically plausible than standard

backpropagation because inference and credit assignment rely on identical neural dynamics and local synaptic updates. Competitive performance has been demonstrated on benchmarks such as CIFAR-10 [64]. However, classical equilibrium propagation still depends on two distinct phases, although recent work has begun exploring phase-free variants [65, 66].

Another approach focuses on dendritic computation in pyramidal neurons. It has been proposed that apical dendrites receive feedback signals from higher cortical areas, which are compared with locally generated predictions via interneuron circuits [67, 68]. In the dendritic error model [24], this comparison generates an error signal that drives local synaptic plasticity, thereby approximating backpropagation while preserving locality of updates. Although biologically grounded, this framework relies on specific connectivity assumptions and more complex compartmental dynamics. Moreover, most demonstrations have been limited to shallow networks with one or two hidden layers.

More recently, models such as Greedy et al. [69] have proposed that error signals are represented in distal dendrites and combined with burst-dependent plasticity to approximate backpropagation. These models propagate errors through deep networks within a single learning phase, avoiding explicit separation between prediction and teaching states. Under certain conditions, they approximate backpropagation-derived gradients and can learn tasks such as MNIST and CIFAR-10 while remaining consistent with known cortical motifs. This method, has not yet been demonstrated to scale to very deep architectures, with existing results limited to networks of around five layers.

Together, these approaches suggest that biologically plausible approximations to backpropagation may be achievable. However, each introduces trade-offs in terms of architectural assumptions, learning phases, or scalability, and a fully satisfactory account of deep, efficient, and local learning in the brain remains an open challenge.

1.2 Predictive coding

Predictive coding is a different theoretical framework that aims to explain how the brain acquires and updates knowledge through experience. At its core, predictive coding proposes that the brain continually minimises the error between actual sensory

inputs and the sensory inputs the brain predicts [70]. Unlike the previously discussed approaches, predictive coding satisfies the three key criteria of learning in the brain: it is inherently hierarchical, it relies on strictly local learning, and it supports effective learning across a wide range of tasks. Moreover, predictive coding has proved highly successful as a comprehensive theoretical framework for modelling brain function. Historically, predictive coding was first developed in the context of visual processing, where it offered principled explanations for phenomena such as extra-classical receptive field effects [70] and the emergence of Gabor-like filters in early visual cortex [71]. Since then, its scope has broadened considerably. Predictive coding has been used to model audition [72], motor control [73, 74], and attention [75]. Predictive coding has also been used to shed light on certain neurological and psychiatric conditions [76].

This section is organised into three parts. The first part introduces the core hypothesis of predictive coding, which postulates that the brain learns an internal model of how its environment produces sensory inputs. By inverting this model, the brain can determine the current state of its environment from incoming sensory activity to produce optimal decision-making [77]. The second part shows how predictive coding provides a concrete neural implementation in a hierarchical network in which both neuronal dynamics and synaptic updates are local. The final part offers a critical assessment of predictive coding, evaluating the extent to which it satisfies the biological and computational criteria for learning in the brain outlined in the previous section.

1.2.1 Perception and learning with an internal world model

The brain does not have direct access to the world itself, but only to the sensory signals it produces. To make sense of these signals, the brain must rely on an inference mechanism to understand the underlying causes of sensory stimuli. An internal world model makes this possible. It encodes the relationship between events in the world and the patterns they produce in perception. For example, if it is raining (event in the world), you are likely to observe dark clouds, hear thunder and smell petrichor (sensory patterns). By inverting the internal model, the brain can use sensory inputs to infer their probable cause. Thus, when you next notice darkening clouds or related signs, you can conclude that rain is imminent and act accordingly by reaching for an umbrella (Figure 1.3A). Importantly, such a model enables prior knowledge to be integrated optimally with incoming evidence, supporting stable perception and

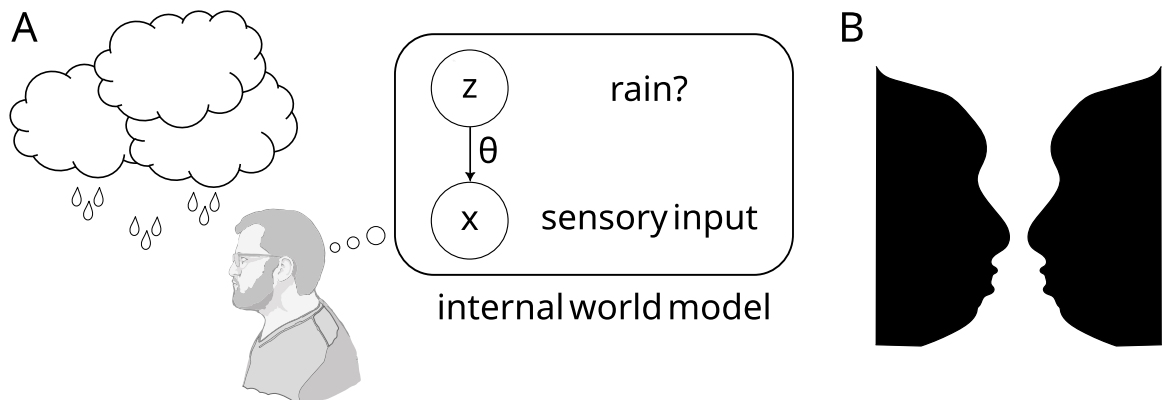


Figure 1.3: **Bayesian brain combines prior information with sensory stimuli to infer the state of the environment.** A: An internal world model relates hidden causes in the environment to sensory stimuli. The causes are hidden because the brain can not sense them directly and needs to infer them from its stimuli. B: Visual illusions where two likely interpretations, a chandelier or two faces, of the stimuli compete, causing perception to switch between the interpretations. Head side view illustration in A is adapted from NIAID NIH BIOART [93].

adaptive behaviour even under noisy or ambiguous conditions. For example, a subtle shift in the colour of the sky might be inconclusive on its own. However, knowing that it is February in the United Kingdom and that it has rained throughout the day increases the likelihood that precipitation will continue, prompting you to seek shelter. This broader hypothesis that the brain learns an internal world model is often referred to as the Bayesian brain hypothesis. This hypothesis was first proposed by Helmholtz [78] and later formalised in computational neuroscience [79, 80, 81]. The Bayesian perspective has been used successfully to explain perception and motor behaviour [82, 83, 84, 85, 86, 87]. For example, multistable perception of visual illusions can be understood as the brain identifying several plausible interpretations for an image from prior expectations (Figure 1.3B). More generally, probabilistic accounts of neural computation have also provided compelling interpretations of neural activity [88, 89, 90, 91, 92].

To combine prior knowledge with incoming sensory information, an internal model of the world assigns probabilities to sensory inputs, x , by relating them to hidden environmental states, z . To achieve this, it uses two probability distributions that must be jointly learned by the brain. The first is the prior distribution, which represents how likely different environmental states are before any sensory input is observed. The second is the likelihood function, which specifies how probable a particular sensory input x is, given a specific environmental state z . Together, these define the internal

generative model:

$$p(x, z; \theta) = \underbrace{p(x|z; \theta)}_{\text{likelihood}} \underbrace{p(z; \theta)}_{\text{prior}},$$

where θ denotes model parameters.

Free energy as a tractable alternative to model evidence

The model parameters must be learned, to obtain an accurate representation of the environment. A well-adapted internal model assigns high probability to the sensory inputs it encounters regularly and low probability to rare inputs. For this reason, the brain should learn parameters that maximise the model evidence:

$$p(x; \theta) = \int p(x, z; \theta) dz,$$

This quantity is obtained by integrating over the states z . This integration is necessary because multiple hidden environmental states can give rise to the same sensory input. Therefore, the brain must evaluate how well its model explains the observed input after accounting for uncertainty about the underlying causes. An equivalent learning objective is to minimise negative log-likelihood $-\ln p(x; \theta)$, more commonly used in the Bayesian literature as an optimisation objective [94].

However, computing the model evidence is generally intractable because it requires integrating over all possible latent states z . Instead, the brain can minimise an upper bound on the negative log-likelihood known as the free energy $\mathcal{F}$ [95, 96] defined as:

$$\begin{aligned} \mathcal{F}(q, \theta) &= -E_{q(z)}[\ln p(x, z; \theta)] + E_{q(z)}[\ln q(z)] \\ &= -\ln p(x; \theta) + \text{KL}(q(z) \parallel p(z|x; \theta)). \end{aligned} \tag{1.4}$$

Here, $q(z)$ is an approximate posterior of the posterior distribution $p(z|x; \theta)$, and $E_{q(z)}$ denotes the expectation with respect to $q(z)$. The term $\text{KL}(q(z) \parallel p(z|x; \theta))$ is the Kullback–Leibler divergence between the approximate posterior $q(z)$ and the true posterior $p(z|x; \theta)$ [97]. This divergence is always non-negative and equals zero only when the two distributions are identical. Consequently, the free energy equals the negative log-likelihood when $q(z) = p(z|x; \theta)$, and is strictly greater otherwise. The advantage of this formulation is that the free energy depends only on the brain’s internal model $p(x, z; \theta)$ and a tractable approximate posterior $q(z)$. By choosing $q(z)$ such that the expectations are computable, the brain can evaluate and minimise free energy even when the true model evidence is intractable.

Perception and learning through free energy minimisation

The brain can learn optimal model parameters by minimising the free energy, as a tractable alternative to the negative log-likelihood, using a two-step process called expectation maximisation [98]. The first step minimises the free energy with respect to the approximate posterior $q(z)$, while keeping the model parameters fixed. This step allows the brain to perform perception: it infers which hidden states of the environment are likely to have caused a sensory input. This is crucial because directly computing the true posterior distribution of environmental states is typically intractable. Instead, free energy minimisation provides a practical and tractable way to approximate the posterior. More formally, this step reduces the divergence $\text{KL}(q(z) \parallel p(z|x; \theta))$, thereby pushing the approximate posterior $q(z)$ closer to the true posterior $p(z|x; \theta)$. The second step minimises the free energy with respect to the model parameters while keeping the approximate posterior $q(z)$ fixed. This minimisation drives the model parameters toward values that better account for the sensory inputs under the current approximate posterior. The optimisation guarantees that the upper bound on the negative log likelihood $-\ln p(x; \theta)$ decreases, improving how well the internal model of the brain accounts for the probability of sensory inputs.

1.2.2 Neural implementation of an internal world model

Predictive coding posits that the brain implements an internal model of the world via a hierarchical neural network, in which local interactions govern both neural activity and synaptic plasticity [99]. We have already seen that free energy provides a manageable objective that the brain could, in principle, use to learn and carry out perception with an internal world model. However, for this account to align with biological evidence, the minimisation of free energy must be achieved using neurons whose activity depends only on their inputs, and whose synaptic modifications rely solely on local information. To address this, predictive coding specifies a particular form of internal model and a particular choice of approximate posterior distribution, structured so that each neuron can minimise free energy through purely local computations.

The assumptions of predictive coding

The world model assumed in predictive coding is a hierarchical Gaussian generative

model. This model represents environmental states as a hierarchy in which higher, more abstract variables predict states at the level below, ultimately generating predictions of sensory inputs. This structure reflects the causal organisation of the environment itself, where sensory inputs arise from a hierarchy of underlying causes. To illustrate this idea, we can re-consider the example of rain. A highly abstract cause might be the time of year (e.g. February), which increases the likelihood of low-pressure weather systems. Such systems are associated with clouds, which in turn give rise to rain droplets that may fall on the skin and be sensed. Thus, sensory data can be understood as the consequence of progressively more concrete causes descending a hierarchical chain. Formally, the internal generative model in predictive coding is defined as

$$p(x, z; \theta) = \mathcal{N}(x; W_1 f(z_2), I) \prod_{l=2}^{L-1} \mathcal{N}(z_l; W_l f(z_{l+1}), I) \mathcal{N}(z_L; \mu, I),$$

where the latent variables z range from the most abstract causes z_L to lower-level causes z_2 . The model parameters θ include the weight matrices W_1 to W_{L-1} , which map predictions between adjacent levels of the hierarchy, and the bias term μ , which specifies a prior mean for the highest-level latent variable z_L . The function $f(\cdot)$ denotes an element-wise non-linearity, and $\mathcal{N}(\cdot; m, \Sigma)$ denotes a Gaussian distribution with mean m and covariance Σ . For simplicity, identity covariance matrices I are assumed, as is common in practice [3, 100]. More generally, each layer may instead be equipped with diagonal or full covariance matrices without altering the derivation or the main conclusions [101, 99, 102].

In predictive coding, $q(z)$ is assumed to be a Dirac delta function. Rather than representing a full probability distribution, the posterior is reduced to a single point estimate. This point can be interpreted as the most probable value of the hidden state, corresponding to maximum a posteriori (MAP) inference under the assumed generative model. Formally,

$$q(z_l) = \delta(z_l - x_l),$$

where x_l denotes the neural activity at layer l . The variable x_l thus specifies the value at which the approximate posterior over the latent variable z_l concentrates all its probability mass. In other words, neural activity parametrises the approximate posterior $q(z)$ in predictive coding, and it encodes the current estimate of the most likely state of the environment.

Free energy as the sum of prediction errors

Together, these assumptions reduce the free energy to a sum of local objectives that all neurons can minimise independently. Specifically, the free energy reduces to a sum of local prediction errors between the neurons' actual activities and the activity predicted by the neurons in the layer above. This contrasts with backpropagation, which has a global objective function that prevents local learning. This result is obtained by using predictive coding's specified distributions into the expression for free energy given in equation (1.4) to obtain

$$\begin{aligned}
\mathcal{F}(q, \theta) &= -E_q[\ln p(x, z; \theta)] + E_q[\ln q(z)] \\
&= -\ln \left(\mathcal{N}(x; W_1 f(x_2), I) \prod_{l=2}^{L-1} \mathcal{N}(x_l; W_l f(x_{l+1}), I) \mathcal{N}(x_L; \mu, I) \right) + c \\
&= \frac{1}{2} \sum_{l=1}^{L-1} \|x_l - W_l f(x_{l+1})\|_2^2 + \frac{1}{2} \|x_L - \mu\|_2^2 + c,
\end{aligned} \tag{1.5}$$

where x_1 is fixed to the sensory input x , and c collects all terms that do not depend on the parameters θ or the neural activities x_l . This simplification follows because (i) expectations under a Dirac delta reduce to evaluating the integrand at its point mass, and (ii) the entropy term $E_q[\ln q(z)]$ is constant for a Dirac delta distribution and can therefore be absorbed into c . In this form, the free energy is entirely expressed as a sum of squared prediction errors: each term measures the mismatch between a neuron's activity x_l and the top-down prediction $W_l f(x_{l+1})$ generated by the layer above. In this thesis, the free energy objective minimised by predictive coding is denoted as E_{gen} and is obtained by omitting the constant term:

$$E_{gen} = \frac{1}{2} \sum_{l=1}^{L-1} \|x_l - W_l f(x_{l+1})\|_2^2 + \frac{1}{2} \|x_L - \mu\|_2^2. \tag{1.6}$$

We use the subscript *gen* to differentiate predictive coding with top-down predictions from another form of predictive coding with bottom-up predictions introduced later.

Perception and learning with local computations

The two-step minimisation of free energy required for learning can be implemented through local computations in a neural network. As discussed earlier, the model parameters must be learned so that the world model accurately captures the latent causes underlying the sensory data. In predictive coding, this learning is achieved by minimising the free energy via two optimisation steps, both of which can be carried out locally in the neural network shown in Figure 1.4. In this network, each layer

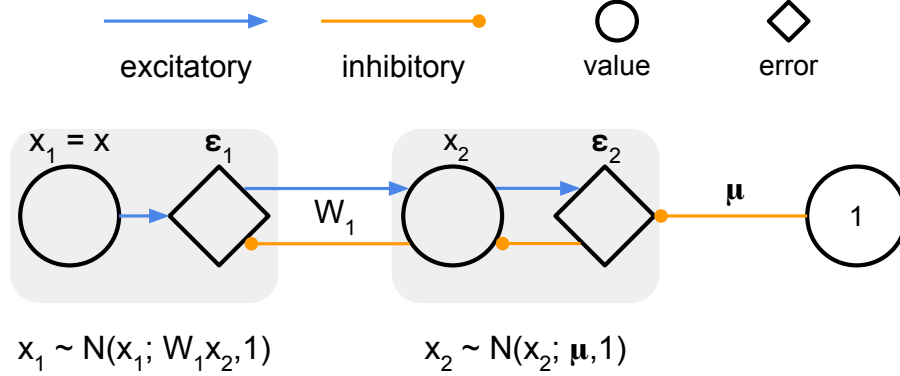


Figure 1.4: **Example neural implementation of predictive coding.** The neural implementation of predictive coding using local synaptic connections for a linear Gaussian model with one sensory input and one latent state.

comprises two neural populations: value neurons encoding the activities x_l , and error neurons encoding prediction errors given by

$$\epsilon_l = x_l - W_l f(x_{l+1}), \quad \epsilon_L = x_L - \mu.$$

The network performs the first optimisation step, corresponding to perception, using local activity dynamics. In this step, the free energy is minimised with respect to $q(z)$, while keeping the model parameters fixed. Because predictive coding represents the approximate posterior through neural activities, this corresponds to updating the activities to reduce free energy. This minimisation is implemented by neural dynamics that perform gradient descent on the predictive coding energy until convergence. For layers $l = 2, \dots, L$,

$$\frac{\partial x_l}{\partial t} \propto -\nabla_{x_l} E_{gen} = -\epsilon_l + f'(x_l) W_{l-1}^\top \epsilon_{l-1}. \quad (1.7)$$

These dynamics are local because they depend only on the activities of neighbouring neurons and the synaptic connections between them. Error neurons receive excitatory input from their corresponding value neurons and inhibitory top-down input reflecting predictions from the layer above, consistent with equation (1.2.2). Value neurons, in turn, receive excitatory bottom-up input from error neurons in the layer below, as well as inhibitory input from error neurons within the same layer, consistent with equation (1.7). After the neural activity has converged, the network updates the model parameters using local computations. In this second step, the free energy is minimised with respect to the model parameters while holding $q(z)$ fixed, which corresponds to keeping the activities x_l fixed. A single gradient-descent step yields

the synaptic update rule

$$\Delta W_l = -\alpha \nabla_{W_l} E_{gen} = \alpha \epsilon_l f(x_{l+1})^\top, \quad (1.8)$$

and the update for the top-level prior mean

$$\Delta \mu = -\alpha \nabla_\mu E_{gen} = \alpha \epsilon_L, \quad (1.9)$$

where $\alpha > 0$ is a learning rate. This form of synaptic plasticity is Hebbian [51], as it depends only on locally available pre- and post-synaptic signals. Each weight update is proportional to the product of the presynaptic activity $f(x_{l+1})$ and the postsynaptic error ϵ_l . This strict locality makes predictive coding consistent with biological constraints on cortical computation and plasticity.

For simplicity, Figure 1.4 illustrates a minimal architecture and does not explicitly depict the role of non-linearities. However, the same locality principles extend to deep non-linear networks with multiple neurons per layer, as well as to formulations in which layer-wise covariances are learned, as discussed by Bogacz [99].

An alternative bottom-up world model

The formulation of predictive coding presented so far assumes top-down predictions, in which abstract latent causes generate predictions of lower-level sensory inputs. However, this is not the only possible arrangement. In Chapters 3 and 4 of this thesis, we also consider an alternative formulation based on bottom-up predictions, in which activity at level $l - 1$ predicts the latent state at level l [103]. This bottom-up formulation achieves substantially improved learning performance on discriminative tasks such as classification, while maintaining predictive coding’s local neural implementation [103, 3]. Furthermore, it establishes a clear conceptual relationship between predictive coding and backpropagation. In particular, predictive coding approximates backpropagation when the output neurons are weakly nudged towards their target values [103]. This correspondence extends beyond simple multilayer networks to arbitrary computational graphs [104]. Moreover, Innocenti et al. [100] showed that, in a linear model, learning with predictive coding is equivalent to backpropagation on a modified mean-squared-error objective that includes a weight-dependent rescaling.

Throughout this thesis, the formulations are distinguished by two notation conventions. First, bottom-up prediction weights are denoted by V_l , whereas top-down

prediction weights are denoted by W_l . Following this notation, the world model for predictive coding with bottom-up predictions is given by

$$p(x, z; \theta) = \prod_{l=3}^L N(z_l; V_{l-1}f(z_{l-1}), I) N(z_2; V_1f(x), I), \quad (1.10)$$

Second, we denote the energy function for predictive coding with bottom-up predictions as E_{disc} , whereas the formulation with top-down predictions is denoted as E_{gen} . The learning objective E_{disc} , given by the predictive coding free energy, takes the form

$$E_{\text{disc}} = \frac{1}{2} \sum_{l=2}^L \|x_l - V_{l-1}f(x_{l-1})\|_2^2, \quad (1.11)$$

which can be derived analogously to the top-down case presented in the previous section. As before, the sensory layer is clamped to the input, $x_1 = x$. In supervised settings, the top layer is additionally clamped to a desired target activity, $x_L = y$. Learning in this formulation can likewise be implemented through local neural dynamics for inference and local synaptic plasticity for parameter updates, following the same derivational steps as described earlier.

1.2.3 Constraints of learning in the brain

Having introduced predictive coding as an implementation of the Bayesian brain, we now assess whether it satisfies the key biological constraints outlined in Section 1.1.2.

Constraint 1: Hierarchical organisation of neural circuits

Cortical computations are organised hierarchically, and any candidate learning mechanism must scale to multi-level processing. Predictive coding learn hierarchical neural networks that meet the hierarchy constraint. Moreover, stable learning has been reported in predictive coding architectures with over one hundred layers [105], suggesting that predictive coding can scale to very deep hierarchies.

Constraint 2: Local computation of activity and plasticity

Neural learning is constrained by locality, meaning that synaptic updates must be computed from pre- and postsynaptic activity available at the synapse. Predictive coding provides a local implementation of both inference and learning [101, 106, 99]. In standard formulations, plasticity depends on locally available activity and

prediction-error signals, addressing a central limitation of backpropagation, whose updates require non-local information.

Several practical concerns on the locality of predictive coding are often raised. First, predictive coding circuits (see Figure 1.4) assume shared weights between value and error pathways. For instance, W_l connects x_l and ϵ_{l-1} , and its transpose $W_l^\top$ connects ϵ_{l-1} and x_l . Non-local weight sharing is often referred to as the weight transport problem [107]. While sharing weight across synaptic connections is not fully local, in predictive coding these connections are spatially localised and involve the same pre- and postsynaptic neuron pair. This allows both directions to be learned independently via local Hebbian rules, maintaining biological plausibility [108]. Second, the neural circuit of predictive coding relies on identity connections between some neurons. These connections can be learned, and in doing so, the network implicitly captures the covariance structure of the activities within each layer [102]. Alternatively, the identity connections can be replaced with random connections without significantly affecting learning performance [108]. Third, predictive coding appears to require two phases during learning: inference followed by weight updates. However, this separation can be implemented by slower plasticity timescales relative to neural dynamics. Finally, standard predictive coding formulations describe separate value and error neurons. There is empirical evidence for error-like signals in the cortex [109, 110], supporting the core computational requirement. Moreover, explicit error-encoding neurons are not strictly necessary: equivalent error representations can be implemented within the dendritic compartments of pyramidal neurons [111].

Constraint 3: Learning performance and versatility

Beyond plausibility, a learning theory must support the brain’s learning efficiency. Predictive coding has been applied across a wide range of learning settings, including memory formation [112], learning correlation structure [113, 95, 102], unsupervised learning [70], supervised learning [103], temporally varying data [114], novelty detection [115], denoising and associative learning [112], and representation learning [116]. Moreover, because inference minimises a shared energy, a single trained model can often be repurposed for multiple tasks. For example, models trained without supervision can perform denoising, missing-data completion, novelty detection, and low-dimensional inference without retraining [112, 115, 3]. Recent work also indicates competitive learning performance in larger hierarchical models. Specifically, predictive coding networks achieve competitive results relative to backpropagation

on benchmarks including MNIST and CIFAR-10, with promising scaling pathways established on more complex datasets like TinyImageNet [3, 4, 117]. Importantly, predictive coding can outperform backpropagation in biologically relevant regimes where catastrophic forgetting is a key limitation, including continual learning, online learning, few-shot learning, and reinforcement learning with sparse rewards [118]. Finally, experiments suggest that predictive coding may learn faster than backpropagation [119, 120, 118], potentially because its dynamics can escape saddle regions of the loss landscape more rapidly [100].

In summary, predictive coding provides a unifying framework for modelling diverse forms of learning in the brain, and has been shown to achieve learning performance comparable to, or exceeding that of, backpropagation.

1.3 Thesis research and overview

This thesis investigates how predictive coding can be extended to capture a broader set of computations within a single, biologically plausible network. While standard formulations of predictive coding already provide a compelling account of hierarchical inference and local learning, they (i) approximate posteriors with point estimates, (ii) specialise either in generative (top-down) or discriminative (bottom-up) modelling, and (iii) leave open why predictive coding often appears to learn robustly in regimes where catastrophic forgetting is prominent. The core of this thesis, therefore, develops and analyses predictive coding along three complementary directions: probabilistic inference, bidirectional learning, and interference-free learning.

1.3.1 Probabilistic inference and learning

Predictive coding models perform inference by converging to a single pattern of neural activity that minimises prediction error. From a probabilistic perspective, this corresponds to representing the posterior over latent causes with a point estimate, rather than a full distribution, and therefore does not explicitly encode uncertainty. However, within the Bayesian brain framework, representing uncertainty is essential for robust perception and decision-making [77]. In particular, access to the uncertainty of the posterior enables a principled weighting of prior beliefs against incoming

sensory evidence, allowing the brain to optimally attenuate the influence of noisy inputs.

In Chapter 2, we propose Monte Carlo predictive coding (MCPC), which augments standard predictive coding neural dynamics with noise to enable sampling-based inference. Adding stochasticity to the activity dynamics allows the network to approximate the full posterior distribution rather than only its mode, while preserving the same local circuit motif and local plasticity rules. This extension broadens the functional scope of predictive coding as it supports learning of probability distributions, full Bayesian inference via posterior sampling, and generation of samples from the learned internal model. More generally, MCPC bridges predictive coding and neural sampling theories that hypothesise that the brain inverts world models using sampling-based inference [88, 89].

1.3.2 Bidirectional learning

Predictive coding has evolved along two complementary lines. Top-down formulations support flexible unsupervised functions, yet underperform on discriminative tasks such as classification [112]. In contrast, bottom-up variants achieve strong supervised performance [3], but lack core generative capabilities. Biological vision, however, appears to integrate accurate bottom-up recognition with top-down completion and contextual modulation, motivating unified models that combine both directions of processing. Although preliminary work has augmented top-down predictive coding with bottom-up inference pathways to accelerate inference convergence [121], such hybrid approaches remain limited in their ability to support a high discriminative performance.

In Chapter 3, we introduce bidirectional predictive coding (bPC), where each layer forms both bottom-up and top-down predictions, and inference minimises a single objective that combines the corresponding prediction errors. This yields a unified model that retains the biological plausibility of predictive coding, while supporting both supervised and unsupervised learning within the same network. bPC also inherits the fast inference benefits of hybrid approaches. As a result, the same trained model achieves competitive performance on both supervised and unsupervised learning tasks.

1.3.3 Interference-free learning

A recurring empirical observation is that predictive coding can perform well in biologically relevant learning regimes such as online updates, continual learning, and learning from limited data, where catastrophic interference is a major challenge for backpropagation. One hypothesis is that predictive coding weight updates induce changes in the network’s predictions that are better aligned with the output prediction error, thereby reducing unintended changes to previously learned behaviour. Yet, it has remained unclear when this advantage should appear, and which properties of the model determine its magnitude.

In Chapter 4, we provide an analytical account of this phenomenon using deep linear predictive coding models, whose inference and learning dynamics can be characterised in closed form. We derive expressions for the change in model predictions induced by single backpropagation and predictive coding updates, and relate these changes to target alignment. The analysis shows that backpropagation updates can suffer from parameter-dependent interference, whereas predictive coding naturally filters part of this interference, yielding improved alignment. Furthermore, we identify regimes in which these differences are most pronounced, including architectures with information bottlenecks and settings with poorly conditioned weight matrices (as may occur in pre-trained systems). Finally, we propose principled modifications, including layer-dependent learning rates for single-sample learning and activity-dependent decorrelation for multi-sample learning, that guarantee interference-free updates within the predictive coding framework.

Chapter 2

Probabilistic inference with Monte Carlo Predictive coding

The Bayesian brain hypothesis states that the brain learns and updates probabilistic generative models of its sensory inputs. By learning efficient generative models, the brain establishes the causal relationship between environmental states and sensory inputs [81, 122, 96]. The brain also mitigates the effect of sensory noise through generative models by optimally integrating prior knowledge with new sensory data. Several studies have successfully employed probabilistic generative models to explain behavior [82, 83, 84, 85, 86, 87], and interpret neural activity [88, 89, 90, 91, 92].

To elucidate how the brain represents generative models, we seek a neural network capable of learning generative models, while adhering to the brain’s intrinsic characteristics. These characteristics include (i) the brain’s ability to infer posterior distributions of environmental states given sensory inputs [123, 82, 83, 84, 85], (ii) its proficiency in constructing efficient and generalisable generative models using hierarchical neural networks [29], and (iii) its reliance on localized computation and plasticity within these networks [124, 125].

Multiple models implementing the Bayesian brain principle have been proposed that capture some of the above characteristics of the brain. Below we review two categories of models that focus on describing learning of probabilistic models and representing the posterior probabilities of the latent states respectively.

An influential theory describing how the cortex learns the generative models is predictive coding, reviewed in the previous chapter. It hypothesises that the brain learns

the generative models by minimising the error between actual sensory inputs and the sensory inputs predicted by its model [70, 113, 99]. To support this theory, several neural networks have been proposed to illustrate how the brain might implement predictive coding [101, 99, 106]. These networks are hierarchically structured and are local in computation and plasticity. Moreover, since the time predictive coding was first proposed in neuroscience to explain retinal processing [126], it has evolved into a comprehensive framework for understanding attention [75], a range of neurological disorders [76], and various neural phenomena [70, 127, 128]. However, predictive coding has demonstrated limited learning performance for generative tasks [129]. Recent work extended predictive coding to improve its learning performance using lateral inhibition and sparse priors [129, 116], however the resulting neural network is unable to infer posterior distributions or generate sensory samples. In addition to predictive coding, other models have been proposed to describe learning of probabilistic models in the brain. For example, a recent study has employed generative adversarial networks to explain delusions observed in some mental disorders [130]. However, no biologically plausible neural implementation of the adversarial objective function has been identified.

On the other hand, a wide range of neural sampling models have also been proposed that infer the posterior distributions using Monte Carlo sampling methods [131, 90, 132, 133, 134, 135, 136]. In these models, the fluctuations of neural activity over time sample the probability distributions the brain is trying to infer. Some studies show that neural variability in the brain exhibits characteristics consistent with neural sampling processes [137, 138, 139]. Despite this, present neural sampling models lack learning capabilities, local learning rules, or depth in their neural architectures. Recent work incorporated neural sampling into a sparse coding model that can learn generative models with local plasticity [133]. However, the sparse coding model does not include a hierarchical architecture that can support learning of complex generative models.

In this Chapter, we bring together the above work on predictive coding and neural sampling by proposing Monte Carlo predictive coding (MCPC). MCPC follows the approach of variational filtering [140] by integrating neural sampling into predictive coding, albeit in a simplified variant that disregards the dynamics of stimuli. This simplification offers two advantages: (i) the simplified model differs from the predictive coding framework proposed by Rao and Ballard [70] by incorporating additional noise into its inference dynamics, allowing the application of recent advancements in

neural implementations [111] and modeling of a broad range of brain learning tasks [103, 118, 114, 102] to MCPC; (ii) it facilitates a more thorough evaluation of inference, generation, and learning performance of the model, given that most benchmarks and metrics are designed for static inputs.

MCPC presents a biologically plausible neural implementation of generative learning in the brain. It infers full posteriors and learns hierarchical generative models by relying solely on local computation and plasticity. Furthermore, MCPC can generate sensory inputs using local neural dynamics, and its neural activity captures the variability in cortical activity during perceptual tasks. MCPC effectively learns generative models that generalise from data and robustly learns the data (co)variance structure across noise types and intensities as well. Overall, MCPC offers a comprehensive theoretical framework for understanding neural computation and capturing key characteristics of cortical activity.

2.1 Results

This section presents MCPC, and it is organized into subsections discussing the following properties of the model:

- MCPC uses neural networks with local computation and plasticity to learn hierarchical generative models.
- MCPC’s neural dynamics infer full posterior distributions of latent variables in the presence of sensory inputs.
- MCPC’s neural dynamics sample from the learned generative model in the absence of sensory inputs.
- MCPC learns efficient generative models that generalise from sensory data.
- MCPC captures the variability of neural activity observed in perceptual experiments.
- MCPC achieves robust learning of data (co)variance across noise types and intensities.

Throughout our experiments, we consider two tasks: learning a simple probability distribution of Gaussian sensory data, and learning a more complex distribution of handwritten digit images from the MNIST dataset [141]. We compare the properties of our model to predictive coding following the formulation by Rao and Ballard [70] and Bogacz [99] that we refer to with PC. This is because MCPC’s neural dynamics are closely related to this implementation of PC. A comparison between MCPC and other formulations of predictive coding that use techniques such as divisive input modulation [106], free-energy minimisation combined with the Laplace approximation [101], or variational filtering [140] is provided in the discussion.

2.1.1 MCPC implementation with local computation and plasticity

To describe MCPC, we first define a hierarchical generative model MCPC assumes, next present its inference and learning algorithm, and then show how it can be implemented through local computation and plasticity.

MCPC learns a hierarchical Gaussian model of sensory input x with latent variables z . The model consists of L layers, the input layer and $L - 1$ layers of latent variables. We denote the activity of sensory neurons by x_1 , and when the sensory input is present, they are fixed to it, i.e., $x_1 = x$. All latent variables are left free to vary and the latent variables in layer l are denoted with x_l , $z = \{x_2, \dots, x_L\}$. Sensory input x is predicted by the first layer x_1 while variables x_l in layer l are predicted by the layer above. The resulting joint distribution over sensory inputs and latent variables is given by:

$$p(x, z; \theta) = \prod_{l=1}^{L-1} N(x_l; W_l f(x_{l+1}), \sigma^2 I) N(x_L; \mu, \sigma^2 I) \quad (2.1)$$

where parameters θ comprise weights W_l and the prior mean μ describing the mean activity in the top layer, $f(\cdot)$ stands for an activation function, I represents an identity matrix, and σ^2 denotes a scalar variance. This is the same joint distribution as the one learned by PC (see Section 1.2.2). A simple example of such a probabilistic model is illustrated in Figure 2.1A, and it includes one sensory input and one latent state. Such model could, for instance, be used by an organism to infer the size of a food item based on observed light intensity [99]. We will use this model throughout this thesis chapter to provide intuition before considering more complex models.

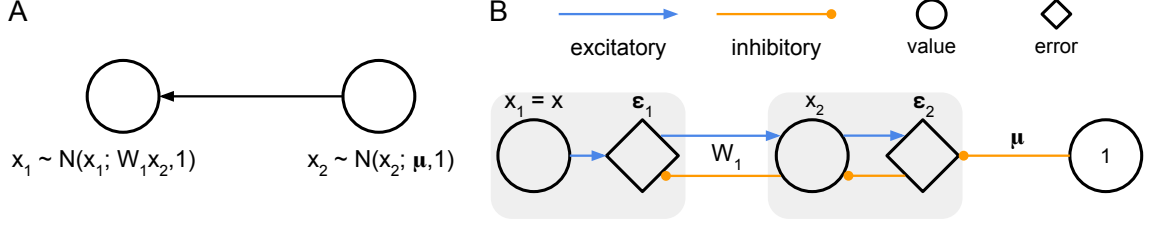


Figure 2.1: **Example of a probabilistic model and its corresponding neural implementation for MCPC.** A: Linear Gaussian model with one sensory input and one latent state. B: The neural implementation of MCPC using local synaptic connections for this model.

MCPC learns a hierarchical Gaussian model by iterating over two steps that descend the negative joint log-likelihood

$$E_{gen} = -\ln p(x, z; \theta) = \frac{1}{2} \sum_{l=1}^{L-1} \frac{\|x_l - W_l \cdot f(x_{l+1})\|^2}{\sigma^2} + \frac{1}{2} \frac{\|x_L - \mu\|^2}{\sigma^2} \quad (2.2)$$

In the first step, MCPC leverages Markov chain Monte Carlo techniques to approximate the full posterior distribution by using the following Langevin dynamics [142]:

$$\frac{\partial x_l(t)}{\partial t} = -\nabla_{x_l} E_{gen} + n_l(t) \quad (2.3)$$

Thus, we modify the latent variables to reduce E_{gen} , but add a zero-mean noise $n_l(t)$ (in the next subsection, we will show explicitly that such dynamics lead to x_l sampling from its posterior distribution). The noise needs to be uncorrelated over time, i.e. with covariance $E[n_l(t)n_l(t')^\top] = 2\sigma_n^2\delta(t-t')I$, where δ is the Dirac delta function, σ_n^2 is the noise variance, and I the identity matrix. The noise variance σ_n^2 is set to one unless otherwise stated.

Evaluating the gradient in equation (2.3), we see below that these neural dynamics give rise to prediction errors ϵ_l encoding the mismatch between the predicted latent state $W_l f(x_{l+1})$ and the inferred latent state x_l .

$$\frac{\partial x_l(t)}{\partial t} = -\epsilon_l + f'(x_l)W_{l-1}^\top\epsilon_{l-1} + n_l(t), \quad (2.4)$$

$$\epsilon_l = x_l - W_l f(x_{l+1}); \quad \epsilon_L = x_L - \mu \quad (2.5)$$

In the second step, MCPC uses the noisy neural activities to update its parameters

as follows:

$$\Delta W_l \propto - \int_{t_0}^{t_0+T} \nabla_{W_l} E_{gen} dt = \int_{t_0}^{t_0+T} \epsilon_l(t) f(x_{l+1}(t))^\top dt \quad (2.6)$$

$$\Delta \mu \propto - \int_{t_0}^{t_0+T} \nabla_{\mu} E_{gen} dt = \int_{t_0}^{t_0+T} \epsilon_L(t) dt \quad (2.7)$$

with t_0 being the time point where the noisy dynamics have converged to their steady-state distribution, and T is large to ensure that the dynamics sample from the whole steady-state distribution. Repeating these two steps enables inference of latent variables and learning of model parameters.

The above algorithm has a direct implementation in a neural network. Such a network has two classes of neurons: value neurons encoding latent states and error neurons encoding prediction errors. The weights of synaptic connections in such a network encode the parameters of the generative model. This is illustrated in Figure 2.1B through a simple network implementing probabilistic inference in the model from Figure 2.1A.

The neural network of MCPC relies on local computation. All neurons perform computations solely based on the activity of their input neurons and the synaptic weights related to these inputs. Specifically, the rate of change of value neurons in equation (2.4) depends on their own activity, the activity of the error neurons connected to them, the weights of these connections, and local noise. Similarly, the activity of error neurons in equation (2.5) can be computed using the activity of connected value neurons and corresponding synaptic weights.

The network also exhibits local plasticity. Synaptic plasticity in MCPC (equations (2.6) and (2.7)) relies exclusively on the product of the activity of pre-synaptic and post-synaptic neurons. The integral in MCPC's synaptic plasticity can also be approximated using local plasticity. This could be achieved by continuously updating synaptic weights with a large time constant.

The neural dynamics and parameter updates of MCPC prescribe the same local neural circuits as existing implementations of PC [95, 99] discussed in the previous chapter, with the addition of a noise term. Hence, MCPC shares the focus of PC on minimizing prediction errors. The additional noise term, however, leads to significant benefits as discussed below.

2.1.2 MCPC infers posterior distributions

Here, we show that MCPC’s neural activity infers full posterior distributions of latent variables in the presence of sensory inputs. We prove that MCPC’s neural activity samples the posterior $p(z|x; \theta)$ at its steady-state for an input x . Moreover, we confirm that MCPC’s neural activity approximates the posterior in the linear model of Figure 2.1A and in a model trained on MNIST digits.

Proposition 1 demonstrates that the neural activity x_2 to x_L prescribed by MCPC samples from the posterior $p(z|x; \theta)$ over latent states z when the dynamics in equation (2.3) have converged.

Proposition 1 *The posterior $p(z|x; \theta)$ is the steady-state distribution $p^{ss}(z)$ of the inference dynamics of MCPC:*

$$p^{ss}(z) = \frac{e^{-E_{gen}}}{Z} = \frac{e^{\ln p(x,z;\theta)}}{Z} = \frac{p(x;\theta)}{Z} p(z|x; \theta) = p(z|x; \theta) \quad (2.8)$$

where Z is the partition function.

The proof is given in the transformations in equation (2.8), which we now explain. It follows from a classical result in statistical physics that the steady-state distribution $p^{ss}(z)$ of a variable z described by the Langevin equation $\frac{\partial z}{\partial t} = -\nabla_z E_{gen} + n(t)$ is given by $p^{ss}(z) = e^{-E_{gen}}/Z$ when the variance of the noise $\sigma_n^2 = 1$ [143]. The Langevin dynamics of MCPC minimise the negative joint log-likelihood $E_{gen} = -\ln p(x, z; \theta)$. The distribution $p^{ss}(z)$ can therefore be rewritten as $p(x, z; \theta)/Z$. Employing the conditional probability formula allows $p^{ss}(z)$ to be subsequently expressed as $p(x; \theta)p(z|x; \theta)/Z$. Given that the distribution $p(x; \theta)$ remains constant for a particular stimulus x , $\frac{p(x; \theta)}{Z}$ forms the partition function of the posterior $p(z|x; \theta)$. However, the posterior distribution integrates to one, $\int p(z|x; \theta) dz = 1$. This implies that $\frac{p(x; \theta)}{Z}$ equals one and that the steady-state distribution $p^{ss}(z)$ effectively simplifies to the posterior distribution $p(z|x; \theta)$. This result for MCPC is analogous to the use of Langevin dynamics for posterior inference in other models [90, 144, 133].

To verify this property, we validate that MCPC samples from the posterior distribution within the simple model from Figure 2.1A, which is tractable. Figure 2.2A illustrates the activity of latent state x_2 of this model during inference under a constant input for both MCPC and PC. While the activity converges to a single value for

PC, activity for MCPC fluctuates around this value representing the uncertainty in its inference. As MCPC’s activity continuously fluctuates, it requires more inference steps to reach steady-state behaviour than PC, whose activity converges to a single point. MCPC inference involves a higher computational burden than PC. Characterising MCPC’s steady-state activity required approximately 10,000 inference steps, compared to 2,000 steps for PC to converge. This higher computational cost, and how it scales with model size, is examined further in Section 2.2.6.2 and Appendix A.2. Figure 2.2B displays a histogram of the latent state’s activity over time throughout the MCPC inference. The inference of PC at its convergence point is also illustrated, as well as the posterior distribution $p(x_2|x;\theta)$ for the specified input. MCPC’s latent state activity accurately samples the posterior of the linear model. In contrast, PC’s inference converges to the mode of the posterior. This result confirms that MCPC samples from the posterior $p(z|x;\theta)$, whereas PC infers the maximum a posteriori (MAP) estimate (see section 1.2.2).

Next, we visually confirm that MCPC infers latent states correctly in a non-linear model with three latent layers trained on MNIST digits. Visualising the latent states during inference shows that both MCPC and PC infer the correct digit when provided with a full-digit image (Figure 2.2C). However, when prompted with an ambiguous masked-digit image, MCPC identifies different possible interpretations, while PC only infers one possible interpretation (Figure 2.2d). This result indicates that MCPC approximates the posterior distribution more accurately than PC. The visualisations are obtained by employing a linear classifier to interpret the latent states. This classifier decodes the latent state x_L and generates a probability distribution over the ten-digit categories. This distribution can then be visualised by mapping it onto ten evenly spaced unit vectors within a circle [145](see Methods section 2.3.2.2 for details). The activity of the latent layer x_L is visualised here. However, similar results are observed for all latent layers, as detailed in Figure A.1.

Finally, we show quantitatively that MCPC indeed approximates the posterior better than a MAP estimate in a non-linear model trained on MNIST. Figure 2.2E shows the Kullback–Leibler (KL) divergence between the posterior across digit classes inferred by a ResNet-9-based ideal observer and the distributions inferred by MCPC, and PC for half-masked images. The KL divergence for a random baseline obtained with shuffled distributions is also shown. This figure shows that the KL divergence between the distributions inferred by the ideal observer and by MCPC is smaller than the one for PC and for the baseline. The lower KL divergence confirms that

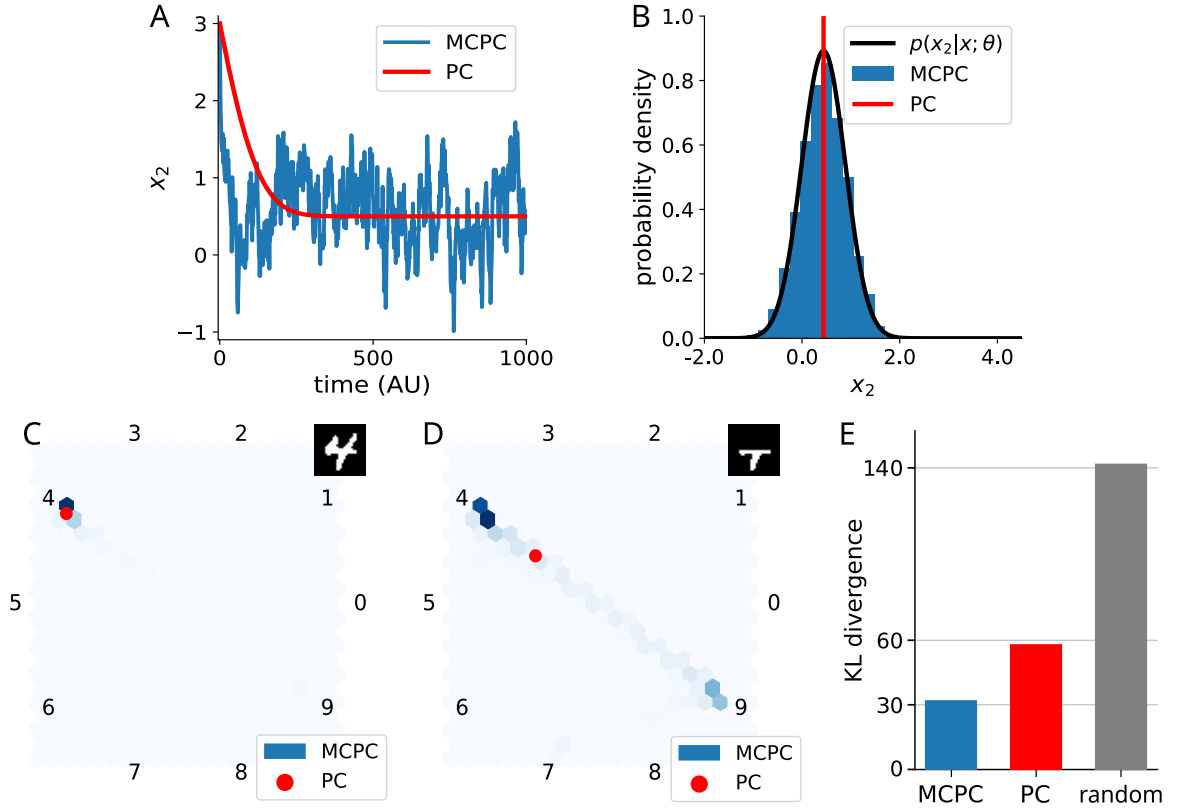


Figure 2.2: **Neural activity of MCPC infers posterior distributions in the presence of inputs.** A, B: Latent state activity x_2 of MCPC and PC in the linear model shown in Figure 2.1A with parameters $\{W_1 = 2, \mu = 0.5\}$ and input $x = 1$. C, D: Latent state activity of MCPC and PC in a model trained on MNIST with a digit image and a half-masked digit image (see top-right) as input. Plots B, C, and D show a histogram of MCPC’s activity over 10,000 timesteps and PC’s activity at convergence. E: KL divergence between the digit class distribution inferred by an ideal ResNet-9 observer and the class distribution decoded from the latent state x_L inferred by MCPC and PC for masked digit images. The KL divergence for shuffled distributions is also provided. Animations of the MCPC’s latent activity in plots B to D can be found in A1, A2 and A3 Videos.

MCPC’s inferred latent states capture the posterior distribution more accurately than PC’s MAP estimate. In this experiment, ResNet-9 is a classifier that achieves over 99% classification accuracy on MNIST [146]. The probability distributions across digit classes of MCPC and PC inferences are obtained with the linear classifier used for interpreting the latent states. Moreover, the random baseline is calculated by averaging the KL divergence between the inferences of the ideal observer and the shuffled distributions inferred by MCPC and by PC.

2.1.3 MCPC samples from its generative model in the absence of sensory inputs

Here, we show that in the absence of sensory inputs, MCPC spontaneously samples from its learned generative model of sensory inputs. We prove that the activity of the unclamped input neurons sample from probability distributions of sensory inputs learned by MCPC. Experiments confirm that the unclamped neural activity generates sensory inputs learned by MCPC in the simple model of Figure 2.1A and in a model trained on MNIST.

To model a scenario in which no sensory input is provided, instead of clamping the input neurons x_1 to a sensory stimulus x , we let these neurons follow similar Langevin dynamics as all other neurons:

$$\frac{\partial x_1}{\partial t} = -\nabla_{x_1} E_{gen} + n_1(t) = -\epsilon_1 + n_1(t). \quad (2.9)$$

Proposition 2 shows that when input neurons are not fixed to sensory stimuli, MCPC spontaneously samples from the learned probability distribution of sensory inputs. In this proposition, we demonstrate that the steady-state of MCPC’s unclamped activity is equal to the marginal likelihood $p(x_1; \theta)$.

Proposition 2 *The marginal likelihood $p(x_1; \theta)$ is the steady-state distribution $p^{ss}(x_1)$ of the Langevin dynamics given in equation (2.9):*

$$p^{ss}(x_1) = \int \frac{e^{-E_{gen}}}{Z} dz = \int \frac{e^{\ln p(x=x_1, z; \theta)}}{Z} dz = p(x_1; \theta) \int \frac{p(z|x_1; \theta)}{Z} dz \quad (2.10)$$

$$= p(x_1; \theta) \quad (2.11)$$

The proof of proposition 2 is similar to that of proposition 1. The steady-state distributions of the neural activity in MCPC in the absence of an input $p^{ss}(x_1, x)$ is given by $e^{-E_{gen}}/Z$. This is a consequence of the Langevin dynamics of MCPC minimizing the negative joint log-likelihood E_{gen} while subjected to a noise variable with variance $\sigma_n^2 = 1$. This distribution can be marginalised over the latent states $z = \{x_2, \dots, x_L\}$ to find the steady-state distribution of the sensory input neurons $p^{ss}(x_1)$. The joint log-likelihood E_{gen} equals $-\ln p(x, z; \theta)$, where $x = x_1$ when input neurons are unclamped. The expression for $p^{ss}(x_1)$ is therefore reformulated as $\int e^{\ln p(x=x_1, z; \theta)} / Z dz$.

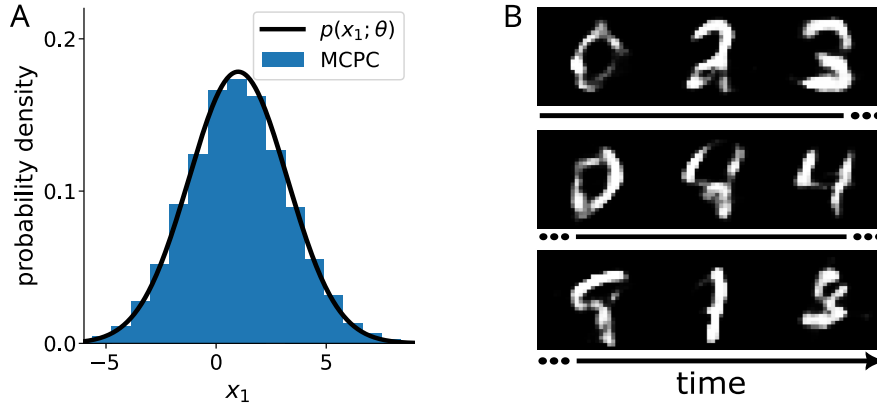


Figure 2.3: **Neural activity of MCPC samples its generative model in the absence of inputs.** A: Histogram of the MCPC activity of the unclamped input neuron x_1 in the linear model given in Figure 2.1A with parameters $\{W_1 = 2, \mu = 0.5\}$ obtained over 10,000 timesteps. B: Activity patterns generated by a model trained on MNIST displayed for time points separated by 3,000 timesteps. The samples of the MNIST-trained model display the probability of a sensory neuron being equal to one. Animation of the MCPC’s unclamped input activity can be found in A4 and A5 Videos.

This expression can be rewritten as $p(x_1) \int p(z|x_1)/Z dz$. Given that the expression $\int p(z|x_1)/Z dz$ remains constant for a specific activity x_1 , this expression forms the partition function of the marginal likelihood $p(x_1; \theta)$. However, the marginal likelihood integrates to one, $\int p(x_1; \theta) dx_1 = 1$. This implies that the partition function $\int p(z|x_1)/Z dz$ equals one and that the steady-state distribution $p^{ss}(x_1)$ effectively simplifies to $p(x_1; \theta)$.

Figures 2.3A and 2.3B experimentally confirm that MCPC generates accurate samples of the generative distribution in the absence of sensory inputs. Figure 2.3A demonstrates this for the linear model by showing that the activity of the unclamped input neuron matches the model’s generative distribution $p(x_1; \theta)$. Similarly, Figure 2.3B illustrates that the unclamped neural activity of a deep non-linear model trained on MNIST produces activity patterns that resemble the digit images used in training.

2.1.4 MCPC learns efficient and generalisable generative models

We show here that MCPC learns precise generative models of sensory data. We demonstrate the precise learning of MCPC by first proving that MCPC is guaranteed

to converge to a local optimum of the marginal likelihood $p(x; \theta)$. Afterwards, we experimentally confirm that MCPC learns efficient generative models of Gaussian sensory data and handwritten digit images. In the process, MCPC outperforms PC and approaches the performance of Deep Latent Gaussian models (DLGMs) on the digit learning task. DLGMs are the standard machine learning approach for training hierarchical Gaussian models (equation (2.1)) using backpropagation [147], and are therefore used as a benchmark. Lastly, we show that MCPC learns hierarchies of abstractions, indicating its capacity for generalisation.

MCPC learns locally optimal generative models of sensory data by implementing the Monte Carlo expectation-maximization algorithm (see proposition 3). This algorithm guarantees that the model parameters converge to a local optimum of the marginal likelihood $p(x; \theta)$ when given enough sampling time during inference [148].

Proposition 3 *MCPC implements the Monte Carlo expectation-maximization algorithm by iterating over:*

1. *E-step: MCPC's inference, $z(t)$, approximates the posterior distributions using an MCMC method for a given input x*

$$z(t) \sim p(z|x; \theta),$$

2. *M-step: MCPC's parameter update maximizes the Monte Carlo expectation of joint log-likelihood*

$$\Delta\theta \propto \nabla_{\theta} \int_{t_0}^{t_0+T} \ln p(x, z(t); \theta) dt \propto \nabla_{\theta} E_{p(z(t))} \{\ln p(x, z(t); \theta)\}.$$

Proposition 3 relies on proving that MCPC's inference samples the posterior distribution for a given input and proving that MCPC's parameter updates maximize the Monte Carlo expectation of the joint log-likelihood. Proposition 1 shows that MCPC's inference samples the posterior distribution. This provides half of the proof for proposition 3. The second part of the proof can be shown by identifying that the expressions E_{gen} in equations (2.6) and (2.7) equal the negative joint log-likelihood. This allows MCPC's parameter updates to be rewritten as $\int_{t_0}^{t_0+T} \nabla_{\theta} \ln p(x, z(t); \theta) dt$. The partial derivatives can be taken out of the integrals to obtain the parameter update $\nabla_{\theta} \int_{t_0}^{t_0+T} \ln p(x, z(t); \theta) dt$. In this expression, $\int_{t_0}^{t_0+T} \ln p(x, z(t); \theta) dt$ is the Monte

Carlo expectation of the joint log-likelihood. Consequently, MCPC parameter updates maximise the Monte Carlo expectation of joint log-likelihood.

Additionally, we show that MCPC learns the distribution of Gaussian sensory data with the linear model of Figure 2.1A. Figure 2.4A illustrates the distribution learned by MCPC after 375 parameter updates. This distribution models the Gaussian data distribution used for training. We obtain the samples of the distribution learned by MCPC using ancestral sampling. In a hierarchical Gaussian model, ancestral sampling consists of first sampling the top latent layer x_L from its Gaussian distribution $N(x_L; \mu I)$. Each layer is then sampled sequentially using the conditional Gaussian distribution $N(x_l; W_l f(x_{l+1}), I)$. Figure 2.4B verifies that MCPC learns an accurate model of Gaussian data for different model initialisations. This demonstrates that each parameter trajectory converges to the parameters for ideal data modeling. This convergence can also be validated analytically by first calculating the curves where the parameter update for the weight or the prior mean parameter equals zero (nullclines shown in green and purple in Figure 2.4B). The intersection of these curves provides the equilibrium points for the parameter values. For MCPC, this intersection is located at the model parameters that perfectly capture the Gaussian data distribution (see A.3 for full derivation).

In contrast to MCPC, PC learns a strikingly poor generative model of the Gaussian data, as shown in Figure 2.4A. PC learns a Gaussian distribution with the correct mean but with an excessive variance. This high variance is caused by the divergence of PC’s weight to $\pm\infty$ during training, as shown in Figure 2.4C for different model initialisations. The variance of the model learned by PC equals $W_1^2 + 1$ (see equation (2.19) in Methods). Consequently, the variance of PC’s generative distribution grows toward infinity as training progresses, leading to a model that becomes increasingly inaccurate. PC’s parameter W_1 diverges to $\pm\infty$ additionally validating the suboptimal learning performance of PC (see Appendix A.3). The underlying cause of this undesirable behavior of PC is that it uses the maximum a-posteriori estimate of x_l in its parameter updates, instead of the full posterior distribution. This learning strategy is equivalent to variational expectation maximization [149], with a Dirac-delta as variational approximation to the true posterior. Crucially, the Dirac-delta ignores uncertainty and introduces an infinite entropy to the free-energy, causing the free-energy to become an arbitrarily loose bound on $\ln p(x; \theta)$ (see Olshausen [150] and A.4 for additional details). As a consequence, the marginal likelihood that needs to be optimised cannot be evaluated, which in practise leads to PC’s weights diverging.

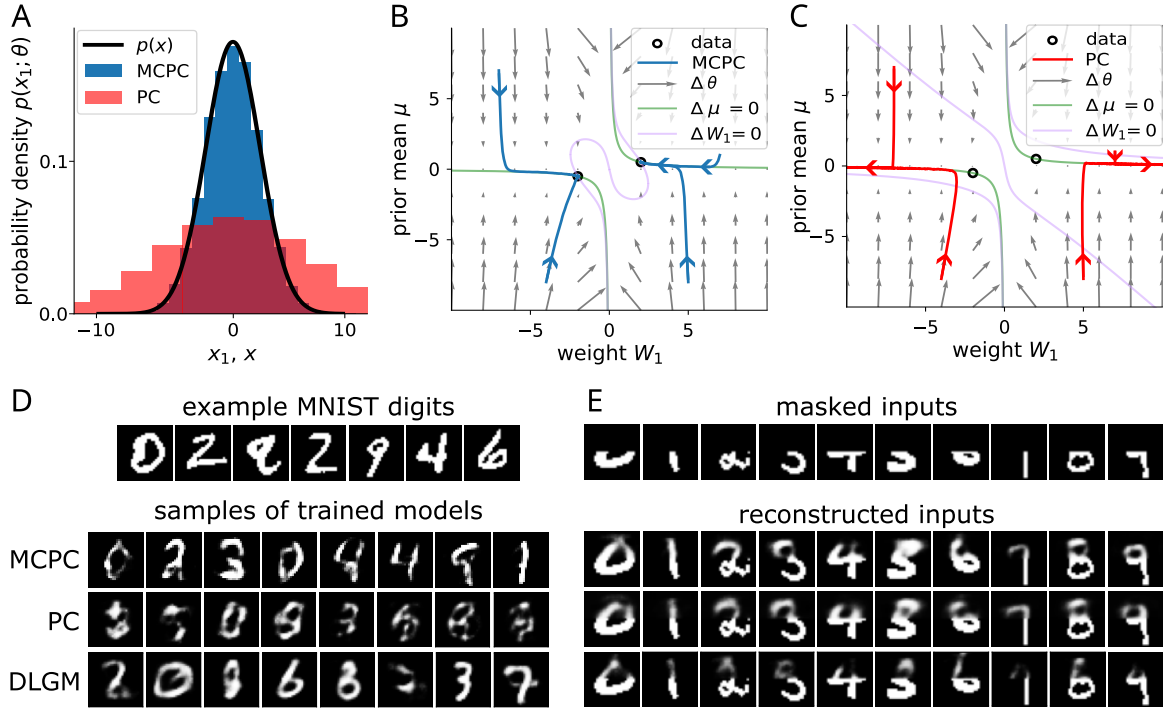


Figure 2.4: **MCPC learns efficient generative models of sensory inputs.** A: Distributions learned by MCPC and PC in the linear model given in Figure 2.1A after 375 parameter updates. B, C: Evolution of the weight W_1 and prior mean μ parameter of the linear model during training with MCPC in B and PC in C. The optimal model parameter values are marked as hollow dots. The vector field shows the expected gradient flow of the parameters. The additional curves reveal nullclines where the parameter update for the weight or the prior mean parameter equals zero (see Appendix A.3 for derivations). D: Comparison between samples obtained from models trained with MCPC and PC on MNIST, as well as from a DLGM trained on MNIST. The samples are obtained by ancestrally sampling the models for PC and the DLGM and by sampling the spontaneous neural activity for MCPC. E: Comparison between masked images reconstructed by MCPC, PC, and a DLGM. We reconstruct the images by obtaining a maximum a-posteriori estimate of the missing pixel values.

In contrast, MCPC implements the Monte Carlo expectation-maximisation algorithm that optimises the marginal log likelihood $\ln p(x; \theta)$ (proposition 3). Interestingly, a range of other theories for learning in the brain [103, 24, 151] are based on a similar energy as in PC, posing the question of whether they suffer from similar failure modes as we uncover here for PC.

Next, we show that MCPC learns accurate hierarchical Gaussian models of MNIST handwritten digit images [141]. For this learning task, we consider non-linear models with three latent layers. We train these models using MCPC, PC, and the DLGM

approach (refer to the Methods section 2.3.2.2 for details). Figure 2.4D presents samples generated from the trained models. The quality of samples generated from an MCPC-trained model approaches that of samples produced by a DLGM. However, the samples obtained from training with PC are of significantly poorer quality, even when we apply weight decay to mitigate PC’s exploding variance. To quantify the difference in performance, we compute three metrics. First, we calculate the Fréchet inception distance (FID) of the generated samples which measures the similarity between generated data and actual data [152]. Second, we approximate the marginal log-likelihood of test data $\ln p(x_{eval}; \theta)$ using Monte Carlo sampling. This metric evaluates the generalization performance of a trained generative model. Third, we compute the mean squared error (MSE) associated with reconstructing masked digits as illustrated in Figure 2.4E. This assesses the ability to learn and retrieve associative memories [112]. Table 2.1 summarises the results and shows that a model trained with MCPC generates significantly better samples than PC and that it has better generalization performance. Additionally, MCPC approaches the generative learning performance of DLGM. Table 2.1 also shows that MCPC can reconstruct masked digits as well as PC, and that both perform significantly better than DLGMs.

Table 2.1: **Comparison of learning performance between MCPC, PC and DLGM.**

Model	FID	$-\ln p(x_{eval})$	Reconstruction MSE (10^{-2})
PC	115.2 ± 3.0	168.9 ± 0.2	8.73 ± 0.03
MCPC	60.6 ± 2.9	144.6 ± 0.7	8.29 ± 0.05
DLGM	45.4 ± 0.7	126.0 ± 0.3	12.04 ± 0.08

We report the FID, the marginal log-likelihood, and the reconstruction error on an MNIST evaluation set (the closer to zero the better for all metrics). We set in bold the best score across the models. Mean $\pm$ standard deviation computed over three seeds.

Finally, we demonstrate that MCPC effectively learns hierarchical abstractions from data. Figure 2.5 illustrates the features learned by all latent neurons in a model trained using MCPC on MNIST. These features are extracted by setting all the neurons’ activities in a layer to zero except for one neuron, which is set to a high activity level. The latent activity of that layer is then propagated forward through the model until the input layer is reached. This process is equivalent to finding the activity pattern that minimizes the negative joint log-likelihood conditioned on the manipulated layer of neurons. The features learned by the first latent layer, x_2 ,

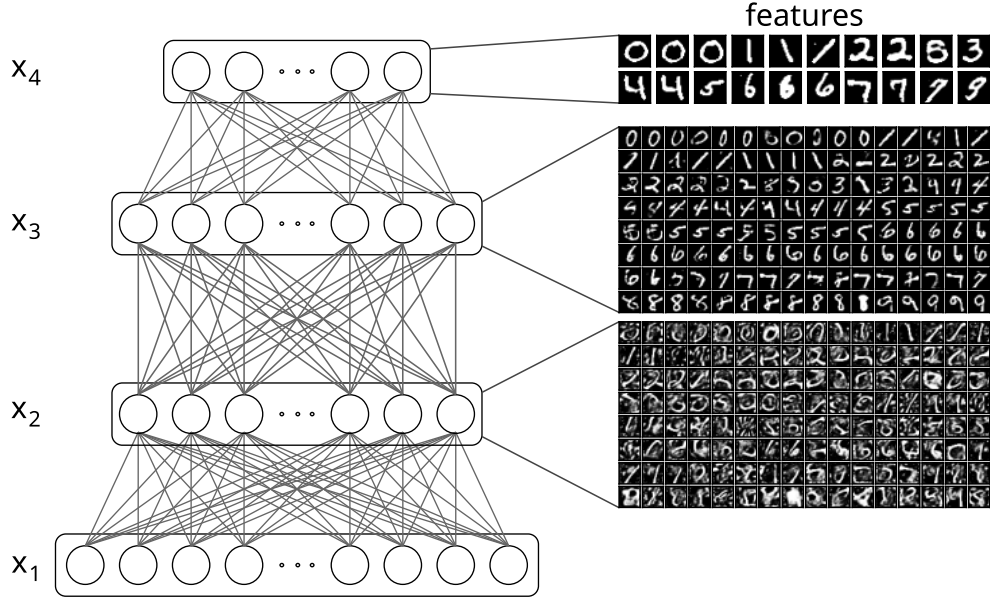


Figure 2.5: **MCPC learns hierarchies of abstractions.** Features of each neuron in an MCPC model trained on MNIST. The features are sorted using a ResNet-9 classifier for each layer.

consist of 128 low-level features. The features learned by the second latent layer, x_3 , include 128 digit representations that vary in orientation, shape, and style. The features learned by the final latent layer, x_4 , consist of 20 digits encompassing most classes with minimal within-class variation. This progression indicates that the model learns increasingly abstract features from the input layer to the deepest latent layer. Furthermore, this demonstrates the model’s ability to transition from representing pixel-level information in the lower layers to capturing semantic information in the higher layers, thereby showcasing its capacity for generalization.

2.1.5 MCPC captures the variability of cortical activity

MCPC captures the key characteristics of the variability of cortical activity during perceptual tasks that PC fails to capture. Specifically, MCPC accounts for the suppression of neural variability at stimulus onset and the increase in similarity between spontaneous and evoked neural activities during development.

MCPC exhibits a decrease in temporal variability of neural activity at stimulus onset as observed in multiple electrophysiology studies [153, 154, 155, 156, 157, 158, 159, 160]. These studies have shown that neural variability is smaller after stimulus onset

than before stimulus onset. This finding holds when measured with intracellular or extracellular recordings and when an animal is task-engaged, awake, or anesthetized. Figure 2.6A illustrates the neural variability experimentally observed by Churchland et al. [156] and the neural variability of MCPC’s latent states at stimulus onset. This figure shows that MCPC’s neural activity captures the decrease in neural variability at stimulus onset for an MNIST-trained model. Appendix A.5 provides additional proof that this observation generally holds for MCPC. This proof shows that the variability of MCPC’s steady-state activity before stimulus onset is in expectation larger than the variability after stimulus onset.

MCPC displays an increase in similarity between spontaneous and average evoked neural activities that is specific to natural scenes as observed during learning for ferrets [137]. Berkes et al. [137] recorded the spontaneous and average evoked neural activity in the primary visual cortex (V1) of ferrets for natural stimuli, sinusoidal gratings, and random noise. They observed that, as development progressed, the spontaneous activity increasingly resembled the average activity evoked by natural stimuli. Additionally, this increase in similarity was not observed for the sinusoidal gratings and random noise. Figure 2.6B compares the similarity between spontaneous and evoked neural activities for natural stimuli, noise, and image gratings reported by Berkes et al. [137] and observed for MCPC in MNIST-trained models. MCPC displays an increase in similarity between spontaneous and evoked neural activities that is specific to the digit stimuli on which it was trained (that are analogous to natural scenes to which the visual systems of animals were exposed). Such an increase in the similarity between spontaneous activity and the average response to stimuli on which the model was trained holds for MCPC in general. This is because the steady-state distribution of MCPC’s spontaneous neural activity becomes more similar to the average steady-state distribution of MCPC’s evoked activity as MCPC’s generative model improves (see Appendix A.5 for the detailed proof). In our experiment, the similarity in neural activities is measured using the KL divergence for neurons in layer x_2 and natural images are MNIST images (see Methods section 2.3.2.2 for a detailed explanation of the experiment). The natural stimuli-specific similarity increase is present across the model’s latent layers. However, as shown in Figure A.1, the KL divergence is only significantly smaller for natural stimuli in layer x_2 .

Both the above characteristics of cortical activity are not reproduced by PC. The spontaneous and evoked neural activities of PC converge to a constant neural activity without neural variability. Consequently, the temporal variability of individual

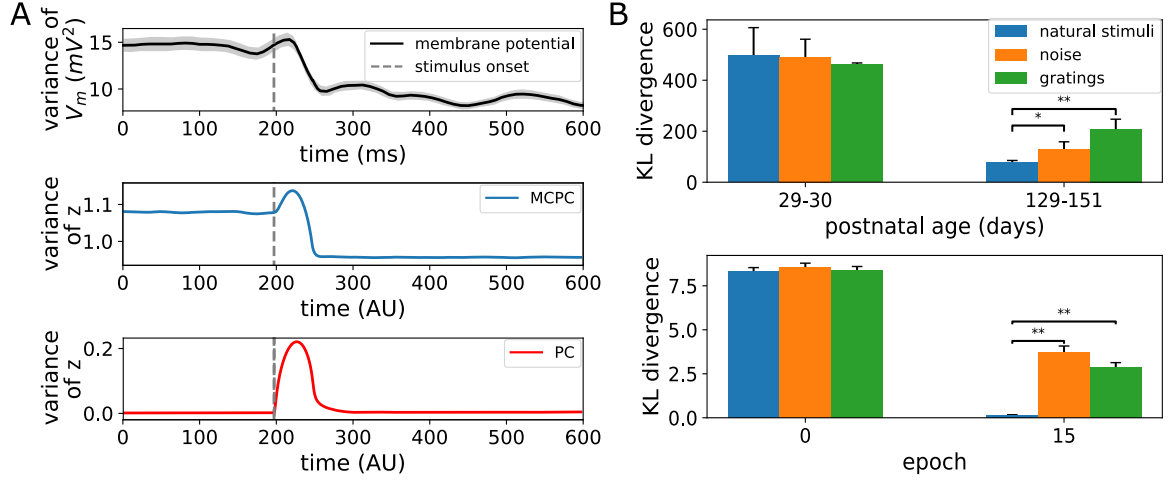


Figure 2.6: MCPC captures two key features of cortical activity. A: MCPC displays the decrease in neural variability at stimulus onset observed in the primary visual cortex (V1) of cats. The top plot recreates the neural quenching observed in the cortex (data re-plotted from figure 2c in Churchland et al. [156]). The middle and bottom plots show the mean temporal variability at stimulus onset of the latent state for MCPC and PC in a model trained on MNIST. Shaded regions give the standard error of the mean (s.e.m.). Note that for MCPC and PC, these shaded regions are not visible due to their minimal magnitude. B: MCPC displays the similarity increase between spontaneous and evoked neural activities specific to natural stimuli observed in V1 of ferrets during development. The similarity is measured using the KL divergence between the distribution of spontaneous activity and the average distribution of evoked neural activities (the closer to zero the more similar). The average distribution is obtained for natural stimuli, noise stimuli, and gratings. The top plot recreates the similarity increase observed in awake ferrets (data re-plotted from figure 4a in Berkes et al. [137]). The bottom plot demonstrates a parallel increase in similarity specific to the training stimuli for MCPC. The MCPC model was trained on MNIST and evaluated using noise, image gratings, and MNIST digits (analogous to natural stimuli). In both plots, the error bars give the s.e.m. and * or ** indicate $p < 0.05$ or $p < 0.01$ respectively for a one-tailed paired samples t-test based on the KL divergences obtained for $n = 10$ MCPC models with the same architecture but different initializations.

neurons is not suppressed at stimulus onset in PC. Instead, the variability temporarily increases above zero at stimulus onset, after which it returns to zero as illustrated in Figure 2.6A. Moreover, training does not enhance the similarity between the distributions of spontaneous and evoked activities in PC. The distribution of PC’s spontaneous activity is a Dirac delta distribution as the activity has no variability. Similarly, the distribution of PC’s average evoked activity is a Dirac mixture distribution. Consequently, the KL divergence between these two non-identical Dirac-based distributions is always infinite.

These two signatures have also been reproduced by previous sampling-based models [161]. What is distinctive about MCPC is that they arise within a hierarchical generative model learned through local plasticity, as we discuss in Section 2.2.4.

2.1.6 MCPC robustly learns the data (co)variance across noise types and intensities

We demonstrate that MCPC effectively learns the variance and covariance structure of data. MCPC is also flexible in accommodating any type of noise distribution and variance in its Langevin dynamics, thereby avoiding the introduction of biologically unrealistic assumptions in the model.

MCPC learns the variance and covariance structure of data for both Gaussian data and the MNIST dataset by capturing the data covariation in its model weights. Figures 2.7A and 2.7B demonstrate that the MCPC model in Figure 2.1A learns a distribution with the same variance as its Gaussian training data across a range of data variances, Σ_{data} . However, MCPC cannot learn the data variance when the data variance is smaller than its layer variance σ^2 and the Langevin noise variance σ_n^2 equals one. For the considered linear model, the marginal likelihood $p(x_1; \theta)$ equals $N(x_1; \mu W_1, \sigma^2(W_1^2 + 1))$. Therefore, the model’s variance is directly encoded in the model’s weight W_1 and the model cannot learn the data distribution when the data variance is smaller than the layer variance σ^2 . This result is experimentally confirmed in Figure 2.7C which shows that the learned weight W_1 approximately equals $\sqrt{\Sigma_{data}/\sigma^2 - 1}$ only for data variance larger than σ^2 . MCPC also learns the covariation structure of data as shown in Figure 2.7D. This figure compares the absolute correlation between non-zero pixels in the MNIST dataset and image samples generated by MCPC, confirming that MCPC accurately learns the pixel correlations.

An additional noteworthy characteristic of the MCPC is its flexibility in accommodating any type of noise distribution and variance. The only requirements on the noise variable $n_l(t)$ in MCPC’s dynamics are as follows: (1) the noise has a zero mean, (2) it is uncorrelated in time and across neurons, and (3) the variance of the noise needs to be constant over time. These requirements follow from the fluctuation-dissipation theorem in statistical mechanics that determines the first two moments of $n_l(t)$ (equations (2.12a) and (2.12b)) and the resulting steady-state distribution of

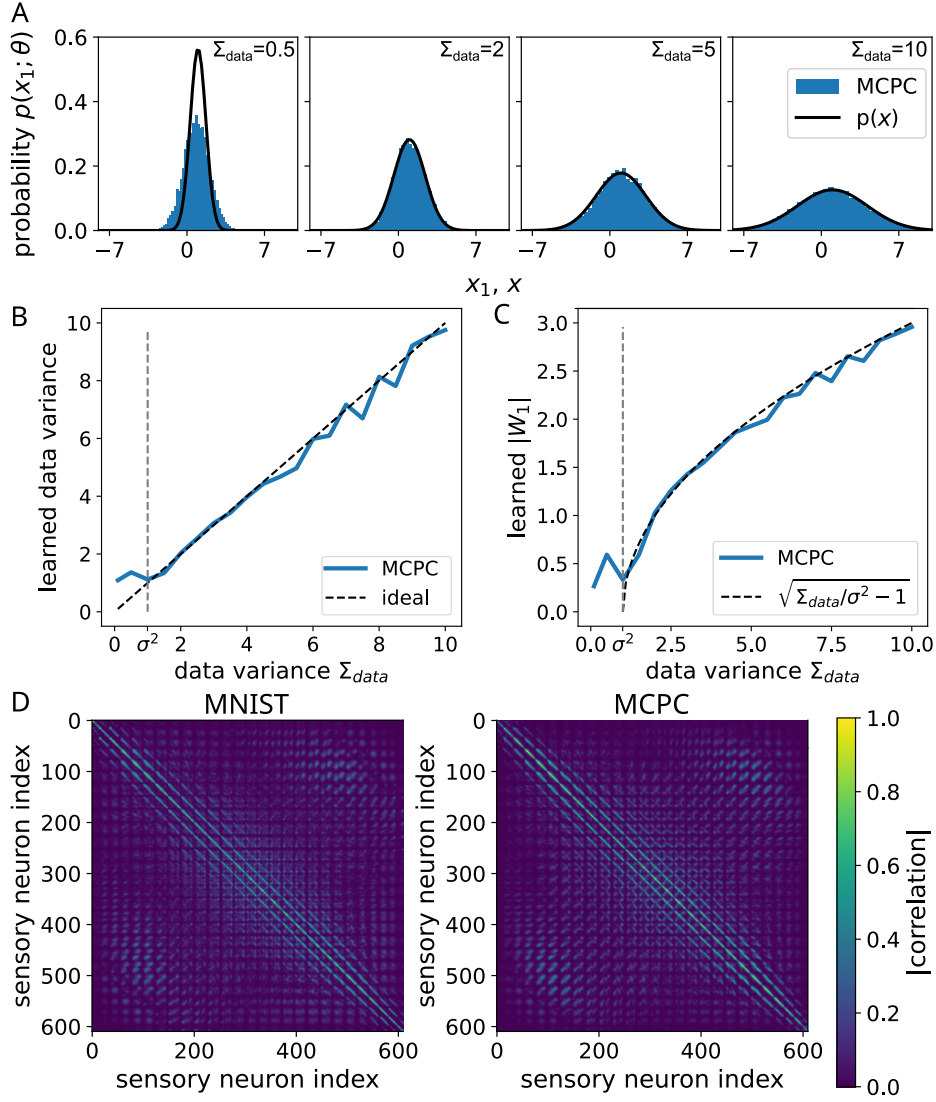


Figure 2.7: **(Co)variance learning in MCPC.** A-C: MCPC model of Figure 2.1A trained on Gaussian data for a range of training data variances. A: Comparison between the data distribution and the distribution generated by the trained model. The distributions learned by MCPC are obtained using MCPC’s spontaneous activity after 10,000 timesteps. B: Variance of distribution generated by trained MCPC model for a range of training data variances. C: Absolute weight, W_1 , of trained MCPC model for a range of training data variances. The ideal weight W_1 equals $\pm\sqrt{\Sigma_{data}/\sigma^2 - 1}$ where $\sigma^2 = 1$ in our experiment. Moreover, the vertical dashed line shown in B and C indicates where the variance of the Gaussian input layer of the MCPC model $\sigma^2 I$ becomes larger than the variance of the data distribution. D: Comparison between the correlation of pixels in the MNIST dataset and in 4000 image samples generated by an MNIST-trained MCPC model. Pixels that are always equal to zero in our MNIST evaluation set are excluded.

the stochastic dynamics (equation (2.13)) where σ_n^2 scales the variance of the noise [143].

$$E[n_i(t)] = 0 \quad (2.12a)$$

$$E[n_i(t)n_i(t')] = 2\sigma_n^2 \delta(t - t')I \quad (2.12b)$$

$$p^{ss}(z) = \frac{e^{E_{gen} \sigma_n^2}}{Z} = \frac{e^{\sum_{l=1}^L \frac{1}{2} \|x_l - W_l \cdot f(x_{l+1})\|^2 / (\sigma^2 \sigma_n^2)}}{Z} \quad (2.13)$$

The fluctuation-dissipation theorem does not impose any specific constraints on the exact distribution of MCPC's noise [162]. This absence of assumption regarding the specific noise distribution ensures that MCPC does not hinge on potentially biologically implausible noise distributions.

The scalar variance of the noise, σ_n^2 , is also not specified in the requirements of the fluctuation-dissipation theorem. As a result, it can be equal to values other than identity (default value used in experiments). However, altering σ_n^2 affects the variance of the layers in MCPC's generative model, as it changes the steady-state distribution of the Langevin dynamics. Specifically, this adjustment scales the variance of the generative layers to $\sigma^2 \sigma_n^2 I$, as indicated in equation (2.13). Therefore, σ_n^2 must remain constant over time to ensure consistency during learning and subsequent inferences.

To verify that MCPC learns the variance of data for non-identity scalar noise variances, we train the linear model from Figure 2.1A on Gaussian data with various levels of noise in its dynamics. Figures 2.8A and 2.8B show that MCPC accurately learns the data variance and distribution when the noise variance, σ_n^2 , is below an upper limit. The learned distributions are generated using MCPC's unclamped neural activity while maintaining the level of noise used during training. For the model of Figure 2.1A, the marginal likelihood $p(x_1; \theta)$ equals $N(x_1; \mu W_1, \sigma_n^2 \sigma^2 (W_1^2 + 1))$ for non-identity noise variances. The model weight, W_1 , should therefore equal $\sqrt{(\Sigma_{data} / \sigma_n^2 \sigma^2 - 1)}$ to capture the data variance and there should exist a learning limit at $\sigma_n^2 = \Sigma_{data} / \sigma^2$ above which no weight values exist to capture the data variance. Figure 2.8C confirms that the model weight changes according to $\sqrt{(\Sigma_{data} / \sigma_n^2 - 1)}$, to capture the data variance. Moreover, a learning limit exists at $\sigma_n^2 = \Sigma_{data} / \sigma^2$, above which MCPC maximally reduces the model's variance by setting the weight W_1 close to zero.

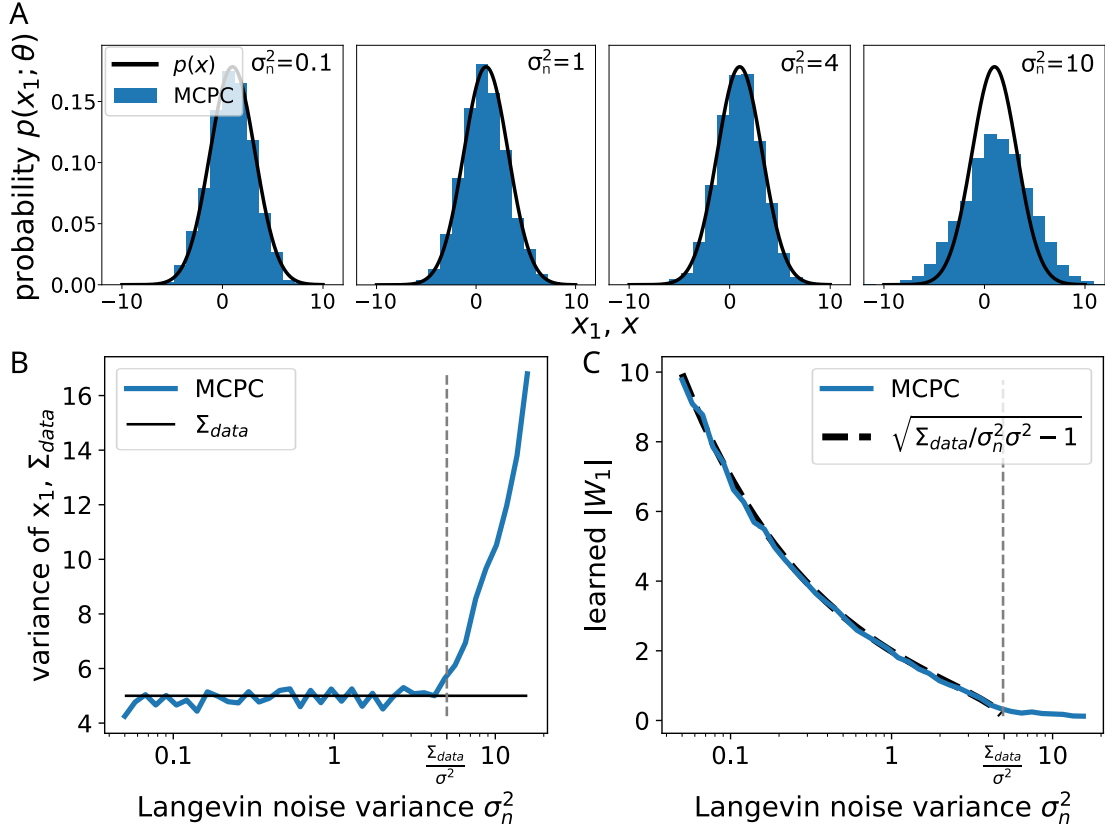


Figure 2.8: **MCPC is compatible with a range of noise levels.** A: Distributions learned by MCPC for the linear model shown in Figure 2.1A when trained on Gaussian data with four different levels of noise. B: Comparison between the variance of the data distribution and the variance of the distribution learned by MCPC with a range of noise levels. C: Comparison between the weight parameter W_0 learned by MCPC and the ideal weight for different levels of noise. The ideal weight parameter is given by $\pm\sqrt{\Sigma_{data}/\sigma_n^2\sigma^2 - 1}$, which can be found by comparing the marginal likelihood of the model to the data distribution as shown in section 2.3.2.1. The Gaussian data used for training in all panels has a variance of five. The distributions learned by MCPC in A and B are obtained using MCPC’s spontaneous activity over 10,000 timesteps after training, while maintaining the level of noise used during training. Moreover, the vertical dashed line shown in B and C indicates where the variance of the Gaussian input layer of the MCPC model $\sigma_n^2\sigma^2 I$ becomes larger than the variance of the data distribution. In these experiments, $\Sigma_{data} = 5$, $\mu_{data} = 1$ and $\sigma^2 = 1$.

2.2 Discussion

This work establishes how the brain could learn probability distributions of sensory inputs by relying solely on local computations and plasticity. We propose Monte Carlo predictive coding, a neural model that learns probability distributions of sen-

sory inputs using a hierarchical neural network with local computation and plasticity. MCPC introduces neural sampling to predictive coding using Langevin dynamics which enables: (i) the inference of full posteriors, (ii) the sampling of learned sensory inputs analogous to the brain imagining sensory stimuli, (iii) learning accurate generative models of sensory inputs, (iv) an ability to explain the variability of cortical activity and (v) learning data variance robustly across noise types and intensities.

2.2.1 Benefits from computational abilities of MCPC

The identified neural dynamics of MCPC infer posterior distributions and generate data samples. These abilities would provide great benefits to organisms supporting them. On one hand, the ability of MCPC to infer posterior distributions reflects the brain’s ability to infer statistically optimal representations of the environment. Such representations are key for survival through optimal perception [163] and decision-making [77]. On the other hand, our model’s ability to generate samples from learned sensory inputs is essential for offline replay. Cognitive functions that rely on offline replay include memory consolidation [164], planning of future actions [165], reinforcement learning [166], visual understanding [167], predictions [168], and decision-making [169]. Taken together, the neural activity of MCPC provides the basis upon which a wide array of other brain functions depend. This implies that MCPC might be useful not only for understanding generative learning but also for unraveling the brain functions that potentially depend on its neural activity patterns.

2.2.2 Unified theory of cortical computation

MCPC integrates the strengths of PC and neural sampling, providing a unified theory of cortical computation.

MCPC, as a form of PC, utilises prediction error minimisation for inference and learning in a hierarchical model. This alignment with PC enables the application of its potential cortical microcircuit implementations [170] and its implementation using dendritic errors [111] to MCPC. Additionally, MCPC can be applied to various learning tasks, similar to PC. For example, PC shows promising results in various classical tasks such as supervised learning, associative learning, representational learning, and reinforcement learning [116, 171, 118]. We expect MCPC to surpass PC in these

tasks, owing to its enhanced inference dynamics that more closely approximate posterior distributions.

Concurrently, MCPC embodies neural sampling by employing neural dynamics to sample posterior distributions. Neural sampling was first proposed by Hoyer and Hyvärinen [88]. Since then, different implementations of sampling-based computations by the brain have been proposed [134, 90, 135, 132, 131, 139, 133, 136]. These models have offered valuable insights that could be applied to MCPC. For instance, the sampling efficiency of MCPC could be improved through the use of excitatory and inhibitory recurrent networks, as suggested by Hennequin et al. [172]. Ultimately, MCPC opens new possibilities for a more comprehensive understanding of cortical computation and of the interplay between prediction-based learning and stochastic sampling mechanisms within the brain.

As a theory of cortical computation, MCPC can provide an account for a broad spectrum of cortical phenomena. This theory could bridge the explanatory scopes of both PC and neural sampling. Predictive coding has played a pivotal role in providing a unified framework for explaining perception and attention [75]. It simultaneously offers insights into a range of neurological disorders such as schizophrenia, epilepsy, post-traumatic stress disorder, and chronic pain [76]. Predictive coding has also explained diverse neural phenomena ranging from retinal information encoding [127], alpha oscillations [128], and non-classical receptive fields [70]. Neural sampling has provided significant insights into explaining dynamic features of cortical activity. These features include the stimulus-dependence of neural variability [156, 161] and oscillations in the gamma band [173], strong transients at stimulus onset [174], and the spatiotemporal dynamics of bi-stable perception [85, 84]. By integrating PC with neural sampling, MCPC is poised to offer a comprehensive model capable of bridging the explanatory scopes of both PC and neural sampling.

2.2.3 Relationship to implementations of predictive coding

Here, we compare MCPC to several formulations of inference and learning using PC.

Our experiment compares MCPC to PC as described by Rao and Ballard [70] and Bogacz [99]. This PC model uses a Dirac delta function to approximate the true

posterior during inference. It relies on local computation and plasticity but is limited to MAP inference and does not support effective learning, unlike MCPC.

PC/BC-DIM is another version of PC that aligns with biased competition theories of cortical function [175, 176]. PC/BC-DIM uses divisive input modulation [106] to update error and prediction neuron activations. This method supports local computation and offers faster inference, but in contrast to MCPC, it does not infer the uncertainty of its inferences.

Another PC implementation by Friston and Kiebel [101] uses normal distributions to approximate the true posterior during inference. This approach utilises the same neural dynamics as Rao and Ballard [70], initially performing MAP inference. Later, it uses the Laplace approximation to approximate the posterior as a normal distribution around the inferred mode. While the MAP inference remains local, estimating the variance of the Laplace approximation in multivariate models involves non-local computation. Compared to MCPC, it allows for faster inference. However, this method approximates the posterior rather than fully sampling it, which can cause problems in complex models with multimodal posteriors.

Finally, variational filtering represents another PC implementation that utilizes Langevin dynamics for inference, similar to MCPC. This approach operates within generalized coordinates, which is beneficial for learning dynamic latent variables and temporal structures in data. MCPC can be considered a zero-order version of variational filtering, where sensory input remains static over time. However, in its current form, MCPC cannot learn dynamic inputs. To extend MCPC to accommodate time-varying inputs, a scheme similar to the recently developed temporal PC [114, 177] could be employed, which uses an additional set of weights to predict future latent states from past latent states.

2.2.4 Relationship to other models introducing Langevin dynamics to brain-inspired generative models

Several brain-inspired generative models using Langevin dynamics have been proposed, and here we discuss their similarities and differences from MCPC.

Langevin dynamics were initially proposed as a sampling strategy for posterior inference that is neurally implementable [90, 135]. This research paved the way for other

models that elucidate diverse facets of perception and cortical functions using neural sampling [172, 131, 139]. For instance, the two experimental observations of neural variability captured by MCPC, as demonstrated in our work in Figure 2.6, have been explained using neural sampling [161]. Nevertheless, the proposed neural sampling models are either devoid of learning capabilities or rely on non-local plasticity mechanisms for weight adjustment. The key distinction is what each model must assume to reproduce these signatures. Previous sampling accounts [161, 139] use a probabilistic model whose parameters are hand-designed, fit offline, or optimised non-locally. They thus explain the variability that arises given a model, but not how the model, and with it the structure of the variability, is acquired. MCPC reproduces the same signatures while learning its hierarchical generative model through local plasticity. What predictive coding specifically contributes is the substrate: the same local circuit that samples the posterior also computes prediction errors, so that learning, inference, and cortical variability follow from a single mechanism.

Langevin dynamics have also been applied in sparse coding models for posterior inference [133]. These models leverage local Langevin dynamics for inference and employ local plasticity rules specific to sparse coding for learning. When trained on patches of natural images, these models successfully learn simple-cell receptive fields. Unlike MCPC, these sparse coding models do not possess hierarchical structures. Additionally, their learning capabilities have only been evaluated on relatively simple datasets, such as oriented bars.

Several machine learning studies have shown that generative models with Langevin dynamics learn accurate generative models of complex machine learning tasks [144, 178]. The studies show that the model with Langevin dynamics can outperform Variational Autoencoder and Generative adversarial networks on datasets such as MNIST, CIFAR-10, and CelebA. These studies confirm that models with Langevin dynamics can learn accurate generative models. However, in contrast to MCPC, these studies considered models that learn using non-local plasticity.

Since our initial presentation of MCPC [179], subsequent research [180, 181] has further validated that generative models employing Langevin dynamics learn precise generative models on complex tasks. Zahid et al. [180] also proposed the use of Langevin dynamics in PC. However, their investigation focused on biologically implausible models with a singular latent layer, trained via backpropagation, diverging from MCPC’s

approach. On the other hand, Dong and Wu [181] incorporated Langevin dynamics into generative models that leverage local computation and plasticity, showcasing capabilities for posterior inference and data generation using local neural dynamics akin to MCPC. Dong and Wu [181] employ exponential-family energy-based models, which differ from the hierarchical Gaussian models used in PC. As a result, their proposed models are less directly linked to PC than MCPC.

2.2.5 Experimental prediction

The core prediction of MCPC posits that the brain concurrently performs predictive error computations and sampling processes. Experimentation to substantiate MCPC would therefore involve detecting simultaneous prediction errors and neural sampling. This requirement distinguishes MCPC from sampling-based accounts of cortical variability, which attribute neural variability to posterior sampling alone. MCPC instead predicts that this variability is accompanied by prediction-error signals within the same circuit. Observing sampling-like variability together with concurrent prediction-error activity, rather than either signature in isolation, would therefore favour MCPC over accounts based on sampling or prediction-error minimisation alone. According to PC theories, prediction errors can be measured in the activity of error neurons, as discussed in this thesis, or in the activity of dendrites [111]. Notably, this activity intensifies in response to unanticipated sensory inputs. Additionally, a measurable signature of sampling is a change in neural variability of value-encoding neurons as a result of a change in uncertainty associated with sensory inputs. An experimental approach to test MCPC’s prediction could, therefore, involve training animals to classify visual stimuli. Following their training, the experiment would measure the neural responses in the animals’ primary visual cortex when they are shown ambiguous and unambiguous stimuli. MCPC predicts that: (i) Neurons or dendrites that encode prediction errors will exhibit greater activity in response to ambiguous stimuli compared to non-ambiguous stimuli, and (ii) value-encoding neurons involved in sampling will display increased variability when processing ambiguous stimuli as opposed to unambiguous stimuli. An observed increase in both error-encoding activity and variability in value neurons in response to ambiguous stimuli, compared to unambiguous ones, would suggest the brain’s use of principles similar to those in MCPC for generative learning.

Additionally, MCPC also predict how signatures of sampling vary across the cortical hierarchy. For example, the increase in similarity between spontaneous and evoked activity is specific to particular layers in our model (Appendix A.1). These predictions could be tested by measuring the similarity between spontaneous and evoked activity across cortical layers in animals trained on specific stimuli.

2.2.6 Limitations and future work

2.2.6.1 Extending MCPC to learn precision weighted prediction errors

While MCPC effectively learns data variance and covariance over a wide range of data parameters, it encounters learning limitations with narrow data distributions relative to its layer variance. To address this issue, the model could be extended to parameterise and learn the precision matrices of the Gaussian layers within MCPC’s generative model, where the precision matrix is the inverse of the covariance matrix. This extension would be particularly significant for the predictive coding field, as existing literature emphasises the importance of minimising precision-weighted prediction errors for both computation and neuropathology [75, 76]. Various schemes for PC models have been proposed to learn precision matrices through local computations [99, 102]. These schemes could be applied to MCPC by appropriately modifying the negative log-likelihood function E_{gen} .

2.2.6.2 Improving the sampling speed of Langevin dynamics

Despite the promising results in this study, sampling using MCPC’s Langevin dynamics requires long inference times [142]. The inference duration could be significantly shortened by relying on advancements in neuroscience and machine learning. For instance, Hennequin et al. [172] showed how the cortex could increase the sampling efficiency of neural circuits using excitatory and inhibitory recurrent networks. Adding higher-order terms to the Langevin dynamics, such as momentum, also dramatically improves convergence speed [182]. Additionally, Ma et al. [183] proposed a general framework for improving the sampling efficiency of Langevin-based sampling. By applying these principles to MCPC, the sampling speed is expected to increase, reaching a value that resembles the fast sampling of the brain [81]. Importantly, the

learning performance of MCPC is then anticipated to improve, as the inferences will capture the posterior more effectively.

The convergence speed of MCPC’s Langevin dynamics also increases with the number of latent dimensions. In our experiments, the model trained on MNIST (over 200 dimensions) requires significantly more inference steps to reach a steady-state than the model trained on Gaussian data (one dimension), as shown in Figure A.2. Scaling MCPC to large models might therefore be limited by the number of inference steps required to reach a steady-state, which may exceed practical limits.

This computational burden has direct implications for both neuroscience and machine learning. From a neuroscience perspective, the brain performs perceptual inference faster than the thousands of sequential update steps that MCPC’s Langevin dynamics require to reach steady-state. The compatibility of MCPC with acceleration mechanisms, such as the excitatory and inhibitory recurrent circuits of Hennequin et al. [172] or momentum-like higher-order dynamics [182], is required for MCPC to be a plausible account of the fast sampling observed in the brain [81]. From a machine learning perspective, the number of inference steps per data point is the principal obstacle to scaling MCPC, as every parameter update is preceded by a full sampling phase. This contrasts with PC and with amortised models such as DLGMs, whose inference is comparatively inexpensive. The same bottleneck is shared by other Langevin- and energy-based generative models [144].

2.2.6.3 Mapping MCPC’s noise to sources of noise in the brain

Currently, mapping the noise variable in MCPC’s neural dynamics to distinct noise sources in the brain remains a challenge. Cortical circuits have various forms of stochasticity that could support the random dynamics of MCPC [184, 185]. However, the constraints on the noise variable within the dynamics of MCPC are minimal. MCPC does not mandate that the noise follow a particular distribution, nor does it specify a required noise level. Consequently, predicting which types of noise in the brain could facilitate the stochastic dynamics of MCPC proves challenging. To establish a more direct link between MCPC’s noise and its potential physiological origins, a spiking implementation of MCPC could be identified. Rethinking MCPC using spiking neural networks, as done in the spiking models of PC [186], might add

constraints on the location and type of noise needed. These constraints could create a clear connection to the physiological sources of noise.

2.3 Methods

2.3.1 Models

In this thesis chapter, we compare three methods for learning hierarchical Gaussian models: Monte Carlo predictive coding, predictive coding, and backpropagation in a deep latent Gaussian model (DLGM).

2.3.1.1 Monte Carlo predictive coding

Algorithm 1 shows the complete implementation of MCPC used for all the simulations. This algorithm is a discrete-time equivalent of MCPC where the dynamics of MCPC given in equation (2.3) are discretised using the Euler–Maruyama method. The algorithm contains a MAP inference before the MCPC inference to shorten MCPC’s mixing time during inference. This additional MAP inference is, however, not always beneficial, as discussed in Figure A.3. Moreover, the algorithm learns using mini-batches of data. Inference is performed independently for each element in a mini-batch and parameters are updated using the sum of the parameter updates across the mini-batch.

2.3.1.2 Predictive coding

We briefly review the predictive coding framework and its implementation used in this thesis. Following the formulation of predictive coding by Bogacz [99] which we refer to with PC, predictive coding learns a hierarchical Gaussian model. The model is learned by iterating over two steps that minimise the joint log-likelihood $E_{gen} = -\ln p(x, z; \theta) = \frac{1}{\sigma^2} \sum_{l=1}^L \|x_l - W_l \cdot f(x_{l+1})\|^2$ where x_1 is clamped to an observation x . First, PC uses neural dynamics that follow the gradient flow on E_{gen} to infer the Maximum a-posteriori estimate of the latent states conditioned on the observation:

$$\frac{\partial x_l(t)}{\partial t} = -\nabla_{x_l} E_{gen} = -\epsilon_l + f'(x_l) W_{l-1}^\top \epsilon_{l-1} \quad (2.14)$$

$$\epsilon_l = \frac{1}{\sigma^2} (x_l - W_l f(x_{l+1})); \quad \epsilon_L = \frac{1}{\sigma^2} (x_L - \mu) \quad (2.15)$$

Algorithm 1: Monte Carlo predictive coding (MCPC)

Require: L layers, activities x_1 to x_L , noise variance σ_n^2 , weights W_1 to W_{L-1} , joint log-likelihood E_{gen} , dataset $\{x^p\}_{p=1}^P$ with P mini-batches of B elements, number of epochs E , Euler step h , number of Euler steps K , number of mixing steps M , number of sampling steps S , and learning rate α .

```
for  $e = 1$  to  $E$  do
  for  $p = 1$  to  $P$  do
    // Independent inference for each sample in batch
     $x_1^b \leftarrow x^{p,b}, \quad 1 \leq b \leq B$ 
     $x_l^b \leftarrow n, \quad n \sim N(0, I), \quad 2 \leq l \leq L \text{ and } 1 \leq b \leq B$ 
    // MAP inference for faster steady-state
    for  $k = 1$  to  $K$  do
       $x_l^b \leftarrow x_l^b - h \frac{\partial E_{gen}^b}{\partial x_l^b}, \quad 2 \leq l \leq L \text{ and } 1 \leq b \leq B$ 
    // MCPC inference
    for  $i = 1$  to  $M + S$  do
       $x_l^b \leftarrow x_l^b - h \frac{\partial E_{gen}^b}{\partial x_l^b} + \sqrt{2h} n_l^b,$ 
       $n_l^b \sim N(0, \sigma_n^2 I), \quad 2 \leq l \leq L \text{ and } 1 \leq b \leq B$ 
       $z^{b,i} \leftarrow \{x_2^b, \dots, x_L^b\} \quad 1 \leq b \leq B$ 
    // Sum of parameter updates for batch
     $W_l \leftarrow W_l - \frac{\alpha}{S} \sum_{i=M+1}^{M+S} \sum_b^B \frac{\partial E_{gen}(x^{p,b}, z^{b,i}, \{W, \mu\})}{\partial W_l}, \quad 1 \leq l \leq L - 1$ 
     $\mu \leftarrow \mu - \frac{\alpha}{S} \sum_{i=M+1}^{M+S} \sum_b^B \frac{\partial E_{gen}(x^{p,b}, z^{b,i}, \{W, \mu\})}{\partial \mu}$ 
```

Algorithm 2: Predictive coding (PC)

Require: L layers, activities x_1 to x_L , weights W_1 to W_{L-1} , joint log-likelihood E_{gen} , dataset $\{x^p\}_{p=1}^P$ with P mini-batches of B elements, number of epochs E , Euler step h , number of Euler steps K and learning rate α .

```
for  $e = 1$  to  $E$  do
  for  $p = 1$  to  $P$  do
    // Independent inference for each sample in batch
     $x_1^b \leftarrow x^{p,b}, \quad 1 \leq b \leq B$ 
     $x_l^b \leftarrow n, \quad n \sim N(0, I), \quad 2 \leq l \leq L \text{ and } 1 \leq b \leq B$ 
    for  $k = 1$  to  $K$  do
       $x_l^b \leftarrow x_l^b - h \frac{\partial E_{gen}^b}{\partial x_l^b}, \quad 2 \leq l \leq L \text{ and } 1 \leq b \leq B$ 
    // Sum of parameter updates for batch
     $W_l \leftarrow W_l - \alpha \sum_b^B \frac{\partial E_{gen}^b}{\partial W_l}, \quad 1 \leq l \leq L - 1$ 
     $\mu \leftarrow \mu - \alpha \sum_b^B \frac{\partial E_{gen}^b}{\partial \mu}$ 
```

Second, PC updates the parameters with a gradient step on E_{gen} , evaluated on the converged MAP estimate x^* with error ϵ^* :

$$\Delta W_l \propto -\nabla_{W_l} E_{gen} = \epsilon_l^* f(x_{l+1}^*)^\top; \quad \Delta \mu \propto -\nabla_\mu F_{pc} = \epsilon_L^*. \quad (2.16)$$

These computations can be implemented in the same neural network with local computation and plasticity as MCPC. This is because PC only differs from MCPC through an additional noise term in the neural dynamics and an integral in the weight updates. Algorithm 3 shows the complete implementation of PC used for all the simulations. This algorithm is a discrete-time equivalent of PC, where the dynamics of PC given in equation (2.14) are discretized using Euler’s method, which consists of taking small discrete steps in the derivative direction.

2.3.1.3 Deep latent Gaussian models

We implement DLGMs as a benchmark model because they are the standard machine learning model for learning hierarchical Gaussian models. DLGMs, which were first proposed by Rezende et al. [147], consist of two main components: a generative model and an inference model. Each of these models is represented by a separate neural network. The generative model is responsible for generating samples, while the inference model approximates the posterior distribution over the latent variables given the observed data. To train this model, we utilize the reparameterization trick [147, 187]. This technique allows for the backpropagation of gradients through stochastic nodes, enabling efficient and accurate gradient-based optimization to learn the parameters of both the generative and inference models. However, DLGMs are not a biologically plausible model of generative learning in the brain. One major shortcoming is that the plasticity mechanisms used by DLGMs are not local. This study uses a modified version of the DLGMs implementation by Zhuo [188]. The implementation is modified so that the inference network learns a rank-one approximation of the covariance matrices of the posterior. This reduces the number of parameters of the inference network without significantly affecting the learning performance of DLGMs. To ensure a fair comparison with PC and MCPC, we employ DLGMs with generative models possessing a parameter count equivalent to that of PC and MCPC. Furthermore, the inference networks of DLGMs are constrained to maintain a parameter count equal to their generative counterparts.

2.3.2 Learning tasks

Two generative learning tasks are studied: a Gaussian learning task and a handwritten digit image learning task.

2.3.2.1 Gaussian learning task

In this task, the data has a Gaussian distribution that can be learned by the model in Figure 2.1A. This model is tractable, facilitating a comparison of MCPC’s steady-state inference with the marginal likelihood $p(x; \theta)$ and the posterior distribution $p(z|x; \theta)$. The model’s tractability also enables a direct comparison of the optimal parameters to the parameters learned by MCPC and PC. Equations (2.17) and (2.18) provide the data distribution and the model used for this task.

$$p(x) = N(x; \mu_{data} = 1, \Sigma_{data} = 5) \quad (2.17)$$

$$p(x, z; \theta) = N(x; W_1 x_2, \sigma^2 \sigma_n^2 I) N(x_2; \mu, \sigma^2 \sigma_n^2 I) \quad (2.18)$$

The marginal likelihood and the posterior distributions are given in equations (2.19) and (2.20).

$$p(x; \theta) = N(x; W_1 \mu, \sigma^2 \sigma_n^2 (W_1^2 + I)) \quad (2.19)$$

$$p(z|x; \theta) = \frac{p(x, z; \theta)}{p(x; \theta)} = \frac{N(x; W_1 x_2, \sigma^2 \sigma_n^2 I) N(x_2; \mu, \sigma^2 \sigma_n^2 I)}{I} N(x; W_1 \mu, \sigma^2 \sigma_n^2 (W_1^2 + I)) \quad (2.20)$$

Unless otherwise stated, both σ^2 and σ_n^2 are set to one. To evaluate steady-state neural activity with and without inputs, we employ 10,000 inference steps for MCPC and 2,000 steps for PC. The optimal parameter values, $\{W_{1,opt} = \pm \sqrt{\Sigma_{data}/\sigma^2 \sigma_n^2} - 1, \mu_{opt} = \pm \mu_{data} / \sqrt{\Sigma_{data}/\sigma^2 \sigma_n^2} - 1\}$, are identified by comparing the marginal likelihood to the data distribution. We train an MCPC and a PC model on this task using the parameters in Table 2.2.

2.3.2.2 MNIST learning task

In this task, the dataset comprises 28x28 binary images representing handwritten digits across ten categories. The model architecture and training parameters used for this task are determined using a hyperparameter search. Moreover, the model has been adapted to have a Bernoulli input layer. In contrast to the Gaussian learning

task, the model is intractable due to its hierarchical structure and non-linear activation functions, necessary for accurate learning. Consequently, direct assessment of inferences and model parameters is not feasible. Instead, we visualize the neural activity of MCPC and PC with and without inputs, we compare the neural activity of MCPC and PC with inputs to the inferences of an artificial ideal observer, and we measure the learning performance of the models using three metrics.

Model parameters. The model architecture and training parameters used for this task are determined using a hyperparameter search summarised in Table 2.2. The dataset includes 60,000 training images and 10,000 testing images, of which 6,000 images are used for hyperparameter tuning and 4,000 for evaluation. Appendix A.6 compiles the search results.

Bernoulli input layer. The model used for this task has been adapted to have a Bernoulli input layer. For binary images, the model’s input layer is transformed into a multivariate Bernoulli distribution. This modification yields the joint log-likelihood $E_{Bernoulli} = -[x^\top \ln(s(W_1 f(x_2))) + (1 - x^\top) \ln(1 - s(W_1 f(x_2)))] + \sum_{l=2}^L (x_l - W_l \cdot f(x_{l+1}))^2$ where s is a sigmoid function. This change does not compromise the biological plausibility of MCPC because it only introduces an additional non-linearity in the inference dynamics of x_2 and the parameter update for W_1 . This modification is shown below:

$$\begin{aligned} \frac{\partial x_2}{\partial t} &= -\nabla_{x_2} E_{Bernoulli} + n_0(t) \\ &= -\epsilon_2 + W_1^\top \epsilon_1 + n_1(t) \quad \text{with } \epsilon_1 = [x - s(W_1 f(x_2))] \\ \Delta W_1 &\propto \int_{t_0}^{t_0+T} -\nabla_{W_1} E_{Bernoulli} dt \\ &\propto \int_{t_0}^{t_0+T} \epsilon_1(t) f(x_2(t))^\top dt. \end{aligned}$$

Visualisation of MCPC’s and PC’s neural activity. We visualize MCPC’s and PC’s neural activity to assess the inference with and without inputs. We record and display the input neurons’ activity over time to visualize inferences without inputs. For the model trained on MNIST, the input Bernoulli sensory layer is discrete and cannot utilize Langevin dynamics. Therefore, we record the neural activity of the model excluding the Bernoulli sensory layer and display the input to the Bernoulli layer predicted by the first latent layers. Mathematically, this is represented as $s(W_1 f(x_2))$. To visualize the neural activity with inputs, we use a linear classifier that decodes

the digit class distribution from the latent state x_L . The classifier is trained on full images of the training data to transform MAP inferences of the latent state x_L to the corresponding digit classes. The digit classes are then assigned coordinates using a convex combination of 10 evenly spaced points on a unit circle [145], resulting in a two-dimensional visualization. In Figure 2.2, we visualize the inference for a full image part of the evaluation data and a partially masked version of the same image. For both visualizations, we use 10,000 inference steps for MCPC and 2,000 for PC.

Quantification of neural activity of MCPC and PC with inputs. We compare the neural activity inferred by MCPC and PC with inputs to the posterior inferred by an artificial ideal observer. This comparison quantifies how well MCPC and PC approximate the posterior distribution. The artificial ideal observer is a ResNet-9 classifier. We employ the same linear classifier as used for the visualizations to decode a digit class distribution from the neural activity of MCPC and PC. The decoded class distribution can then be compared to the digit class distribution inferred by the ideal observed. For PC, the digit class distribution is obtained by decoding the inferred latent state x_L at convergence. For MCPC, this distribution is obtained by decoding the fluctuating latent state x_L at steady-state and averaging the decoded distributions across MCPC samples. MCPC and PC are compared to the ideal observer by computing the KL divergence between the digit class distributions on the MNIST evaluation set with the top half of the images masked. For the random baseline, we compute the Kullback-Leibler divergence between the posterior distribution inferred by the ideal observer and the distributions inferred by both MCPC and PC, after these have been randomly shuffled. This shuffling results in the distributions inferred by PC and MCPC being associated with random inputs.

Performance metrics. We assess the learning accuracy of MCPC and PC using three metrics. Firstly, the Frechet Inception Distance (FID) evaluates the quality and diversity of generated images [152]. The FID is computed by comparing the evaluation images with 5000 generated images using a public FID implementation [189]. Secondly, we approximate the marginal log-likelihood for the evaluation images to assess a model’s generalization performance. The marginal log-likelihood is approximated

using the following Monte Carlo estimate from 5000 latent state samples:

$$\begin{aligned} -\ln p(x^{eval}; \theta) &= -\ln \prod_{i=1}^{4000} \int p(x^{i,eval}, z; \theta) dz \\ &\approx -\sum_{i=1}^{4000} \ln \sum_{s=1}^{5000} p(x^{i,eval}, z^s; \theta), \quad z^s \sim p(z; \theta) \end{aligned}$$

The samples to compute both the FID and the marginal log-likelihood are obtained using ancestral sampling. Finally, the reconstruction MSE measures the error between images and the images reconstructed by a model when inputted with the bottom half of the images. We calculate the error as the mean squared error between the top half of the original and reconstructed images for the MNIST evaluation set. The reconstructed images are obtained by performing MAP inferences of the missing image pixel values.

Visualising model features. We visualize the model features of all latent neurons for a model trained on MNIST with MCPC with hyperparameters that optimize the model’s FID. The feature represented by a neuron is determined by setting the activity of that neuron to 10 while keeping all other neurons in the same layer at zero activity. This approach is due to the use of ReLU non-linearities in the model, which disregard any activities less than or equal to zero. The chosen activity level of 10 for the analyzed neuron is based on experimental findings showing that activities of 10 or higher produce the same generated pattern. The neuron’s feature is obtained by propagating the modified layer activity forward through the model’s non-linearities and parameters until it reaches the input layer. For example, after setting the activity in layer x_3 the feature would be calculated as $s(W_1 f(W_2 f(x_3)))$.

Pixel Correlation We visualize and compare the absolute values of the Pearson correlation coefficients between pixels in MNIST images from our evaluation set and those generated by a trained MCPC model optimized for the best FID score. Pixels that are consistently zero in the MNIST evaluation set are excluded, as the correlation coefficient is undefined for these pixels. We compute the correlation coefficients for 4000 image samples generated using ancestral sampling from the MCPC model trained on MNIST.

2.3.3 Measuring cortical-like properties of MCPC’s neural activity

We measure two properties of MCPC models: the neural variability at stimulus onset, and the similarity between spontaneous and average evoked activity during training.

2.3.3.1 Evaluating neural variability

We first measure the temporal neural variability of the latent state activity at stimulus onset for MCPC and PC. This mimics the neural variability recording in the V1 region of cats done by Churchland et al. [156]. The model used for this experiment is trained on MNIST with MCPC to optimize the model’s FID. Moreover, we measure the neural variability around 256 stimuli onsets from the MNIST evaluation set. We measure the temporal neural variability by computing the standard deviation of the activity of all the latent states over a sliding window of 1000 timesteps. Then, we average the neural variability over all latent states for the 256 stimuli onsets. Churchland et al. [156] employed a 50-ms sliding window to estimate the variance in membrane potential of individual neurons. Then, they averaged the neural variability across the 52 recorded neurons and all stimuli to plot the mean change in neural variability at stimulus onset. Our approach replicates the experimental approach of Churchland et al. [156] for measuring neural variability of membrane potentials from cat V1.

2.3.3.2 Similarity of spontaneous and average evoked neural activity

Our method to measure the similarity of spontaneous and average evoked activity follows the approach used to measure this similarity in the V1 region of ferrets. Berkes et al. [137] recorded the spontaneous and average evoked neural activity with a linear array of 16 electrodes implanted in V1 of 16 ferrets (approximately four ferrets per age group). They measured the evoked activity for natural stimuli, sinusoidal gratings, and random noise. Moreover, they quantified the similarity in activity using the KL divergence. Our approach relies on similar sensory inputs and uses the same quantification metric. We measure the similarity between the spontaneous activity and the average evoked activity using a KL divergence in 10 MCPC models. We compute the KL divergence at different steps during training on MNIST as follows: First, we record the evoked activity of an MCPC model to (i) 256 samples from MNIST’s

evaluation set (natural stimuli), (ii) 256 samples of sinusoidal gratings with 16 possible orientations, and (iii) 256 samples of random binary images. We record from five randomly selected latent states in the first latent layer ($l = 2$) for 9,500 inference steps. Recording from five latent states reduces the computation time while maintaining representative results for the whole network. Moreover, recording from the first latent layer mirrors the V1 region, which is the first cortical area that processes visual information. Second, we record the spontaneous activity of the model for the same five latent states for 9,500 activity updates. Third, for each type of evoked activity, we compute the average experimental distribution of evoked activities across samples. Finally, we compare the three average distributions to the distribution of spontaneous activity using the KL divergence as implemented by Pérez-Cruz [97]. We repeat this procedure for 10 models trained on MNIST with MCPC to optimize the model’s FID. These models have the same architecture and learning parameters but they are initialized using a different seed. After, the KL divergence for natural stimuli is compared using paired samples t-tests to the KL divergence for gratings and for noise.

Table 2.2: **Parameters of MCPC, PC, and DLGMs.**

	Gaussian	MNIST
MCPC Inference		
optimizer	SGD	SGD
lr	0.02	{0.003, 0.01, 0.03, 0.1, 0.3}
mixing steps T	150	50
sampling steps M	1	100
PC Inference		
optimizer	Adam	Adam
lr	0.02	{0.03, 0.1, 0.3, 0.7}
max_steps	150	250
Architecture		
input dimension	1	748
number of latent layers L	2	4
dimension of x_2 to x_{L-1}	-	{128, 256, 360}
dimension of x_L	1	{10, 15, 20, 25, 30}
activation function	linear	{ReLU, tanh}
layer variance σ^2	1	1
noise variance σ_n^2	1	1
Learning		
optimiser	Adam	Adam
lr	0.02	{0.001, 0.003, 0.01, 0.03}
decay	0	{0, 0.01, 0.1, 1}
num_epochs	75	50
batch_size	256	{64, 128, 256}

The parameters are given for the linear task and the hyperparameter search space for the MNIST task. The parameters under MCPC Inference are only used for MCPC inference. The parameters under PC Inference are used for PC inference and the MAP inference in algorithm 1 implementing MCPC. The parameters under Architecture and Learning are shared by MCPC, PC, and DLGMs.

Chapter 3

Bidirectional predictive coding

3.1 Introduction

Visual inference plays a critical role in the brain, providing information processing for interpreting and interacting with the environment. Two frameworks have emerged to explain how the brain could implement visual inference. The first describes vision as a bottom-up discriminative process, where sensory stimuli are progressively filtered through layered neural architectures, predicting behaviourally relevant outputs [190]. This framework resembles inference in feedforward neural networks commonly employed in machine learning for image classification. The other framework formalises vision as a generative process, where the brain constructs a probabilistic model of sensory inputs [81]. From this perspective, the brain learns a top-down generative model with priors over incoming sensory activity, and neural responses arise from the Bayesian inversion of this model, estimating posterior probabilities of brain states given sensory information. Various neural implementations of this inversion have been proposed, including variational methods [95, 96] or sampling approaches [89, 161, 191]. Experimental evidence suggests that visual perception may be a combination of both frameworks [192, 193]. Discriminative models explain the rapid initial responses to visual stimuli through efficient bottom-up processing of the brain [193], whereas generative models capture the probabilistic computations critical for optimally integrating noisy sensory inputs with prior knowledge, as displayed in perception and behaviour [82, 83, 86, 87].

One computational model capable of capturing both generative and discriminative

processing modes in the brain is predictive coding (PC), as introduced in the first Chapter. In its generative formulation, PC accounts for a broad range of biological data from the visual system [126, 70, 127] and is successful in tasks such as associative memory [112] and image generation [1]. In its discriminative formulation, PC matches backpropagation-trained neural networks in image classification [103, 3] while also explaining neural data [118]. Crucially, PC relies on local computations and Hebbian learning rules consistent with biological constraints [51, 124, 125] and implementable in realistic neural circuits [99], making it a strong candidate for modelling both generative and discriminative learning in the brain. However, current PC formulations remain restricted to a single inference mode. Hybrid approaches that combine generative and discriminative versions exist [121] but sacrifice performance in at least one domain, leaving open how a single PC model could flexibly support both inference modes as required for biological vision.

In this Chapter, we propose bidirectional predictive coding (bPC), a novel model of biological vision. bPC provides a biologically grounded neural mechanism that explains how the brain can simultaneously perform generative and discriminative inference based on PC. We focus on evaluating and understanding bPC’s performance in a range of computational tasks, where PC models with either generative or discriminative inference have been individually successful. We propose bPC as a model of visual perception, but bPC can equally model other modalities with bidirectional processing. Our key contributions can be summarised as follows:

- We propose bidirectional predictive coding, a biologically plausible model of visual perception employing both generative and discriminative inference that naturally arises from minimising a single energy function;
- We show that bPC performs as well as its purely discriminative or generative counterparts on both supervised classification tasks and unsupervised representation learning tasks, and outperforms precedent hybrid models;
- We provide an explanation for the superior performance of bPC in both tasks, by showing that it learns an energy landscape that better captures the training data distribution than its unidirectional counterparts;
- We further show that bPC outperforms other PC models in two biologically relevant tasks, including learning in a bimodal model architecture and inference

with occluded visual scenes, indicating its potential as a more faithful model for visual inference in the brain.

3.2 Background and related work

Discriminative predictive coding. Recent research has explored PC models that employ bottom-up predictions from sensory inputs to latent neural activities [103, 118], analogous to traditional feedforward neural networks. These models are structured as hierarchical Gaussian models consisting of L layers, each characterized by neural activity x_l , where x_1 corresponds to the sensory input and x_L represents a label. PC learns this hierarchical Gaussian structure by minimizing an energy function corresponding to the negative joint log-likelihood of the model:

$$E_{disc}(x, V) = \sum_{l=2}^L \frac{1}{2} \|x_l - V_{l-1}f(x_{l-1})\|_2^2, \quad (3.1)$$

where x denotes the set of neural states from x_1 to x_L , the parameters V include the bottom-up weights V_l of each layer l , and f represents an activation function. We refer to this model as discriminative predictive coding (discPC) and illustrate it in the top panel of Figure 3.1A. In discPC, inference is performed by a forward pass from x_1 to x_l 's, and the learning rule of this model approximates BP, using only local computations and Hebbian plasticity [103]. Recent work showed that discPC performs comparably to BP in classifying MNIST, Fashion-MNIST and CIFAR-10 [3] and outperforms BP at learning scenarios encountered by the brain, such as online and continual learning [118]. However, discPC lacks unsupervised learning capabilities due to the non-uniqueness of the solution to its generative dynamics [129].

Generative predictive coding. Classically, PC uses top-down predictions from neural activities to sensory data, based on the hypothesis that the brain learns by minimizing the error between its predicted sensory inputs and the actual sensory input [70, 95]. With neurons arranged hierarchically, the negative joint log-likelihood can be written as:

$$E_{gen}(x, W) = \sum_{l=1}^{L-1} \frac{1}{2} \|x_l - W_l f(x_{l+1})\|_2^2, \quad (3.2)$$

where x_1 is set to sensory data and x_L can be clamped to a label for supervised learning or remain free for unsupervised learning. The parameters W_l are top-down

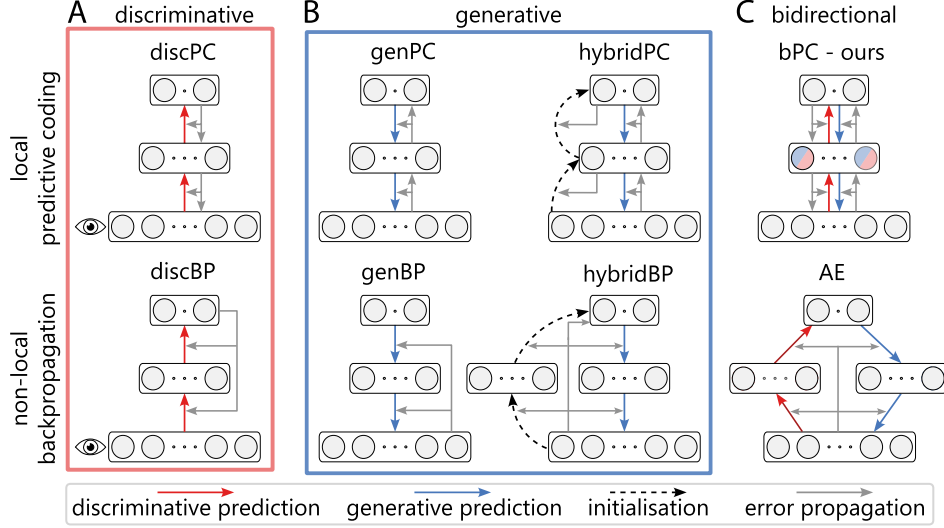


Figure 3.1: **Model architectures used in our experiments.** Arrows indicate the direction of prediction and error propagation. Dashed arrows represent initialisation. Discriminative models (red) are parametrised by bottom-up mappings from sensory inputs to brain states, generative models (blue) are parametrised by top-down mappings from brain states to sensory inputs, and bidirectional models incorporate both directions. Models in the top row employ local computations and error propagation that are considered biologically plausible, whereas those in the bottom row utilise non-local, backpropagation-based error computations that lack biological plausibility.

weights. We refer to this formulation of predictive coding as generative predictive coding (genPC). In genPC, inference of latent activities x_l is performed by clamping x_1 to sensory inputs and iteratively updating x_l 's via gradient descent on the energy E_{gen} . genPC's top-down predictions are illustrated in the top-left panel of Figure 3.1B. genPC has explained various visual phenomena, such as extra-classical receptive field effects and repetition suppression [70, 194, 195]. More recently, genPC has been employed to model associative memory [112] and unsupervised image generation [1, 180]. The computational framework of genPC can be implemented within neural networks that utilize local computations and Hebbian plasticity [99]. However, genPC has poor performance in supervised learning tasks [121].

Predictive coding models with mixed inference modes. In this work, we benchmark our bPC model primarily against hybrid predictive coding (hybridPC) [121]. In hybridPC, iterative inference proceeds in the same way as in genPC. However, an additional bottom-up network is introduced to provide a feedforward initialisation of neural activities. This network only sets the initial states and does not influence the subsequent dynamics, as illustrated in the top-right panel of Figure 3.1B. Although hybridPC performs both supervised and unsupervised learning,

its supervised performance falls short of discPC. In this work, we show that bPC performs supervised learning on par with discPC, and we provide an explanation of hybridPC’s inferior performance through the lens of bPC (see Appendix B.5). Sun and Orchard [129] noted the energy-minimising nature of PC models could theoretically allow image generation in discPC (equation 3.1), by clamping x_L to a label and iteratively updating x_1 . However, the generated images appear nonsensical due to the non-unique inferential dynamics solutions. Salvatori et al. [196] generalised the idea of Sun and Orchard [129] to a PC model where all neurons are interconnected with each other. However, it only slightly outperforms the classification performance of a linear classifier. Finally, Qiu et al. (2023) proposed a bidirectional PC model in which the connections between separate layers share the same weights, for instance, the bottom-up weights from layer $l - 1$ to l equal the top-down weights from layer l to $l + 1$. It is unlikely that the brain shares synaptic connections between separate layers of processing.

Discriminative and generative models of the brain. Cortical processing models are often divided into discriminative, which use feedforward networks to filter inputs [197, 198], and generative, grounded in the Bayesian Brain hypothesis [78, 81] and theories such as PC [70, 95] or adaptive resonance [199]. Growing evidence suggests the brain combines these approaches [200, 192, 193]: for example, it might employ a discriminative approach to rapidly extract important features, and perform Bayesian inference on these features to form high-level perceptions such as object categories [201]. Few models explicitly integrate both. The wake-sleep algorithm used to train the Helmholtz machine [79] is analogous to a generative-discriminative process. The Symmetric Predictive Estimator [202] models bidirectional processing, though only on toy tasks. A related bidirectional model based on information maximization [203] improves biological plausibility but requires two inference phases per update. Interactive activation models [204] also perform bidirectional inference, but do not incorporate a learning mechanism.

Discriminative and generative models in machine learning. Research in machine learning has likewise sought to integrate discriminative and generative pathways. Encoder–decoder architectures such as U-Nets [205] combine bottom-up and top-down processing via skip connections, while models including VAAE [206] and

VAR [207] unify both pathways within a shared latent representation. Other approaches, exemplified by PredNet [208], take more explicit inspiration from cortical circuitry to implement bidirectional information flow. Despite their strong empirical performance, these models depend on non-local learning signals and therefore do not provide a biologically plausible account of bidirectional learning in the brain.

3.3 Bidirectional predictive coding

In contrast to genPC and discPC, bPC neurons perform both top-down and bottom-up predictions, as shown in the top panel of Figure 3.1C. bPC achieves this bidirectional inference by unifying the energy functions of genPC and discPC into a single formulation, enabling both generative and discriminative prediction within the same circuit. Using the notations previously introduced, the energy function of bPC given by:

$$E(x, W, V) = \sum_{l=1}^{L-1} \frac{\alpha_{gen}}{2} \|x_l - W_l f(x_{l+1})\|_2^2 + \sum_{l=2}^L \frac{\alpha_{disc}}{2} \|x_l - V_{l-1} f(x_{l-1})\|_2^2, \quad (3.3)$$

where W_l are the top-down weights and V_l are the bottom-up weights. α_{gen} and α_{disc} are scalar weighting constants, which are needed to account for magnitude differences in the errors of bottom-up and top-down predictions. The weighting constants can be viewed as learnable precision parameters [95]; however, they are tuned and kept constant in our implementation for simplicity.

In each trial of learning, we first initialise the layers of neural activity using a feedforward sweep from layer x_1 to x_L along the bottom-up predictions, similar to discPC [103]. For instance, the second and third layers are initialised as $x_2 = V_1 f(x_1)$ and $x_3 = V_2 f(V_1 f(x_1))$ respectively. This initialisation strategy can be interpreted as a mechanism for fast amortised inference when a sensory input is initially encountered, which is also observed in the brain [209, 200]. All bPC experiments used this activity initialisation scheme. After, we update the neural activities to minimise E via several gradient descent steps (neural dynamics) following:

$$\frac{dx_l}{dt} \propto -\nabla_x E = -\epsilon_l^{gen} - \epsilon_l^{disc} + f'(x_l) \odot \left(W_{l-1}^\top \epsilon_{l-1}^{gen} + V_l^\top \epsilon_{l+1}^{disc} \right) + \mathcal{N}(0, \sigma^2 I), \quad (3.4)$$

where

$$\epsilon_l^{gen} := \alpha_{gen}(x_l - W_l f(x_{l+1})), \quad \epsilon_l^{disc} := \alpha_{disc}(x_l - V_{l-1} f(x_{l-1})) \quad (3.5)$$

denote the top-down and bottom-up prediction errors of neurons in layer l respectively. f' denotes the derivative of the function f , and $\odot$ is the element-wise product. The normally distributed noise $\mathcal{N}$ is zero-mean, temporally uncorrelated, and independent across neurons. By default, we set $\sigma^2 = 0$, yielding deterministic dynamics that converge to the maximum a posteriori estimate. Unless stated otherwise, all experiments in this chapter use these inference dynamics. Setting $\sigma^2 = 1$ induces stochastic dynamics that sample from the model's posterior [1]. These dynamics enable learning the distributions of sensory inputs as proposed in the previous chapter.

After updating neural activities, the weights are updated to minimise E via a single gradient descent step:

$$\Delta W_l \propto -\nabla_{W_l} E = \epsilon_l^{gen} f(x_{l+1})^\top, \quad \Delta V_l \propto -\nabla_{V_l} E = \epsilon_{l+1}^{disc} f(x_l)^\top. \quad (3.6)$$

3.3.1 Neural implementation

The computations described in equations (3.4) and (3.6) can be implemented in a neural network with fully local computation and plasticity, as illustrated in Figure 3.2. This network contains value neurons, which encode x_l , error neurons, which represent prediction errors, and synaptic connections, which encode the model parameters. All computations are local, relying solely on pre- and post-synaptic activity. Value neuron dynamics depend on local error signals, their own activity, and incoming synaptic weights. Similarly, error neuron activity depends only on adjacent value neurons and synaptic weights. The plasticity of the weights is also Hebbian, as it is a product of pre- and post-synaptic activity. The local implementation of bPC inherits that of genPC or discPC (shown in Figure 3.2 right and reviewed in the first Chapter), although it has two distinct error neurons per value neuron for bottom-up and top-down prediction errors.

3.3.2 Flexible learning

bPC can be trained both in supervised and unsupervised settings. In all cases, neurons in intermediate layers (layer 2 to $L - 1$) are un-clamped and evolve according to the neural dynamics in equation (3.4). In supervised settings, the first layer x_1 is clamped to the input, while the top layer x_L is clamped to the target labels. In unsupervised

settings, x_L is left unclamped, allowing bPC to learn compressed representations of the input. A mixed setting is also possible, where only a subset of neurons in x_L are clamped to label information, while others remain free. In this setting, the model can jointly infer labels and learn an compressed representation.

3.4 Experiments

3.4.1 bPC performs simultaneously well in classification and generation

In this experiment, we assessed bPC’s capacity for both classification (discriminative) and class-conditional image generation (generative). We compared bPC with discPC, genPC, hybridPC, and their backpropagation (BP) equivalents on MNIST and Fashion-MNIST, using identical architectures with two hidden layers of 256 neurons each. Additional baselines, including bPC with shared bottom-up and top-down weight as proposed by Qiu et al. [210], are provided in the Appendix B.1.

During training, the input layer x_1 was clamped to images and the output layer $x_L = x_4$ to their corresponding one-hot labels (Figure 3.3A). After training, discriminative performance was assessed by fixing x_1 to an input image and inferring the label at x_L over 100 inference steps. Generative performance was evaluated by clamping x_L to a class label and measuring the root mean squared error (RMSE) between the inferred image at x_1 (after 100 inference steps) and the average image for that class.

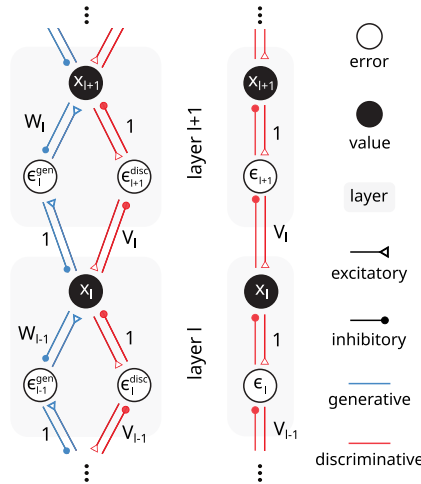


Figure 3.2: Neural implementation of bPC (left) and discPC (right).

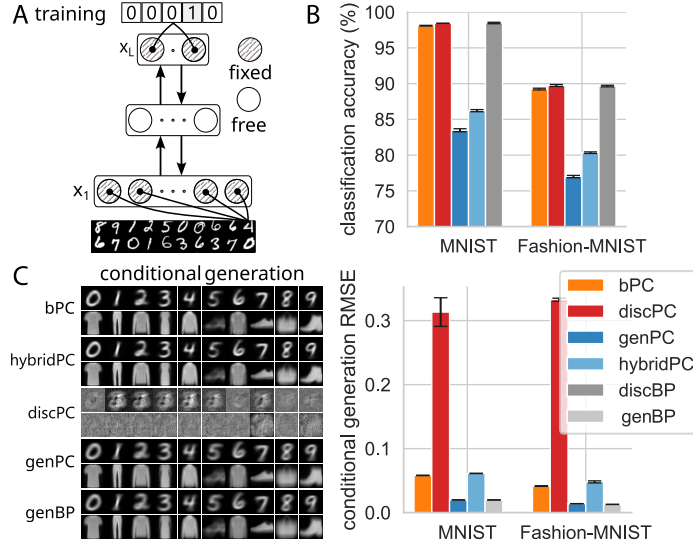


Figure 3.3: **bPC accurately classifies and generates class average images on MNIST and Fashion-MNIST.** A: Training set-up. The models are trained with x_1 fixed to images and x_L fixed to labels. B: Classification accuracy of models. C: Examples of the generated images conditional on class labels (left) and RMSEs between generated images and mean images of each class (right). Error bars denote the standard error of the mean (s.e.m.) across 5 seeds.

The results are shown in Figure 3.3B. On both datasets, bPC achieved classification accuracy comparable to discPC and discBP, while genBP and hybridPC performed less well. For generation, bPC obtained RMSE scores similar to genPC, hybridPC, and genBP, whereas discPC exhibited much higher errors. Visualizations in Figure 3.3C further illustrate these differences: discPC generated images with little class-relevant structure, while bPC and other bidirectional models produced clear and representative samples.

These findings align with previous reports that unidirectional PC models excel only in their specialized domain [129, 121], and demonstrate that bPC can integrate both discriminative and generative capabilities within a single framework. Notably, this integration uses the same number of error neurons but only half the value neurons required for maintaining separate unidirectional pathways for the two tasks. This indicates that bPC is more energy-efficient, especially as error signals can be handled in the dendrites of neurons rather than by a separate neuron population [111]. We observed that the discriminative weighting parameter (α_{disc}) must be set higher than the generative weighting (α_{gen}) due to the larger magnitude of top-down prediction errors. An exploration of the trade-off between α_{disc} and α_{gen} is provided in Appendix

B.2.

3.4.2 bPC performs unsupervised representation and distribution learning

Next, we show that bPC learns compressed representations and data distributions in the absence of supervision. We compared bPC, genPC, hybridPC, and their BP equivalents on MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100, excluding discPC since it cannot perform unsupervised learning. We clamped only x_1 to input images, leaving all other layers free (see Figure 3.4A). For MNIST and Fashion-MNIST, models used two hidden layers (256 neurons each) and a 30-neuron representation layer x_L , trained with only eight inference steps per update to test fast inference. CIFAR models had five convolutional layers and a 256-neuron representation layer, trained with 32 inference steps. An activity decay term at x_L stabilised learning and regularised representations. After training, representations were obtained from x_L using 100 inference steps. We evaluated these representations in two ways. First, reconstruction quality was measured by reinitialising the layers x_1 - x_{L-1} , clamping x_L , and running 100 inference steps, with the RMSE computed against the input. Second, linear readout classification accuracy was measured from x_L .

To assess distribution learning, we compared bPC, genPC, hybridPC, and a VAE [187] on MNIST, extending all PC models with stochastic dynamics to infer full posteriors ($\sigma^2 = 1$ in equation (3.4), see Section 3.7.5 for more details). Architectures matched those above, trained with 50 or 250 inference steps. Models were evaluated using Fréchet Inception Distance (FID). Samples were generated ancestrally by following top-down predictions starting from sampling x_L from its Gaussian prior and propagating downward through conditional Gaussians.

As shown in Figure 3.4B-E, bPC consistently outperformed genPC, yielding lower reconstruction RMSE, higher decoding accuracy, and better FID. It matched hybridPC across most tasks, and on CIFAR datasets significantly surpassed it in reconstruction error, approaching BP-based baselines. Reconstructions and generated samples (Figures 3.4C, E) illustrate these differences.

These results highlight that bPC outperforms genPC when inference steps are limited, and matches hybridPC despite the latter’s amortized pathway. This suggests

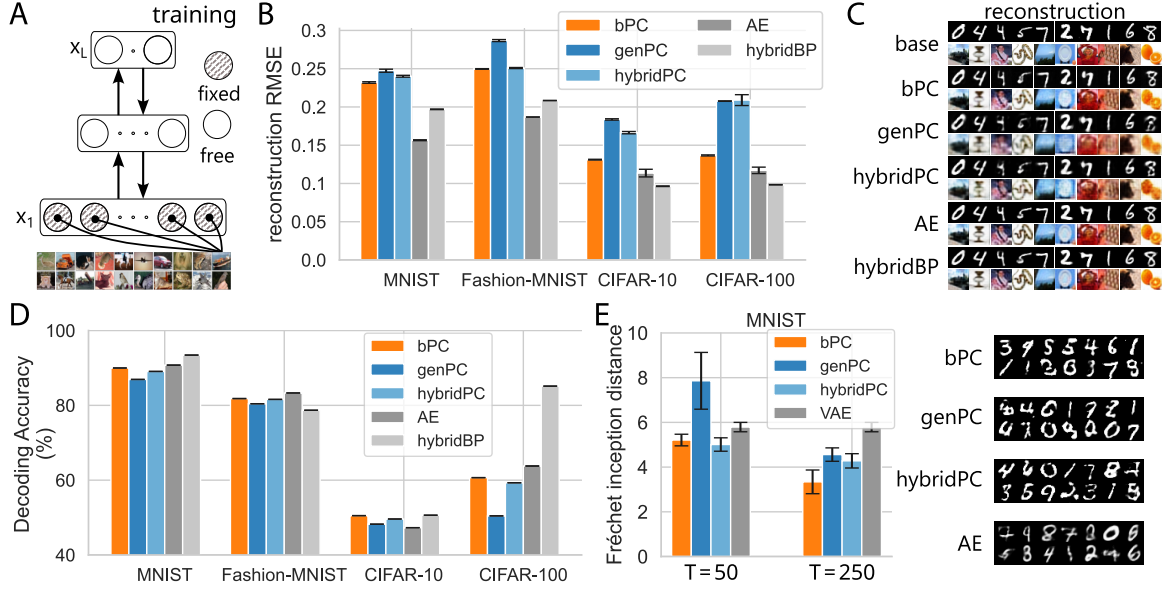


Figure 3.4: **bPC matches the performance of hybridPC at unsupervised learning.** A: Training set-up, where only x_1 is clamped to input images. B: Reconstruction RMSE from representations. C: Example reconstructions. D: Linear decoding accuracy from representations. E: Fréchet Inception Distance of samples generated by models with stochastic dynamics, with examples shown. T denotes the number of inference steps used during training. Error bars show s.e.m. across 5 seeds.

that bPC’s bottom-up pathway also serves as amortized inference, rapidly initializing neural activities toward the optimal state. The slight edge of bPC over hybridPC likely arises from the active involvement of bottom-up weights V_l during iterative inference, which continuously deliver sensory information to latent neurons. This effect becomes more pronounced with complex inputs such as CIFAR.

3.4.3 bPC performs combined supervised and unsupervised learning

In this experiment, we combined the supervised and unsupervised settings described above to test whether bPC can simultaneously develop discriminative capabilities and compact representations. This setting is motivated both computationally and biologically: real-world learning rarely occurs in isolation, and cortical circuits appear to integrate categorical supervision (e.g., from higher-order areas) with unsupervised structure learning from sensory input.

We trained bPC, hybridPC, and their BP equivalents on MNIST, Fashion-MNIST,

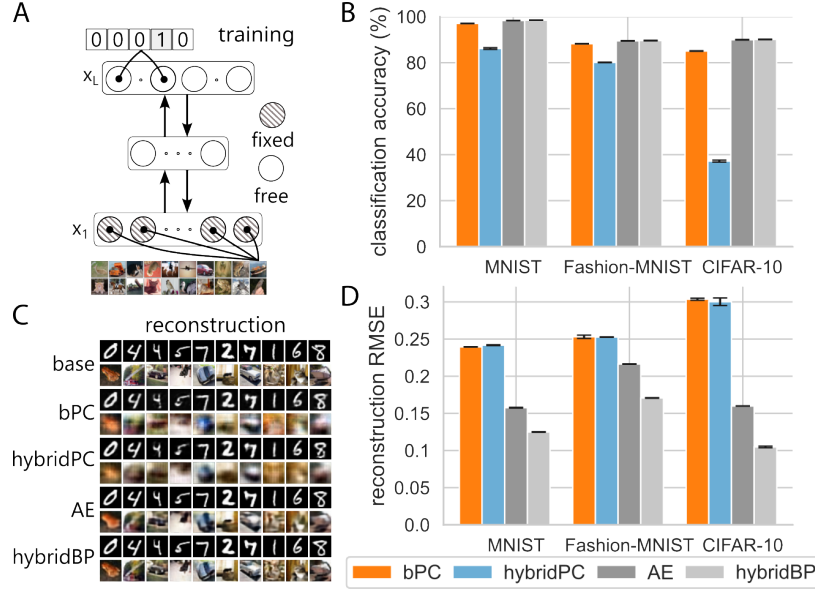


Figure 3.5: **bPC is the only PC model that can jointly learn low-dimensional representations of images and accurately classify them.** A: Training set-up, where the latent layer is only partially clamped to class labels. B: Classification accuracy. C: Example reconstructions on MNIST and CIFAR10. D: Reconstruction RMSEs. Error bars show s.e.m. across 5 seeds.

and CIFAR-10. During training, the input layer x_1 was clamped to images, while the top layer x_L was partially clamped to one-hot labels, leaving the remaining neurons free to learn complementary representations (Figure 3.5A). For MNIST and Fashion-MNIST, models had two hidden layers of 256 neurons each, and x_L comprised 40 neurons (10 for labels, 30 for representations). For CIFAR-10, we used a convolutional architecture with four hidden convolutional layers and a 266-neuron latent layer (x_L , with 10 label and 256 representation neurons). Activity decay was applied to the representational subpopulation in x_L to regularize learning. Classification accuracy was evaluated as in Section 3.4.1, by presenting only images and inferring labels at x_L . Generative quality was assessed as in Section 3.4.2, but with reconstructions conditioned jointly on the inferred representation and the label.

Figure 3.5B reveals that bPC achieves classification accuracy on par with BP models across all datasets. Furthermore, it significantly surpasses hybridPC, particularly on CIFAR-10, where hybridPC exhibits more than 45% lower accuracy than bPC. In terms of generative quality, Figure 3.5D shows that bPC achieves reconstruction errors similar to hybridPC across all datasets. The reconstructed examples in Figure 3.5C illustrate that bPC goes beyond generating class-average images, capturing

features such as shape, color, and spatial locations (e.g., it generates ‘4’s in different styles). However, fine details are absent in bPC’s CIFAR-10 reconstructions due to artefacts introduced by max-pooling operations in the discriminative bottom-up pathway, affecting the generative top-down reconstruction process. Consequently, bPC shows stronger grid-like artefacts than hybridPC, which uses only max-pooling for initialization, and than BP models with separate generative and discriminative layers. While removing max-pooling removes artefacts, it reduces classification accuracy (see Appendix B.3). Nonetheless, even without max-pooling, bPC’s classification accuracy significantly outperforms hybridPC, while generative performance remains comparable. Future research could refine bPC’s architecture to balance classification accuracy with artefact-free generation.

3.4.4 bPC’s shared latent layers prevent biased or overconfident energy landscape

In this experiment, we investigated why bPC can perform both discriminative and generative tasks effectively, focusing on the role of its shared latent layers in shaping the energy landscape (equation (3.3)). As a toy example, we considered the XOR problem, where the landscape can be directly visualized. We trained bPC, discPC, and genPC (each with two hidden layers of 16 neurons) by clamping x_1 to 2D inputs of XOR and x_4 to scalar outputs (Figure 3.6A). bPC used the same number of parameters as discPC and genPC combined but half the neurons.

An ideal XOR landscape has well-localized minima only at valid input-label pairs ($[0,1]$ and $[1,0]$ for one class; $[0,0]$ and $[1,1]$ for the other). Figure 3.6B top shows that discPC instead develops broad low-energy regions, indicating overconfidence even for implausible or out-of-distribution (OOD) inputs. genPC collapses each class into a single mean, failing to capture the true structure. A combined discPC+genPC model inherits both flaws. In contrast, bPC learns sharp, class-specific minima centred on valid inputs (Figure 3.6B bottom).

To test generality, we evaluated the models from Section 3.4.1 by sampling MNIST digits. We clamped x_L to a label, initialized x_1 randomly, and iterated inference until reaching energies within the lowest 10% of those observed on test images. For the combined discPC+genPC model, we required the generated samples to satisfy this low-energy criterion for both components. This set-up serves as a sampling process,

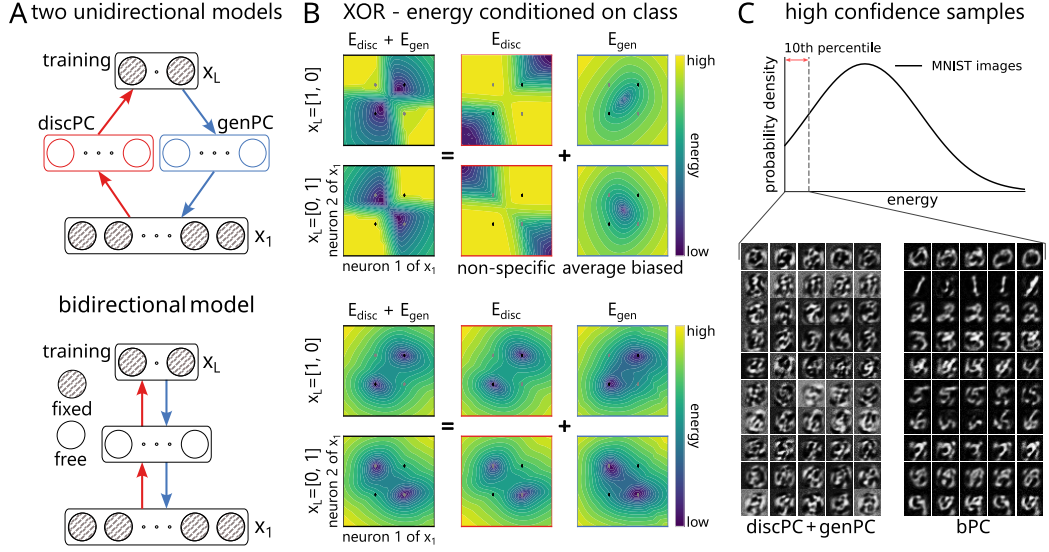


Figure 3.6: **bPC develops energy landscape suitable for both generation and classification.** A: We compare bPC (bottom) to training a discPC and a genPC separately (top). B: Visualisation of the models’ energy landscape after training on XOR. Shown are the discriminative, generative, and summed energies. C: MNIST samples considered highly likely (bottom 10% energy) by the models.

allowing us to inspect images considered as highly likely by the models. As shown in Figure 3.6C, bPC produced realistic digits, while the combined model yielded poorly formed shapes. Quantitatively, bPC achieved higher Inception Scores (6.05 ± 0.17 vs. 3.62 ± 0.03) and lower FID (44.4 ± 2.2 vs. 140.5 ± 2.1). Similar trends held for BP-based baselines (see Appendix B.4).

These results clarify the observations in Section 3.4.1. discPC fails at generation because it produces broad, overconfident minima that admit many nonsensical and OOD inputs, leading to the noisy conditional generations in Figure 3.3, similarly to Sun and Orchard [129]. genPC, by contrast, learns narrow minima around class means, which rejects OOD inputs but identifies deviations from training-set mean as high-energy states, ultimately sacrificing discriminative precision. bPC integrates the two: bidirectional predictions effectively regularize each other, sharpening discriminative minima while anchoring them to specific data points through the generative process. Shared latent layers thus prevent solutions overly biased to training mean or those that are overconfident, yielding superior performance in both classification and generation.

3.4.5 Learning in biologically relevant tasks

In this section, we demonstrate that bPC outperforms other predictive coding models in two biologically relevant scenarios: (1) learning in a model architecture with two input streams analogous to two sensory modalities, and (2) classifying images from partially occluded inputs, similar to how human vision includes regions with missing sensory information, such as the retinal blind spot.

Bimodal model architecture

The brain often develops neural representations through associations across modalities, such as linking spoken names to visual objects [211]. To test whether bPC can form such associations, we trained bPC and a bimodal variant of genPC on MNIST and Fashion-MNIST, with one latent layer connected to two inputs: an image and its one-hot label (Figure 3.7A; note that bPC is naturally bimodal and thus requires no restructuring). bPC incorporated both bottom-up and top-down connections for each modality, whereas genPC relied only on top-down pathways. After training, we evaluated cross-modal transfer by providing input to one modality and measuring inference quality in the other, classification accuracy for labels and RMSE for reconstructed images. As shown in Figure 3.7B-D, bPC significantly outperformed bimodal genPC on both tasks. This result is consistent with Sections 3.4.1: bPC’s bidirectional pathways naturally support associative coding, while genPC must be restructured to handle multimodal inputs, where one pathway predicts the image (similar to genPC) and the other predicts the label (similar to discPC) and inherits the weaknesses of both genPC and discPC.

Robustness to missing information

The cortex can recognize objects even when sensory input is incomplete or occluded [212]. We therefore tested the classification performance of the trained models from Section 3.4.1 (bPC, genPC, discPC, and discBP) under progressive occlusion of MNIST and Fashion-MNIST inputs, masking up to 90% of pixels (Figure 3.7E). Observed pixels were clamped in x_1 , while missing ones were left free and initialized to zero. We then run inference on the missing pixels and in latent layers, with extended iterations to accommodate slower convergence. Results (Figure 3.7F-G) show that bPC maintained high classification accuracy even with 80% missing pixels. genPC also retained some robustness but at consistently lower accuracy. In con-

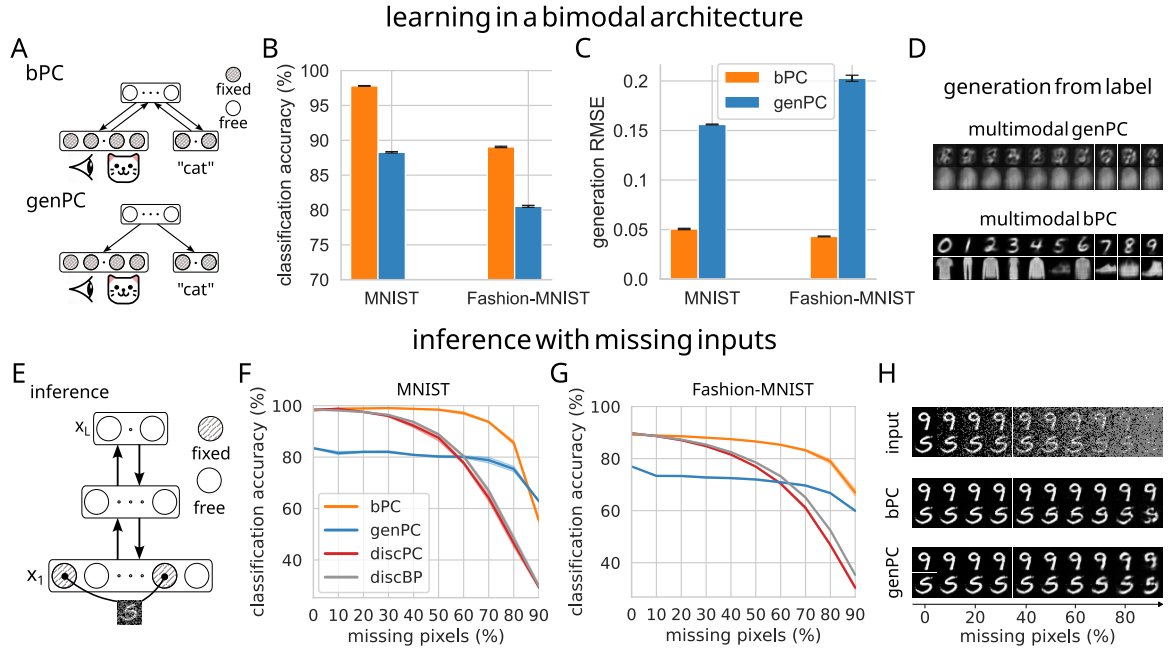


Figure 3.7: **bPC performance in a bimodal architecture and robustness against missing information.** A: Bimodal bPC and genPC. Note that bimodal bPC is fully equivalent to unimodal bPC in Section 3.4.1; it is bent here to serve visual purposes. B: Classification accuracy of bimodal PC models. C & D: Generation RMSE and examples. E: Model evaluation with partially occluded sensory inputs. F & G: Classification accuracy against percentage of missing pixels with MNIST and Fashion-MNIST respectively. H: Activity of x_1 after inference for genPC and bPC. Error bars and shaded region show s.e.m. across 5 seeds.

trast, discPC and discBP collapsed beyond 50% occlusion. Figure 3.7H reveals that bPC actively reconstructed missing inputs via top-down priors, integrating them with observed evidence. genPC shared this generative capacity but lacked strong discriminative accuracy, while purely discriminative models could not compensate for missing information at all.

3.4.6 Bidirectional processing in deep models

To investigate whether bPC scales effectively to deeper models and more complex datasets, we trained bPC, discPC and discBP using the following architecture-dataset combinations: VGG-5 on CIFAR-10, VGG-9 on CIFAR-100, and VGG-16 on Tiny-ImageNet. After training, we evaluated discriminative performance by measuring the model’s classification accuracy. Additionally, we assessed generative capabilities by evaluating classification accuracy when 30% and 50% of the input pixels were

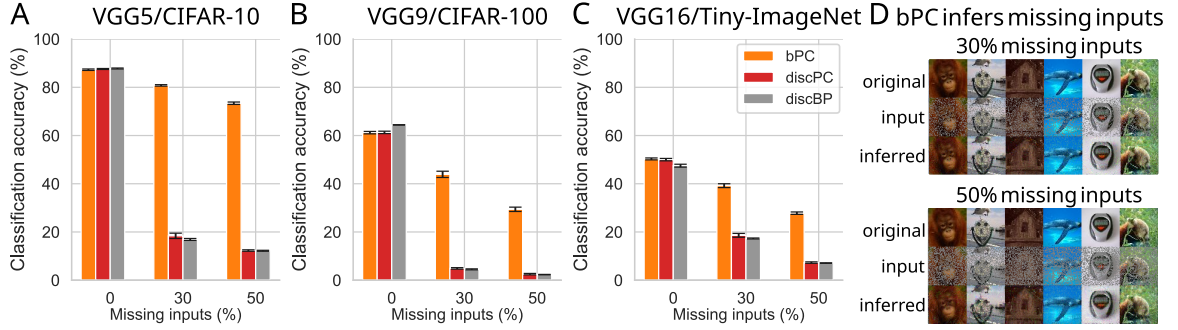


Figure 3.8: **bPC’s discriminative and generative properties scale to deep networks.** A & B & C: Classification accuracy against the percentage of missing pixels for bPC, discPC and discBP on the architecture-dataset combinations: VGG-5 on CIFAR-10, VGG-9 on CIFAR-100, and VGG-16 on Tiny-ImageNet. D: Activity of input neurons x_1 after inference for bPC trained on Tiny-ImageNet. Error bars show s.e.m. across 5 seeds.

missing, following the methodology outlined in Section 3.4.5. To efficiently simulate the predictive coding models in this experiment, we employed error optimisation [4]. This approach prevents energy decay in predictive coding models and enables the training of larger architectures. Refer to Section 3.9 for more implementation details.

Figure 3.8 confirms that bPC successfully combines discriminative and generative processing in deep networks. bPC achieves a classification accuracy comparable to discPC and discBP when full images are presented and significantly outperforms discPC and discBP on images with missing input information. For example, when 50% of pixels are missing, bPC classifies more than 60% more accurately than discPC and discBP on CIFAR-10 (Figure 3.8A). This improvement stems from bPC’s generative processing, which fills in missing information. Figure 3.8D illustrates this effect: after inference, the activity of the input layer x_1 reveals that neurons lacking inputs have been updated to predict the missing values. Once the missing information is inferred, bPC classifies images accurately. Overall, even in larger models, bPC effectively balances discriminative and generative performance by jointly minimising bottom-up and top-down prediction errors.

3.5 Discussion

3.5.1 Summary

Inspired by theoretical and empirical findings on visual processing, we propose bidirectional predictive coding, a biologically plausible computational model of visual inference that integrates generative and discriminative processing. We demonstrate that bPC performs effectively across both supervised classification and unsupervised representation learning tasks. Across both settings, it consistently outperforms or matches traditional predictive coding models. Our experiments reveal that the performance of bPC emerges from its ability to develop an energy landscape that is optimized for both discriminative and generative tasks. This dual optimisation improves its robustness to out-of-distribution data. Furthermore, we show bPC’s effectiveness in biologically relevant scenarios such as multimodal integration and inference with partially missing inputs. Overall, bPC offers a hypothesis for how flexible inference could emerge in the brain, while also providing a method to enhance the robustness of discriminative models in machine learning applications.

3.5.2 Hybrid model of the visual system

A central claim of this thesis is that bPC provides a unified model of vision that integrates discriminative and generative processing within a single architecture. This hybrid organisation mirrors a core principle of biological vision which combines two modes of computation: rapid initial inference followed by knowledge-guided refinement.

Human perception is characterised by both speed and flexibility. Familiar objects can be recognised extremely quickly, often within a single feedforward sweep through the visual hierarchy [209, 213]. However, when stimuli are degraded, noisy, or ambiguous recognition becomes slower [214, 215]. This dissociation suggests that visual processing combines both feedforward discrimination and top-down generation. On one hand, discriminative mechanisms enable fast mappings from sensory input to behaviourally relevant states through sequential bottom-up transformations. They are efficient and well suited for clear, unambiguous inputs. On the other hand, generative mechanisms incorporate prior knowledge to resolve ambiguity and fill in missing

structure. Through iterative refinement, they support more robust interpretation under challenging conditions. The brain might therefore be a hybrid model combining the complementary strengths of both approaches. Converging evidence reinforces the view that the visual system integrates both discriminative and generative processes [193].

bPC translates this hybrid principle into a computational model architecture. When a stimulus is presented, neural activity is first initialised via a bottom-up predictive stream, enabling rapid inference comparable to feedforward discriminative models. The model then engages generative dynamics that iteratively refine internal representations by incorporating accumulated knowledge about the input. The functional consequences of this hybrid design are most evident when information is incomplete. For partially occluded images, the initial feedforward pass yields limited accuracy, as illustrated by the low classification accuracy of the purely discriminative models, reflecting the constraints of purely discriminative inference. However, with iterative refinement, bPC progressively infers missing structure, sharpens its latent representations, and substantially improves performance. In doing so, it outperforms models restricted to feedforward computation alone. By integrating these complementary modes of computation, bPC captures the key organising principle of biological vision.

3.5.3 Populations of error neurons

Recordings from the mouse visual cortex provide important constraints on theories of predictive processing, which bPC might explain.

In a series of experiments, Jordan and Keller [216] performed whole-cell recordings from layer 2/3 (L2/3) excitatory neurons in the mouse’s primary visual cortex while animals navigated a virtual reality environment. Under normal conditions, visual flow was coupled to locomotion speed. Brief interruptions were introduced in which visual flow and locomotion were decoupled, generating a visuomotor mismatch. In a subsequent open-loop phase, locomotion and visual flow were decoupled so that visual stimuli could be presented independently of movement, and locomotion could occur without visual feedback. These manipulations revealed two functional response profiles within L2/3. One group of neurons showed depolarising responses to visuomotor mismatch. In the open-loop condition, these neurons depolarised with increasing locomotion speed but hyperpolarised in response to visual flow presented in isolation.

The second group exhibited the complementary pattern: they hyperpolarised during mismatch, depolarised to isolated visual flow, and hyperpolarised to increased locomotion without visual coupling. Together, these findings indicate that L2/3 neurons integrate locomotion-related and visual inputs with opposing signs, consistent with a subtractive interaction between top-down motor-related signals and bottom-up visual input. Jordan and Keller [216] interpreted these findings as evidence that L2/3 neurons encode prediction errors between top-down motor predictions and bottom-up sensory inputs. This conclusion is broadly consistent with PC models, such as those by Rao and Ballard [70]. In this view, the two populations correspond to neurons encoding positive and negative prediction errors separately. Alternative interpretations have shown that these results can be explained by temporally self-supervised mechanisms between cortical layers [217].

bPC offers a different explanation. In our framework, each layer contains two distinct error populations, but these do not encode positive and negative errors. Instead, they correspond to errors arising from two computational streams: top-down (generative) and bottom-up (discriminative) processing. As illustrated in Figure 3.2, top-down error neurons receive inhibitory inputs from top-down signals and excitatory inputs from bottom-up signals. Conversely, bottom-up error neurons receive excitatory input from top-down signals and inhibitory input from bottom-up signals. This organisation naturally produces two neuronal populations with opposing response profiles, consistent with the observations of Jordan and Keller [216]. Importantly, these two types of neurons encode errors from distinct processing streams rather than separate representations of positive and negative prediction errors. The data may therefore be interpreted as evidence for a hybrid architecture in which generative and discriminative computations coexist within the same cortical layer.

3.5.4 Future work

A key direction for future research is extending bPC to explicitly model temporal structure. The brain operates in a dynamically changing environment, where survival depends not only on interpreting current sensory inputs but also on anticipating future states. Integrating bPC with temporal PC [114] would enable the model to capture time-varying sensory dynamics. This extension would introduce an additional set of weights that predict future neural activity from past latent states, allowing the energy function to incorporate temporal prediction errors alongside spatial generative and

discriminative terms. Such a framework would provide an account of perception across time, and could be evaluated on video understanding, sequence modelling, and temporally structured multimodal tasks.

Another promising avenue concerns self-supervised learning. Self-supervised learning leverages the structure or patterns within the data as the target for learning. For example, the brain may learn largely without explicit external labels, but instead by drawing on structure shared across different modalities [218] and over time [217]. In this case, one sensory modality may serve as a teaching signal for another, and past sensory information may be used to predict future inputs. bPC is particularly well-suited to such settings because its generative–discriminative energy formulation avoids the trivial solutions often encountered in machine learning self-supervised objectives [219]. Therefore, exploring cross-modal and temporal self-supervised training within the bPC framework could bridge biologically plausible learning with the strong empirical performance of modern self-supervised models.

Overall, these extensions would bring bPC closer to modelling the full richness of biological learning while broadening its applicability to real-world machine learning problems.

3.6 Methods

This section is organised as follows: We begin with a detailed explanation of the methods underlying the experiments presented in this chapter. Next, we provide summary tables containing the exact values plotted in the figures. Finally, we evaluate whether bPC learns effectively in deep models.

3.7 Details of experiments

Here, we detail the training setup and the evaluation procedure of the experiments. Our implementation of the predictive coding models was adapted from Pinchetti et al. [3].

3.7.1 Datasets

We evaluate the models on four standard image classification benchmarks: MNIST[141], Fashion-MNIST[220], CIFAR-10, and CIFAR-100[221]. Below, we summarise their properties and how they were subdivided in our experiments for training, validation and testing. The images of all datasets were normalised so that pixel values varied between -1 and 1 .

- **MNIST**: A dataset of grayscale handwritten digits (0–9).
- **Fashion-MNIST**: A dataset of grayscale images of clothing items.
- **CIFAR-10**: A dataset of colour images categorised into 10 classes of everyday objects.
- **CIFAR-100**: Similar to CIFAR-10, but with 100 fine-grained object categories.

Table 3.1: Summary of dataset characteristics. The evaluation set of each dataset is split 50/50 into validation and test subsets.

Dataset	Image Size	Channels	Classes	Train	Validation	Test
MNIST	28×28	1	10	60,000	5,000	5,000
Fashion-MNIST	28×28	1	10	60,000	5,000	5,000
CIFAR-10	32×32	3	10	50,000	5,000	5,000
CIFAR-100	32×32	3	100	50,000	5,000	5,000

In addition to these benchmarks, we also consider the XOR learning task. XOR is a simple, non-linearly separable binary classification problem, often used to test the capacity of neural networks to learn non-linear functions. The input-output relationship of XOR is given in Table 3.2.

Table 3.2: XOR truth table with scaled inputs.

Input 1	Input 2	Output (XOR)
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

3.7.2 Model architectures

We consider five different neural architectures in our experiments, selected based on the dataset and the learning setting.

MLP for MNIST and Fashion-MNIST For MNIST and Fashion-MNIST in Sections 3.4.1, 3.4.2 and 3.4.3, we use a network with neural layers of dimension [784, 256, 256, latent_size]. The latent layer size is set to 10 for supervised experiments, 30 for unsupervised experiments, and 40 for experiments with partial clamping of the top-layer activity (x_4). Both generative and discriminative predictions between layers follow the form $f(Wx + b)$, where f is a non-linearity, W is a weight matrix, b is a bias vector, and x is the activation of the previous layer. This is the transformation of a layer in a multilayered perceptron (MLP). Consequently, unidirectional models (discPC, genPC, discBP, genBP) have the architecture of an MLP. hybridPC and bPC have the same number of parameters as two MLPs (one generative and one discriminative).

CNN for unsupervised learning with CIFAR-10 and CIFAR-100 In the unsupervised representational learning experiments (Section 3.4.2) on CIFAR-10 and CIFAR-100, we use a model architecture with convolutions and a representational layer $x_L = x_6$ of 256 neurons. In this model, discriminative predictions consist of five strided convolution layers, with a stride of two, and one MLP layer. Generative predictions mirror the discriminative layers with five transposed convolution layers, with a stride of two, and one MLP layer. The convolutional-based layers perform the transformation $f(c(x))$, where c is a (transposed) convolution layer and f is a non-linearity. A full description of the convolution-based layers can be found in Table 3.3.

VGG-5 for combined supervised and unsupervised learning with CIFAR-10

For the CIFAR-10 experiments that combine supervised and unsupervised representational learning in Section 3.4.3, we use a VGG-5-style network with a 266-dimensional latent layer $x_L = x_6$ (10 neurons for class labels and 256 for representations). In this model, discriminative predictions consist of four convolutional layers with max pooling and one MLP layer. The generative predictions consist of four transposed convolution layers and one MLP layer. The discriminative convolution layers have a stride of one, and the transpose convolutions have a stride of two to compensate for the dimensionality reduction of the max-pooling. A full description of the convolution-based layers

can be found in Table 3.3.

For the above three models, the activation function used in the discriminative prediction from x_{L-1} to x_L is an identity layer (no activation). The generative prediction from layer x_2 to x_1 uses a $\tanh(\cdot)$ activation to constrain outputs to the range $[-1, 1]$, matching normalised pixel values. The other activation functions are the same for the whole network and are determined using a hyperparameter search.

MLP for XOR Task For the XOR task in Section 3.4.4, we use a network with neural layers of dimension $[2, 16, 16, 1]$. The generative and discriminative predictions are equivalent to the ones used for MNIST and Fashion-MNIST, with the exception that: a sigmoid activation is used for the prediction from x_3 to x_4 and an identity activation is used for the prediction from x_2 to x_1 . These changes stabilise training and help avoid bias because the tanh activation used for (Fashion-)MNIST saturates near ± 1 , but never reaches these values, causing the model to systematically under-predict XOR inputs.

MLP for Bimodal Learning Task In the bimodal generation task of Section 3.4.5, the network consists of two input layers, with 784 (image) and 10 (label) neurons, and one latent layer with 256 neurons. The predictions between the latent layer and the input layers are MLP layers. The bimodal bPC model is equivalent to the bPC model trained on (Fashion-)MNIST in Section 3.4.1, except it only has one hidden layer. In contrast, the bimodal genPC is different from the models considered in Section 3.4.1, and it can be re-interpreted as a unidirectional model with one generative MLP layer and one discriminative MLP layer. For fair comparison, we include an additional non-linearity in the prediction layers of the genPC variant. The MLP layers become $f_1(Wf_2(x) + b)$. For the image modality, $f_1 = \tanh(\cdot)$; for the label modality, f_1 is the identity. The additional non-linearity, f_2 , ensures parity with bPC models, which include non-linear discriminative projections.

3.7.3 Training procedures

We followed the procedures outlined in Algorithm 3 to train predictive coding models and hybridBP, and the procedure outlined in Algorithm 4 for the remaining back-propagation models. For a given task, both algorithms were trained on the same dataset for the same number of epochs.

Algorithm 3: Training procedure of PC models.

Require: Model with neural activities x , parameters θ , energy E and initialisation function $init(\cdot)$. Dataset $\{y_p\}_{p=1}^P$ with P mini-batches of B elements. Number of epochs N . Activity optimiser $optim_x(\cdot)$, number of activity updates K , and parameter optimiser $optim_\theta(\cdot)$

```

for  $n = 1$  to  $N$  do
  for  $p = 1$  to  $P$  do
    // Independent inference for each sample in batch
     $x_b \leftarrow init(y_{p,b}), \quad 1 \leq b \leq B$ 
    for  $k = 1$  to  $K$  do
       $x_b \leftarrow optim_x(\frac{\partial E_b}{\partial x_b}), \quad 1 \leq b \leq B$ 
    // Sum of parameter updates for batch
     $\theta \leftarrow optim_\theta(\frac{1}{B} \sum_b^B \frac{\partial E_b}{\partial \theta})$ 

```

Algorithm 4: Training procedure of BP models.

Require: Model with parameters θ and forward pass $fp(\cdot, \theta)$. Energy function E . Dataset $\{y_p\}_{p=1}^P$ with P mini-batches of B elements. Number of epochs N . Parameter optimiser $optim_\theta(\cdot)$

```

for  $n = 1$  to  $N$  do
  for  $p = 1$  to  $P$  do
    // Independent forward pass for each sample in batch
     $\hat{y}_b \leftarrow fp(y_{p,b}, \theta), \quad 1 \leq b \leq B$ 
    // Sum of parameter updates for batch
     $\theta \leftarrow optim_\theta(\frac{1}{B} \sum_b^B \frac{\partial E(\hat{y}_b, y_{p,b})}{\partial \theta})$ 

```

Table 3.3: Detailed architectures of convolution-based models. Convolution and transposed convolution have the same kernel sizes and paddings. The values in the brackets give the parameter value for each convolution-based layer in the model, starting from the layer closest to the input image.

	CNN	VGG-5
Channel Sizes	[32, 64, 128, 256, 512]	[128, 256, 512, 512]
Kernel Sizes	[3, 3, 3, 3, 3]	[3, 3, 3, 3]
Paddings	[1, 1, 1, 1, 1]	[1, 1, 1, 1]
Strides conv.	[2, 2, 2, 2, 2]	[1, 1, 1, 1]
Strides transposed conv.	[2, 2, 2, 2, 2]	[2, 2, 2, 2]
Output Paddings transposed conv.	[1, 1, 1, 1, 1]	[0, 0, 0, 0]
Pooling Window	-	2×2
Pooling Stride	-	2

Optimisers

The optimisation procedures differ between activity updates and parameter updates. In Algorithm 3, neural activities are updated using stochastic gradient descent with momentum. This optimiser outperforms Adam in PC models [3]. In contrast, Algorithm 4 does not include activity updates and only optimises parameters. For parameter updates in both algorithms, we use the AdamW optimiser [222]. The optimiser hyperparameters were determined via a hyperparameter search.

Initialisation functions

The initialisation procedure in Algorithm 3 generally consists of (1) clamping the activity x_1 and x_L to images and labels depending on the learning task and (2) performing a feedforward sweep. This sweep initialises the activity of each layer by propagating inputs through the model in the direction of prediction. For example, a bottom-up feedforward sweep for a three-layer model initialises the second and third layers as: $x_2 = f(x_1)$ and $x_3 = V_2 f(V_1 f(x_1))$. This approach, originally proposed by Whittington and Bogacz [103], significantly reduces the time required for inference and improves learning. The following models use bottom-up feedforward initialisation: discPC, hybridPC, bPC, and hybridBP. The genPC model, by contrast, uses a top-down feedforward initialisation. The difference between hybridBP and the other models initialised using a bottom-up sweep is that hybridBP contains a single layer of neural activities, x_L , that can be initialised using a full bottom-up forward pass.

The bimodal genPC model from Section 3.4.5 differs from the other PC models in its

initialisation procedure. Because there are no direct prediction paths from the two modality-specific input layers to the shared latent layer, a standard feedforward sweep cannot be used to initialise the latent layer. Instead, we tested two alternatives for initialising the latent layer: zero-initialisation and Xavier uniform initialisation. The best initialisation depended on the task and was determined with a hyperparameter search. For the bimodal bPC model, we retained the feedforward sweep used for other bPC models, using a forward sweep from the image modality to the latent layer.

Energy functions

The energy functions for all PC and BP models are listed in Table 3.4. For PC models, the energy functions follow the formulations described earlier in this Chapter, except that we use a generalised notation: f_l denotes the transformation applied at layer l , allowing for models beyond standard MLPs. BP model energy functions use squared errors to remain consistent with PC formulations.

The hybridBP model includes a stop-gradient operation in its energy function. This ensures that its second loss term is only applied during parameter updates, not iterative inference. This loss term objective is to improve the discriminative initialisation of the model’s latent layer. In the broader machine learning literature, hybridBP can be viewed as a generative model that performs Bayesian inference by combining amortised inference with iterative refinement of the latent variables [223].

In the case of the autoencoder (AE), the energy function departs from standard autoencoders to accommodate experiments described in Section 3.4.3, where part of the latent state x_L is fixed to class labels. The AE energy includes an additional term: $\|x_{L,ae} - f_{p,disc}(x_1, V)\|_2^2$ where $x_{L,ae} := [x_{L,0:k-1}, f_{p,disc}(x_1, V)_{k:D}]$ that combines k fixed neurons and discriminative predictions. This term penalises the error between the fixed components of $x_{L,ae}$ and their prediction from the discriminative part of the model. Notably, this term is only non-zero for the fixed neurons. The additional energy term is also equal to zero in the representational learning experiments of Section 3.4.2 because none of the neurons of $x_{L,ae}$ are fixed. For reference, the discriminative part of our AE is usually referred to as the encoder and the generative part as the decoder.

In the combined supervised and unsupervised learning task (Section 3.4.3), we apply separate energy scaling to different parts of the models. Unsupervised learning benefits from equal weighting of discriminative and generative energies, while supervised

tasks perform better when the generative energy is down-scaled. To balance these, we set α_{disc} of the free (unclamped) neurons in x_L to α_{gen} . This allows for the generative energy to be down-scaled in the whole network, except for the free neurons of x_L , that specialise in unsupervised representational learning. This adjustment was used for bPC, hybridBP, and AE, but not for hybridPC. In hybridPC, the discriminative weights predicting free and fixed neurons are independent due to the use of local learning rules. Moreover, these discriminative weights have minimal influence on the generative weights, as inference is fully driven by the generative energy. As a result, scaling the discriminative and generative energies in different parts of the model merely scales the corresponding gradients. This effect has no impact on learning because the AdamW optimiser normalises gradients. In contrast, in hybridBP, some parameters are shared in predicting both free and fixed neurons. This introduces a dependency between the scaling of energies associated with free and fixed neurons in x_L . Therefore, we introduced explicit scaling factors α_{disc} and α_{gen} in hybridBP's energy function for this task.

Model	Loss Function
discPC	$E_{disc}(x, V) = \sum_{l=2}^L \frac{1}{2} \ x_l - f_l(x_{l-1}, V_{l-1})\ _2^2$
discBP	$\mathcal{L}_{disc}(x_1, x_L, V) = \frac{1}{2} \ x_L - f_{p, disc}(x_1, V)\ _2^2$
genPC	$E_{gen}(x, W) = \sum_{l=1}^{L-1} \frac{1}{2} \ x_l - f_l(x_{l+1}, W_l)\ _2^2$
genBP	$\mathcal{L}_{gen}(x_1, x_L, W) = \frac{1}{2} \ x_1 - f_{p, gen}(x_L, W)\ _2^2$
hybridPC	$E_{hybrid}(x, W, V) = \sum_{l=1}^{L-1} \frac{1}{2} \ x_l - f_l(x_{l+1}, W_l)\ _2^2 + \sum_{l=2}^L \frac{1}{2} \ \text{sg}(x_l) - f_l(\text{sg}(x_{l-1}), V_{l-1})\ _2^2$
hybridBP	$\mathcal{L}_{hybrid}(x_1, x_L, W, V) = \frac{1}{2} \ x_1 - f_{p, gen}(x_L, W)\ _2^2 + \frac{1}{2} \ \text{sg}(x_L) - f_{p, disc}(x_1, V)\ _2^2$
bPC	$E(x, W, V) = \sum_{l=1}^{L-1} \frac{\alpha_{gen}}{2} \ x_l - f_l(x_{l+1}, W_l)\ _2^2 + \sum_{l=2}^L \frac{\alpha_{disc}}{2} \ x_l - f_l(x_{l-1}, V_{l-1})\ _2^2$
AE	$\mathcal{L}(x_1, x_L, W, V) = \frac{\alpha_{gen}}{2} \ x_1 - f_{p, gen}(x_{L, ae}, W)\ _2^2 + \frac{\alpha_{disc}}{2} \ x_{L, ae} - f_{p, disc}(x_1, V)\ _2^2,$ $x_{L, ae} := [x_{L, 0:k-1}, f_{p, disc}(x_1, V)_{k:D}]$

Table 3.4: Energy functions for the predictive coding and backpropagation models considered in this chapter. In the PC models, $f_l(x, \theta)$ is the transformation between layers with input x and parameters θ . In the BP models, f_p denotes a forward pass and $x_{L, ae}$ is an activity vector composed of k fixed neurons and fills the remaining outputs of the discriminative forward pass of the AE.

3.7.4 Training hyperparameters

In this Section, we report the hyperparameter search space for the experiments in Sections 3.4.1, 3.4.2, 3.4.3, and 3.4.5, as well as the training parameters for the models in Section 3.4.4. All hyperparameter tuning was performed using Bayesian optimisation from Weights and Biases [224].

Tables 3.5, 3.6, and 3.7 list the hyperparameter search spaces for the models trained in Sections 3.4.1, 3.4.2, and 3.4.3, respectively. The hyperparameter search was conducted using the validation sets of each dataset. In Section 3.4.1, discPC and genPC were tuned for classification accuracy and generation RMSE separately, while bPC and hybridPC were jointly tuned on both metrics. This ensures a fair comparison, as bPC and hybridPC have twice as many parameters as discPC and genPC. For joint tuning, we combined the two metrics using the objective $2 \cdot (1 - \text{accuracy}/100) + \text{RMSE}$. In Section 3.4.2, the models were tuned for the image reconstruction MSE, linear decoding accuracy and generation FID separately. For the reconstruction, the RMSE was reported instead of the MSE for consistency with the generation RMSE. In Section 3.4.3, all models were jointly optimised for classification accuracy and reconstruction MSE. We combined the two metrics using $2 \cdot (1 - \text{accuracy}/100) + \text{MSE}$ for MNIST and Fashion-MNIST, and $(1 - \text{accuracy}/100) + 4 \cdot \text{MSE}$ for CIFAR-10. The RMSE was also reported after tuning instead of the MSE. The weightings in the combined objectives compensate for the scale difference between the metrics.

Table 3.8 reports the training parameters for the models used to generate MNIST samples in Section 3.4.4. The leaky ReLU activation function was used because it generated the best samples across model types, and the other parameters were set to default values that ensured stable learning across models.

Table 3.9 lists the hyperparameter search space for the bimodal bPC and genPC models trained in Section 3.4.5. The genPC model was tuned separately for classification accuracy and reconstruction RMSE, while the bPC model was tuned jointly on both metrics. The joint tuning followed the same combined metric as described above for MNIST. During training and evaluation, more inference steps were used for genPC than bPC, as genPC lacks an effective feedforward initialisation scheme, which slows inference convergence.

Table 3.5: Hyperparameter search configuration for experiments in Section 3.4.1 for both MNIST and Fashion-MNIST datasets.

Parameter	bPC	hybridPC	discPC	genPC	genBP	discBP
Epoch				25		
Batch Size				256		
Activation				[leaky relu, tanh, gelu]		
lr_x		$(1e-3, 5e-1)^2$		$(1e-4, 5e-2)^2$		-
momentum _x		[0.0, 0.5, 0.9]				-
lr_θ		$(1e-5, 1e-3)^2$		$(1e-6, 1e-4)^2$		$(1e-5, 1e-3)^2$
weight_decay _θ			$(1e-5, 1e-2)^2$			
T			8			-
T eval			100			-
α_{gen}	$1e[0,-4;-1]^1$		-			-
α_{disc}	1		-			-

¹: “[a, b; c]” denotes a sequence of values from a to b with a step size of c.

²: “(a, b)” represents a log-uniform distribution between a and b.

Table 3.6: Hyperparameter search configuration for experiments in Section 3.4.2 for MNIST , Fashion-MNIST, CIFAR-10, and CIFAR-100 datasets. Parameters unique to the models trained on MNIST and Fashion-MNIST are indicated with MLP. Parameters unique to the CIFAR datasets are indicated with CNN.

Parameter	bPC	genPC	hybridPC	hybridBP	AE
Epoch (MLP)			25		
Epoch (CNN)			50		
Batch Size			256		
Activation			[leaky relu, tanh, gelu]		
lr_x			$(1e-3, 5e-1)^1$		-
momentum _x			[0.0, 0.5, 0.9]		-
lr_θ (MLP)			$(1e-5, 1e-3)^1$		
lr_θ (CNN)			$(1e-5, 1e-3)^{1,2}$		
weight_decay _θ			$(1e-5, 1e-2)^1$		
T (MLP)			8		-
T (CNN)			32		-
T eval			100		-
α_{gen}	1		-	-	1
α_{disc}	1		-	-	1

¹: “(a, b)” represents a log-uniform distribution between a and b.

² Learning rates of θ were scaled with warmup-cosine-annealing scheduler without restart.

3.7.5 Evaluation

In our experiments, we consider six different evaluation procedures discussed below.

Table 3.7: Hyperparameter search configuration for experiments in Section 3.4.3. Parameters unique to the models trained on MNIST and Fashion-MNIST are indicated with MLP. Parameters unique to the CIFAR-10 dataset are indicated with VGG. Two activity optimisers and two parameter optimisers are used for the VGG models. This allows the models to have different learning rates for different task-specific parts of the networks. One activity optimiser was used for the free neurons of x_L with learning rate $lr_{x,free}$, and another for all other neurons with learning rate lr_x . Both activity optimisers use the same momentum parameter. One parameter optimiser with learning rate $lr_{\theta,gen}$ was used for the generative parameters and the discriminative parameters predicting the free neurons of x_L . The other parameter optimiser was used for the remaining discriminative parameters. Both parameter optimisers use the same weight decay.

Parameter	bPC	hybridPC	hybridBP	AE
Epoch (MLP)		25		
Epoch (VGG)		50		
Batch Size		256		
Activation		[leaky relu, tanh, gelu]		
lr_x		(1e-3, 5e-1) ¹		-
$lr_{x,free}$ (VGG)		(1e-3, 5e-1) ¹		-
momentum _x		[0.0, 0.5, 0.9]		-
lr_{θ} (MLP)		(1e-5, 1e-3) ¹		
$lr_{\theta,disc}$ (VGG)		(1e-5, 1e-3) ^{1,3}		
$lr_{\theta,gen}$ (VGG)		(1e-5, 1e-2) ^{1,3}		
weight_decay _{θ}		(1e-5, 1e-2) ¹		
T (MLP)		8		-
T (VGG)		32		-
T eval		100		-
α_{gen} (MLP)	1e[0,-4;-1] ¹	-	1e[0,-4;-1] ¹	1e[0,-4;-1] ¹
α_{gen} (VGG)	1e[-4,-7;-1] ¹	-	1e[0,-8;-1] ¹	1e[0,-8;-1] ¹
α_{disc}	1	-	1	1

¹: “(a, b)” represents a log-uniform distribution between a and b.

²: “[a, b; c]” denotes a sequence of values from a to b with a step size of c.

³ Learning rates of θ were scaled with warmup-cosine-annealing scheduler without restart.

Classification accuracy We evaluate classification performance using accuracy, defined as the percentage of input images correctly classified by each model.

For the PC models, classification is performed by first setting x_1 to the input image. We initialise the remaining layers using a bottom-up feedforward sweep, followed by 100 steps of iterative inference. The predicted class is determined by identifying the neuron in the output layer x_L (corresponding to class encoding units) with the highest activity. The bottom-up initialisation significantly reduces the time required to reach

Table 3.8: Hyperparameters for models used to sample MNIST images in Section 3.4.1.

Parameter	bPC	genPC	discPC
Epoch		25	
Batch Size		256	
Activation		leaky relu	
lr_x		0.01	
$momentum_x$		0.0	
lr_θ		1e-4	
$weight_decay_\theta$		5e-3	
T		8	
T eval		100	
α_{gen}	1e-4		-
α_{disc}	1		-

Table 3.9: Hyperparameter search configuration for bimodal models in Section 3.4.5 for both MNIST and Fashion-MNIST datasets.

Parameter	bPC	genPC
Epoch		25
Batch Size		256
Activation	[leaky relu, tanh, gelu]	
lr_x	(1e-3, 5e-1) ²	
$momentum_x$	[0.0, 0.5, 0.9]	
lr_θ	(1e-5, 1e-3) ²	
$weight_decay_\theta$	(1e-5, 1e-2) ²	
T	8	20
T eval	100	1000
α_{gen}	1e[0,-4;-1] ¹	-
α_{disc}	1	-

¹: “[a, b; c]” denotes a sequence of values from a to b with a step size of c.

²: “(a, b)” represents a log-uniform distribution between a and b.

steady state during inference. However, bottom-up initialisation does not apply to genPC and bimodal genPC due to the absence of bottom-up predictive pathways. In genPC, we initialise x_L to zero and perform a top-down feedforward sweep instead. For bimodal genPC, the latent layer is initialised identically to the procedure used during training.

For the BP models, classification is performed via a standard discriminative forward pass from the input image, with the predicted class taken as the one corresponding to the highest output activation.

Conditional generation of mean class images We evaluate the generative performance of the models by computing the RMSE between images generated for each class and the corresponding class-average image. We obtain the class-average image by computing the average image across all images belonging to that class in the evaluation set. To compute the RMSE, we calculate the squared error for each pixel between the generated image and the average image for the corresponding class. These errors are averaged across all pixels and classes, and the square root of this mean is reported as the final RMSE.

To generate an image from a PC model conditioned on a class label, we set the top-layer activity x_L to the one-hot encoding of the target label. The remaining layers are then initialised using a top-down feedforward sweep. One exception is the discPC model, which does not support top-down initialisation. For discPC, we initialise the input layer x_1 to zero activity and then perform a bottom-up feedforward sweep. Following initialisation, we run 100 steps of iterative inference and extract the final activity of the input layer x_1 as the generated image.

For the BP models, we generate an image by performing a generative forward pass starting from a one-hot label vector.

Image reconstruction from representations

We assess the quality of learned representations by measuring the RMSE between reconstructed images and their original inputs. RMSE is computed in the same manner as for conditional image generation.

For PC models and hybridBP, the reconstruction procedure is as follows: (1) clamp x_1 to the input image, (2) initialise the remaining layers via a bottom-up feedforward sweep, (3) perform 100 steps of iterative inference, (4) clamp x_L to its activity after the inference, (5) re-initialise the other layers using a top-down feedforward sweep, (6) run 100 additional inference steps, and (7) record the final activity in x_1 as the reconstructed image.

For the AE, an image is reconstructed by performing a discriminative forward pass using the input image to obtain a representation, followed by a generative forward pass from the representation to get the reconstructed image.

In Section 3.4.3, where labels are also provided during reconstruction, the above

procedure is slightly modified. For PC models and hybridBP, certain neurons of x_L are additionally clamped to the ground truth label during the first step. During the fourth step, the label components and the remaining latent representation (recorded after inference) are clamped in x_L . For the AE, we retain the standard discriminative forward pass, but replace the predicted label portion of its representation with the true label before passing it through the generative model, to produce the reconstructed image.

Linear readout/decoding accuracy

We further evaluate the quality of the learned representations by measuring their linear readout accuracy. The representations are obtained in the same way as for image reconstruction. A linear classifier head is then trained with backpropagation for each model to classify images based on their representations. This evaluation tests whether representations of different classes are linearly separable. Higher decoding accuracy reflects better representations.

FID of generated image samples

We compare the ability of bPC, genPC, and hybridPC models to learn probability distributions using stochastic extensions of PC. PC can be equipped with stochastic neural dynamics to model distributions of sensory inputs in an unsupervised manner by injecting noise into the inference process. This idea was originally proposed in [1] for the genPC model, and here we apply it to genPC, hybridPC, and bPC.

When stochastic dynamics are applied, the noisy inference process can be shown to generate samples from the posterior distribution of the probabilistic model defined by PC, conditioned on the sensory input. These posterior samples can then be used for parameter learning through Monte Carlo Expectation Maximisation.

The training procedure mirrors that of the deterministic experiments: each iteration consists of several (now noisy) activity updates followed by a single parameter update. This is equivalent to using a single posterior sample per parameter update, a common practice in related generative models such as variational AE.

To accelerate inference, we further incorporate momentum into the neural dynamics, as introduced for PC in [3]. The resulting dynamics correspond to a discrete-time second-order Langevin process. In practice, this is implemented by adding Gaussian

noise to the gradients of the energy function and updating activities with stochastic gradient descent with momentum:

$$\Delta x_l = lr_x r_l, \quad (3.7)$$

$$\Delta r_l = -lr_x \nabla_{x_l} E - lr_x (1 - m) r_l + \sqrt{2(1 - m)lr_x n_l}. \quad (3.8)$$

After training, the models can be used to generate input samples. For a top-down predictive coding model, this is done by ancestral sampling: first sampling the top latent layer $x_L \sim N(0, I)$, and then recursively sampling each lower layer from its conditional Gaussian distribution $N(x_l; f_l(x_{l+1}, W_{l+1}), I)$. The prior distribution on x_L emerges from the additional activity decay applied to x_L for the unsupervised learning experiments.

To evaluate generative performance, we compute the Fréchet Inception Distance between generated and real samples. We use the open-source library pytorch-fid [189], adapted to MNIST by replacing the standard Inception network with a ResNet-18 trained on MNIST as the feature extractor.

As a baseline, we include a VAE trained on the same learning task. The VAE architecture and parameter count are matched to those of hybridPC and bPC to enable a fair comparison. Training follows the standard VAE procedure: the encoder network approximates the posterior distribution over latent variables, while the decoder parametrises the generative model. The model is optimised end-to-end via backpropagation using the standard VAE objective.

Visualisation of Energy Landscapes on XOR

We visualise the energy landscapes of trained genPC, discPC, and bPC models on the XOR task. To generate the landscape, we clamp the input layer neurons to a 2D coordinate within the range $[-3, 3]$, sampled at intervals of 0.25 along both axes. Simultaneously, we clamp the output layer x_L to the one-hot encoding of one of the two classes. For each coordinate-label combination, we run 10,000 steps of iterative inference to ensure convergence to equilibrium and record the final energy of the model. This procedure is repeated over the grid of 2D inputs and both class labels, allowing us to plot the full energy landscape. For fair comparison, we also include the combined energy landscape of discPC and genPC. Together, the combined parameter count of discPC and genPC matches the total parameter count of bPC.

Equal weighting is applied to the energy values of the discPC and genPC models when adding their energy, as we use equal weighting for the discriminative and generative components in the bPC model ($\alpha_{disc} = \alpha_{gen} = 1$).

MNIST image generation with PC models

We evaluate the generative capabilities of the bPC model and a combined genPC and discPC model on the MNIST dataset. The goal is to assess how likely these models are to assign low energy to implausible, label-inconsistent samples.

As a baseline, we first estimate the energy distribution over the test set for each model. For each test image, we clamp x_1 to the image and x_L to its associated label. The model is then initialised as during training, and we run 50,000 steps of iterative inference to reach a steady state. We compute the 10th percentile from the resulting energy values as an energy threshold for high-quality, in-distribution samples. This process was repeated independently for bPC, genPC and discPC.

To generate images conditioned on a label, we use the following procedure: (1) randomly initialise the input layer x_1 by sampling each neuron’s activity uniformly from the interval $[-1,1]$, (2) clamp x_L to the target label, (3) initialise all hidden layers to zero, (4) run iterative inference until the median energy of the batch falls below the previously determined 10th percentile threshold, (5) retain the 50% of samples within the batch that fall below or equal to the energy threshold and discard the rest. Using this method, we generate 256 samples per model.

In the combined genPC and discPC model, the input layer x_1 is shared by both models, and its updates during inference are influenced by both energy functions. However, genPC typically exhibits energy magnitudes significantly larger than those of discPC. To prevent one component from dominating the updates to x_1 , we scale the generative energy by a factor equal to the ratio of the discPC energy threshold to the genPC energy threshold. Additionally, inference is only terminated once the generative and the discriminative energies fall below their respective thresholds.

The models used for sample generation are trained separately from those used in the supervised experiments from Section 3.4.1. During preliminary testing, we found that generative quality strongly depends on the choice of activation function. In particular, leaky ReLU yielded superior image samples. As a result, we train all models using leaky ReLU activations and a shared set of training hyperparameters.

These parameters are selected for their stability and effectiveness across all model types for classification and generation tasks.

We assess the quality and diversity of the generated samples using two standard metrics:

- **Fréchet Inception Distance:** The FID measures the distance between the distributions of real and generated images [152]. We use the public implementation of FID from Seitzer [189], but modify it to use a ResNet classifier trained on MNIST, instead of the original Inception model trained on ImageNet. This ensures that the FID score better reflects visual quality and diversity on the MNIST domain.
- **Inception Score (IS):** The Inception Score evaluates how easily a classifier can identify the class of a generated image [225]. We used a publicly available implementation for the MNIST dataset [226].

Classification accuracy with partially missing inputs

To assess model robustness to missing input data, we evaluate classification accuracy under varying levels of input occlusion. This evaluation follows the same procedure as standard classification (described above), but for images missing a random subset of pixels. The proportion of missing pixels ranges from 10% to 90%, in increments of 10%. Missing pixels are selected uniformly at random, independently of their spatial location within the image. For all models, missing pixels in x_1 are initialised to zero activity and are left unclamped, allowing the model to update their values during inference. We repeat the same classification procedure as before, but we increased the number of inference steps to 600,000 to ensure convergence. This is necessary because PC models converge more slowly when input information is incomplete [102]. In addition to measuring classification accuracy, we also record the post-inference activity of neurons in x_1 to qualitatively assess the model’s ability to fill in the missing input.

3.8 Tabular results

Tables 3.10 to 3.15 report the result illustrated in Figures 3.3, 3.4, 3.5, and 3.7. These results are obtained on the test set of the datasets for five different weight initialisations.

Table 3.10: Classification accuracy and class average image generation for models considered in Section 3.4.1. Higher accuracy and lower RMSE are better. We report the mean +/- sem over five seeds. Results indicated with * are significantly worse in performance than bPC determined using an independent-samples t-test ($n=5$, $p < 0.05$).

Model	Acc %		RMSE	
	MNIST	Fashion-MNIST	MNIST	Fashion-MNIST
bPC	98.10 $\pm$ 0.05	89.24 $\pm$ 0.12	0.0581 $\pm$ 0.0004	0.0415 $\pm$ 0.0005
hybridPC	86.22 $\pm$ 0.15*	80.34 $\pm$ 0.11*	0.0612 $\pm$ 0.0003	0.0480 $\pm$ 0.0016
genPC	83.48 $\pm$ 0.21*	77.00 $\pm$ 0.16*	0.0198 $\pm$ 0.0001	0.0140 $\pm$ 0.0001
discPC	98.43 $\pm$ 0.01	89.74 $\pm$ 0.14	0.3133 $\pm$ 0.0224*	0.3326 $\pm$ 0.0024*
BP	98.48 $\pm$ 0.10	89.66 $\pm$ 0.11	0.0198 $\pm$ 0.0001	0.0128 $\pm$ 0.0001

Table 3.11: Image reconstruction RMSE from latent representations for models considered in Section 3.4.2. Lower RMSE is better. We report the mean +/- sem over five seeds. Results indicated with * are significantly worse in performance than bPC determined using an independent-samples t-test ($n=5$, $p < 0.05$).

Model	MNIST	Fashion-MNIST	CIFAR-10	CIFAR-100
bPC	0.2320 $\pm$ 0.0010	0.2497 $\pm$ 0.0004	0.1311 $\pm$ 0.0005	0.1366 $\pm$ 0.0007
genPC	0.2473 $\pm$ 0.0020*	0.2868 $\pm$ 0.0013*	0.1837 $\pm$ 0.0009*	0.2077 $\pm$ 0.0003*
hybridPC	0.2401 $\pm$ 0.0012*	0.2508 $\pm$ 0.0007	0.1664 $\pm$ 0.0015*	0.2089 $\pm$ 0.0071*
AE	0.1565 $\pm$ 0.0006	0.1868 $\pm$ 0.0001	0.1135 $\pm$ 0.0050	0.1171 $\pm$ 0.0042
BP	0.1969 $\pm$ 0.0004	0.2084 $\pm$ 0.0002	0.0964 $\pm$ 0.0004	0.0983 $\pm$ 0.0002

3.9 Scaling bPC to deeper models

To investigate whether bPC scales effectively to deeper models and more complex datasets, we trained discPC, discBP, and bPC using the following architecture-dataset combinations: VGG-5 on CIFAR-10, VGG-9 on CIFAR-100, and VGG-16 on Tiny-ImageNet.

Table 3.12: Linear decoding accuracy (%) across datasets for different models considered in Section 3.4.2. Higher is better. We report the mean $\pm$ sem over five seeds. Results indicated with * are significantly worse in performance than bPC determined using an independent-samples t-test ($n=5$, $p < 0.05$).

Model	MNIST	Fashion-MNIST	CIFAR-10	CIFAR-100
bPC	89.99 $\pm$ 0.01	81.82 $\pm$ 0.01	50.52 $\pm$ 0.01	60.68 $\pm$ 0.01
genPC	86.93 $\pm$ 0.01*	80.44 $\pm$ 0.01*	48.23 $\pm$ 0.02*	50.45 $\pm$ 0.02*
hybridPC	89.07 $\pm$ 0.01*	81.59 $\pm$ 0.01	49.61 $\pm$ 0.01*	59.31 $\pm$ 0.01*
AE	90.76 $\pm$ 0.01	83.31 $\pm$ 0.01	47.29 $\pm$ 0.02	63.78 $\pm$ 0.04
hybridBP	93.44 $\pm$ 0.01	78.69 $\pm$ 0.018*	50.63 $\pm$ 0.01	85.14 $\pm$ 0.01

Table 3.13: FID scores for different models trained with 50 and 250 activity updates before each weight update considered in Section 3.4.2. Lower is better. We report the mean $\pm$ sem over five seeds. The VAE only has one value because it does not have iterative inference.

Model	FID @ 50 updates	FID @ 250 updates
bPC	5.21 $\pm$ 0.26	3.34 $\pm$ 0.53
genPC	7.86 $\pm$ 1.27	4.56 $\pm$ 0.30
hybridPC	5.01 $\pm$ 0.30	4.28 $\pm$ 0.32
VAE	5.79 $\pm$ 0.21	

Table 3.14: Classification accuracy and image reconstruction RMSE from latent representations for models considered in Section 3.4.3. Higher accuracy and lower RMSE are better. We report the mean $\pm$ sem over five seeds. Results indicated with * are significantly worse in performance than bPC determined using an independent-samples t-test ($n=5$, $p < 0.05$).

Model	Acc %			RMSE		
	MNIST	Fashion-MNIST	CIFAR-10	MNIST	Fashion-MNIST	CIFAR-10
bPC	97.06 $\pm$ 0.08	88.24 $\pm$ 0.08	85.06 $\pm$ 0.11	0.2394 $\pm$ 0.0002	0.2531 $\pm$ 0.0022	0.3036 $\pm$ 0.0014
hybridPC	86.15 $\pm$ 0.36*	80.11 $\pm$ 0.11*	37.20 $\pm$ 0.43*	0.2418 $\pm$ 0.0006	0.2527 $\pm$ 0.0003	0.3003 $\pm$ 0.0051
AE	98.32 $\pm$ 0.06	89.49 $\pm$ 0.10	89.97 $\pm$ 0.12	0.1576 $\pm$ 0.0005	0.2164 $\pm$ 0.0003	0.1599 $\pm$ 0.0002
hybridBP	98.48 $\pm$ 0.03	89.61 $\pm$ 0.13	90.11 $\pm$ 0.11	0.1249 $\pm$ 0.0004	0.1707 $\pm$ 0.0004	0.1048 $\pm$ 0.0011

After training, we evaluated the discriminative performance of the models by measuring classification accuracy. Additionally, we assessed generative capabilities by evaluating classification accuracy when 30% and 50% of the input pixels were missing, following the methodology outlined in Section 3.4.5.

Data Preparation

Table 3.15: Classification accuracy and class average image generation for bimodal genPC and bPC considered in Section 3.4.5. Higher accuracy and lower RMSE are better. We report the mean $\pm$ sem over five seeds. Results indicated with * are significantly worse in performance than bPC determined using an independent-samples t-test ($n=5$, $p < 0.05$).

Model	Acc %		RMSE	
	MNIST	Fashion-MNIST	MNIST	Fashion-MNIST
bPC	97.80 $\pm$ 0.05	89.05 $\pm$ 0.10	0.0506 $\pm$ 0.0007	0.0431 $\pm$ 0.0004
genPC	88.28 $\pm$ 0.10*	80.53 $\pm$ 0.14*	0.1561 $\pm$ 0.0004*	0.2027 $\pm$ 0.0031*

We used the CIFAR-10 and CIFAR-100 datasets [221]. For discPC and discBP, images were rescaled to the range $[0, 1]$ and normalized using the mean and standard deviation shown in Table 3.16, consistent with Pinchetti et al. [3]. For bPC, images were rescaled to the range $[-1, 1]$ to align with the effective output range of the tanh activation function, which served as the output activation for the top-down predictions across all bPC models.

Table 3.16: Data normalization

	Mean (μ)	Std (σ)
CIFAR-10	[0.4914, 0.4822, 0.4465]	[0.2023, 0.1994, 0.2010]
CIFAR-100	[0.5071, 0.4867, 0.4408]	[0.2675, 0.2565, 0.2761]
Tiny-ImageNET	[0.485, 0.456, 0.406]	[0.229, 0.224, 0.225]

VGG Architectures

We used deep convolutional neural network architectures from the VGG family [227]. Table 3.17 summarises the specific architectures for VGG-5, VGG-9, and VGG-16. Following convolutional layers, a single linear layer was used for classification. We used the GeLU activation function. Batch normalisation was also used in the VGG-9 and VGG-16 models after each convolutional layer to stabilise training.

In the bPC models, the top-down architecture mirrored the discriminative (bottom-up) layers. Each convolutional layer was paired with a transposed convolution sharing identical parameters. However, when convolutional layers were immediately followed by max-pooling operations, the corresponding transposed convolutional layers used a stride of two and a padding of one (instead of zero) to compensate for the change in channel width and height introduced by max pooling. No batch normalisation was used in the top-down predictions.

Table 3.17: Detailed architectures of VGG models. The locations of the pooling layers correspond to the indices of the convolutional layers after which the max-pooling operations are applied.

	VGG-5	VGG-9	VGG-16
Channel Sizes	[128, 256, 512, 512]	[128, 128, 256, 256, 512, 512, 512, 512]	[64, 64, 128, 128, 256, 256, 512, 512, 512, 512, 512, 512]
Kernel Sizes	[3, 3, 3, 3]	[3, 3, 3, 3, 3, 3, 3, 3]	[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
Strides	[1, 1, 1, 1]	[1, 1, 1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Paddings	[1, 1, 1, 1]	[1, 1, 1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Pool location	[0, 1, 2, 3]	[0, 2, 4, 6]	[1, 4, 7, 10, 13]
Pool window	2×2	2×2	2×2
Pool stride	2	2	2

Learning Rate Schedule

The learning rate schedule was structured as follows:

1. During the initial 10% of training, the learning rate linearly increased from w_lr to $1.1 \times w_lr$.
2. Subsequently, a cosine decay schedule reduced the learning rate smoothly to $0.1 \times w_lr$ over the remaining epochs.

where w_lr refers to the tuned weight learning rate.

Simulating PC using Error optimisation

To efficiently simulate the predictive coding models, we employed error optimisation as described in Goemaere et al. [4]. This approach prevents exponential energy decay in predictive coding models and enables the training of larger architectures. discPC directly follows the formulation introduced in this chapter. bPC can likewise be expressed using an error reparameterisation: this is done by rewriting the inference energy function of its discriminative component in the same way as for discPC, while leaving the top-down generative prediction error loss unchanged, i.e., computed directly from the layer activities. We validated this reformulation of bPC by confirming that its iterative inference converges to the same equilibrium point as the neural dynamics of bPC described in this chapter, but does so more quickly when using error reparameterisation.

Model Hyperparameters

The hyperparameters of discPC and discBP for the VGG5 and VGG9 model were adopted from Goemaere et al. [4], where they were tuned using Hyperband Bayesian

optimization (via the Weights & Biases platform) across the combinations listed in Table 3.18. For the VGG16 models, we repeated the tuning summarised in 3.18.

For bPC models, no additional hyperparameter tuning was performed. Instead, we directly applied the optimal hyperparameters obtained for discPC, while setting bPC’s $\alpha_{disc} = 1$, and $\alpha_{gen} = 1e^{-5}$ for VGG-5, and $1e^{-8}$ for VGG-9 and VGG-16.

Table 3.18: Summary of hyperparameter search space from Goemaere et al. [4].

Method	Tuned hyperparameter range	Optimizer	Optim steps (T)	Epochs (sweep/final)
discPC	e.lr: fixed at 0.001 e.momentum: fixed at 0.0 w.lr: log-uniform [1e-5, 1e-2] w_decay: log-uniform [1e-6, 1e-3]	SGD (error) Adam (weights)	5 (all models)	25/25
discBP	w.lr: log-uniform [1e-5, 1e-2] w_decay: log-uniform [1e-6, 1e-3]	Adam (weights)	NA	25/25

Glossary: w.lr: base weight learning rate (see learning rate schedule below), w_decay: weight decay, {e,s}.lr: error / state learning rate, {e,s}.momentum: error / state momentum, T: nr. of optimization steps

Evaluation

We evaluated the models under three conditions: with 0%, 30%, and 50% of input pixels set to zero across all image channels. Missing pixels were selected uniformly at random, independent of their spatial locations.

For discPC and discBP models, we obtained classification accuracy directly from a single feedforward pass, as these models inherently yield zero reconstruction loss for any given input.

For bPC models, missing pixel values in x_1 were initialised to zero and left unclamped, allowing the network to iteratively infer their values. While 600,000 inference steps were previously used to guarantee convergence for MNIST-trained MLPs, this approach is computationally infeasible for our larger models. To accelerate convergence, we adopted a two-stage inference process. First, we performed 1,000 warm-up inference steps with $\alpha_{disc} = 0$, facilitating faster completion of missing pixel values. Subsequently, we restored $\alpha_{disc} = 1$ and carried out an additional 2,000 inference steps to determine final classification accuracy.

Besides measuring classification accuracy, we qualitatively assessed the ability of bPC models to reconstruct missing inputs by examining post-inference neuron activity in x_1 .

Results

Table 3.19 shows that bPC achieves classification accuracy comparable to discPC and discBP when full images are presented, even in deeper VGG architectures.

Tables 3.20 and 3.21 demonstrate that bPC significantly outperforms discPC and discBP on images with missing inputs. This improvement arises from bPC’s iterative inference, which fills in missing information. Figure 3.9 illustrates this effect: after inference, the activity of the input layer x_1 reveals that neurons lacking sensory input have been updated to predict the missing values. Once the missing information is inferred, bPC classifies images accurately. This result holds for both 30% and 50% missing inputs, highlighting bPC’s robustness to incomplete data.

Overall, even in larger models, bPC effectively balances discriminative and generative performance by jointly minimising bottom-up and top-down prediction errors.

Table 3.19: Classification accuracy of bPC, discPC and discBP when whole images are presented.

Model	bPC	discPC	discBP
VGG5/CIFAR10	$87.3^{\pm 0.3} / 98.9^{\pm 0.1}$	$87.6^{\pm 0.2} / 98.8^{\pm 0.1}$	$87.8^{\pm 0.2} / 98.8^{\pm 0.1}$
VGG9/CIFAR100	$61.2^{\pm 0.5} / 84.9^{\pm 0.3}$	$61.3^{\pm 0.5} / 84.6^{\pm 0.2}$	$64.4^{\pm 0.1} / 81.9^{\pm 0.2}$
VGG16/Tiny-ImageNet	$50.3^{\pm 0.4} / 73.6^{\pm 0.2}$	$50.0^{\pm 0.5} / 72.6^{\pm 0.2}$	$47.5^{\pm 0.6} / 70.8^{\pm 0.3}$

Table 3.20: Classification accuracy of bPC, discPC and discBP when 30% of presented images are missing. Best result shown in bold.

Model	bPC	discPC	discBP
VGG5/CIFAR10	$80.8^{\pm 0.3} / 97.2^{\pm 0.1}$	$18.4^{\pm 1.1} / 67.8^{\pm 1.5}$	$16.9^{\pm 0.4} / 63.8^{\pm 1.8}$
VGG9/CIFAR100	$43.9^{\pm 1.3} / 64.4^{\pm 1.1}$	$4.9^{\pm 0.3} / 18.1^{\pm 0.7}$	$4.5^{\pm 0.2} / 12.8^{\pm 0.9}$
VGG16/Tiny-ImageNet	$39.2^{\pm 0.8} / 55.3^{\pm 0.5}$	$18.6^{\pm 0.8} / 37.4^{\pm 0.9}$	$17.3^{\pm 0.2} / 36.5^{\pm 0.2}$

Table 3.21: Classification accuracy of bPC, discPC and discBP when 50% of the presented images are missing. Best result shown in bold.

Model	bPC	discPC	discBP
VGG5/CIFAR10	$73.4^{\pm 0.5} / 95.6^{\pm 0.1}$	$12.3^{\pm 0.3} / 56.4^{\pm 1.0}$	$12.2^{\pm 0.2} / 54.9^{\pm 1.7}$
VGG9/CIFAR100	$29.4^{\pm 0.9} / 48.0^{\pm 0.8}$	$2.5^{\pm 0.3} / 10.4^{\pm 0.5}$	$2.3^{\pm 0.1} / 8.2^{\pm 0.2}$
VGG16/Tiny-ImageNet	$27.8^{\pm 0.5} / 41.7^{\pm 1.1}$	$7.3^{\pm 0.3} / 19.3^{\pm 0.2}$	$7.1^{\pm 0.1} / 19.2^{\pm 0.3}$

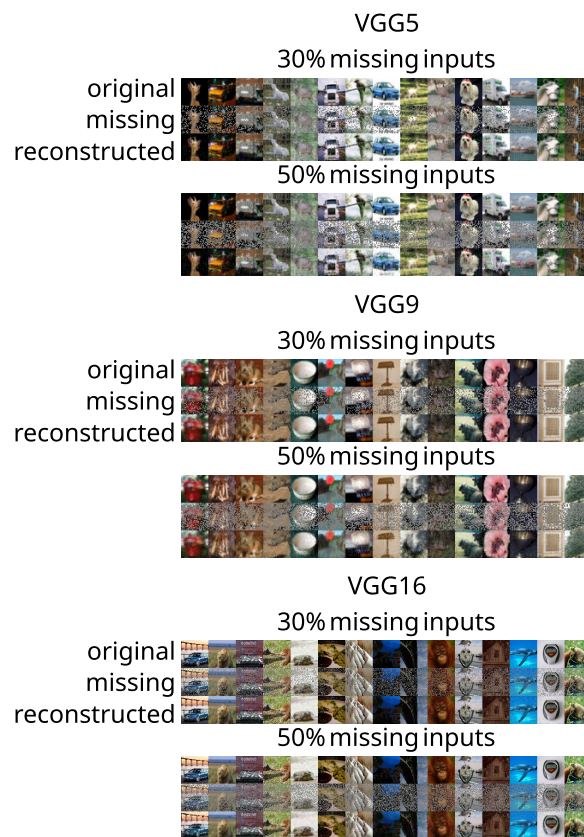


Figure 3.9: Image reconstruction through iterative inference by bPC.

Chapter 4

Optimal learning with target alignment in predictive coding

4.1 Introduction

Learning in the brain differs markedly from backpropagation (BP), the dominant training paradigm in modern machine learning [46, 50, 47]. On one hand, the brain can acquire useful representations using only a handful of stimulus exposures [228, 21, 229] and integrate new information without disrupting previously stored memories [230, 231]. On the other hand, neural networks trained with BP rely on non-local credit assignment and typically require extensive repetition of inputs [232]. Moreover, BP-trained networks are well known to suffer from catastrophic forgetting, where new learning can rapidly overwrite earlier knowledge [233, 234].

Predictive coding (PC) offers an influential account of cortical learning based on minimising prediction errors through local dynamics [70, 113, 99], reviewed in the first Chapter. Recent work [118] has suggested that PC outperforms BP on learning regimes that are both biologically relevant and particularly prone to catastrophic interference, such as learning from limited data, online parameter updates, and continual learning. Song et al. [118] argue that these advantages arise because PC parameter updates induce changes in the model’s predictions that are better aligned with the output error than those induced by BP. As illustrated in Figure 4.1, changes in the model’s predictions that are misaligned with the output error cause interference, by introducing errors in directions the model already predicts accurately. Target alignment quantifies this interference. The more effective learning performance of PC

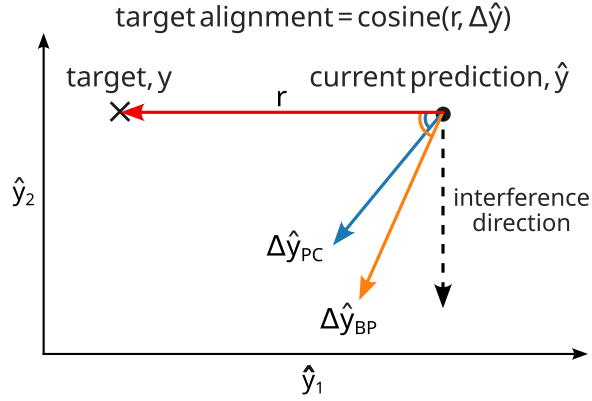


Figure 4.1: **Measuring interference using target alignment.** A change in model prediction, $\Delta\hat{y}$, that is partially perpendicular to the prediction error $r = y - \hat{y}$ introduces interference by creating errors in directions the model already predicts correctly. In the illustrated deep linear network (DLN), the second output dimension is already correctly predicted. The illustrated changes in prediction, which are not fully aligned with the residual, r , degrade the correct prediction of the second output dimension, thereby causing interference. Target alignment quantifies this interference by measuring the cosine similarity between changes in prediction $\Delta\hat{y}$ and model output prediction error r , with a maximum value of one when they are aligned.

over BP observed by Song et al. [118] could therefore be explained by PC’s higher target alignment, which indicates reduced interference between incorporating new information and preserving prior knowledge.

An additional perspective arises from the observation that optimal target alignment is equivalent to performing learning along natural gradients [235]. In this regime, the change in the model’s predictions is exactly aligned with the output error, yielding the steepest descent in output error. Natural gradient methods are known to accelerate learning [236, 237, 238], reduce sensitivity to parameterisation [239], and improve learning stability [240, 241]. In deep linear networks (DLNs), they also render online and batch learning equivalent [235]. Interpreted through this lens, target alignment provides a measure of how closely learning dynamics approximate natural gradient descent, thereby indicating potential benefits of PC over BP by operating closer to the natural gradient regime.

However, a principled analytical explanation of when, why, and on what classes of problems PC should outperform BP, has remained elusive. Moreover, existing evidence of potential benefits of PC has largely been confined to small-scale models [118].

This Chapter provides an analytical account of the interference-reducing properties of PC. We show that, in deep linear networks, PC reduces overall interference by mitigating the weight-dependent interference that arises in BP. As a consequence, PC exhibits improved target alignment, especially in regimes where model weights are poorly conditioned, including architectures with information bottlenecks and pre-trained models. We derive closed-form expressions for the change in prediction induced by PC and BP updates in DLNs and compare their target alignment. We also characterise how PC’s update can be modified to guarantee interference-free learning for learning with individual samples and for a batch of inputs. Our key contributions can be summarised as follows:

- We derive and empirically validate analytical expressions for the change in model predictions induced by BP and PC in DLNs.
- We show that BP suffers from parameter-dependent interference, whereas PC reduces this interference, leading to improved target alignment.
- We demonstrate that PC with layer-specific learning rates guarantees interference-free single-sample learning, independently of architecture or weight conditioning.
- We extend the analysis to multi-sample learning, showing how interference arises between samples and how activity-dependent decorrelation of weight updates enables PC to recover interference-free learning.

4.2 Background

4.2.1 Deep linear networks

Deep linear networks (DLNs) are fully connected linear networks with one or more hidden layers. DLNs have been extensively used to analytically describe the learning dynamics of BP [242, 243, 236, 244] and PC [245, 100]. Despite learning linear models, DLNs exhibit non-linear learning dynamics, as illustrated in Figure 4.2, where the BP (orange) and PC (blue) trajectories follow curved paths that deviate from the linear ideal solution (green). These learning dynamics have proved useful for understanding non-linear networks. For instance, Saxe et al. [242] showed that the learning dynamics

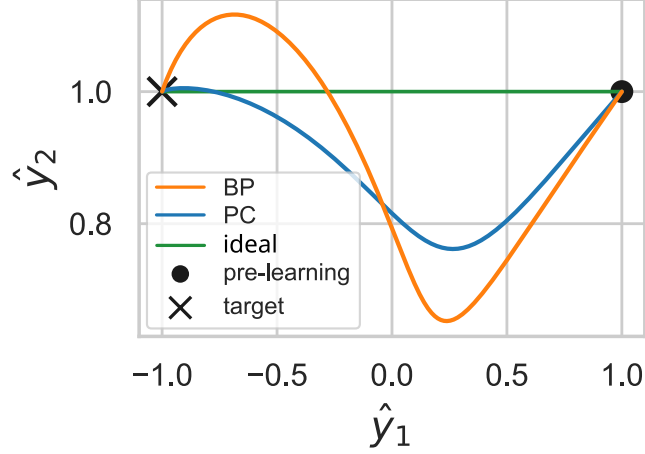


Figure 4.2: **Non-linear learning dynamics of PC and BP in DLNs.** Evolution of a DLN’s predictions during training with BP and PC. The DLN has one input neuron, one hidden neuron and two output neurons. The prediction trajectory shown in green is ideal, as the model’s prediction changes only along the direction of the error, remaining unchanged in directions orthogonal to it and thereby avoiding interference with previously learned information. PC achieves higher target alignment than BP in this model, as its changes in prediction are more closely aligned with the prediction error than BP, confirming previous results [118].

of neural networks include long plateaus followed by rapid transitions to lower error solutions.

Similarly, DLNs have been used to analyse both the neural activity dynamics and weight updates of PC. For instance, it has been shown that PC’s equilibrium activity interpolates between the feedforward dynamics of BP and backwards-propagated targets, such as those used in target propagation [245]. Furthermore, Innocenti et al. [100] showed that PC’s loss landscape exhibits greater robustness to vanishing gradients.

For a hierarchical model with L activity layers, a DLN learns with BP by minimising a standard squared error energy function:

$$\mathcal{L}_{BP}(x, y, V) = \frac{1}{2} \|y - V_{L-1:1}x\|_2^2 \quad (4.1)$$

where x is an input activity, y is an output activity, the parameter V includes the weight matrices V_l of each layer l , and $V_{L-1:1}$ is the shorthand for the product of weights from layer 1 to $L - 1$, $V_{L-1:1} = V_{L-1} \dots V_1$.

4.2.2 Learning with predictive coding

DLNs can also learn using PC with bottom-up predictions, reviewed in the first Chapter. In a linear PC model with L layers, each characterised by neural activity x_l , PC trains a DLN by minimising the energy function

$$E_{disc}(x, V) = \sum_{l=2}^L \frac{1}{2} \|x_l - V_{l-1}x_{l-1}\|_2^2 \quad (4.2)$$

where x denotes the set of neural states from x_1 to x_L , x_1 corresponds to the input and x_L represents the output. As previously described, this energy function is minimised through two steps for a given data input-output pair. First, the neural activity of the network is updated to minimise the energy until convergence. Then, the weights are updated using one gradient step on the energy function, evaluated at the activity minimizing energy found during inference.

For a DLN, PC's inference and learning dynamics are tractable and analyzable. The converged neural activity can be calculated analytically. PC's inference equilibrium in DLNs is an average of BP's feedforward pass activity values and the targets computed by target propagation [245]. The activity equilibrium is therefore given for layer 2 to $L - 1$ by

$$x_l^* = \underbrace{\hat{x}_l}_{\text{BP feed forward activity}} + \underbrace{\left(I + \sum_{k=2}^{l-1} V_{l-1:k} V_{l-1:k}^\top \right) V_{L-1:l}^\top S^{-1} r}_{\text{target propagation targets}} \quad (4.3)$$

where $\hat{x}_l = V_{L-1:1}x$ is the feed-forward activity of a DLN at layer l , $r = y - V_{L-1:1}x$ is the output error of a DLN's feed-forward prediction, and S is given by

$$S = I + \sum_{l=1}^{L-2} V_{L-1:l+1} V_{L-1:l+1}^\top. \quad (4.4)$$

The weight updates of a DLN trained with PC are also equivalent to a BP-trained model with a squared error energy function with a weight-dependent rescaling matrix [100] given by

$$E_{disc}(x, y, V) = \frac{1}{2} (y - V_{L-1:1}x)^\top S^{-1} (y - V_{L-1:1}x) \quad (4.5)$$

This energy function is equivalent to the energy given in equation (4.2) when the neural states are at their activity equilibrium, as given in equation (4.3).

4.2.3 Target alignment of learning

The extent to which a learning rule follows the ideal update direction can be quantified using target alignment. Target alignment measures how strongly the change in a model’s prediction induced by a weight update is aligned with its prediction error, using cosine similarity:

$$\text{Target alignment} = \cos(\angle(r, \Delta\hat{y})) = \frac{r^\top \Delta\hat{y}}{\|r\| \|\Delta\hat{y}\|} \quad (4.6)$$

Here, r denotes the output prediction error $r = y - V_{L-1:1}x$, $\hat{y} = V_{L-1:1}x$ is the feedforward prediction, and $\Delta\hat{y}$ represents the change in prediction resulting from a weight update. For DLNs, $\Delta\hat{y} = (V_{L-1:1}^{\text{post}} - V_{L-1:1}^{\text{pre}})x$, with pre and post indicating the weights’ state before and after the update, respectively. A target alignment of one indicates perfect alignment, zero indicates orthogonality, and minus one indicates opposite directions. Throughout, we define interference as a target alignment smaller than one. Target alignment for PC and BP in an example DLN is illustrated in Figure 4.2.

Learning updates that follow the shortest path between the current prediction and the target reduce interference [118, 246] and lead to faster convergence [235, 247]. As shown in Figure 4.2, neither BP nor PC follows this shortest path. The DLN comprises one input neuron, one hidden neuron, and two output neurons. All weights are initially set to one. The network receives an input of 1 with a target output of $[-1, 1]$. Training is carried out separately using PC and BP until the model output matches the target. The figure shows that PC learning trajectories are more closely aligned with the ideal path than those of BP. This behaviour has been observed in both small linear models and deeper non-linear architectures [118].

The reasons why PC exhibits higher target alignment than BP remain unclear. It has been shown that PC inference combines BP-like feedforward activity with signals resembling target propagation [245], and that models trained with target propagation achieve optimal target alignment [118]. From this perspective, PC may attain higher target alignment than BP by effectively integrating BP with target-propagation-like updates [118]. However, this explanation remains empirical and lacks analytical proof.

Existing theoretical understanding indicates that, for shallow linear models, BP and PC are equivalent: there is no hidden-layer activity to infer, and both methods achieve perfect target alignment [118]. In contrast, for deep models, parameter updates across

layers can interfere, causing the resulting change in the model’s prediction to deviate from the direction of the prediction error. An important exception arises in deep networks with orthogonal initialisation, where both BP and PC achieve ideal target alignment [242, 118]. This benefit, however, comes at the cost of reduced expressivity.

Alternative learning algorithms with higher target alignment than BP have been studied extensively in machine learning. In particular, natural gradient methods [235] guarantee maximally aligned learning updates by rescaling BP parameter updates with the inverse Fisher information matrix, yielding updates of the form:

$$\Delta V = \propto -F^{-1} \nabla_V \mathcal{L}_{BP} \quad (4.7)$$

where F denotes the Fisher information matrix. By accounting for interactions between parameters in their effect on the model output, the Fisher matrix reconditions the gradient to produce updates that achieve optimal target alignment and maximise the reduction of the objective function per update. Natural gradient methods have also been investigated in the context of DLNs [236]. Despite these attractive theoretical properties, natural gradient methods are often impractical in modern machine learning due to the computational cost associated with estimating and inverting the Fisher matrix. Several computationally cheaper approximations have been proposed [238, 248], but these approaches still offer limited insight into how interference-free learning could arise from local, biologically plausible computations. As a result, the relevance of natural gradients to large-scale deep learning and to neuroscience remains limited.

4.3 Results

4.3.1 Origin of interference in BP

To understand when PC yields higher target alignment than BP, we first derive the change in output prediction of a network trained with BP. Specifically, we analyse how changes in the network’s prediction relate to the output residual, r . We consider a DLN trained with a squared error objective function $\mathcal{L}_{BP}$ as given in equation (4.1).

In BP, the weight updates follow the steepest descent of the model’s objective function with respect to the weights. Although the weight updates occur in discrete steps,

we analyse the corresponding continuous-time limit, which holds as long as the learning rate is sufficiently small. This approach aligns with prior studies on the weight dynamics of DLNs under BP [242, 236] and PC [100]. The weight dynamics for an input x and output y are given by

$$\frac{\partial V_l}{\partial t} = -\nabla_{V_l} \mathcal{L}_{BP} = V_{L-1:l+1}^\top (y - V_{L-1:1}x)(V_{l-1:1}x)^\top = V_{L-1:l+1}^\top r(V_{l-1:1}x)^\top. \quad (4.8)$$

The change in prediction resulting from the weight dynamics can then be calculated by differentiating using the product rule, as follows:

$$\frac{\partial \hat{y}}{\partial t} = \frac{\partial V_{L-1:1}}{\partial t} x \quad (4.9)$$

$$= \sum_{l=1}^{L-1} V_{L-1:l+1} \frac{\partial V_l}{\partial t} V_{l-1:1} x \quad (4.10)$$

$$= \sum_{l=1}^{L-1} V_{L-1:l+1} V_{L-1:l+1}^\top r(V_{l-1:1}x)^\top V_{l-1:1} x \quad (4.11)$$

$$= \sum_{l=1}^{L-1} V_{L-1:l+1} V_{L-1:l+1}^\top r \hat{x}_{l-1}^\top \hat{x}_{l-1} \quad (4.12)$$

where the feed forward activity at layer l equal to $V_{L-1:1}x$ is referred to as $\hat{x}_l$, and $V_{L-1:L}$ and $V_{0:1}$ are equal to the identity matrix. This result is equivalent to previous derivations for DLNs [236], and it was numerically validated in Appendix C.1.

When training on a single input-output pair, the derived expression for the change in prediction under BP reveals a single source of interference. In the ideal case, the change in prediction would equal the target residual r . However, the actual update is given by a sum in which each term consists of r pre-multiplied by $V_{L-1:l+1} V_{L-1:l+1}^\top$ and post-multiplied by $\hat{x}_{l-1}^\top \hat{x}_{l-1}$. The matrix $V_{L-1:l+1} V_{L-1:l+1}^\top$ acts directly on r , thereby distorting its direction and introducing interference. In contrast, the factor $\hat{x}_{l-1}^\top \hat{x}_{l-1}$ is a scalar rather than a matrix. Therefore, it only rescales the update magnitude and does not contribute to interference.

Interference is therefore expected to be large when $V_{L-1:l+1} V_{L-1:l+1}^\top$ deviates substantially from the identity matrix. This occurs when $V_{L-1:l+1}$ is rank-deficient, for example, due to architectural bottlenecks such as those in autoencoders, or from highly correlated features that emerge during training. In either case, the resulting product becomes singular or is poorly approximated by the identity. Interference is also exacerbated by poor conditioning of $V_{L-1:l+1}$, which can occur when training amplifies

certain feature directions while suppressing others. In contrast, interference vanishes when the weights are orthogonal. Under orthogonal initialisation, each weight matrix satisfies $V_l V_l^\top = I$, which implies $V_{L-1:l+1} V_{L-1:l+1}^\top = I$, by repeated cancellation. This observation is consistent with prior analyses of DLNs [242]. Nevertheless, maintaining strict orthogonality imposes a strong constraint on the parameter space and is generally impractical, as it limits the model’s expressive capacity during learning.

4.3.2 PC attenuates interference

Interference in a network trained with PC differs from that in BP due to an additional scaling matrix. This additional term counteracts distortions induced by the weights, attenuating interference.

We derive the change in prediction $\frac{\partial \hat{y}}{\partial t}$ of a DLN trained with PC’s objective function E_{disc} given in equation (4.2). This objective is first minimised with respect to the neural activities until convergence to their equilibrium at x_l^* for layer l given in equation (4.3). Afterwards, the weight updates follow the steepest-descent direction on PC’s objective function evaluated at the converged activities. We show in the Methods section 4.5.7 that such weight dynamics of a DLN trained with PC are given by:

$$\frac{\partial V_l}{\partial t} = -\nabla_{V_l} E_{disc} = V_{L-1:l+1}^\top S^{-1} r x_{l-1}^{*\top} \quad (4.13)$$

The change in prediction resulting from PC’s weight dynamics can be calculated by differentiating using the product rule, as follows:

$$\frac{\partial \hat{y}}{\partial t} = \frac{\partial V_{L-1:1}}{\partial t} x \quad (4.14)$$

$$= \sum_{l=1}^{L-1} V_{L-1:l+1} \frac{\partial V_l}{\partial t} V_{l-1:1} x \quad (4.15)$$

$$= \sum_{l=1}^{L-1} V_{L-1:l+1} V_{L-1:l+1}^\top S^{-1} r x_{l-1}^{*\top} V_{l-1:1} x \quad (4.16)$$

$$= \sum_{l=1}^{L-1} V_{L-1:l+1} V_{L-1:l+1}^\top S^{-1} r x_{l-1}^{*\top} \hat{x}_{l-1} \quad (4.17)$$

which was numerically validated in Appendix C.1.

Importantly, this expression differs from the BP result in equation (4.12) through the presence of the scaling matrix S^{-1} . Recalling its definition, S equals $I + \sum_{l=1}^{L-2} V_{L-1:l+1} V_{L-1:l+1}^\top$.

Related terms, $V_{L-1:l+1}V_{L-1:l+1}^\top$, appear within the expression for the change in prediction. This similarity suggests that the additional factor S^{-1} introduced by PC may cancel with these terms under certain conditions, and prevent all interference.

When the scalar factor $x_{l-1}^{*\top}\hat{x}_{l-1}$ is identical across layers, the layer-wise interference matrices cancel, resulting in a change in prediction that is perfectly aligned with the residual r . Assuming that $x_{l-1}^{*\top}\hat{x}_{l-1}$ equals a constant c for all layers (the conditions under which this assumption holds will be analysed in more detail below), the expression for PC target alignment simplifies to

$$\begin{aligned}
\frac{\partial \hat{y}}{\partial t} &= \sum_{l=1}^{L-1} V_{L-1:l+1} V_{L-1:l+1}^\top S^{-1} r x_{l-1}^{*\top} \hat{x}_{l-1} \\
&= \sum_{l=1}^{L-1} V_{L-1:l+1} V_{L-1:l+1}^\top S^{-1} r c \\
&= \underbrace{\left(I + \sum_{l=1}^{L-2} V_{L-1:l+1} V_{L-1:l+1}^\top \right)}_S S^{-1} r c \\
&= r c,
\end{aligned}$$

where we used the convention $V_{L-1:L} = 1$ introduced earlier, together with the definition of S given in equation (4.4).

This simplification indicates that PC has an intrinsic mechanism for minimising interference, which can be advantageous when neural activity exhibits similar properties across layers. Unlike BP, this property does not depend on the network architecture or the stage of training. Similar to BP, PC’s interference factor also vanished for orthogonal weights. While the assumption that $x_{l-1}^{*\top}\hat{x}_{l-1}$ is constant across layers does not generally hold, as long as these quantities do not vary excessively, the scaling matrix S^{-1} is expected to substantially reduce interference in PC over BP.

4.3.3 Is PC’s interference reduction mechanism effective?

We experimentally validate that PC’s prediction changes are more closely aligned with the output residual r than those produced by BP, even when the assumption of constant activity norm across layers is not met.

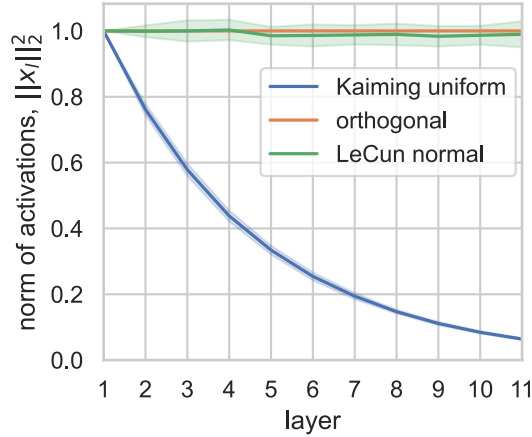


Figure 4.3: **LeCun and orthogonal initialisation maintain stable expected activation norms in DLNs.** Feedforward activity norm, $\hat{x}_l^\top \hat{x}_l$, across depth of a DLN initialised with Kaiming, orthogonal and LeCun initialisation. Activity decays with depth for Kaiming initialisation, while orthogonal and LeCun initialisations maintain the expected activity norm. The DLN has input, hidden, and output widths of 512 neurons. Inputs are sampled from a standard normal distribution and rescaled to unit norm. The activity norm is computed at each depth for 10 input samples.

We evaluate target alignment in randomly initialised DLNs after a single weight update using either BP or PC. We vary network depth and width, and additionally examine the effect of introducing ReLU non-linearities. Unless stated otherwise, we consider a DLN with input, hidden, and output widths all set to 512, a single hidden layer, and no non-linearities. Inputs and targets are sampled independently from standard normal distributions, the batch size is one, and results are averaged over ten random seeds.

We compare three widely used weight initialisation schemes: Kaiming uniform, orthogonal, and LeCun normal. Kaiming uniform causes the activity norm of a network to decay rapidly with depth, whereas orthogonal and LeCun normal initialisations preserve stable activation norms (Figure 4.3). Evaluating PC with Kaiming initialisation places it outside its ideal operating regime, allowing the assessment of PC’s reduced interference relative to BP when the assumptions required to minimise interference are not satisfied.

Figure 4.4 shows that PC achieves better target alignment than BP under both Kaiming and LeCun initialisations. These results indicate that PC’s interference-reducing mechanism remains effective even when the scalar quantity $x_{l-1}^{*\top} \hat{x}_{l-1}$ is not constant across layers. As expected, orthogonal initialisation yields perfect alignment for both

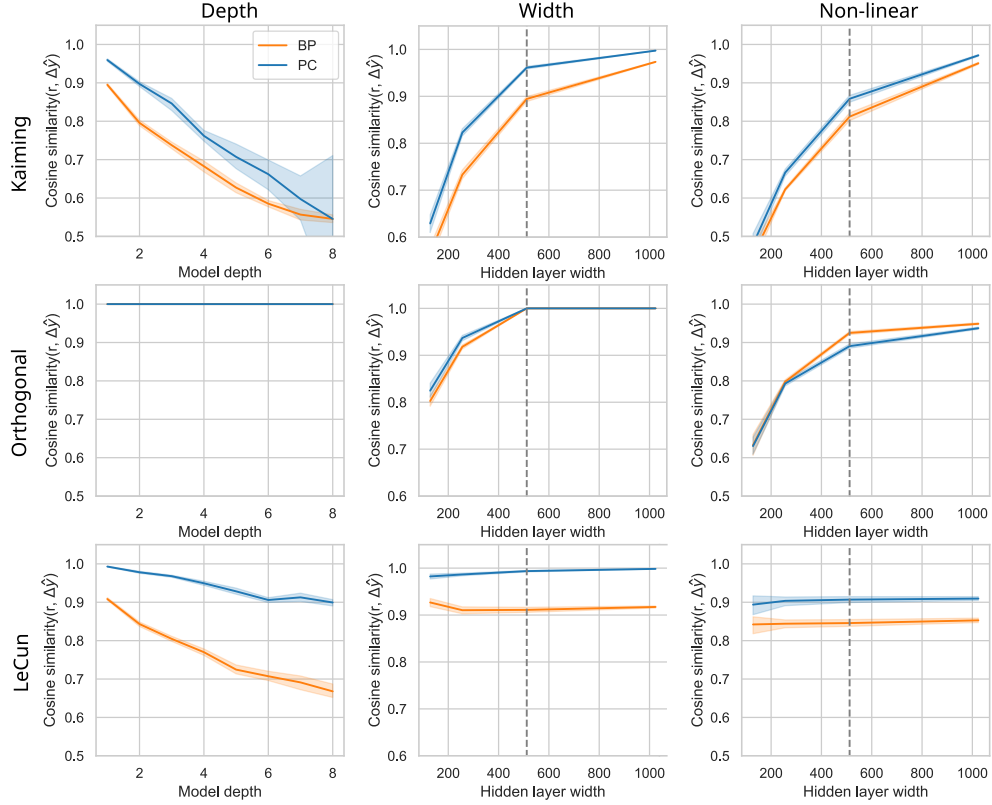


Figure 4.4: **PC’s interference reduction mechanism is effective across initialisation in DLNs.** Target alignment of a DLN trained on a single sample for one weight update, comparing BP and PC across three weight initialisation methods. Model depth and width are varied independently, with default values of one hidden layer and a hidden layer width of 512. Target alignment in a deep non-linear model with ReLU activation is also measured for comparison. Inputs and targets are sampled from a standard normal distribution. Vertical grey dashed lines indicate the width at which the hidden layer matches the input and output dimensionality; smaller widths correspond to an information bottleneck. Results are reported as mean and standard deviation over 10 random seeds.

PC and BP in linear networks without information bottlenecks. When a bottleneck is introduced, however, alignment degrades substantially, since rank constraints prevent the weights from being fully orthogonal. LeCun initialisation attains near-perfect alignment across settings. Although alignment decreases slightly with increasing depth, it remains largely insensitive to changes in network width. In contrast, Kaiming initialisation results in the poorest overall alignment. Under Kaiming initialisation, PC alignment deteriorates markedly as depth increases and exhibits greater variability across model configurations. Introducing ReLU non-linearities reduces target alignment across all initialisation schemes.

Overall, these results demonstrate that PC’s interference-attenuating mechanism consistently improves target alignment relative to BP. The results also indicate that Le-Cun initialisation could be adopted as a new default initialisation method for PC research. This change would be especially significant when working with deep networks. Moreover, increasing the network hidden layer width reduces the alignment gap between PC and BP, whereas the introduction of non-linearities generally degrades alignment for both methods.

4.3.4 Interference-free learning in PC with layer specific learning rates

Although our previous results show that PC exhibits better alignment than BP, its standard formulation does not achieve optimal target alignment. Introducing layer-specific learning rates, ensures that PC achieves optimal alignment for linear models trained on a single sample, irrespective of neural activities, model weights or model architecture.

To show this, we modify PC’s weight update to include a layer-specific scaling factor α_l :

$$\frac{\partial V_l}{\partial t} = -\alpha_l \nabla_{V_l} E_{disc} \quad (4.18)$$

In practice, this is equivalent to assigning different learning rates for each layer. Choosing α_l equal to the inverse of PC’s scalar factor $x_{l-1}^{*\top} \hat{x}_{l-1}$ removes the dependency of PC’s change in prediction on this scalar factor. This allows the expression for PC’s change in prediction to be aligned with the residual r , leading to maximal target alignment as shown below:

$$\frac{\partial \hat{y}}{\partial t} = \sum_{l=1}^{L-1} V_{L-1:l+1} \frac{\partial V_l}{\partial t} V_{l-1:1} x = - \sum_{l=1}^{L-1} V_{L-1:l+1} \alpha_l \nabla_{V_l} E_{disc} \hat{x}_{l-1} \quad (4.19)$$

$$= \sum_{l=1}^{L-1} V_{L-1:l+1} (1/x_{l-1}^{*\top} \hat{x}_{l-1}) V_{L-1:l+1}^\top S^{-1} r x_{l-1}^{*\top} \hat{x}_{l-1} \quad (4.20)$$

$$= \underbrace{\sum_{l=1}^{L-1} V_{L-1:l+1} V_{L-1:l+1}^\top}_{=I} S^{-1} r \quad (4.21)$$

$$= r \quad (4.22)$$

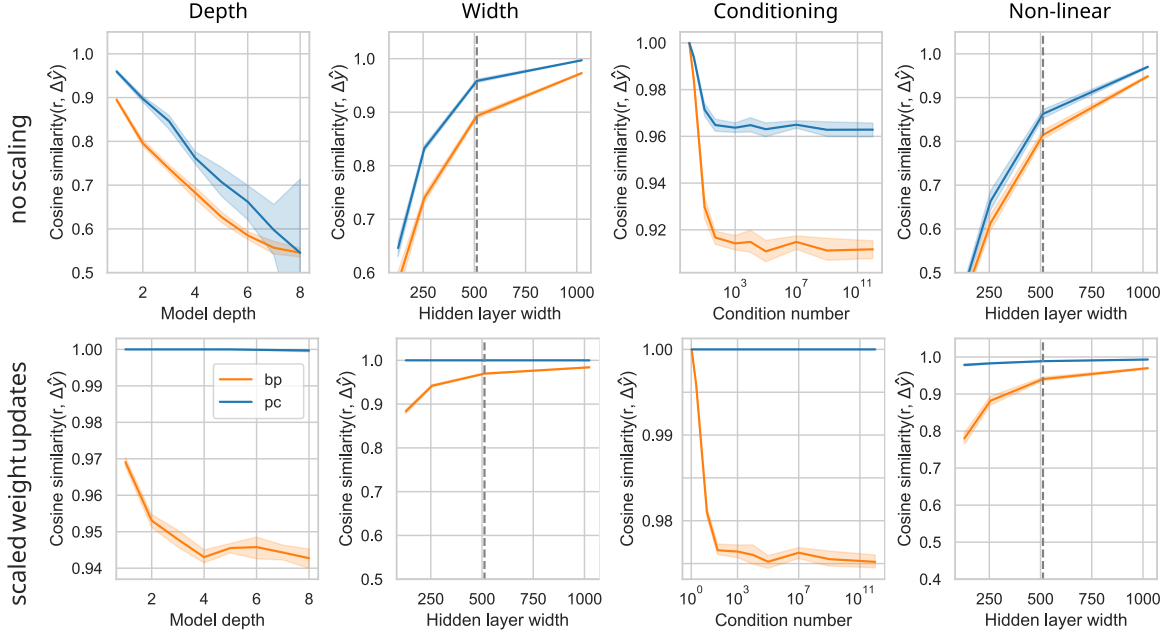


Figure 4.5: **Interference-free learning in PC with layer-dependent learning rates for training with individual samples.** Target alignment of a DLN trained on one sample for one weight update, comparing BP and PC without (top) and with (bottom) adaptive learning rates. Model depth, width, and conditioning, and activation function are varied independently, with defaults of one hidden layer, a width of 512, Kaiming initialisation and no activation function. Inputs and targets are sampled from a standard normal distribution. Vertical grey dashed lines indicate the location where hidden layer width matches input and output dimensionality. Results are reported as mean and standard deviation over 10 random seeds.

To evaluate layer-specific learning rates, we measure target alignment in a DLN trained with and without scaling under both BP and PC. For BP, we scale the BP weight updates by the inverse of BP’s scaling factors, $\hat{x}_{l-1}^\top \hat{x}_{l-1}$, ensuring a fair comparison. We vary both the depth and width of the model, as well as introduce ReLU non-linearities. Additionally, we vary the conditioning of the model weights (see details in Section 4.5.6.2), quantified using the condition number. This number is the ratio of the largest to the smallest singular value. Larger condition numbers indicate poorer conditioning, and orthogonal weights have a condition number of one. All other experimental settings are set to the default values of the previous experiments, and models were initialised using Kaiming uniform initialisation.

Figure 4.5 confirms that introducing layer-specific learning rates ensures that PC learns with maximal target alignment. These results are observed across model architectures and weight conditioning levels. Non-linearities only slightly reduce alignment.

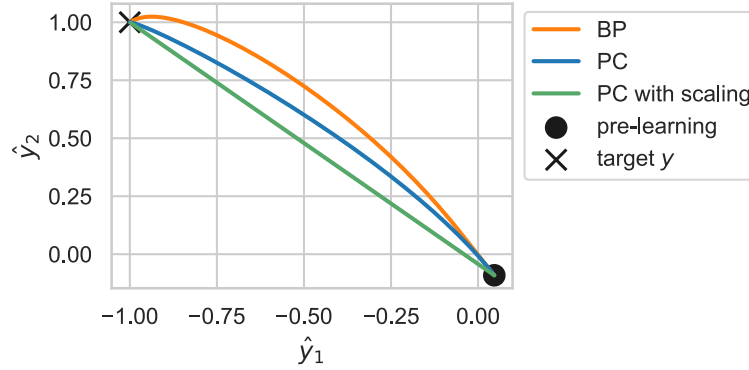


Figure 4.6: **Interference-free learning trajectory of PC with layer-specific learning rates trained on a single sample.** Evolution of a DLN’s predictions during training with BP, PC and PC with adaptive learning rates. The DLN has two input neurons, two hidden neurons and two output neurons. The model’s weights are initialised using Kaiming initialisation.

BP benefits from layer-specific scaling factors as well, but does not achieve maximal alignment or independence from architecture. Increasing the condition number of the weight matrices reduces target alignment for both PC and BP, except for PC with activity-dependent learning rates. This behaviour is expected, as a higher condition number corresponds to stronger interference from the weight product $V_{L-1:l+1}V_{L-1:l+1}^\top$.

Figure 4.6 further illustrates the interference-free learning of PC with layer-specific learning rates. We consider a DLN with two neurons per layer, one hidden layer and randomly initialised weights. The model is trained with the input $[1, 1]$ for the target $[-1, 1]$. The figure shows that standard PC and BP cause interference during learning, shown by the curved paths of their learning trajectories. PC with layer-specific learning rates, however, produces a trajectory that follows the shortest path between the initial prediction and the target. Each update is aligned with the model output error, leaving the model prediction unaltered in directions already learned.

4.3.5 Training on a batch of samples introduces inter-sample interference

So far, we have shown that PC minimises interference in a single-sample learning regime. However, learning in the brain often requires the coexistence of multiple memories within shared neural circuitry [249, 250]. Here, we show that weight updates across a batch of samples cause additional interference.

4.3.5.1 Change in prediction for batch training

We begin by re-deriving the change in predictions of a DLN when training on a mini-batch of data. For batch training, weight updates are the average of the per-sample updates across the batch.

Backpropagation

For BP with a mini-batch of size B , the weight update for layer l is

$$\frac{\partial V_l}{\partial t} = -\frac{1}{B} \sum_{b=1}^B \nabla_{V_l} \mathcal{L}_{BP} = \frac{1}{B} \sum_{b=1}^B V_{L-1:l+1}^\top r_b \hat{x}_{l-1,b}^\top = \frac{1}{B} V_{L-1:l+1}^\top R \hat{X}_{l-1}^\top. \quad (4.23)$$

where x_b and y_b denote the input and target output for the b -th data point in the batch, and r_b is the corresponding output prediction residual. For compactness, we define $R = [r_1, \dots, r_B]$ as the matrix of residuals and $\hat{X}_{l-1} = [\hat{x}_{l-1,1}, \dots, \hat{x}_{l-1,B}]$ as the matrix of feedforward activities at layer $l-1$.

Following the derivation in equations (4.10–4.12), the resulting change in the predictions across the batch is

$$\frac{\partial \Delta \hat{Y}}{\partial t} = \frac{1}{B} \sum_{l=1}^{L-1} V_{L-1:l+1}^\top V_{L-1:l+1}^\top R \hat{X}_{l-1}^\top \hat{X}_{l-1}, \quad (4.24)$$

where $\Delta \hat{Y} = [\Delta \hat{y}_1, \dots, \Delta \hat{y}_B]$ collects the prediction changes for all batch elements.

Predictive coding

For PC trained on a mini-batch, the weight update for layer l is given by

$$\frac{\partial V_l}{\partial t} = -\frac{1}{B} \sum_{b=1}^B \nabla_{V_l} E_{disc} = V_{L-1:l+1}^\top S^{-1} R X_{l-1}^{*\top}. \quad (4.25)$$

with $X_{l-1}^* = [x_{l-1,1}^*, \dots, x_{l-1,B}^*]$.

Following the derivation in equations (4.15–4.17), the corresponding change in PC's predictions is

$$\frac{\partial \Delta \hat{Y}}{\partial t} = \frac{1}{B} \sum_{l=1}^{L-1} V_{L-1:l+1}^\top V_{L-1:l+1}^\top S^{-1} R X_{l-1}^{*\top} \hat{X}_{l-1}. \quad (4.26)$$

4.3.5.2 Batch training causes additional interference

As indicated by the expressions derived above, training on a mini-batch introduces additional sources of interference in both BP and PC. In particular, the matrix factors $\hat{X}_{l-1}^\top \hat{X}_{l-1}$ for BP and $X_{l-1}^{*\top} \hat{X}_{l-1}$ for PC act directly on the residual matrix R , distorting its direction and thereby inducing interference across samples.

Figure 4.7 confirms that batch training introduces additional interference. We train a DLN with input, hidden, and output widths all set to 512 and a single hidden layer using both BP and PC. The model is trained on batches of data with randomly sampled inputs x , and outputs defined as $y = W_{data}x$, where W_{data} is a random matrix mapping the input to the output dimension. We vary the batch size and report the mean target alignment across batch elements. We consider Kaiming, orthogonal, and LeCun initialisation schemes. Additionally, we evaluate training with the layer-dependent learning rates introduced in Section 4.3.4, set to the mean learning rates across the batch. Figure 4.7 shows that batch-trained models exhibit substantially lower target alignment than models trained with batch size one. Moreover, incorporating layer-specific learning rates does not alleviate the interference induced by larger batch sizes.

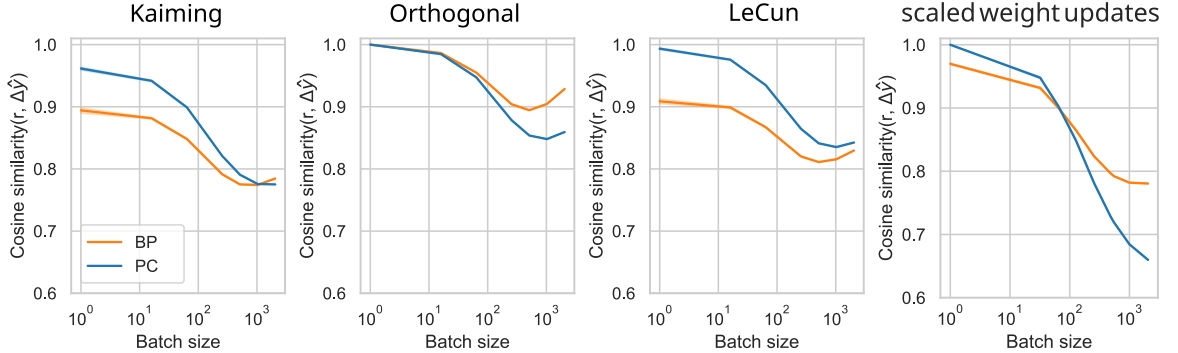


Figure 4.7: **Training on batches of samples introduces inter-sample interference.** Target alignment of a DLN trained on a varying number of samples for one weight update, comparing BP and PC across initialisations and with scaled learning rates. The model has 512 input, hidden, and output neurons. Inputs are sampled from a standard normal distribution. The outputs are generated by a random mapping of the inputs. Results are reported as mean and standard deviation over 10 seeds.

4.3.5.3 Mitigating batch inference in PC

Standard PC does not have a mechanism to reduce inter-sample interference. However, PC can maximise target alignment during multi-sample learning by multiplying its weight updates by a matrix that removes correlations between samples.

To achieve this, we multiply the weight update of each layer by a matrix A_l

$$\frac{\partial V_l}{\partial t} = -\frac{1}{B} \sum_{b=1}^B \nabla_{V_l} E_{disc} A_l \quad (4.27)$$

The decorrelation factors ensure that $X_{l-1}^{*\top} A_l \hat{X}_{l-1} = I$, thereby removing interference induced across samples. As a result, PC's change in prediction remains aligned with R

$$\frac{\partial \hat{y}}{\partial t} = \frac{1}{B} \sum_{l=1}^{L-1} V_{L-1:l+1} V_{L-1:l+1}^\top S^{-1} R \underbrace{X_{l-1}^{*\top} A_l \hat{X}_{l-1}}_{=I} \quad (4.28)$$

$$= \frac{1}{B} \sum_{l=1}^{L-1} \underbrace{V_{L-1:l+1} V_{L-1:l+1}^\top}_{=I} S^{-1} R \quad (4.29)$$

$$= \frac{1}{B} R \quad (4.30)$$

An example decorrelation factor is

$$A_l = (E_B[x_{l-1,b}^{*\top} \hat{x}_{l-1,b}])^{-1} \quad (4.31)$$

where E_B is the expectation across the batch of samples. While there are other options for A_l , this factor is closely linked to the precision of neural activities, which can be computed by PC's neural circuits [99]. Note that, due to rank constraints, A_l can exactly eliminate inter-sample interference only when the number of samples is smaller than the width of the activity layers, from layers 1 to $L - 1$.

We repeat the target alignment experiments for a DLN trained with a single weight update using PC and BP, both with and without the proposed layer-wise decorrelation matrices. We vary the model's depth, width, weights' condition number, and batch size, as well as include a ReLU non-linearity. All default experimental settings are the same as in previous experiments, except for the default batch size which is equal

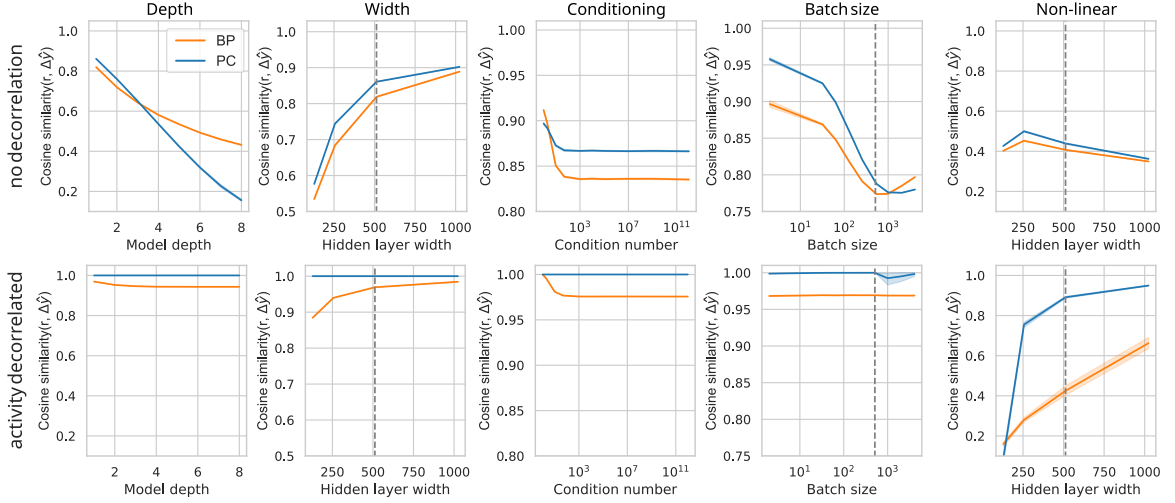


Figure 4.8: **Interference-free learning in PC with matrix factors for batch training.** Target alignment of a DLN trained on a batch of samples for one weight update, comparing BP and PC without (top row) and with (bottom row) decorrelation matrices. Model depth, width, conditioning and batch size are varied independently, with default values of one hidden layer, a hidden layer width of 512, conditioning set by Kaiming initialisation, and a batch size of 128. A deep non-linear model with ReLU activation is also considered. Inputs are sampled from a standard normal distribution. The outputs are generated by a random mapping of the inputs. Vertical grey dashed lines indicate the width where the hidden layer matches the input and output dimensionality, and when the batch size exceeds the model’s smallest width. Results are reported as mean and standard deviation over 10 seeds.

to 128 samples. For BP, we employ a decorrelation matrix A_l equal to the precision matrix of the activity layer $l - 1$, $A_l = (E_B[\hat{x}_{l-1,b}^\top \hat{x}_{l-1,b}])^{-1}$. To ensure numerical stability, we employ Moore–Penrose pseudoinverses.

Introducing layer-wise decorrelation matrices enables PC to achieve optimal target alignment for batch training (Figure 4.8), independently of model architecture, parameter values, and batch size up to a limit. BP also exhibits improved alignment with decorrelation matrices, but still fails to reach optimal alignment and consistently underperforms PC. Introducing non-linearities reduces target alignment and reintroduces a dependence on architectural choices.

Learning with optimal target alignment on a batch of data has additional benefits. Figure 4.9 illustrates the learning trajectories for four samples of a DLN trained on a batch of 32 samples. For two illustrated samples, the target coincides with the pre-learning prediction, indicating that the model has already learned them. In both BP and standard PC, interference from other samples causes previously learned

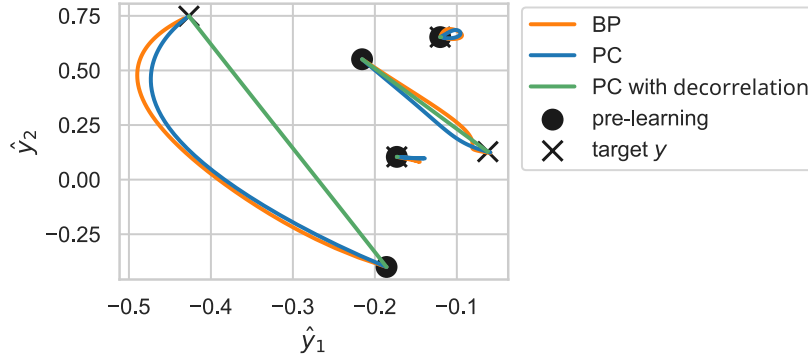


Figure 4.9: **Interference-free learning trajectory of PC with decorrelation factors trained on 32 samples.** Evolution of a DLN’s predictions during training with BP, PC, and PC with decorrelation factors. The DLN has 512 input, hidden, and output neurons, and the weights are initialised using Kaiming initialisation. Inputs are sampled from a standard normal distribution. Of the 32 samples, 30 have outputs sampled from a standard normal distribution, while the remaining two have outputs fixed to the model’s initial predictions. The trajectories of four samples are visualised: two with random inputs and outputs, and two corresponding to the samples whose outputs are set to the model’s initial prediction.

samples to be temporarily unlearned and subsequently relearned. In contrast, PC with decorrelation preserves the corresponding predictions, leaving them unchanged and thereby preventing forgetting. This characteristic may support continual learning in the brain by reducing forgetting during memory replay.

4.4 Discussion

4.4.1 Summary

Inspired by early empirical reports of reduced destructive interference in PC relative to BP, we set out to provide a principled analytical account of when and why such advantages should arise. By deriving closed-form expressions for the changes in model predictions induced by BP and PC in DLNs, we show that the two learning rules differ fundamentally in how interference emerges during learning. In particular, BP updates are distorted by parameter-dependent interference arising from correlated and poorly conditioned weights, whereas PC attenuates interference, providing improved target alignment. We further show that, although PC remains subject to conditions for optimal reduction of interference, this limitation can be overcome through

layer-dependent learning rates. This guarantees interference-free learning for single samples, independently of the architecture or weight conditioning. Finally, extending the analysis to multi-sample learning reveals an additional source of cross-sample interference. In PC, this can be mitigated through activity-dependent decorrelation of weight updates, recovering interference-free learning trajectories.

4.4.2 When does PC significantly outperform BP?

BP updates distort the direction of the output error by introducing correlated products of weights. These correlations reduce alignment with the target signal and increase interference with previously learned representations. PC intrinsically counteracts these distortions, producing prediction updates that remain more closely aligned with the error.

PC is expected to significantly outperform BP in learning regimes with substantial parameter-dependent interference. Our analysis shows that such interference arises most prominently in deep or narrow architectures, in the presence of information bottlenecks, and when weights are poorly conditioned. These characteristics are common in autoencoders, pretrained models, later stages of learning and continual learning. In highly overparameterised models with wide hidden layers, weight conditioning improves and BP interference diminishes, making the performance difference between PC and BP likely negligible.

4.4.3 Benefits of natural gradients could extend to PC

Maximising target alignment and natural gradient methods share the same objective: parameter updates that produce the most efficient change in model predictions for reducing output error. From this perspective, the improved target alignment observed in PC suggests a close functional relationship with natural gradient descent.

This connection is important because many of the theoretical advantages established for natural gradients should therefore extend to PC, at least qualitatively. In particular, natural gradient methods are known to achieve theoretically optimal learning speed. In DLNs, natural gradients lead to exponential convergence to the global optimum [235, 236]. Moreover, natural gradients eliminate the distinction between online

and batch learning in DLNs. Learning from a batch of samples, one sample at a time, yields the same parameter update as learning from the whole batch simultaneously [235]. If PC approximates natural gradient updates through its interference-filtering mechanisms, this provides a possible explanation for how biologically plausible, online learning could inherit the stability and efficiency typically associated with batch optimisation. Under this view, PC offers a mechanistic route by which the brain could realise fast and stable learning.

4.4.4 Future work

4.4.4.1 Whole training trajectories

The analysis performed in this chapter should be evaluated during training on real data. While such work is beyond this thesis, we expect the analysis to naturally extend to learning over entire training runs. Mechanisms that improve target alignment at each update accumulate over time, leading to faster convergence and reduced interference across successive learning steps. Consequently, the improved target alignment achieved by PC should translate into faster learning and more stable optimisation dynamics.

The experiments demonstrate that weight initialisation has a significant impact on target alignment in PC, and should therefore be chosen carefully. For whole training trajectories, choosing the appropriate initialisation will improve target alignment of the early learning dynamics.

Layer-dependent learning rates and activity-dependent decorrelation matrices are designed to improve target alignment throughout training by reducing interference at every update. Layer-dependent learning rates are effective in single-sample or small-batch settings. More generally, PC with decorrelation matrices should improve target alignment across entire training runs and large batch sizes.

4.4.4.2 Biological plausibility

While the learning rate scaling and decorrelation matrices are expected to accelerate learning in PC, they can introduce biologically unrealistic computations. These modifications to PC are, therefore, better interpreted as defining a baseline for optimal

performance rather than faithfully capturing how biological learning operates. In reality, neural systems likely employ approximate strategies that trade off optimality for computational cost and simpler circuitry.

The main biologically implausible requirement of layer-dependent learning rates is temporal delays. The learning rates depend on the activity of all neurons within a layer, both before and after PC inference, making the process non-local in time. Implementing such factors would therefore require mechanisms that store and compare activity across time. However, these delays may not be necessary in practice. As training progresses and the residual, r , tends to zero, the converged activity, x_l^* , approaches the feedforward estimate, $\hat{x}_l$, (see equation (4.3)). This limit effectively eliminates the need for temporal delays. Alternatively, biological systems might rely on inhibitory or homeostatic processes that keep the neural activity norm approximately constant, removing the need for explicit layer-dependent learning rates.

Similarly, the computations of the decorrelation matrices A_l appear incompatible with strictly local biological mechanisms. Accurately deriving them would require coordinated temporal delays alongside lateral connections spanning all neurons within a layer. More biologically plausible approximations can be considered. One option is to define A_l using either pre- or post-inference PC activity alone. In this case, A_l is a precision matrix over activities, potentially implemented with local lateral connectivity as proposed for PC [99]. As a further simplification, PC may encourage decorrelated activities via mutual inhibition between neurons. If activities within a layer are uncorrelated and maintain a roughly constant norm, A_l reduces to a multiple of the identity. The explicit decorrelation matrices can then be omitted while still preserving near-optimal alignment across batches of inputs.

4.5 Methods

4.5.1 Training data

All experiments in this Chapter use synthetically generated data. Input vectors are sampled independently from a standard normal distribution, $x \sim \mathcal{N}(0, I)$, unless stated otherwise. Target outputs are generated via a fixed linear mapping,

$$y = W_{\text{data}}x,$$

where W_{data} is sampled once per experiment seed with entries drawn independently from a zero-mean normal distribution with variance $1/n_{\text{in}}$, where n_{in} refers to the number of input neurons. This scaling ensures that output magnitudes remain comparable across different input dimensionalities. For experiments involving mini-batches, each sample within a batch is generated independently using the same procedure. Inputs and targets are resampled for each experimental run and random seed.

An exception to this data generation procedure is the experiment measuring the propagation of activation norms across network depth in Figure 4.3. In this case, inputs are sampled from a standard normal distribution and subsequently normalised to unit Euclidean norm to ensure a norm of one for all samples

$$x \leftarrow \frac{x}{\|x\|}.$$

This normalisation isolates the effect of network depth and weight initialisation on the evolution of feedforward activity magnitudes.

4.5.2 Training BP models

Linear models

DLNs trained with BP are optimised by minimising a mean squared error objective using stochastic gradient descent. For a mini-batch of B input-output pairs $\{(x_b, y_b)\}_{b=1}^B$, The continuous-time gradient descent dynamics for the weights of layer l are given by

$$\frac{\partial V_l}{\partial t} = -\frac{1}{B} \sum_{b=1}^B \nabla_{V_l} \mathcal{L}_{\text{BP}} = \frac{1}{B} \sum_{b=1}^B V_{L-1:l+1}^\top (y_b - V_{L-1:1} x_b) (V_{l-1:1} x_b)^\top, \quad (4.32)$$

where $V_{L-1:l+1}$ and $V_{l-1:1}$ denote the products of weight matrices above and below layer l , respectively, with the convention $V_{L-1:L} = V_{0:1} = I$.

In practice, these dynamics are discretised using an explicit Euler step, yielding the weight update

$$\Delta V_l = \alpha \frac{1}{B} \sum_{b=1}^B V_{L-1:l+1}^\top (y_b - V_{L-1:1} x_b) (V_{l-1:1} x_b)^\top, \quad (4.33)$$

where α denotes the learning rate. We use a fixed learning rate of $\alpha = 10^{-4}$ in all experiments. This learning rule is commonly employed for training deep neural networks. Varying the learning rate does not affect the results.

We perform only one weight update step in all experiments except the experiments illustrating learning trajectories in Figures 4.6 and 4.9. For these experiments on learning trajectories, we update the weights 100 times using a learning rate of 0.1. These parameters ensure that the model learns the input-output mappings of the samples.

Non-linear models

Training in deep non-linear networks follows the same procedure, but with a non-linear forward mapping. The sample-wise loss function is given by

$$\mathcal{L}_{\text{BP}}(x, y, V) = \frac{1}{2} \|y_b - f_V(x_b)\|_2^2, \quad (4.34)$$

where the network output is defined as

$$f_V(x) = V_{L-1}f(V_{L-2}f(\cdots f(V_1x))), \quad (4.35)$$

and $f(\cdot)$ denotes the ReLU activation function applied elementwise. Gradients are computed via standard BP through the non-linear activations, and the resulting weight updates are applied using the same discrete-time update rule and learning rate as in the linear case.

4.5.3 Training PC models

Linear models

DLNs trained with PC minimise local prediction errors by first iteratively updating the neural activity until convergence, followed by a single weight update with stochastic gradient descent. In the experiments, the steady-state activity for each DLN layer x_l^* is computed using the expression given in equation (4.3). This is equivalent to iterative inference but is less computationally intensive. For a mini-batch of B input-output pairs, the continuous-time gradient descent dynamics for the weight of layer l are given by

$$\frac{\partial V_l}{\partial t} = -\nabla_{V_l} E_{\text{disc}} = \frac{1}{B} \sum_{b=1}^B V_{L-1:l+1}^\top S^{-1}(y_b - V_{L-1:1}x_b) x_{l-1,b}^{*\top}.$$

In the experiments, these dynamics are discretised using an explicit Euler step, yielding

$$\Delta V_l = \alpha \frac{1}{B} \sum_{b=1}^B V_{L-1:l+1}^\top S^{-1}(y_b - V_{L-1:1} x_b) x_{l-1,b}^{*\top}. \quad (4.36)$$

where α denotes the learning rate. We use a fixed learning rate of $\alpha = 10^{-4}$ in all experiments, except stated otherwise. Varying the learning rate does not affect the results.

In all experiments, we perform a single iteration of the two-step training procedure consisting of inference followed by one weight update, except for the experiments illustrating learning trajectories in Figures 4.6 and 4.9. For these experiments, we repeat the two-step training procedure 100 times using a learning rate of 0.1, ensuring that the model learns the input-output mappings.

Non-linear models

For deep non-linear networks, PC’s energy function is given by

$$E_{disc}(x, V) = \frac{1}{B} \sum_{b=1}^B \sum_{l=2}^L \frac{1}{2} \|x_{l,b} - f(V_{l-1} x_{l-1,b})\|_2^2.$$

where $f(\cdot)$ is a ReLU activation function. The feedforward pass of this model is identical to that of the corresponding non-linear model trained with BP. The training procedure otherwise follows that used for PC in DLNs. However, inference is no longer analytically tractable in the non-linear setting. Consequently, the converged neural activities x_l^* are obtained via iterative inference. Inference is performed by minimising PC’s energy function using gradient descent for 10,000 steps, with an inference step of 0.05. We use a large number of inference steps to ensure that the neural activity reaches steady-state.

4.5.4 Mechanisms for improving alignment

In several experiments, we modify the standard PC and BP weight updates by introducing additional multiplicative factors. These factors are designed to reduce interference, thereby improve target alignment. Two classes of factors are considered: (i) scalar, activity-dependent learning factors for single-sample training, and (ii) matrix-valued decorrelation factors for batch training.

4.5.4.1 Activity-dependent learning rates

We first introduce layer-specific scalar factors that rescale the weight updates at each layer. For PC, the continuous-time weight update at layer l is modified as

$$\frac{\partial V_l}{\partial t} = -\alpha_l \nabla_{V_l} E_{disc}. \quad (4.37)$$

The scaling factor α_l is chosen to cancel the activity-dependent scalar term appearing in PC's change in prediction and is defined as

$$\alpha_l = \frac{1}{E_B[x_{l-1}^{*\top} \hat{x}_{l-1}]}, \quad (4.38)$$

where $\hat{x}_{l-1}$ denotes the feedforward activity at layer $l-1$, x_{l-1}^* denotes the converged activity following PC inference, and $E_B[\cdot]$ denotes the expectation across the batch elements.

For comparison, we also apply analogous activity-dependent scaling to BP. In this case, the scaling factor at layer l is given by

$$\alpha_l = \frac{1}{E_B[\hat{x}_{l-1}^\top \hat{x}_{l-1}]}, \quad (4.39)$$

which removes the corresponding activity-dependent scalar factor from BP's prediction update.

Introducing these activity-dependent learning factors is equivalent to using layer-dependent learning rates. For instance, for networks trained with PC, the discretised weight updates given in equation (4.36) become

$$\Delta V_l = \alpha_l \alpha \frac{1}{B} \sum_{b=1}^B V_{L-1:l+1}^\top S^{-1} (y_b - V_{L-1:1} x_b) x_{l-1,b}^{*\top}.$$

Here, α denotes the global learning rate and α_l is a layer-specific scaling factor. Their product can therefore be interpreted as an effective learning rate for layer l .

In all experiments employing these scaling factors, the layer-specific coefficients α_l are recomputed for each batch of samples and applied only for the duration of a single weight update. Apart from the inclusion of these factors, the training procedure is identical to that used without scaling. In particular, all experiments use the same global learning rate and the same number of inference steps.

4.5.4.2 Decorrelation matrix factors

We next introduce layer-specific matrix-valued factors that rescale the weight updates to remove inter-sample interference. These factors are designed to cancel the activity-dependent matrix terms that appear in the change in prediction when learning from multiple samples, thereby improving target alignment.

For PC, the continuous-time weight update at layer l is modified as

$$\frac{\partial V_l}{\partial t} = -\frac{1}{B} \sum_{b=1}^B \nabla_{V_l} E_{disc} A_l, \quad (4.40)$$

where A_l is a decorrelation matrix applied to the right of the gradient.

The decorrelation matrix is chosen to cancel the batch-averaged activity correlation term appearing in PC’s change in prediction. Specifically, we define

$$A_l = \left(E_B [x_{l-1,b}^{*\top} \hat{x}_{l-1,b}] \right)^{-1}, \quad (4.41)$$

where $\hat{x}_{l-1,b}$ denotes the feedforward activity at layer $l-1$ for sample b , $x_{l-1,b}^*$ denotes the corresponding converged activity following PC inference. $E_B[\cdot]$ denotes the expectation over batch elements calculated with the empirical mean across the batch.

For comparison, we also apply an analogous decorrelation factor to BP. In this case, the matrix factor is defined using feedforward activities only:

$$A_l = \left(E_B [\hat{x}_{l-1,b}^\top \hat{x}_{l-1,b}] \right)^{-1}. \quad (4.42)$$

These matrix-valued factors are recomputed separately for each layer at every weight update. Matrix inversion is performed using the Moore–Penrose pseudoinverse to maintain numerical stability. The training procedure remains unchanged with identical training parameters to those used in the standard settings.

4.5.5 Experimental model configurations

All experiments use fully connected DLNs. The default architecture consists of a linear model with 512 input neurons, a single hidden layer of 512 neurons, and 512 output neurons. This model is initialised using Kaiming uniform initialisation. The

models is trained either on a single sample (Figures 4.4 and 4.5) or on batches of 128 samples (Figure 4.8), depending on the experiment.

Several initialisations are considered in Section 4.3.3. Under Kaiming uniform, each weight matrix is sampled as $V_l \sim U\left(-\sqrt{\frac{1}{n_l}}, \sqrt{\frac{1}{n_l}}\right)$, where n_l denotes the input dimensionality of layer l . Under LeCun normal initialisation, weights are sampled according to $V_l \sim \mathcal{N}\left(0, \frac{1}{n_l}\right)$. Orthogonal initialisation is performed by setting the condition number of one, as described in the next section.

Architectural variations are introduced by independently varying network depth and width. Network depth is defined as the number of hidden layers and is varied over values $1, \dots, 8$. When varying depth, all hidden layers have equal width. Network width is varied over 128, 256, 512, 1024 neurons, with input and output dimensions set to 512 neurons. The settings with 128 and 256 hidden neurons cause an information bottleneck. When width is varied, the model contains a single hidden layer.

To examine the effect of weight conditioning, we explicitly control the condition number of the weight matrices. Conditioning values are varied over $1, 2, 10, 50, 10^3, 10^4, 10^5, 10^7, 10^9, 10^{12}$. Conditioning is imposed by rescaling the singular values of initially Kaiming-initialised weight matrices while preserving their singular vectors.

Batch size is varied to study inter-sample interference effects. Batch sizes are varied across 1, 32, 64, 128, 256, 480, 550, 1000, 2048. For experiments involving decorrelation matrices, batch sizes exceeding the smallest layer width are included to probe the limits of exact interference cancellation.

Across all experiments, parameters that are not explicitly varied are fixed to their default values. Results are averaged over multiple random seeds as specified in the corresponding figure captions.

4.5.6 Metrics

4.5.6.1 Target alignment

Target alignment quantifies how closely a learning update follows the direction of the output prediction error. Following equation (1.6), target alignment is defined as the cosine similarity between the output residual $r = y - \hat{y}$ and the change in the

model’s prediction induced by a weight update, $\Delta\hat{y}$ (see equation (4.6)). A value of one indicates perfect alignment with the residual, zero indicates orthogonality, and negative values indicate updates that move the prediction away from the target. In experiments involving mini-batches, target alignment is computed separately for each sample and reported as the mean across samples. All reported results correspond to the alignment induced by a single parameter update.

4.5.6.2 Condition number

The conditioning of a model’s weights is quantified using the condition number of each weight matrix. For a weight matrix V_l , the condition number is defined as

$$\kappa(V_l) = \frac{\sigma_{\max}(V_l)}{\sigma_{\min}(V_l)}, \quad (4.43)$$

where $\sigma_{\max}$ and $\sigma_{\min}$ denote the largest and smallest singular values, respectively. A condition number of one corresponds to perfectly conditioned (orthogonal) weights, while larger values indicate poorer conditioning and increased susceptibility to interference. When conditioning is varied experimentally, all layers are assigned the same condition number.

We set the conditioning by modifying the singular value spectrum of each weight matrix, while preserving its singular vectors. For a given layer l , we compute the singular value decomposition

$$V_l = U_l \Sigma_l V_l^\top, \quad (4.44)$$

where $\Sigma_l = \text{diag}(\sigma_1, \dots, \sigma_k)$ contains the singular values in descending order. We then replace the original singular values by a linearly spaced spectrum

$$\sigma'_i = \sigma_{\max} \left(1 - \frac{i-1}{k-1} \left(1 - \frac{1}{\kappa} \right) \right), \quad (4.45)$$

where $\sigma_{\max}$ is the largest original singular value, k is the rank of V_l , and κ is the desired condition number. This construction ensures $\sigma'_{\max}/\sigma'_{\min} = \kappa$.

To isolate the effect of conditioning from changes in the overall weight scale, the modified singular values, Σ'_l , are rescaled to preserve the Frobenius norm of the original weight matrix,

$$\Sigma'_l \leftarrow \Sigma'_l \frac{\|\Sigma_l\|_F}{\|\Sigma'_l\|_F}. \quad (4.46)$$

The conditioned weight matrix is then reconstructed as $V_l \leftarrow U_l \Sigma_l' V_l^\top$. This procedure alters only the conditioning of the weights while keeping their total energy and singular directions fixed.

4.5.7 Derivation of PC's weight updates

PC updates its weights by following the negative gradient of its energy function with respect to the synaptic parameters. For the PC energy function

$$E_{disc}(x, V) = \sum_{l=2}^L \frac{1}{2} \|x_l - V_{l-1} x_{l-1}\|_2^2,$$

the weight dynamics are given by

$$\frac{\partial V_l}{\partial t} = -\nabla_{V_l} E_{disc} = (x_{l+1}^* - V_l x_l^*) x_l^{*\top} = \epsilon_{l+1}^* x_l^{*\top},$$

where x_l^* denotes the equilibrium activity of layer l , and ϵ_l^* is the corresponding prediction error at equilibrium.

At the activity equilibrium reached during inference, the gradient of the energy with respect to the neural activities vanishes. This yields, for $l = 2, \dots, L-1$,

$$\nabla_{x_l} E_{disc} = (x_l^* - V_l x_{l-1}^*) - V_{l+1}^\top (x_{l+1}^* - V_{l+1} x_l^*) = 0.$$

Rearranging this expression gives

$$\begin{aligned} (x_l^* - V_{l-1} x_{l-1}^*) &= V_l^\top (x_{l+1}^* - V_{l+1} x_l^*), \\ \epsilon_l^* &= V_l^\top \epsilon_{l+1}^*. \end{aligned}$$

By recursively applying this relation, the prediction error at layer l can be expressed in terms of the output-layer error. Substituting into the weight update yields

$$\frac{\partial V_l}{\partial t} = \epsilon_l^* x_{l-1}^{*\top} = V_{L-1:l+1}^\top \epsilon_L^* x_{l-1}^{*\top}, \quad (4.47)$$

where $V_{L-1:l+1} = V_{L-1} \cdots V_{l+1}$.

The equilibrium activity can be written equivalently as

$$x_l^* = V_{l-1} x_{l-1}^* + \epsilon_l^*.$$

Using this recursive form, the output activity can be expanded as

$$\begin{aligned}
x_L^* &= V_{L-1}x_{L-1}^* + \epsilon_L^* \\
&= V_{L-1}(V_{L-2}x_{L-2}^* + \epsilon_{L-1}^*) + \epsilon_L^* \\
&= V_{L-1:1}x_1^* + V_{L-1:2}\epsilon_1^* + V_{L-1:3}\epsilon_2^* + \cdots + \epsilon_L^* \\
&= V_{L-1:1}x_1^* + \sum_{l=2}^L V_{L-1:l}\epsilon_{l-1}^* + \epsilon_L^*.
\end{aligned}$$

Since the input and output layers are clamped to the input x and target y , respectively, and using the recursive expression for the prediction errors, the above expression becomes

$$\begin{aligned}
x_L^* &= V_{L-1:1}x_1^* + \sum_{l=2}^{L-1} V_{L-1:l}\epsilon_{l-1}^* + \epsilon_L^*, \\
y &= V_{L-1:1}x + \sum_{l=2}^{L-1} V_{L-1:l}V_{L-1:l}^\top \epsilon_L^* + \epsilon_L^*, \\
y - V_{L-1:1}x &= \left(I + \sum_{l=2}^{L-1} V_{L-1:l}V_{L-1:l}^\top \right) \epsilon_L^*.
\end{aligned}$$

Using the definitions of S and r introduced in the background, this simplifies to

$$r = S \epsilon_L^*, \quad (4.48)$$

$$\epsilon_L^* = S^{-1}r. \quad (4.49)$$

Finally, combining equations (4.66) and (4.65), we obtain the expression used in the manuscript for the PC weight updates:

$$\boxed{\frac{\partial V_l}{\partial t} = V_{L-1:l+1}^\top \epsilon_L^* x_{l-1}^{*\top} = V_{L-1:l+1}^\top S^{-1}r x_{l-1}^{*\top}.}$$

4.5.8 Derivation of PC's weight updates

PC's weight updates follow the negative gradient of its energy function with respect to its weights. For PC's energy function given by

$$E_{disc}(x, V) = \sum_{l=2}^L \frac{1}{2} \|x_l - V_l x_{l-1}\|_2^2, \quad (4.50)$$

the weight updates are given by

$$\frac{\partial V_l}{\partial t} = -\nabla_{V_l} E_{disc} = (x_l^* - V_l x_{l-1}^*) x_{l-1}^{*\top} = \epsilon_l^* x_{l-1}^{*\top} \quad (4.51)$$

where x_l^* refers to the equilibrium activity of PC, and ϵ_l^* refers to the prediction error at equilibrium.

The derivative of PC's energy function with respect to the activities is equal to zero at the equilibrium of the activity inference. Consequently:

$$\nabla_{x_l} E_{disc} = 0 = (x_l^* - V_l x_{l-1}^*) - V_{l+1}^\top (x_{l+1}^* - V_{l+1} x_l^*) \quad (4.52)$$

By rearranging above expression, you get

$$\begin{aligned} (x_l^* - V_l x_{l-1}^*) &= V_{l+1}^\top (x_{l+1}^* - V_{l+1} x_l^*) \\ \epsilon_l^* &= V_{l+1}^\top \epsilon_{l+1}^* \end{aligned}$$

Using recursion the PC's weight update is given by

$$\frac{\partial V_l}{\partial t} = \epsilon_l^* x_{l+1}^{*\top} = V_{L-1:l+1}^\top \epsilon_L^* x_{l-1}^{*\top} \quad (4.53)$$

The expression for the layer prediction errors can be rearranged as follows:

$$x_l^* = V_l x_{l-1}^* + \epsilon_l^*$$

Using this rearranged expression the output of the model can be written as:

$$x_L^* = V_L x_{L-1}^* + \epsilon_L^* \quad (4.54)$$

$$= V_L (V_{L-1} x_{L-2}^* + \epsilon_{L-1}^*) + \epsilon_L^* \quad (4.55)$$

$$= V_{L-1:1} x_1^* + V_{L-1:2} \epsilon_1^* + V_{L-1:3} \epsilon_2^* + \dots + \epsilon_L^* \quad (4.56)$$

$$= V_{L-1:1} x_1^* + \sum_{l=2}^L V_{L-1:l} \epsilon_{l-1}^* + \epsilon_L^* \quad (4.57)$$

$$(4.58)$$

By using the facts that the input neurons and output neurons are clamped to the input x and target y , and using above's recursive formula for the prediction errors,

above's expression can be rewritten to:

$$x_L^* = V_{L-1:1}x_1^* + \sum_{l=2}^L V_{L-1:l}\epsilon_{l-1}^* + \epsilon_L^* \quad (4.59)$$

$$y = V_{L-1:1}x + \sum_{l=2}^L V_{L-1:l}V_{L-1:l}^\top \epsilon_L^* + \epsilon_L^* \quad (4.60)$$

$$y - V_{L-1:1}x = \left(I + \sum_{l=2}^L V_{L-1:l}V_{L-1:l}^\top \right) \epsilon_L^* \quad (4.61)$$

$$(4.62)$$

By using the expressions for S and r defined in the manuscript, the expression can be further simplified to

$$r = S \epsilon_L^* \quad (4.63)$$

$$S^{-1}r = \epsilon_L^* \quad (4.64)$$

$$(4.65)$$

By combining the expressions given in equations (4.66) and (4.65), we obtain the expression used in the manuscript for the weight updates of PC:

$$\boxed{\frac{\partial V_l}{\partial t} = V_{L-1:l+1}^\top \epsilon_L^* x_{l-1}^{*\top} = V_{L-1:l+1}^\top S r x_{l-1}^{*\top}} \quad (4.66)$$

Chapter 5

General Discussion

5.1 Summary of results

This thesis explores how predictive coding can be extended to support a wider range of computations within a single, biologically plausible network.

Chapter 2 shows how predictive coding can infer full posterior distributions using local noisy neural dynamics, instead of being limited to maximum a posteriori estimates [99]. We introduce Monte Carlo Predictive Coding (MCPC), an implementation of predictive coding with stochastic neural dynamics. These dynamics enable the inference over full posterior distributions, which can be used to learn accurate probabilistic models of sensory inputs. The proposed neural dynamics also account for variability in cortical activity.

Chapter 3 brings together top-down predictive coding [70] and bottom-up predictive coding [103], integrating their respective learning capabilities within a single neural network governed by local computations and plasticity. In the resulting model, termed bidirectional predictive coding (bPC), neurons generate predictions about both the layer above and the layer below within the hierarchy. Empirically, we find that bPC unifies the strengths of these, previously separate, predictive coding formulations without any substantial loss in performance. A single model can, for example, combine the strong supervised classification performance associated with bottom-up predictive coding with the representational learning characteristic of top-down predictive coding. Moreover, it integrates classification ability with the capacity to reconstruct missing inputs, making it far more robust to partially occluded inputs

than conventional classifiers. Overall, bPC provides a potential explanation of how the distinct learning capabilities of multiple predictive coding formulations can be combined within a single model.

Chapter 4 offers a principled analytical account of when and why predictive coding shows less destructive interference than backpropagation, as observed experimentally [118]. We derive closed-form expressions for the changes in model predictions induced by backpropagation and predictive coding in deep linear networks, revealing a fundamental difference in how interference arises during learning. Backpropagation updates are influenced by parameter-dependent interference. Predictive coding updates combine this parameter-dependent interference with a matrix normalisation factor. These two effects counterbalance one another, resulting in improved alignment for predictive coding. This advantage is particularly evident in networks with narrow hidden layers and in those with correlated or poorly conditioned weights. Finally, we introduce methods that ensure interference-free learning in predictive coding when training on single data samples as well as on batches.

5.2 Does the research of the thesis meet the constraints of learning in the brain?

In the introduction, we outline three requirements that learning algorithms in the brain should satisfy, namely these algorithms should be able to: (i) train hierarchical neural networks, (ii) use local computations, (iii) solve complex tasks. Here, we evaluate whether the research presented in this thesis meets these requirements.

5.2.1 Monte Carlo predictive coding

MCPC satisfies both the hierarchy and locality criteria, yet it becomes prohibitively expensive to simulate in high-dimensional settings. For complex datasets, it is therefore better deployed alongside other representational learning models that can first capture the structure of high-dimensional data [3].

Like other predictive coding frameworks, MCPC acquires a hierarchical structure that reflects the organisation of the cortex. It departs from standard predictive coding, by introducing additional noise into the activity dynamics. This noise is local to

each neuron and independent across neurons, ensuring that both neural activity and synaptic plasticity remain entirely local.

Using Gaussian data and small-scale tasks, we show that MCPC can learn accurate distributions of sensory inputs. However, the time required for MCPC inference increases markedly with the number of neurons, limiting its practicality in larger models. It is unlikely to scale effectively to highly complex image distributions, such as those encountered in the visual stream [29].

Nevertheless, the brain may not need to learn full sensory distributions directly, as it relies on the interaction of multiple systems [29]. The sensory system could instead develop compact representations of inputs, reducing them to low-dimensional latent states [251, 252, 253]. In this case, the brain could use MCPC to learn probability distributions over these lower-dimensional representations.

5.2.2 Bidirectional predictive coding

bPC satisfies all the required constraints: it learns hierarchical models through entirely local computations and scales effectively to complex learning tasks.

bPC enables multilayered neural networks to learn using fully local rules, for both neural activity and synaptic plasticity. However, compared to other predictive coding models, bPC imposes additional architectural constraints. Specifically, it includes two populations of error neurons for each value neuron. Dual error-neuron populations have been observed in the cortex [216], mirroring the two error types in bPC. While some accounts attribute these populations to physiological constraints [109], in bPC they emerge naturally from computational principles.

In terms of performance, bPC matches backpropagation on challenging tasks while retaining the full range of predictive learning capabilities within a single framework. We demonstrate that it scales to networks exceeding ten layers and performs effectively on datasets such as CIFAR-10, CIFAR-100, and Tiny-ImageNet. bPC approaches the learning performance of backpropagation on both supervised and unsupervised tasks for these complex datasets.

5.2.3 Target alignment of predictive coding

The research presented in Chapter 4 primarily addresses the third constraint by analysing the conditions under which predictive coding should outperform backpropagation. Given that the analysis focuses on top-down predictive coding in deep linear models, it also satisfies the first two requirements: learning hierarchical neural networks through local computations.

Our analysis identifies two modifications to predictive coding that guarantee interference-free learning. However, neither modification is fully local. The first relies on temporal delays to combine neural activity measured before and after PC inference. The second also depends on temporal delays, in addition to requiring lateral connectivity spanning all neurons within a layer. Further investigation is needed to understand how the brain might approximate these mechanisms using strictly local computations while preserving high target alignment. A likely route is through inhibitory or homeostatic processes. Such mechanisms could regulate the overall scale of neural activity across layers, reducing the need for the first modification. Promoting decorrelated neural representations through lateral inhibition would remove the benefit of the second.

Both our experiments and the analysis of target alignment suggest that predictive coding should generally outperform backpropagation. However, the advantage is likely to be most pronounced in deep models with narrow hidden layers and in networks that are already partially trained, where weights have developed correlated structure. Predictive coding may therefore be particularly well suited to training deep autoencoders, continual learning, and fine-tuning pre-trained models.

It is important to note that the analysis assumes that predictive coding converges during inference. In practice, full convergence is often computationally prohibitive, and only a small number of activity update steps are performed [3]. In this regime, predictive coding approximates backpropagation [254, 104], reducing its distinct advantages. We therefore expect predictive coding to offer a learning benefit over backpropagation, but at the cost of increased computational demands.

5.3 Future work

5.3.1 Robust perception with predictive coding

A promising direction for future work is to combine MCPC with bPC to develop a unified account of robust, uncertainty-aware perception. In such a framework, bPC could learn low-dimensional latent structures from complex sensory data through fully local computations. MCPC would then operate on the resulting low-dimensional latent states, performing probabilistic inference via stochastic neural dynamics. Rather than attempting to sample directly in high-dimensional sensory spaces, MCPC would model uncertainty over compact representations learned by bPC, enabling principled posterior inference about the state of the world. This combination would allow the system not only to classify or reconstruct inputs but also to quantify uncertainty and support decision-making under ambiguity, bringing predictive coding closer to normative accounts of Bayesian perception [81, 83] and optimal decision-making [77].

However, to further align the model with cortical function, this integrated framework should be extended to temporal settings, and its learning method might need to be extended to self-supervised learning. Incorporating temporal predictive coding would enable the network to learn structured dynamics of the environment, capturing how environmental states evolve over time [101, 114]. Substantial evidence also indicates that the brain learns the dynamics of stimuli as illustrated by motion-sensitive neurons [255]. MCPC and bPC could be extended using temporal predictive coding [114]. The resulting models would have an additional set of weights to predict future neural activity from past latent states (see Tang et al. [114] for details), which can be implemented using local computations. A natural learning objective in this regime is temporal self-supervised learning, which may reflect a central learning principle of neocortical learning [217]. Importantly, bPC is well-suited for self-supervised learning: unlike many discriminative architectures used in self-supervised learning, it does not admit trivial representational collapse [219]. This is because top-down predictions require latent variables to explain away sensory inputs across the hierarchy. This learning signal may provide a biologically plausible mechanism for representation learning, while MCPC would furnish uncertainty estimates over both current latent states and predicted futures. Together, these extensions would move predictive coding towards a theory of robust, temporally extended perception and decision-making in the brain.

5.3.2 Scaling predictive coding to brain-sized networks

A central obstacle to scaling predictive coding to brain-sized models is the cost and stability of iterative inference. Although current implementations already handle moderately deep architectures, they remain far removed from the scale, heterogeneity, and dynamical richness of cortical circuits. The requirement to run iterative activity updates substantially increases computational overhead. This will be especially pronounced in regimes where predictive coding is expected to outperform backpropagation. Indeed, the results in Chapter 4 suggest that improved target alignment and reduced destructive interference are most pronounced when inference reaches equilibrium. In practice, however, predictive coding is trained with only a few inference steps [3], thereby diminishing these benefits.

Encouragingly, recent work has begun reducing the computational cost of iterative inference. Reparameterising predictive coding to optimise error variables, rather than neural activities, yields identical fixed points while dramatically accelerating convergence [4]. This reparametrisation reduces the number of inference steps to reach equilibrium by orders of magnitude, such as a 140x decrease in the number of steps in a 20-layer linear network [4]. Other studies demonstrate that appropriate activity initialisation can significantly shorten the trajectory to equilibrium as well [256], suggesting that activity initialisation may be a fruitful direction. Beyond algorithmic acceleration, a deeper theoretical understanding of stability is required. Identifying the conditions under which the predictive coding equilibrium is stable, well-conditioned, and rapidly reachable would inform principled parameterisations of weights. While recent analyses have characterised regimes of stability and instability [105], the geometry of the inference landscape in nonlinear networks remains poorly understood. Clarifying how architectural choices, weight correlations, and layer widths shape convergence dynamics is essential for reliable large-scale deployment.

Finally, substantial progress may depend on advances in hardware. The continuous, relaxation-based dynamics of predictive coding are naturally aligned with analog computation, in which physical processes implement iterative inference directly. Analog substrates promise lower power consumption, massive parallelism, and reduced data movement, all of which are critical at scale [26, 27]. Although predictive coding has not yet been realised in analog hardware, closely related frameworks such as equilibrium propagation [257] and sparse coding [133] have already been implemented in

analog systems. Moreover, intrinsic hardware noise can be harnessed as a computational resource, providing an efficient substrate for stochastic inference [133], which could be used for Monte Carlo predictive coding. Developing hardware that exploits these dynamical and stochastic properties may ultimately be decisive in extending predictive coding to models approaching the size and efficiency of the brain.

5.4 Concluding remarks

More broadly, this thesis speaks to two larger questions. The first concerns neuroscience. The aim is not to pin down the exact mechanism the brain uses to learn. It is to show that effective, scalable learning is possible within the brain’s known constraints: hierarchical processing and local computation. Predictive coding is one proof that it is achievable. The second concerns machine learning. Here, algorithms such as predictive coding matter not only as brain models, but as practical alternatives to backpropagation. They can beat backpropagation on biologically relevant problems, including continual, online, and few-shot learning. Their local, relaxation-based dynamics also suit the massively parallel and analog hardware discussed above [26, 27]. Pursued together, both directions may bring us closer to understanding learning in the brain and to building more efficient learning machines.

Appendix A

Monte Carlo predictive coding

A.1 Supplementary figures

Figures A.1, A.2, and A.3.

A.2 Supplementary videos

A1 Video. MCPC posterior inference in linear model. Animation of the activity of the latent state in a linear model with one latent state during MCPC inference for a constant input. In this animation, the orange dot shows the time-varying activity of the latent state. The blue histogram summarises the activity of the latent state from the beginning of the animation to the time point in the animation being visualized. Finally, the black curve shows the true posterior distribution that can be analytically calculated from the model parameters and the input to the model.

A2 Video. MCPC posterior inference in non-linear model for half masked MNIST digit. Animation of the activity of the latent layer x_L in a non-linear model trained on the MNIST dataset during MCPC inference for a half-masked digit input. The orange dot shows the time-varying activity of the latent state x_L transformed to coordinates using a linear classifier and a convex combination of 10 evenly spaced points on a unit circle. The linear classifier is trained to decode digit class distributions from the latent state x_L . The decoded class distribution can then be transformed to

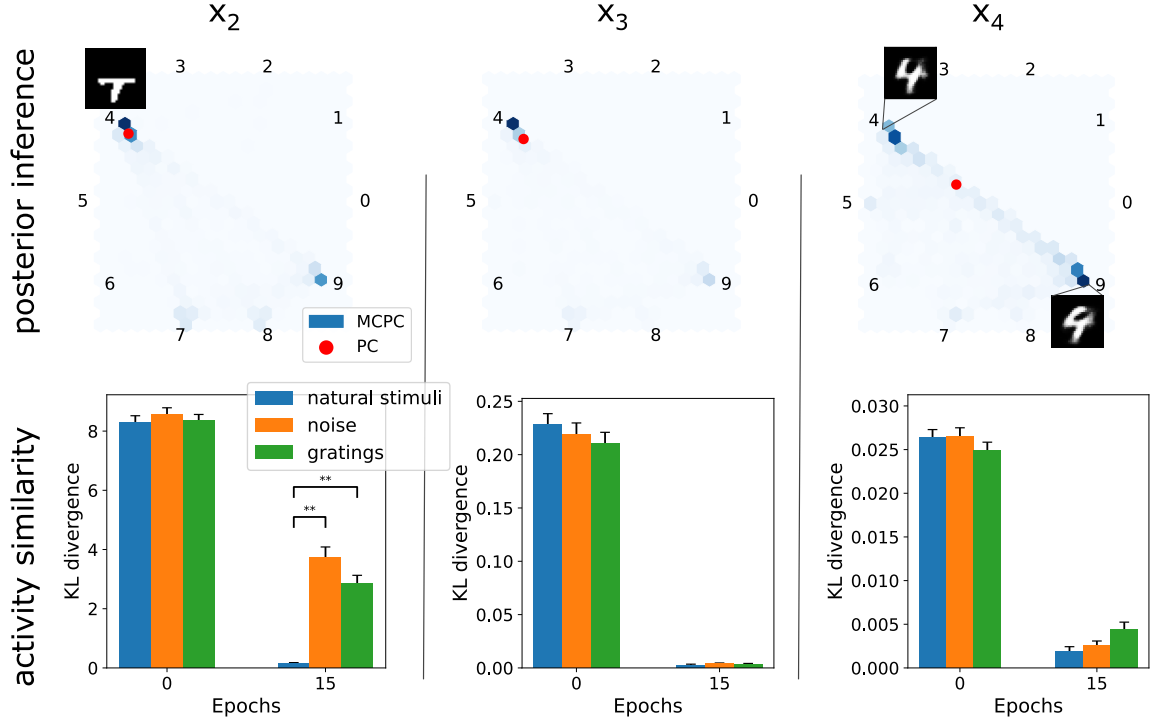


Figure A.1: **Masked inference and similarity increase across model hierarchy.**
top. We visualize the latent layer activity for a masked input (top left) across all latent layers for PC and MCPC, following the method outlined in Section 4.2.2 of the manuscript. The MCPC model identifies different potential interpretations for a given masked input across its latent layers, whereas the PC model infers only one possible interpretation. Additionally, we visualize the reconstructed images by the MCPC model when the latent layers represent two different possible interpretations. The reconstructed digit when the MCPC model infers a “4” resembles the digit four, while the reconstructed digit when the model infers a “9” resembles the digit nine.
bottom. We repeat the analysis to assess the similarity between spontaneous activity and average evoked activity across all latent layers, following the method described in Section 4.3.2. The KL divergence between spontaneous activity and average evoked activity for natural stimuli is lower compared to noise images and image gratings for an MNIST-trained MCPC model across all latent layers. However, this difference is only statistically significant in the first latent layer x_2 and not in the higher latent layers.

a coordinate using the convex combination. The blue hexagons show the probability density of a two-dimensional histogram of the activity of the latent state from the beginning of the animation to the time point in the animation being visualized.

A3 Video. MCPC posterior inference in non-linear model for full MNIST digit. Animation of the activity of the latent layer x_L in a non-linear model trained

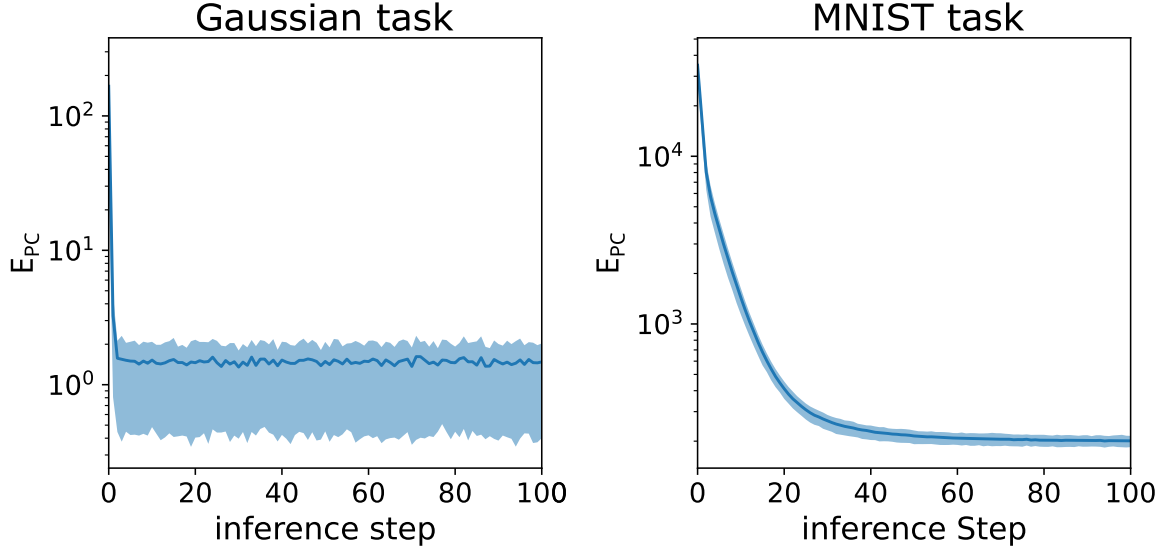


Figure A.2: **Negative joint log-likelihood of model during MCPC inference.** Sampling using Langevin dynamics has been reported to exhibit large mixing times that scale exponentially with the number of dimensions. We employed at least 50 MCPC inference steps along with a PC warmup before sampling for all our experiments. This figure illustrates the negative joint log-likelihood of MCPC models, $E_{gen} = E_{PC} = \frac{1}{2} \sum_{l=1}^{L-1} \frac{\|x_l - W_l \cdot f(x_{l+1})\|^2}{\sigma^2} + \frac{1}{2} \frac{\|x_L - \mu\|^2}{\sigma^2}$, during inference averaged for 256 data samples for a model for **(left)** the Gaussian task $\{W_1 = 2, \mu = 0\}$ and **(right)** of a model trained on MNIST. This figure demonstrates that the average sum of prediction errors of the model converges in fewer than 50 inference steps, indicating that the models have likely reached a steady state. This result suggests that MCPC’s convergence time remains manageable even as the size of the latent state increases from one neuron in the Gaussian task to 276 neurons across three layers for the MNIST task. However, larger models might require a convergence time that is beyond practical limits. All the latent variables are randomly initialised before inference following a uniform distribution between -10 and 10. Both models have a learning rate of 0.1 for the activity updates. The shaded region represents the interquartile range.

on the MNIST dataset during MCPC inference for a full digit input. The orange dot and the blue hexagons have been determined as described in A2 Video.

A4 Video. MCPC unclamped activity of sensory input neuron in a linear model. Animation of the activity of the input neuron in a linear model with one latent state and one input neuron resulting from MCPC dynamics when the input neuron is unclamped. The orange dot shows the input neuron activity over time. The histogram summarises the activity of the input state from the beginning of the animation to the time point in the animation being visualized. The black curve

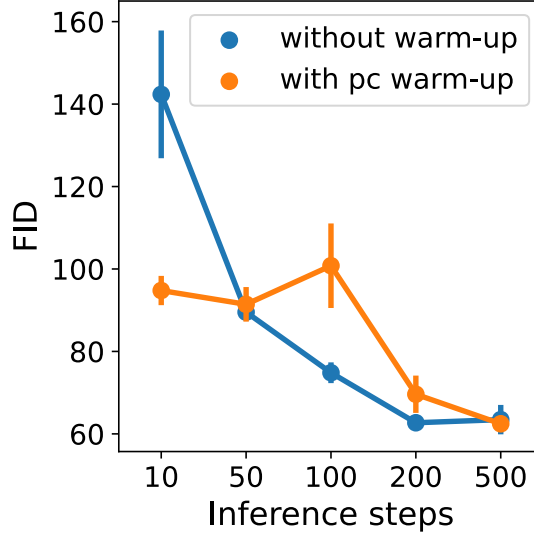


Figure A.3: **MCPC with and without PC warm-up inference steps trained on the MNIST dataset.** We train an MCPC model on the MNIST dataset with and without warm-up steps. Moreover, we evaluate a range of inference step counts. When the model is trained without warm-up steps, the inference process includes MCPC mixing steps followed by a single MCPC sampling step. However, when the model undergoes training with PC warm-up steps, the initial half of the inference steps consist of PC inference steps, while the remaining inference steps are MCPC mixing steps and one MCPC sampling step. The model parameters for training can be found in A.6 and correspond to the parameters that maximise the FID measure. This figure demonstrates that using PC warm-up inference steps results in improved performance with a limited number of total inference steps, diminished performance with 100 or 200 inference steps, and comparable performance with a large number of inference steps. Ultimately, this result shows that warm-up steps are not always beneficial and should be considered for each learning task separately. The results are shown for three initialisation seeds. The dots show the mean result, while the error bars show the standard deviation.

shows the marginal likelihood that can be analytically calculated from the model parameters.

A5 Video. MCPC unclamped activity of sensory input neurons in non-linear model trained on MNIST. Animation of the activity of the input neurons in a non-linear model trained on MNIST resulting from MCPC dynamics when the input neurons are unclamped.

A.3 Learning in a linear model with one input neuron and one latent state using MCPC and PC

We compare training the linear model given in Figure 1a of the main text using MCPC and PC. We find that MCPC only has equilibrium points at the optimal model parameters while PC does not have an equilibrium for finite weights. This demonstrates the theoretical superiority of MCPC over PC. Additionally, this elucidates why the weights in PC become excessively large, which is observed across tasks. Here, we consider a linear model with identity covariance matrices for simplicity. However, the result generally holds for any covariance matrices.

A.3.1 Learning with MCPC

Inference. MCPC infers the posterior distribution $p(x_2|x; \theta)$ which for the considered model equals:

$$\begin{aligned} p(x_2|x; \theta) &= \frac{p(x, z; \theta)}{p(x; \theta)} = \frac{N(x; W_1 x_2, I) N(x_2; \mu, I)}{N(x; W_1 \mu, W_1^2 + 1)} \\ &= N\left(x_2; \frac{W_1 x + \mu}{W_1^2 + 1}, \frac{1}{W_1^2 + 1}\right). \end{aligned}$$

Expected parameter updates. The expected parameter updates can be found using MCPC's equation for parameter updates. For the weight W_1 , the expected parameter update is given by:

$$\begin{aligned} E\{\Delta W_1\} &\propto E\{E_{p(x_2|x; \theta)}\{\epsilon_1 x_2\}\} = E\left\{\int (x - W_1 x_2) x_2 p(x_2|x; \theta) dx_2\right\} \\ &= E\left\{\int (x - W_1 x_2) x_2 N\left(x_2; \frac{W_1 x + \mu}{W_1^2 + 1}, \frac{1}{W_1^2 + 1}\right) dx_2\right\} \\ &= \frac{1}{(W_1^2 + 1)^2} \left(-W_1 (Var\{x\} + E\{x\}^2) + \mu(W_1^2 - 1)E\{x\} + W_1^3 + W_1\mu^2 + W_1 \right). \end{aligned}$$

For the prior mean μ , the expected parameter update is given by:

$$\begin{aligned} E\{\Delta \mu\} &\propto E\{E_{p(x_2|x; \theta)}\{\epsilon_2\}\} = E\left\{\int (x_2 - \mu) p(x_2|x; \theta) dx_2\right\} \\ &= E\left\{\int (x_2 - \mu) N\left(x_2; \frac{W_1 x + \mu}{W_1^2 + 1}, \frac{1}{W_1^2 + 1}\right) dx_2\right\} \\ &= -\frac{W_1}{W_1^2 + 1} (E\{x\} - W_1 \mu). \end{aligned}$$

In these expressions, $E\{x\}$ equals the mean of the data and $Var\{x\}$ equals the variance of the data.

Nullclines. The nullclines of the model can be found by equating the expected parameter updates to zero. Therefore, the nullclines for $E\{\Delta W_1\} = 0$ are given by the expression:

$$\mu = \frac{-(W_1^2 - 1)E\{x\} \pm \sqrt{(W_1^2 - 1)^2 E\{x\}^2 + 4W_1^2(Var\{x\} + E\{x\}^2 - W_1^2 - 1)}}{2W_1}.$$

Similarly, the nullcline for $E\{\Delta\mu\} = 0$ is given by the expression:

$$\mu = \frac{E\{x\}}{W_1}.$$

Equilibrium points. Equilibrium points lie at the intersection between the nullclines $E\{\Delta W_1\} = 0$ and the nullcline $E\{\Delta\mu\} = 0$. By equating the nullclines, we obtain the condition on equilibrium points:

$$W_1^2 = Var\{x\} + 1.$$

The parameter values of the equilibrium points are therefore $\{W_1 = \pm\sqrt{Var\{x\} + 1}, \mu = \pm\frac{E\{x\}}{\sqrt{Var\{x\} + 1}}\}$. These values are optimal because a model with these parameter values has a marginal likelihood $p(x; \theta)$ equal to $N(x; E\{x\}, Var\{x\})$.

A.3.2 Learning with PC

Inference. For the linear model, PC inference converges to the latent state z_2^* for a given input x .

$$\frac{\partial z_2^*(t)}{\partial t} = 0 = -\epsilon_2^* + W_1\epsilon_1^* = -(z_2^* - \mu) + W_1(z_1^* - W_1 z_2^*)$$

$$z_2^* = \frac{\mu + W_1 x}{1 + W_1^2}$$

Expected parameter updates. The expected parameter updates can be found using PC's equation for parameter updates. For the weight W_1 , the expected parameter

update is given by:

$$\begin{aligned}
E\{\Delta W_1\} &\propto E\{e_1^* z_2^*\} = E\{(x - W_1 z_2^*) z_2^*\} \\
&= E\left\{\left(x - W_1 \frac{\mu + W_1 x}{1 + W_1^2}\right) \frac{\mu + W_1 x}{1 + W_1^2}\right\} \\
&= \frac{1}{(1 + W_1^2)^2} \left(W_1 E\{x^2\} + (\mu - W_1^2 \mu) E\{x\} - W_1 \mu^2\right) \\
&= \frac{1}{(1 + W_1^2)^2} \left(W_1 (Var\{x\} + E\{x\}^2) + \mu(1 - W_1^2) E\{x\} - W_1 \mu^2\right).
\end{aligned}$$

For the prior mean μ , the expected parameter update is given by:

$$\begin{aligned}
E\{\Delta \mu\} &\propto E\{\epsilon_2^*\} = E\{z_2^* - \mu\} = E\left\{\frac{\mu + W_1 x}{1 + W_1^2} - \mu\right\} \\
&= \frac{W_1}{1 + W_1^2} E\{x\} - \frac{W_1^2}{1 + W_1^2} \mu.
\end{aligned}$$

Nullclines. The nullclines for $E\{\Delta W_1\} = 0$ are given, therefore, by the expression:

$$\mu = \frac{-(W_1^2 - 1)E\{x\} \pm \sqrt{(W_1^2 - 1)^2 E\{x\}^2 + 4W_1^2 (Var\{x\} + E\{x\}^2)}}{2W_1}.$$

Similarly, the nullcline for $E\{\Delta \mu\} = 0$ is given by the expression:

$$\mu = \frac{E\{x\}}{W_1}.$$

Equilibrium points. We can attempt to find equilibrium points by finding the intersection between the nullclines for $E\{\Delta W_1\} = 0$ and the nullcline for $E\{\Delta \mu\} = 0$ as follows:

$$\begin{aligned}
\frac{E\{x\}}{W_1} &= \frac{-(1 - W_1^2)E\{x\} \pm \sqrt{(1 - W_1^2)^2 E\{x\}^2 + 4W_1^2 (Var\{x\} + E\{x\}^2)}}{-2W_1} \\
E\{x\}(-1 - W_1^2) &= \pm \sqrt{E\{x\}^2(1 - W_1^2)^2 + 4W_1^2 (Var\{x\} + E\{x\}^2)} \\
E\{x\}^2(1 + W_1^4 + 2W_1^2) &= E\{x\}^2(1 + W_1^4 - 2W_1^2) + 4W_1^2 (Var\{x\} + E\{x\}^2) \\
4W_1^2 E\{x\}^2 &= 4W_1^2 (Var\{x\} + E\{x\}^2) \\
Var\{x\} &= 0
\end{aligned}$$

There are, therefore, no equilibrium points except when $Var\{x\} = 0$ where all the points on $\mu = \frac{E\{x\}}{W_1}$ are fixed points. Consequently, the parameter updates of PC do not converge.

A.4 Predictive coding optimizes an infinitely loose bound on the marginal log-likelihood $\ln p(x; \theta)$

Our results suggest that predictive coding fails to learn accurate generative models. This result could be explained by a theoretical shortcoming of predictive coding which is also present in a range of other theories of learning in the brain, as discussed below.

Predictive coding learns a generative model by changing its model parameters to maximize the marginal likelihood:

$$p(x; \theta) = \int p(x, z; \theta) dz$$

Predictive coding uses the variational expectation-maximization algorithm to optimize this marginal distribution which is intractable [149, 258, 99, 259]. This algorithm employs the variational distribution $q(z)$, parametrized by $\{x_2, \dots, x_L\}$, to estimate the posterior $p(z|x; \theta)$. It then establishes the free energy E_{gen} as an upper bound on the negative log-likelihood:

$$\begin{aligned} E_{gen} &= -\mathbb{E}_q\{\ln p(x, z; \theta)\} + \mathbb{E}_q\{\ln q(z)\} \\ &= -\ln p(x; \theta) + KL(q(z)||p(z|x; \theta)) \geq -\ln p(x; \theta), \end{aligned} \tag{A.1}$$

with $KL(\cdot||\cdot)$ the KL divergence which is always positive.

The variational expectation-maximization algorithm learns by iterating over two steps. First, it minimizes the free energy E_{gen} with respect to $\{x_2, \dots, x_L\}$ for the current parameters θ . In other words, it finds an approximation $q(z)$ for the posterior. Second, the algorithm minimizes the free energy with respect to θ for the parameters $\{x_2, \dots, x_L\}$ inferred in the first step.

Predictive coding uses the Dirac delta distribution $q_l(z) = \delta(z_l - x_l)$ as variational distribution [99], enabling an implementation of predictive coding using local computation and plasticity. Importantly, the entropy $-\mathbb{E}_q\{\ln \delta(\cdot)\}$ for this variational distribution is equal to minus infinity, causing a divergence in the weights of the model [150].

Additionally, the free energy E_{gen} minimized by predictive coding is therefore also infinite, making it an infinitely loose bound on the marginal likelihood. This infinitely

loose bound raises doubts about the significance of the objective minimized by predictive coding. This theoretical shortcoming may explain the observed poor learning performance of predictive coding in this study despite the application of weight normalization to prevent diverging weights. Moreover, a range of theories for learning in the brain [24, 103, 151] are based on a similar energy function as PC’s variational free energy, importing the problem of implicitly ignoring an infinite entropy. Future work could therefore explore whether these other learning algorithms exhibit similar suboptimal learning capabilities to those observed in predictive coding.

A.5 Proofs for MCPC capturing the variability of cortical activity

A.5.1 Neural variability decreases at stimulus onset

At stimulus onset, the variance of MCPC’s steady-state activity decreases. This is because MCPC’s neural activity of the latent state switches from representing the marginal distribution $p(z; \theta)$ to the posterior $p(z|x; \theta)$, which in expectation has a lower variance. The steady state of MCPC’s latent state before stimulus onset can be derived similarly to proposition 2 of the main text and equals the marginal distribution $p(z; \theta)$:

$$\begin{aligned} p^{ss}(z) &= \int \frac{e^{-E_{gen}}}{Z} dx = \int \frac{e^{\ln p(x,z;\theta)}}{Z} dx = p(z; \theta) \int \frac{p(x|z; \theta)}{Z} dx \\ &= p(z; \theta) \end{aligned}$$

Moreover, the law of total variance presented in equation (A.2) shows that the posterior $p(z|x; \theta)$ has in expectation a lower variance than the marginal distribution $p(z; \theta)$. Therefore, the neural variability of MCPC’s latent state is expected to decrease at stimulus onset as has been observed in the brain.

$$Var[p(z; \theta)] = E\{Var[p(z|x; \theta)]\} + Var[E\{p(z|x; \theta)\}] \quad (\text{A.2})$$

A.5.2 Natural stimuli specific increase in similarity between spontaneous and evoked neural activity

As the training progresses, the similarity between MCPC’s spontaneous activity (encoding the marginal distribution $p(z; \theta)$) and its average evoked activity increases (encoding the posterior distribution $p(z|x; \theta)$). This increase in similarity, however, is specific to the natural stimuli used during training. It does not extend to data outside the training set’s distribution, indicating a specialized adaptation to the learned stimuli. This phenomenon can be explained by Bayesian statistics. The marginal distribution $p(x; \theta)$ of a latent variable model with latent state x and input y serves as the model’s prior distribution over the latent state, encapsulating prior expectations. Initially, the model parameters are randomly initialized, leading to a discrepancy between the model’s average posterior for its inputs and its prior distributions. However, as training advances, the model refines its generative capabilities, aligning the marginal likelihood $p(x; \theta)$ closer to the data distribution $p(x)$. Consequently, the model’s prior over the latent state begins to mirror the average posterior distribution more accurately. If the model achieves perfect learning, its prior will match exactly with the expected posterior for the training data. This result follows from Bayesian statistics that specifies that the probability distribution $p(z)$ equals $\int p(z|x)p(x)dy$ (see equation A.3). However, the prior of the model is specific to the data distribution. Consequently, when considering posterior distributions for data samples outside the training data distribution, the prior will not equate to the average posterior.

$$p(z) = \int p(z, x)dx = \int p(z|x)p(x)dx \quad (\text{A.3})$$

A.6 Hyperparameter search results

Tables A.1, A.2 and A.3 summarise the optimal model parameters for MCPC, PC, and DLGMs found during the hyperparameter search to maximise the FID, the marginal likelihood $-\ln p(x_{eval})$, and the reconstruction MSE.

Table A.1: **Results of the hyperparameter search to maximize the FID for MCPC, PC, and DLGMs.**

	MCPC	PC	DLGM
Architecture			
input dimension	748	748	748
number of hidden layers	3	3	3
hidden layers dimension	(128, 128, 20)	(128, 128, 20)	(256,256,20)
activation function	ReLU	ReLU	ReLU
PC Inference			
optimizer	Adam	Adam	-
lr	0.7	0.1	-
max_steps	250	250	-
MCPC Inference			
optimizer	SGD	-	-
lr	0.1	-	-
mixing steps	50	-	-
sampling steps	100	-	-
Learning			
optimiser	Adam	Adam	Adam
lr	0.01	0.01	0.001
decay	0.0	0.1	0.01
num_epochs	50	50	50
batch_size	256	128	128

Table A.2: **Results of the hyperparameter search to maximize the marginal likelihood for MCPC, PC, and DLGMs.**

	MCPC	PC	DLGM
Architecture			
input dimension	748	748	748
number of hidden layers	3	3	3
hidden layers dimension	(128, 128, 20)	(128, 128, 25)	(128, 128, 10)
activation function	ReLU	Tanh	ReLU
PC Inference			
optimizer	Adam	Adam	-
lr	0.1	0.3	-
max_steps	250	250	-
MCPC Inference			
optimizer	SGD	-	-
lr	0.03	-	-
mixing steps	50	-	-
sampling steps	100	-	-
Learning			
optimiser	Adam	Adam	Adam
lr	0.003	0.001	0.001
decay	0.1	0.01	0.01
num_epochs	50	50	50
batch_size	64	256	256

Table A.3: **Results of the hyperparameter search to maximize the reconstruction MSE for MCPC, PC, and DLGMs.**

	MCPC	PC	DLGM
Architecture			
input dimension	748	748	748
number of hidden layers	3	3	3
hidden layers dimension	(256, 256, 20)	(256, 256, 30)	(256, 256, 20)
activation function	ReLU	Tanh	ReLU
PC Inference			
optimizer	Adam	Adam	-
lr	0.7	0.7	-
max_steps	250	250	-
MCPC Inference			
optimizer	SGD	-	-
lr	0.03	-	-
mixing steps	50	-	-
sampling steps	100	-	-
Learning			
optimiser	Adam	Adam	Adam
lr	0.001	0.003	0.003
decay	0.01	0.1	0.1
num_epochs	50	50	50
batch_size	64	256	128

Appendix B

Bidirectional predictive coding

B.1 Other types of predictive coding models

In this experiment, we train three additional types of predictive coding models on the supervised task described in Section 3.4.1: (1) PC along arbitrary graphs[196], (2) bPC with shared weights for discriminative and generative predictions[260], and (3) discPC with activity decay during generation[129].

PC along arbitrary graphs (agPC) differs from bPC, in that each neuron has a single energy function, with predictions computed from all incoming connections. In contrast, bPC uses separate energy terms for bottom-up (discriminative) and top-down (generative) predictions. In this formulation, the energy associated with an agPC layer x_l is given by:

$$E_l = \frac{1}{2} \|x_l - f(W_l x_{l+1} + V_{l-1} x_{l-1} + b)\|_2^2,$$

where W_l and V_{l-1} are top-down and bottom-up weights, respectively. We use the same training algorithm, hyperparameter tuning, and evaluation procedure as bPC, modifying only the energy function. Initialisation is done using bottom-up sweeps (for training and classification evaluation) and top-down forward sweeps (for generation evaluation), matching bPC. We also tested zero and Xavier initialisation. However, this resulted in worse learning performance.

bPC with shared weights (shared bPC) uses a single set of weights for both discriminative and generative predictions. Its energy function associated with a layer

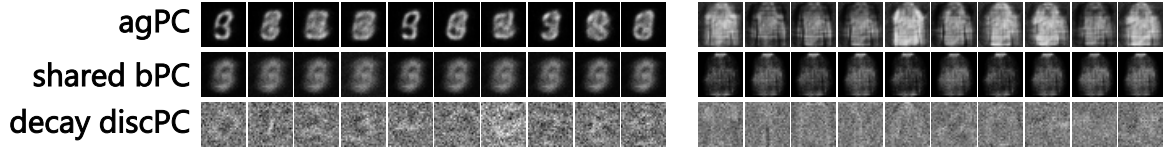


Figure B.1: MNIST (left) and Fashion-MNIST (right) class average image generation for agPC, shared bPC and decay discPC.

x_l is given by:

$$E_l = \frac{\alpha_{\text{gen}}}{2} \|x_l - f(W_l x_{l+1})\|_2^2 + \frac{\alpha_{\text{disc}}}{2} \|x_l - f(W_{l-1}^\top x_{l-1})\|_2^2,$$

where $W_l^\top$ is reused for both directions. This model omits bias terms and relies on non-local computations. We follow the same training, tuning, and evaluation protocol as bPC.

discPC with activity decay (decay discPC) extends standard discPC by adding an activity decay term during generation. We use the same training and hyperparameter tuning protocol as discPC. Additionally, we tune an activity decay rate from a log-uniform distribution over the range $[10^{-5}, 1]$. During generative evaluation, we increase the number of inference steps to 10,000 due to slower convergence and we initialise neurons to zero activity before generation for consistency with Sun and Orchard [129].

Table B.1 reports classification accuracy and reconstruction RMSE for these models. Figure B.1 illustrates the images generated by each model for each MNIST and Fashion-MNIST class. None of the models match bPC in both classification and generative performance.

Table B.1: Classification accuracy and class average image generation for agPC, shared bPC and decay discPC on the supervised training task of Section 3.4.1. Higher accuracy and lower RMSE are better. We report the mean +/- sem over five seeds.

Model	Acc %		RMSE	
	MNIST	Fashion-MNIST	MNIST	Fashion-MNIST
arbitrary graph PC	71.47 $\pm$ 0.27	62.50 $\pm$ 6.05	0.1685 $\pm$ 0.0299	0.2426 $\pm$ 0.0077
shared-bPC	97.39 $\pm$ 0.08	87.99 $\pm$ 0.04	0.1380 $\pm$ 0.0042	0.2560 $\pm$ 0.0093
decay discPC	-	-	0.4047 $\pm$ 0.0005	0.3433 $\pm$ 0.0002

B.2 Balancing discriminative and generative energies in bPC

In this experiment, we investigate how the relative weighting of the discriminative and generative energy terms ($\alpha_{disc}/\alpha_{gen}$) affects bPC’s learning in the supervised and unsupervised settings described in Sections 3.4.1 and 3.4.2. For each setting, we perform hyperparameter tuning of the bPC model while constraining the ratio $\alpha_{disc}/\alpha_{gen}$, and report the test performance of the model with the optimal parameters. We use a grid search over the following hyperparameters: the activity learning rate [0.01, 0.003, 0.001], the parameter learning rate [0.001, 0.0003, 0.0001], and the weight decay [0., 0.0001, 0.0003, 0.001]. Other settings, including a GeLU activation function and an activity momentum of zero, are constant. All remaining parameters, such as the number of epochs, batch size, and number of inference steps during training and evaluation, are kept consistent with the experiments described in Sections 3.4.1 and 3.4.2.

In the supervised case, generation RMSE remains stable across ratios of $\alpha_{disc}/\alpha_{gen}$, but classification accuracy declines as the discriminative energy is down-scaled. We suspect this is due to a mismatch in energy magnitudes. A typical discPC model has an energy of around 0.1 for test data samples after training, while genPC has an energy of approximately 50. Thus, the generative energy dominates when α_{disc} is not significantly larger than α_{gen} , leading to a poor classification like genPC models. In contrast, bPC’s unsupervised learning performance is relatively robust across a wide range of ratios, but degrades sharply when the discriminative energy is too large. We also observed increased training instability when $\alpha_{disc} \gg \alpha_{gen}$.

These results highlight the importance of appropriately scaling the energy terms based on the task. Hybrid tasks such as those in Section 3.4.3 require tailored weighting across the models to ensure effective learning. Future work could make the energy scaling a learnable parameter. This change could make bPC learn optimal scalings autonomously.

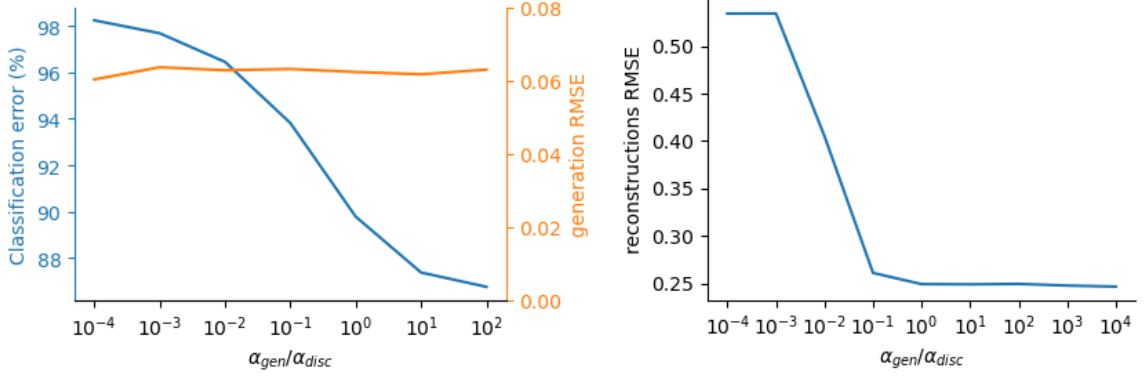


Figure B.2: bPC’s supervised (left) and unsupervised (right) learning performance on MNIST depending on the relative weighting of its discriminative and generative energy term.

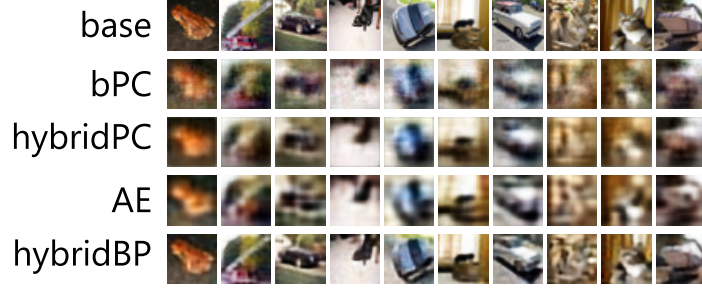


Figure B.3: CIFAR-10 image reconstruction from latent representation and labels for models of Section 3.4.3 without max-pooling.

B.3 Effect of max-pooling on generative artifacts

In this experiment, we modify the VGG-5 architecture of the models from Section 3.4.3 by removing max-pooling layers and increasing the convolutional strides from one to two. We also increase the inference time during training to 48 inference steps. All other aspects of the experiment remain unchanged.

As shown in Figure B.3, this change eliminates the checkerboard artefacts in the reconstructed images and improves reconstruction quality for both bPC and hybridPC. This improvement is reflected in the lower RMSE values reported in Table B.2. However, removing max-pooling reduces classification accuracy across all models, with bPC experiencing a drop of approximately 25%. Despite this, the overall trends remain consistent: bPC shows a substantially better discriminative performance than hybridPC for comparable reconstruction RMSE.

Table B.2: Classification accuracy and image reconstruction RMSE from latent representations for CIFAR-10 models considered in Section 3.4.3 without max-pooling. Higher accuracy and lower RMSE are better. We report the mean $\pm$ sem over five seeds.

Model	Acc %	RMSE
bPC	61.67 $\pm$ 0.29	0.2178 $\pm$ 0.0047
hybridPC	37.36 $\pm$ 0.18	0.2196 $\pm$ 0.0010
AE	84.58 $\pm$ 0.11	0.1777 $\pm$ 0.0005
hybridBP	85.79 $\pm$ 0.26	0.1095 $\pm$ 0.0018

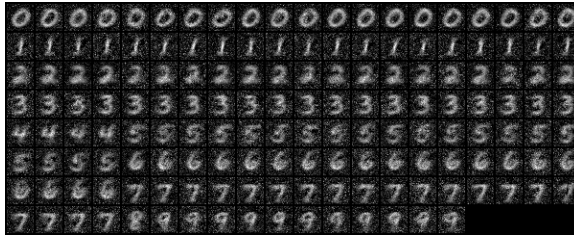


Figure B.4: Samples generated for the combined discBP + genBP model.

B.4 Sample generation of combined discBP + genBP models

In this experiment, we repeat the combined model image generation procedure from Section 3.4.4 using discBP and genBP. The two models share an input layer, which is iteratively updated to minimise their energy functions until the energy falls below the 10th percentile of each model’s energy distribution on the test set. We generate samples using the same procedure as described previously for the discPC + genPC model. We compute the FID and Inception scores to quantify the generation quality.

Figure B.4 displays the generated samples for different classes. While the images resemble the class averages, they also exhibit noise across pixels and less visual diversity than bPC samples. As shown in Table B.3, the backpropagation-based combined model yields higher FID and lower Inception scores, indicating that bPC generates samples more consistent with the MNIST distribution than the backpropagation-based approach.

Table B.3: Fréchet inception distance and inception score for samples generated for the combined discBP + genBP model compared to bPC and discPC + genPC. Results are given for mean +/- sem for three seeds.

Model	Inception	FID
bPC	$6.05^{\pm 0.17}$	$44.4^{\pm 2.2}$
discPC + genPC	$3.62^{\pm 0.03}$	$140.5^{\pm 2.1}$
discBP + genBP	$5.00^{\pm 0.02}$	$99.1^{\pm 3.5}$

B.5 Relationship between hybridPC and bPC

In this work, we benchmark our bPC model primarily against hybrid predictive coding [121]. hybridPC is the only plausible PC model that incorporates both slow, iterative inference and fast, feedforward inference, to date. In hybridPC, fast inference is enabled by a bottom-up network added to the genPC network architecture, which serves solely to initialise neural activities and does not impact their dynamics.

HybridPC was originally defined with two separate objective functions. One of these is the genPC energy function. The other is a loss for the bottom-up initialisation parameters. However, the learning process of hybridPC can also be expressed as a single unified loss function. This formulation is analogous to the energy function in bPC. The neural dynamics and weight updates in hybridPC minimise the following energy function:

$$E_{hybrid}(x, W, V) = \sum_{l=1}^{L-1} \frac{1}{2} \|x_l - W_l f(x_{l+1})\|_2^2 + \sum_{l=2}^L \frac{1}{2} \|\text{sg}(x_l) - V_{l-1} f(\text{sg}(x_{l-1}))\|_2^2 \quad (\text{B.1})$$

Unlike in bPC, a stop gradient $\text{sg}(\cdot)$ operation is applied. This change ensures that only the top-down generative predictions drive the neural dynamics. HybridPC can perform both supervised and unsupervised learning. However, the supervised learning performance of hybridPC is much poorer than that of discPC. This is due to the stop gradient operation in the bottom-up stream of this model, that prevents it from learning a discriminative model that maps well from x_1 to x_L [121]. This caps its supervised learning performance to that of genPC [121].

In bPC, bottom-up predictions contribute directly to the inference process, ensuring that the learned neural activity patterns incorporate discriminative signals from the

bottom-up pathway, thereby significantly enhancing discriminative performance while retaining strong generative capabilities.

With this reformulation of hybridPC, and given the similarity in objective functions, the bottom-up weight updates in bPC can be interpreted as learning an inversion of the activity updates. This inversion provides a mechanism for fast initialisation in bPC, implementing amortised inference.

Appendix C

Target alignment with predictive coding

C.1 Experimental validation

We experimentally validate the expressions for $\frac{\partial \hat{y}}{\partial t}$ for BP and PC. We generate random DLNs with inputs and outputs drawn from a standard normal distribution. We vary the model architecture by changing the network depth and width, and repeat measurements across multiple random seeds. For each setting, we compute the change in the model’s prediction resulting from a single parameter update using either BP or PC. We compare the empirical change in prediction to the corresponding analytical expression. We discretise the analytical expressions and match the learning rate of the experiment to the Euler step of the discretised equations. We quantify the difference in empirical and analytical $\frac{\partial \hat{y}}{\partial t}$ using the 2-norm of the difference, normalised by the 2-norm of the empirical change in the prediction. This yields a relative error metric, with values closer to zero indicating better agreement between theory and experiment.

Figure C.1 shows that the difference between the analytical prediction and the observed change in prediction vanishes as the learning rate approaches zero. This is as expected, since the analytical expression is derived for the continuous-time derivative, which corresponds to the zero step-size limit. Typical learning rates used for model training are around 10^{-3} , for which the difference is minimal (10^{-3}).

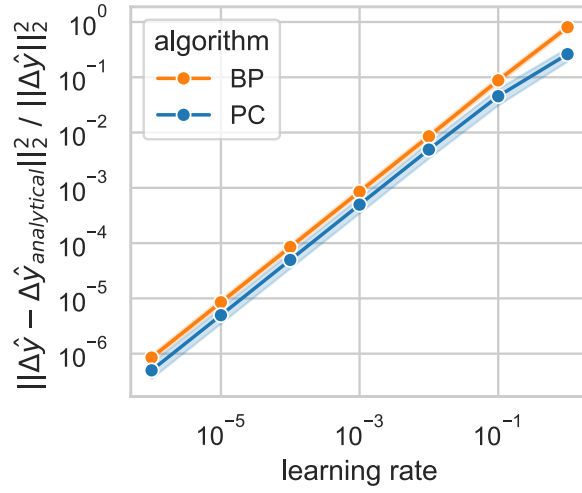


Figure C.1: **The analytical expressions of $\Delta\hat{\mathbf{y}}$ for BP and PC in DLNs predict experimental $\Delta\hat{\mathbf{y}}$.** Normalised difference between the derived analytical expression of the change in prediction of a model trained for one parameter update $\Delta\hat{\mathbf{y}}_{\text{analytical}}$ and the experimental change in prediction for varying learning rates. The results report the mean and standard deviation of the normalised difference for randomly initialised models with random inputs and outputs. Models are evaluated using 256-dimensional inputs and outputs. Batch sizes of 64, 128, and 256 are considered. Network architectures use hidden layer widths of 128, 256, or 512, with depths ranging from two to six hidden layers. All results are averaged over three random seeds.

Bibliography

- [1] G. Oliviers, R. Bogacz, and A. Meulemans. Learning probability distributions of sensory inputs with monte carlo predictive coding. *PLOS Computational Biology*, 20(10):1–34, 10 2024. doi: 10.1371/journal.pcbi.1012532.
- [2] G. Oliviers, M. Tang, and R. Bogacz. Bidirectional predictive coding, 2025. URL <https://arxiv.org/abs/2505.23415>.
- [3] L. Pinchetti, C. Qi, O. Lokshyn, C. Emde, A. M’Charrak, M. Tang, S. Frieder, B. Menzat, G. Oliviers, R. Bogacz, T. Lukasiewicz, and T. Salvatori. Benchmarking predictive coding networks – made simple. In *International Conference on Learning Representations*, 2025.
- [4] C. Goemaere, G. Oliviers, R. Bogacz, and T. Demeester. Error optimization: Overcoming exponential signal decay in deep predictive coding networks, 2025. arXiv preprint arXiv:2505.20137.
- [5] M. G. Craske, M. Treanor, C. C. Conway, T. Zbozinek, and B. Vervliet. Maximizing exposure therapy: An inhibitory learning approach. *Behaviour Research and Therapy*, 58:10–23, 2014. doi: 10.1016/j.brat.2014.04.006.
- [6] C. Pittenger and R. S. Duman. Stress, depression, and neuroplasticity: A convergence of mechanisms. *Neuropsychopharmacology*, 33(1):88–109, 2008. doi: 10.1038/sj.npp.1301574.
- [7] R. S. Duman and G. K. Aghajanian. Synaptic dysfunction in depression: potential therapeutic targets. *Science*, 338(6103):68–72, 2012. doi: 10.1126/science.1222939.
- [8] M. J. Frank, L. C. Seeberger, and R. C. O’Reilly. By carrot or by stick: Cognitive reinforcement learning in parkinsonism. *Science*, 306(5703):1940–1943, 2004. doi: 10.1126/science.1102941.

- [9] E. T. Koch, S. P. Dukelow, and T. Cluff. Motor learning after stroke: what we’ve learned and what lies ahead. *Brain*, 149(2):386–403, 10 2025. ISSN 0006-8950. doi: 10.1093/brain/awaf388.
- [10] P. Howard-Jones. Neuroscience and education: myths and messages. *Nature Reviews Neuroscience*, 15:817–824, 2014. doi: 10.1038/nrn3817.
- [11] A. M. Graybiel. Habits, rituals, and the evaluative brain. *Annual Review of Neuroscience*, 31:359–387, 2008. doi: 10.1146/annurev.neuro.29.051605.112851.
- [12] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017. doi: 10.1016/j.neuron.2017.06.011.
- [13] A. Cleeremans. The radical plasticity thesis: How the brain learns to be conscious. *Frontiers in Psychology*, 2, 2011. ISSN 1664-1078. doi: 10.3389/fpsyg.2011.00086.
- [14] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015. doi: 10.1038/nature14539.
- [15] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *CoRR*, abs/1802.07569, 2018.
- [16] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [17] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, 2015. URL <https://arxiv.org/abs/1412.1897>.
- [18] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1: 206–215, 2019. doi: 10.1038/s42256-019-0048-x.
- [19] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning, 2017. URL <https://arxiv.org/abs/1702.08608>.

- [20] A. M. Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10:3770, 2019. doi: 10.1038/s41467-019-11786-6.
- [21] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
- [22] A. Sandberg. Energetics of the brain and ai, 2016. URL <https://arxiv.org/abs/1602.04019>.
- [23] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton. Back-propagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020. ISSN 1471-0048. doi: 10.1038/s41583-020-0277-3.
- [24] J. Sacramento, R. Ponte Costa, Y. Bengio, and W. Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in Neural Information Processing Systems*, 31, 2018.
- [25] J. C. R. Whittington and R. Bogacz. Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23(3):235–250, 2019. ISSN 1364-6613. doi: 10.1016/j.tics.2018.12.005.
- [26] J. Wu, X. Lin, Y. Guo, J. Liu, L. Fang, S. Jiao, and Q. Dai. Analog optical computing for artificial intelligence. *Engineering*, 10:133–145, 2022. ISSN 2095-8099. doi: 10.1016/j.eng.2021.06.021.
- [27] G. Indiveri and S.-C. Liu. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE*, 103(8):1379–1397, 2015. doi: 10.1109/JPROC.2015.2444094.
- [28] S. Herculano-Houzel. The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, 3, 2009. ISSN 1662-5161. doi: 10.3389/neuro.09.031.2009.
- [29] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1–47, 1991.
- [30] D. S. Bassett and O. Sporns. Network neuroscience. *Nature Neuroscience*, 20: 353–364, 2017. doi: 10.1038/nn.4502.

- [31] G. Stuart and N. Spruston. Dendritic integration: 60 years of progress. *Nature Neuroscience*, 18:1713–1721, 2015. doi: 10.1038/nn.4157.
- [32] M. London and M. Häusser. Dendritic computation. *Annual Review of Neuroscience*, 28:503–532, 2005. doi: 10.1146/annurev.neuro.28.061604.135703.
- [33] B. P. Bean. The action potential in mammalian central neurons. *Nature Reviews Neuroscience*, 8:451–465, 2007. doi: 10.1038/nrn2148.
- [34] M. S. Nielsen, L. N. Axelsen, P. L. Sorgen, V. Verma, M. Delmar, and N.-H. Holstein-Rathlou. Gap junctions. *Comprehensive Physiology*, 2(3):1981–2035, 2012. doi: 10.1002/j.2040-4603.2012.tb00453.x.
- [35] E. R. Kandel. The molecular biology of memory storage: A dialogue between genes and synapses. *Science*, 294(5544):1030–1038, 2001. doi: 10.1126/science.1067020.
- [36] R. C. Malenka and M. F. Bear. Ltp and ltd: An embarrassment of riches. *Neuron*, 44(1):5–21, 2004. doi: 10.1016/j.neuron.2004.09.012.
- [37] N. Caporale and Y. Dan. Spike timing-dependent plasticity: A hebbian learning rule. *Annual Review of Neuroscience*, 31:25–46, 2008. doi: 10.1146/annurev.neuro.31.060407.125639.
- [38] G.-q. Bi and M.-m. Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18(24):10464–10472, 1998. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.18-24-10464.1998.
- [39] NIAID Visual & Medical Arts. Healthy neuron. <https://bioart.niaid.nih.gov/bioart/197>, October 2024. NIAID NIH BioArt Source. Accessed 5 March 2026.
- [40] K. D. Harris and G. M. G. Shepherd. The neocortical circuit: Themes and variations. *Nature Neuroscience*, 18:170–181, 2015. doi: 10.1038/nn.3917.
- [41] J. J. DiCarlo, D. Zoccolan, and N. C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012. doi: 10.1016/j.neuron.2012.01.010.

- [42] J. D. Semedo, A. I. Jasper, A. Zandvakili, A. Krishna, A. Aschner, C. K. Machens, A. Kohn, and B. M. Yu. Feedforward and feedback interactions between visual cortical areas use different population activity patterns. *Nature Communications*, 13(1):1099, 2022.
- [43] P. K. Kuhl. Early language acquisition: Cracking the speech code. *Nature Reviews Neuroscience*, 5(11):831–843, 2004. doi: 10.1038/nrn1533.
- [44] E. Dayan and L. G. Cohen. Neuroplasticity subserving motor skill learning. *Neuron*, 72(3):443–454, 2011. doi: 10.1016/j.neuron.2011.10.008.
- [45] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011. doi: 10.1126/science.1192788.
- [46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [47] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [48] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.
- [49] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL <https://arxiv.org/abs/1712.01815>.
- [50] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, Cambridge, MA, 2016.
- [51] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, 1949.
- [52] E. L. Bienenstock, L. N. Cooper, and P. W. Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience*, 2(1):32–48, 1982. doi: 10.1523/JNEUROSCI.02-01-00032.1982.

- [53] E. Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982. doi: 10.1007/BF00275687.
- [54] P. Mazzoni, R. A. Andersen, and M. I. Jordan. A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10):4433–4437, 1991.
- [55] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.
- [56] H. S. Seung. Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, 40(6):1063–1073, 2003.
- [57] K. P. Unnikrishnan and K. P. Venugopal. Alopex: A correlation-based learning algorithm for feedforward and recurrent neural networks. *Neural Computation*, 6(3):469–490, 1994.
- [58] J. Werfel, X. Xie, and H. S. Seung. Learning curves for stochastic gradient descent in linear feedforward networks. In *Advances in Neural Information Processing Systems*, pages 1193–1200, 2003.
- [59] P. R. Roelfsema and A. Holtmaat. Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19:166–180, 2018. doi: 10.1038/nrn.2018.6.
- [60] R. C. O’Reilly. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, 8(5): 895–938, 1996.
- [61] P. Baldi and F. Pineda. Contrastive learning and neural oscillations. *Neural Computation*, 3(4):526–545, 1991.
- [62] Y. Bengio, T. Mesnard, A. Fischer, S. Zhang, and Y. Wu. Stdp-compatible approximation of backpropagation in an energy-based model. *Neural Computation*, 29(3):555–577, 2017.
- [63] B. Scellier and Y. Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017.

- [64] A. Laborieux, M. Ernoult, B. Scellier, Y. Bengio, J. Grollier, and D. Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in Neuroscience*, 15:633674, 2021.
- [65] M. Ernoult, J. Grollier, D. Querlioz, Y. Bengio, and B. Scellier. Equilibrium propagation with continual weight updates. *arXiv preprint arXiv:2005.04168*, 2020.
- [66] M. J. Falk, A. T. Strupp, B. Scellier, and A. Murugan. Temporal contrastive learning through implicit non-equilibrium memory. *Nature Communications*, 16(1):2163, 2025.
- [67] B. A. Richards and T. P. Lillicrap. Dendritic solutions to the credit assignment problem. *Current Opinion in Neurobiology*, 54:28–36, 2019.
- [68] K. P. Körding and P. König. Learning with two sites of synaptic integration. *Network: Computation in Neural Systems*, 11(1):25–39, 2000.
- [69] W. Greedy, H. W. Zhu, J. Pemberton, J. Mellor, and R. Ponte Costa. Single-phase deep learning in cortico-cortical networks. In *Advances in Neural Information Processing Systems*, volume 35, pages 24213–24225, 2022.
- [70] R. P. N. Rao and D. H. Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.
- [71] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.
- [72] M. I. Garrido, J. M. Kilner, K. E. Stephan, and K. J. Friston. The mismatch negativity: A review of underlying mechanisms. *Clinical Neurophysiology*, 120(3):453–463, 2009.
- [73] J. M. Kilner, K. J. Friston, and C. D. Frith. Predictive coding: An account of the mirror neuron system. *Cognitive Processing*, 8(3):159–166, 2007.
- [74] D. Wolpert, K. Doya, and M. Kawato. A unifying computational framework for motor control and social interaction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1431):593–602, 2003.

- [75] A. Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204, Jun 2013. doi: 10.1017/S0140525X12000477.
- [76] K. Friston. Computational psychiatry: from synapses to sentience. *Molecular Psychiatry*, Sep 2022. doi: 10.1038/s41380-022-01743-z.
- [77] J. Trommershäuser, L. Maloney, and M. Landy. Decision making, movement planning, and statistical decision theory. *Trends in Cognitive Sciences*, 12:291–297, 2008.
- [78] H. von Helmholtz. Concerning the perceptions in general, 1867. In W. Dennis, editor, *Readings in the History of Psychology*, pages 214–230. Appleton-Century-Crofts, 1948. doi: 10.1037/11304-027.
- [79] P. Dayan, G. Hinton, R. Neal, and R. Zemel. The helmholtz machine. *Neural Computation*, 7(5):889–904, 1995.
- [80] T. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, 20(7):1434–1448, 2003.
- [81] D. Knill and A. Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 27(12):712–719, 2004. doi: 10.1016/j.tins.2004.10.007.
- [82] M. O. Ernst and M. S. Banks. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429–433, 2002.
- [83] D. Wolpert, Z. Ghahramani, and M. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880, 1995.
- [84] R. Sundaeswara and P. Schrater. Perceptual multistability predicted by search model for bayesian decisions. *Journal of Vision*, 8:12.1–12.19, 2008.
- [85] S. Gershman, E. Vul, and J. Tenenbaum. Multistability and perceptual inference. *Neural Computation*, 24:1–24, 2012.
- [86] D. Knill and W. Richards. *Perception as Bayesian Inference*. Cambridge University Press, New York, 1996.

- [87] R. van Beers, A. Sittig, and J. Gon. Integration of proprioceptive and visual position-information: An experimentally supported model. *Journal of Neurophysiology*, 81(3):1355–1364, 1999.
- [88] P. Hoyer and A. Hyvärinen. Interpreting neural response variability as monte carlo sampling of the posterior. In *Advances in Neural Information Processing Systems*, volume 15, 2002.
- [89] J. Fiser, P. Berkes, G. Orban, and M. Lengyel. Statistically optimal perception and learning: from behavior to neural representations. *Trends in Cognitive Sciences*, 14(3):119–130, 2010.
- [90] R. Moreno-Bote, D. Knill, and A. Pouget. Bayesian sampling in visual perception. *Proceedings of the National Academy of Sciences*, 108:12491–12496, 2011.
- [91] A. Graf, A. Kohn, M. Jazayeri, and J. A. Movshon. Decoding the activity of neuronal populations in macaque primary visual cortex. *Nature Neuroscience*, 14(2):239–245, 2011.
- [92] P. Berens, A. S. Ecker, R. J. Cotton, W. Ma, M. Bethge, and A. S. Tolias. A fast and simple population code for orientation in primate v1. *Journal of Neuroscience*, 32(31):10618–10626, 2012. doi: 10.1523/JNEUROSCI.1335-12.2012.
- [93] NIAID Visual & Medical Arts. Male head side view. <https://bioart.niaid.nih.gov/bioart/329>, October 2024. NIAID NIH BioArt Source. Accessed 5 March 2026.
- [94] C. M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.
- [95] K. Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360:815–836, 2005.
- [96] K. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010. doi: 10.1038/nrn2787.
- [97] F. Pérez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE International Symposium on Information Theory*, pages 1666–1670. IEEE, 2008. doi: 10.1109/ISIT.2008.4595271.

- [98] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–38, 1977.
- [99] R. Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *Journal of Mathematical Psychology*, 76:198–211, 2017.
- [100] F. Innocenti, E. M. Achour, R. Singh, and C. L. Buckley. Only strict saddles in the energy landscape of predictive coding networks? In *Advances in Neural Information Processing Systems*, volume 37, pages 53649–53683, 2024.
- [101] K. Friston and S. Kiebel. Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1211–1221, 2009.
- [102] M. Tang, T. Salvatori, B. Millidge, Y. Song, T. Lukasiewicz, and R. Bogacz. Recurrent predictive coding models for associative memory employing covariance learning. *PLOS Computational Biology*, 19(4):e1010719, 2023.
- [103] J. Whittington and R. Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Computation*, 29(5):1229–1262, 2017.
- [104] B. Millidge, A. Tschantz, and C. Buckley. Predictive coding approximates backprop along arbitrary computation graphs, 2020. arXiv preprint arXiv:2006.04182.
- [105] F. Innocenti, E. Achour, and C. Buckley. mupc: Scaling predictive coding to 100+ layer networks. *arXiv preprint arXiv:2505.13124*, 2025.
- [106] M. W. Spratling, K. De Meyer, and R. Kompass. Unsupervised learning of overlapping image components using divisive input modulation. *Computational Intelligence and Neuroscience*, 2009:1–19, 2009. doi: 10.1155/2009/381457.
- [107] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, 2016. ISSN 2041-1723. doi: 10.1038/ncomms13276.
- [108] B. Millidge, A. Tschantz, A. Seth, and C. L. Buckley. Relaxing the constraints on predictive coding models, 2020. URL <https://arxiv.org/abs/2010.01047>.

- [109] G. Keller and T. Mrsic-Flogel. Predictive processing: a canonical cortical computation. *Neuron*, 100(2):424–435, 2018.
- [110] S.-p. Ku, E. L. Hargreaves, S. Wirth, and W. A. Suzuki. The contributions of entorhinal cortex and hippocampus to error driven learning. *Communications Biology*, 4(1):618, 2021.
- [111] F. Mikulasch, L. Rudelt, M. Wibral, and V. Priesemann. Where is the error? hierarchical predictive coding through dendritic error computation. *Trends in Neurosciences*, 46(1):45–59, 2023.
- [112] T. Salvatori, Y. Song, Y. Hong, L. Sha, S. Frieder, Z. Xu, R. Bogacz, and T. Lukasiewicz. Associative memories via predictive coding. In *Advances in Neural Information Processing Systems*, volume 34, pages 3874–3886, 2021.
- [113] K. Friston. Learning and inference in the brain. *Neural Networks*, 16(9):1325–1352, 2003.
- [114] M. Tang, H. Barron, and R. Bogacz. Sequential memory with temporal predictive coding. In *Advances in Neural Information Processing Systems*, volume 36, pages 44341–44355. Curran Associates, Inc., 2023.
- [115] T. Li, M. Tang, and R. Bogacz. Predictive coding model detects novelty on different levels of representation hierarchy. *Neural Computation*, 37(8):1373–1408, 2025.
- [116] A. Ororbia and D. Kifer. The neural coding framework for learning generative models. *Nature Communications*, 13(1):2064, 2022.
- [117] C. Qi, M. Forasassi, T. Lukasiewicz, and T. Salvatori. Towards the training of deeper predictive coding neural networks, 2025. arXiv preprint arXiv:2506.23800.
- [118] Y. Song, B. Millidge, T. Salvatori, T. Lukasiewicz, Z. Xu, and R. Bogacz. Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nature Neuroscience*, 27(2):348–358, 2024.
- [119] N. Alonso, B. Millidge, J. Krichmar, and E. Neftci. A theoretical framework for inference learning, 2022. arXiv preprint arXiv:2206.00164.

- [120] F. Innocenti, R. Singh, and C. Buckley. Understanding predictive coding as an adaptive trust-region method, 2023. arXiv preprint arXiv:2305.18188.
- [121] A. Tscshantz, B. Millidge, A. K. Seth, and C. L. Buckley. Hybrid predictive coding: Inferring, fast and slow. *PLOS Computational Biology*, 19(8):1–31, 08 2023. doi: 10.1371/journal.pcbi.1011280.
- [122] K. Doya, S. Ishii, A. Pouget, and R. Rao. *Bayesian Brain: Probabilistic Approaches to Neural Coding*. MIT Press, 2006.
- [123] J. M. Beck, W. J. Ma, R. Kiani, T. Hanks, A. K. Churchland, J. Roitman, M. N. Shadlen, P. E. Latham, and A. Pouget. Probabilistic population codes for bayesian decision making. *Neuron*, 60(6):1142–1152, 2008.
- [124] M. I. Posner, S. E. Petersen, P. T. Fox, and M. E. Raichle. Localization of cognitive operations in the human brain. *Science*, 240:1627–1631, 1988. doi: 10.1126/science.3289116.
- [125] J. Lisman. Glutamatergic synapses are structurally and biochemically complex because of multiple plasticity processes: long-term potentiation, long-term depression, short-term potentiation and scaling. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 372(1715):20160260, 2017. doi: 10.1098/rstb.2016.0260.
- [126] M. V. Srinivasan, S. B. Laughlin, and A. Dubs. Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 216(1205):427–459, 1982. doi: 10.1098/rspb.1982.0085.
- [127] T. Hosoya, S. A. Baccus, and M. Meister. Dynamic predictive coding by the retina. *Nature*, 436(7047):16001064, July 2005. doi: 10.1038/nature03689.
- [128] A. Alamia and R. VanRullen. Alpha oscillations and traveling waves: Signatures of predictive coding? *PLOS Biology*, 17(10):1–26, 10 2019. doi: 10.1371/journal.pbio.3000487.
- [129] W. Sun and J. Orchard. A predictive-coding network that is both discriminative and generative. *Neural Computation*, 32(10):1836–1862, 2020. doi: 10.1162/neco_a.01311.
- [130] S. J. Gershman. The generative adversarial brain. *Frontiers in Artificial Intelligence*, 2:18, 2019. doi: 10.3389/frai.2019.00018.

- [131] L. Aitchison and M. Lengyel. The hamiltonian brain: Efficient probabilistic inference with excitatory-inhibitory neural circuit dynamics. *PLOS Computational Biology*, 12(12):e1005186, 2016. doi: 10.1371/journal.pcbi.1005186.
- [132] C. Savin and S. Denève. Spatio-temporal representations of uncertainty in spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [133] M. Y.-S. Fang, M. Mudigonda, R. Zarccone, A. Khosrowshahi, and B. A. Olshausen. Learning and inference in sparse coding models with langevin dynamics. *Neural Computation*, 34(8):1676–1700, 07 2022. doi: 10.1162/neco_a_01505.
- [134] L. Shi and T. Griffiths. Neural implementation of hierarchical bayesian inference by importance sampling. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.
- [135] A. Grabska-Barwinska, J. Beck, A. Pouget, and P. Latham. Demixing odors - fast inference in olfaction. In *Advances in Neural Information Processing Systems 26*, pages 1968–1976. Curran Associates, Inc., 2013.
- [136] D. J. Rezende and W. Gerstner. Stochastic variational learning in recurrent spiking networks. *Frontiers in Computational Neuroscience*, 8:38, 2014. doi: 10.3389/fncom.2014.00038.
- [137] P. Berkes, G. Orbán, M. Lengyel, and J. Fiser. Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331(6013):83–87, January 2011. doi: 10.1126/science.1195870.
- [138] L. Aitchison and M. Lengyel. With or without you: predictive coding and bayesian inference in the brain. *Current Opinion in Neurobiology*, 46:219–227, Oct 2017. doi: 10.1016/j.conb.2017.08.010.
- [139] R. Echeveste, L. Aitchison, G. Hennequin, et al. Cortical-like dynamics in recurrent circuits optimized for sampling-based probabilistic inference. *Nature Neuroscience*, 23(10):1138–1149, 2020. doi: 10.1038/s41593-020-0671-1.
- [140] K. J. Friston. Variational filtering. *NeuroImage*, 41(3):747–766, 2008. doi: 10.1016/j.neuroimage.2008.03.017.
- [141] Y. LeCun, C. Cortes, and C. J. Burges. Mnist handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.

- [142] R. M. Neal. Mcmc using hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall / CRC Press, 2010.
- [143] W. T. Coffey and Y. Kalmykov. *The Langevin Equation*. World Scientific, Singapore, 3rd edition, 2012.
- [144] E. Nijkamp, B. Pang, T. Han, L. Zhou, S.-C. Zhu, and Y. N. Wu. Learning multi-layer latent variable model via variational optimization of short run mcmc for approximate inference. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 361–378, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58539-6.
- [145] X. Ji, A. Vedaldi, and J. Henriques. Invariant information clustering for unsupervised image classification and segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9864–9873. IEEE Computer Society, Nov 2019. doi: 10.1109/ICCV.2019.00996.
- [146] P. Gavrikov and J. Keuper. Cnn filter db: An empirical investigation of trained convolutional filters. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun 2022. doi: 10.1109/cvpr52688.2022.01848.
- [147] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Beijing, China, 22–24 Jun 2014. PMLR.
- [148] G. C. G. Wei and M. A. Tanner. A monte carlo implementation of the em algorithm and the poor man’s data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.
- [149] R. Neal and G. E. Hinton. *A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants*. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-011-5014-9.
- [150] B. A. Olshausen. Learning linear, sparse, factorial codes. Technical Report AIM-1580, CBCL-138, Massachusetts Institute of Technology, 1996.

- [151] A. Meulemans, N. Zucchet, S. Kobayashi, J. von Oswald, and J. Sacramento. The least-control principle for local learning at equilibrium. In *Advances in Neural Information Processing Systems*, volume 35, pages 33603–33617, 2022.
- [152] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.
- [153] C. Monier, F. Chavane, P. Baudot, L. J. Graham, and Y. Frégnac. Orientation and direction selectivity of synaptic inputs in visual cortical neurons: A diversity of combinations produces spike tuning. *Neuron*, 37:663–680, 2003. doi: 10.1016/S0896-6273(03)00064-3.
- [154] I. M. Finn, N. J. Priebe, and D. L. Ferster. The emergence of contrast-invariant orientation tuning in simple cells of cat visual cortex. *Neuron*, 54:137–152, 2007. doi: 10.1016/j.neuron.2007.02.029.
- [155] J. F. Mitchell, K. A. Sundberg, and J. H. Reynolds. Differential attention-dependent response modulation across cell classes in macaque visual area v4. *Neuron*, 55:131–141, 2007. doi: 10.1016/j.neuron.2007.06.018.
- [156] M. M. Churchland, B. M. Yu, J. P. Cunningham, L. P. Sugrue, M. R. Cohen, G. S. Corrado, W. T. Newsome, A. M. Clark, P. Hosseini, B. B. Scott, D. C. Bradley, M. A. Smith, A. Kohn, J. A. Movshon, K. M. Armstrong, T. Moore, S. W. Chang, L. H. Snyder, S. G. Lisberger, N. J. Priebe, I. M. Finn, D. Ferster, S. I. Ryu, G. Santhanam, M. Sahani, and K. V. Shenoy. Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nature Neuroscience*, 13(3):369–378, 2010. doi: 10.1038/nn.2501.
- [157] A. K. Churchland, R. Kiani, R. Chaudhuri, X.-J. Wang, A. Pouget, and M. N. Shadlen. Variance as a signature of neural computations during decision making. *Neuron*, 69:818–831, 2011. doi: 10.1016/j.neuron.2010.12.037.
- [158] C. Hussar and T. Pasternak. Trial-to-trial variability of the prefrontal neurons reveals the nature of their engagement in a motion discrimination task. *Proceedings of the National Academy of Sciences*, 107:21842–21847, 2010. doi: 10.1073/pnas.1009956107.

- [159] A. Ledberg, A. Montagnini, R. Coppola, and S. L. Bressler. Reduced variability of ongoing and evoked cortical activity leads to improved behavioral performance. *PLoS ONE*, 7:e43166, 2012. doi: 10.1371/journal.pone.0043166.
- [160] X.-L. Qi and C. Constantinidis. Variability of prefrontal neuronal discharges before and after training in a working memory task. *PLoS ONE*, 7:e41053, 2012. doi: 10.1371/journal.pone.0041053.
- [161] G. Orbán, P. Berkes, J. Fiser, and M. Lengyel. Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron*, 92(2):530–543, 2016. doi: 10.1016/j.neuron.2016.09.038.
- [162] N. Grønbech-Jensen. On the application of non-gaussian noise in stochastic langevin simulations. *Journal of Statistical Physics*, 190:96, 2023.
- [163] N. Chater, M. Oaksford, U. Hahn, and E. Heit. Bayesian models of cognition. *WIREs Cognitive Science*, 1(6):811–823, 2010. doi: 10.1002/wcs.79.
- [164] A. Tambini and L. Davachi. Persistence of hippocampal multivoxel patterns into postencoding rest is related to memory. *Proceedings of the National Academy of Sciences*, 110(48):19591–19596, 2013. doi: 10.1073/pnas.1308499110.
- [165] I. Momennejad, A. R. Otto, N. D. Daw, and K. A. Norman. Offline replay supports planning in human reinforcement learning. *eLife*, 7:e32548, 2018. doi: 10.7554/eLife.32548.
- [166] E. Roscow, R. Chua, R. Ponte Costa, M. Jones, and N. Lepora. Learning offline: memory replay in biological and artificial reinforcement learning. *Trends in Neurosciences*, 44(10):808–821, 2021. ISSN 0166-2236. doi: 10.1016/j.tins.2021.07.007.
- [167] P. Schwartenbeck, A. Baram, Y. Liu, S. Mark, T. Muller, R. Dolan, M. Botvinick, Z. Kurth-Nelson, and T. Behrens. Generative replay for compositional visual understanding in the prefrontal-hippocampal circuit. *bioRxiv*, 2021. doi: 10.1101/2021.06.06.447249.
- [168] M. Ekman, P. Kok, and F. P. de Lange. Time-compressed preplay of anticipated events in human primary visual cortex. *Nature Communications*, 8:15276, 2017. doi: 10.1038/ncomms15276.

- [169] Y. Liu, M. G. Mattar, T. E. J. Behrens, N. D. Daw, and R. J. Dolan. Experience replay is associated with efficient nonlocal learning. *Science*, 372(6544), 2021. doi: 10.1126/science.abf1357.
- [170] A. M. Bastos, W. Usrey, R. A. Adams, G. R. Mangun, P. Fries, and K. J. Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.
- [171] B. Millidge, T. Salvatori, Y. Song, R. Bogacz, and T. Lukasiewicz. Predictive coding: Towards a future of deep learning beyond backpropagation? In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5538–5545. IJCAI, 2022. doi: 10.24963/ijcai.2022/774.
- [172] G. Hennequin, L. Aitchison, and M. Lengyel. Fast sampling-based inference in balanced neuronal networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [173] S. Ray and J. Maunsell. Differences in gamma frequencies across visual cortex restrict their possible use in computation. *Neuron*, 67:885–896, 2010.
- [174] B. Haider, M. Häusser, and M. Carandini. Inhibition dominates sensory responses in the awake cortex. *Nature*, 493:97–100, 2013.
- [175] M. W. Spratling. Predictive coding as a model of biased competition in visual selective attention. *Vision Research*, 48(12):1391–1408, 2008.
- [176] M. W. Spratling. Reconciling predictive coding and biased competition models of cortical function. *Frontiers in Computational Neuroscience*, 2(4):4, 2008. doi: 10.3389/neuro.10.004.2008.
- [177] B. Millidge, M. Tang, M. Osanlouy, N. S. Harper, and R. Bogacz. Predictive coding networks for temporal prediction. *PLOS Computational Biology*, 20(4): 1–31, 2024. doi: 10.1371/journal.pcbi.1011183.
- [178] D. An, J. Xie, and P. Li. Learning deep latent variable models by short-run mcmc inference with optimal transport correction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15410–15419, 2021. doi: 10.1109/CVPR46437.2021.01516.

- [179] G. Oliviers, R. Bogacz, and A. Meulemans. Monte carlo predictive coding: Representing the posterior distribution of latent states in predictive coding networks. In *Conference on Cognitive Computational Neuroscience*, Oxford, UK, 2023. doi: 10.32470/CCN.2023.1373-0.
- [180] U. Zahid, Q. Guo, and Z. Fountas. Sample as you infer: Predictive coding with Langevin dynamics. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 58105–58121. PMLR, 21–27 Jul 2024.
- [181] X. Dong and S. Wu. Neural sampling in hierarchical exponential-family energy-based models. *CoRR*, abs/2310.08431, 2023. doi: 10.48550/arXiv.2310.08431.
- [182] W. Mou, Y.-A. Ma, M. J. Wainwright, P. L. Bartlett, and M. I. Jordan. High-order langevin diffusion yields an accelerated mcmc algorithm. *J. Mach. Learn. Res.*, 22(1), Jan 2021.
- [183] Y.-A. Ma, T. Chen, and E. Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [184] A. Faisal, L. P. J. Selen, and D. Wolpert. Noise in the nervous system. *Nature Reviews Neuroscience*, 9:292–303, 2008. doi: 10.1038/nrn2258.
- [185] M. Llera-Montero, J. Sacramento, and R. Ponte Costa. Computational roles of plastic probabilistic synapses. *Current Opinion in Neurobiology*, 54:90–97, 2019. ISSN 0959-4388. doi: 10.1016/j.conb.2018.09.002.
- [186] M. Boerlin, C. K. Machens, and S. Denève. Predictive coding of dynamical variables in balanced spiking networks. *PLOS Computational Biology*, 9(11): e1003258, 2013. doi: 10.1371/journal.pcbi.1003258.
- [187] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv e-prints*, art. arXiv:1312.6114, Dec 2013. doi: 10.48550/arXiv.1312.6114.
- [188] Y. Zhuo. Deep latent gaussian models, 2019. URL <https://github.com/yiyuezhao/Deep-Latent-Gaussian-Models>.
- [189] M. Seitzer. pytorch-fid: Fid score for pytorch, August 2020. URL <https://github.com/mseitzer/pytorch-fid>. Version 0.3.0.

- [190] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *The Journal of Physiology*, 160(1):106, 1962.
- [191] R. M. Haefner, P. Berkes, and J. Fiser. Perceptual decision-making as probabilistic inference by neural sampling. *Neuron*, 90(3):649–660, 2016. ISSN 0896-6273. doi: 10.1016/j.neuron.2016.03.020.
- [192] C. Teufel and P. C. Fletcher. Forms of prediction in the nervous system. *Nature Reviews Neuroscience*, 21(4):231–242, 2020.
- [193] B. Peters, J. J. DiCarlo, T. Gureckis, R. Haefner, L. Isik, J. Tenenbaum, T. Konkle, T. Naselaris, K. Stachenfeld, Z. Tavares, et al. How does the primate brain combine generative and discriminative computations in vision? *arXiv*, pages arXiv–2401, 2024.
- [194] J. Hohwy, A. Roepstorff, and K. Friston. Predictive coding explains binocular rivalry: an epistemological review. *Cognition*, 108(3):687–701, 2008.
- [195] R. Auksztulewicz and K. Friston. Repetition suppression and its contextual determinants in predictive coding. *Cortex*, 80:125–140, 2016.
- [196] T. Salvatori, L. Pinchetti, B. Millidge, Y. Song, R. Bogacz, and T. Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. *CoRR*, abs/2201.13180, 2022. URL <https://arxiv.org/abs/2201.13180>.
- [197] D. L. K. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.
- [198] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.
- [199] S. Grossberg. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. neural. *Neural Networks : the official journal of the International Neural Network Society*, 37, 2012. doi: 10.1016/j.neunet.2012.09.017.

- [200] V. A. F. Lamme and P. R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neurosciences*, 23(11):571–579, 2000.
- [201] I. Yildirim, M. H. Siegel, A. A. Soltani, S. Ray Chaudhuri, and J. B. Tenenbaum. Perception of 3d shape integrates intuitive physics and analysis-by-synthesis. *Nature Human Behaviour*, 8(2):320–335, 2024.
- [202] D. Xu, A. Clappison, C. Seth, and J. Orchard. Symmetric predictive estimator for biologically plausible neural learning. *IEEE Transactions on Neural Networks and Learning Systems*, 29(9):4140–4151, 2017.
- [203] B. Bozkurt, C. Pehlevan, and A. Erdogan. Correlative information maximization: A biologically plausible approach to supervised deep neural networks without weight symmetry. *Advances in Neural Information Processing Systems*, 36: 34928–34941, 2023.
- [204] D. E. Rumelhart and J. L. McClelland. An interactive activation model of context effects in letter perception: Part 2. the contextual enhancement effect and some tests and extensions of the model. *Psychological Review*, 89(1):60–94, 1982. doi: 10.1037/0033-295X.89.1.60.
- [205] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, Cham, 2015. doi: 10.1007/978-3-319-24574-4.28.
- [206] B. Yang and X. Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. pages 15703–15712, 06 2025. doi: 10.1109/CVPR52734.2025.01464.
- [207] K. Tian, Y. Jiang, Z. Yuan, B. Peng, and L. Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. 2024.
- [208] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. 05 2016. doi: 10.48550/arXiv.1605.08104.
- [209] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996.

- [210] S. Qiu, S. Bhattacharyya, D. Coyle, and S. Dora. Deep predictive coding with bi-directional propagation for classification and reconstruction. *arXiv preprint arXiv:2305.18472*, 2023.
- [211] M. L. Rosen, M. A. Sheridan, K. A. Sambrook, M. R. Peverill, A. N. Meltzoff, and K. A. McLaughlin. The role of visual association cortex in associative memory formation across development. *Journal of Cognitive Neuroscience*, 30(3):365–380, 2018.
- [212] H. Komatsu. The neural mechanisms of perceptual filling-in. *Nature Reviews Neuroscience*, 7(3):220–231, 2006. doi: 10.1038/nrn1869.
- [213] M. R. Greene and A. Oliva. The briefest of glances: The time course of natural scene understanding. *Psychological Science*, 20(4):464–472, 2009.
- [214] C. J. Spoeer, T. C. Kietzmann, J. Mehrer, I. Charest, and N. Kriegeskorte. Recurrent neural networks can explain flexible trading of speed and accuracy in biological vision. *PLOS Computational Biology*, 16(10):e1008215, 2020.
- [215] K. Kar, J. Kubilius, K. Schmidt, E. B. Issa, and J. J. DiCarlo. Evidence that recurrent circuits are critical to the ventral stream’s execution of core object recognition behavior. *Nature Neuroscience*, 22(6):974–983, 2019.
- [216] R. Jordan and G. B. Keller. Opposing influence of top-down and bottom-up input on excitatory layer 2/3 neurons in mouse primary visual cortex. *Neuron*, 108(6):1194–1206.e5, 2020. ISSN 0896-6273. doi: 10.1016/j.neuron.2020.09.024.
- [217] K. K. Nejad, P. Anastasiades, L. Hertäg, and R. Ponte Costa. Self-supervised predictive learning accounts for cortical layer-specificity. *Nature Communications*, 16(1):6178, 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-61399-5.
- [218] G. A. Calvert. Crossmodal processing in the human brain: Insights from functional neuroimaging studies. *Cerebral Cortex*, 11(12):1110–1123, 12 2001. ISSN 1047-3211. doi: 10.1093/cercor/11.12.1110.
- [219] A. C. Li, A. A. Efros, and D. Pathak. Understanding collapse in non-contrastive siamese representation learning, 2022. URL <https://arxiv.org/abs/2209.15007>.

- [220] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. URL <https://arxiv.org/abs/1708.07747>.
- [221] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- [222] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [223] J. Marino, Y. Yue, and S. Mandt. Iterative amortized inference. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3403–3412. PMLR, 2018.
- [224] L. Biewald. Experiment tracking with weights and biases, 2020. URL <https://www.wandb.com/>. Software available from wandb.com.
- [225] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016. URL <https://arxiv.org/abs/1606.03498>.
- [226] J. Chen. Inception score, October 2020. URL https://github.com/sundyCoder/IS_MS_SS.git.
- [227] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [228] L. B. Smith, S. S. Jones, B. Landau, L. Gershkoff-Stowe, and L. Samuelson. Object name learning provides on-the-job training for attention. *Psychological Science*, 13(1):13–19, 2002.
- [229] W. Nie, Z. Yu, L. Mao, A. B. Patel, Y. Zhu, and A. Anandkumar. Bongard-logo: A new benchmark for human-level concept learning and reasoning. *Advances in Neural Information Processing Systems*, 33:16468–16480, 2020.
- [230] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

- [231] O. C. González, Y. Sokolov, G. P. Krishnan, J. E. Delanois, and M. Bazhenov. Can sleep protect memories from catastrophic forgetting? *eLife*, 9:e51005, 2020.
- [232] P. A. Tsividis, T. Pouncy, J. L. Xu, J. B. Tenenbaum, and S. J. Gershman. Human learning in atari. In *AAAI Spring Symposium Series*, pages 643–646, 2017.
- [233] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. 1989.
- [234] G. M. van de Ven, N. Soures, and D. Kudithipudi. *Continual learning and catastrophic forgetting*, pages 153–168. 2025.
- [235] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.
- [236] A. Bernacchia, M. Lengyel, and G. Hennequin. Exact natural gradient in deep linear networks and its application to the nonlinear case. In *Advances in Neural Information Processing Systems*, volume 31, 2018.
- [237] D. Huh. Curvature-corrected learning dynamics in deep neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML’20. JMLR.org, 2020.
- [238] J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2020.
- [239] J. Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.
- [240] A. Jnini and F. Vella. Dual natural gradient descent for scalable training of physics-informed neural networks, 2025. URL <https://arxiv.org/abs/2505.21404>.
- [241] J. Müller and M. Zeinhofer. Achieving high accuracy with pinns via energy natural gradients, 2023. URL <https://arxiv.org/abs/2302.13163>.
- [242] A. Saxe, J. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014.

- [243] K. Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, volume 29, 2016.
- [244] C. C. J. Dominé, N. Anguita, A. M. Proca, L. Braun, D. Kunin, P. A. M. Mediano, and A. M. Saxe. From lazy to rich: Exact learning dynamics in deep linear networks. *arXiv preprint arXiv:2409.14623*, 2024.
- [245] B. Millidge, Y. Song, T. Salvatori, T. Lukasiewicz, and R. Bogacz. A theoretical framework for inference and learning in predictive coding networks, 2022. URL <https://arxiv.org/abs/2207.12316>.
- [246] T.-C. Kao, K. T. Jensen, G. M. van de Ven, A. Bernacchia, and G. Hennequin. Natural continual learning: Success is a journey, not (just) a destination. *arXiv preprint arXiv:2106.08085*, 2021.
- [247] G. Zhang, J. Martens, and R. B. Grosse. Fast convergence of natural gradient descent for over-parameterized neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- [248] X. Wu, W. Yu, C. Zhang, and P. Woodland. An improved empirical fisher approximation for natural gradient descent, 2024.
- [249] D. J. Cai, D. Aharoni, T. Shuman, J. Shobe, J. Biane, W. Song, B. Wei, M. Veshkini, M. La-Vu, J. Lou, S. E. Flores, I. Kim, Y. Sano, M. Zhou, K. Baumgaertel, A. Lavi, M. Kamata, M. Tuszynski, M. Mayford, P. Golshani, and A. J. Silva. A shared neural ensemble links distinct contextual memories encoded close in time. *Nature*, 534(7605):115–118, 2016.
- [250] M. Sehgal, M. Zhou, A. Lavi, S. Huang, Y. Zhou, and A. J. Silva. Memory allocation mechanisms underlie memory linking across time. *Neurobiology of Learning and Memory*, 153:21–25, 2018.
- [251] H. B. Barlow. Possible principles underlying the transformations of sensory messages. In *Sensory Communication*, pages 217–234. The MIT Press, 1961.
- [252] J. A. Gallego, M. G. Perich, L. E. Miller, and S. A. Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, 2017. ISSN 0896-6273. doi: 10.1016/j.neuron.2017.05.025.

- [253] J. A. Gallego, M. G. Perich, R. H. Chowdhury, S. A. Solla, and L. E. Miller. Long-term stability of cortical population dynamics underlying consistent behavior. *Nature Neuroscience*, 23(2):260–270, 2020. ISSN 1546-1726. doi: 10.1038/s41593-019-0555-4.
- [254] Y. Song, T. Lukasiewicz, Z. Xu, and R. Bogacz. Can the brain do backpropagation? — exact implementation of backpropagation in predictive coding networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 22566–22579, 2020.
- [255] C. L. Baker. Spatial- and temporal-frequency selectivity as a basis for velocity preference in cat striate cortex neurons. *Visual Neuroscience*, 4(2):101–113, 1990.
- [256] L. Pinchetti, S. Frieder, T. Lukasiewicz, and T. Salvatori. Faster predictive coding networks via better initialization, 2026. URL <https://arxiv.org/abs/2601.20895>.
- [257] J. D. Kendall, R. D. Pantone, K. Manickavasagam, Y. Bengio, and B. Scellier. Training end-to-end analog neural networks with equilibrium propagation. *CoRR*, abs/2006.01981, 2020. URL <https://arxiv.org/abs/2006.01981>.
- [258] D. G. Tzikas, A. C. Likas, and N. P. Galatsanos. The variational approximation for bayesian inference. *IEEE Signal Processing Magazine*, 25(6):131–146, 2008.
- [259] K. Friston, J. Kilner, and L. Harrison. A free energy principle for the brain. *Journal of Physiology-Paris*, 100(1-3):70–87, 2006. doi: 10.1016/j.jphysparis.2006.10.001.
- [260] S. Qiu, S. Bhattacharyya, D. Coyle, and S. Dora. Deep predictive coding with bi-directional propagation for classification and reconstruction. *arXiv preprint arXiv:2305.18472*, 2023. URL <https://arxiv.org/abs/2305.18472>.