Extending Predictive Coding: Space, Time and Memory

Mufeng Tang
St Cross College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity 2025

Acknowledgments

First and foremost, I want to thank my supervisor, Rafal Bogacz. He's taught me so much—not just about science, but how to think like a scientist. From him, I learned to approach big questions by starting from the simplest systems, which often reveal the deepest insights. He also showed me how to communicate science clearly to different audiences. What I've appreciated most is who he is beyond the science: always kind, humble, and supportive. No matter how rough or strange my ideas were, he'd listen with genuine interest and often get excited about them. He truly cares not just about the work, but about the people behind it, and that's made my D.Phil. journey as enjoyable as it could be.

I also want to thank my co-supervisor, Helen Barron, for bringing fresh experimental neuroscience perspectives into my research. Coming from statistics and machine learning, I often thought in terms of models and abstractions, but Helen's insights, and the thoughtful questions from her and her lab, helped ground my thinking in real neural data. Their curiosity thinking often sparked new ideas or pushed me to refine my models in ways I wouldn't have considered on my own. That cross-disciplinary input has been incredibly valuable, and I'm truly grateful for it.

A big thanks to my lab mates—Yuhao, Charlotte, Gaspard, and Nima—for making the office such a warm and fun place. Whether it was lunch chats, random deep dives into weird topics, or just sharing jokes throughout the day, you made the work environment genuinely enjoyable. Being around people who get the research and know how to have fun made this journey all the more memorable.

Lastly, I want to thank my family. Thank you to my parents for supporting me throughout all these years while I am far away from home. I would like to thank my fiancé and best friend, Wan, for her love and support since we met in 2015. Thank you for making me understand what is truely important in life is not the goal that we try to reach, but the journey towards it and the people that we experience the journey with. I am thankful that you've been by my side during all these years and I'm excited for what lies ahead of us.

Statement of Intellectual Contribution

The research in this thesis is largely the work of myself along with my supervisors Prof. Rafal Bogacz and Prof. Helen Barron. Co-authorship is detailed below:

- Chapter 2 is based on the publication *Recurrent predictive coding models for associative memory employing covariance learning*, of which I am the lead author and Tommaso Salvatori, Beren Millidge, Yuhang Song have contributed to the experimentation and editing of the work. Prof. Thomas Lukasiewicz and Prof. Rafal Bogacz have provided supervision. Section 2.3.3 of this chapter is my main theoretical contribution to the preprint *Predictive coding model can detect novelty on different levels of representation hierarchy*, of which Tianjin Li was the first author while he was a Master's student at the University of Oxford, and I co-authored and co-supervised this work with Prof. Rafal Bogacz.

- Chapter 3 is based on the publication *Predictive coding networks for temporal prediction*, of which Beren Millidge and I were co-first author. Specifically, Beren developed the initial idea of temporal predictive coding and its conceptual relationship to Kalman filtering, and designed the linear filtering experiments. My contributions consist of the formalization of the relationship to Bayesian filtering, completing the filtering experiments, investigating the covariance-encoding property of the model, and designed and performed experiments with natural movies. Prof. Nicol Harper and Prof. Rafal Bogacz provided supervision. A special thank you goes to Sebastian Klavinskis-Whiting for providing the natural movie data.

- Chapter 4 is based on the publication *Sequential memory with temporal predictive coding*, of which I am the lead author and Prof. Helen Barron and Prof. Rafal Bogacz provided supervision.

- Chapter 5 is based on the preprint *Learning grid cells by predictive coding*, of which I am the lead author and Prof. Helen Barron and Prof. Rafal Bogacz provided supervision.

Abstract

Predictive coding is one of the most influential theories and computational models of the brain in modern neuroscience. It posits that the brain constructs an internal generative model of the world, continuously predicting sensory input and updating itself by minimizing the mismatch between prediction and reality. Neural activity, in this framework, reflects the brain's attempt to infer the hidden causes behind its sensory experiences. With strong theoretical roots in the Bayesian Brain hypothesis and a concrete implementation through artificial neural networks that resemble biological circuitry, predictive coding has significantly influenced theoretical neuroscience and has begun to inform methods in artificial intelligence.

This thesis aims to push the boundary of predictive coding towards three crucial topics of the brain that are underexplored within the predictive coding framework: space, time and memory. These aspects are important as they support how the continuum of sensory experiences are represented in the brain. They are also intertwined, which requests a single, unified theoretical account. This thesis will start with modeling *memories* of individual events, using an extension of predictive coding models that incorporates recurrent connections observed in the brain's memory system. We will establish a theoretical relationship between this recurrent predictive coding model with other influential memory models in neuroscience, suggesting a shared underlying principle that may reflect real brain mechanisms. We will then step into the realm of *time*, proposing a temporal extension of predictive coding. Applying it to the visual system, we will show that the model develops similar dynamic representations to the visual cortex. We will also apply this model to memory, although this time sequential memories of dynamic events. We will show that it successfully performs sequential memory, and presents evidence for an internal map of the events. Inspired by the latter point, we will further apply predictive coding to *space*. We show that the model gives rise to grid cell–like responses—spatial patterns characterized by striking hexagonal geometry, as seen in the brain's navigation circuits. This discovery is the

first to show that predictive coding can yield biologically plausible representations beyond the sensory cortex.

Together, these contributions suggest that predictive coding may offer a unifying computational framework for understanding how the brain encodes memory, time, and space, providing fresh insights into the computational mechanisms underlying various neural functions and representations.

Contents

List of Figures

Notation Conventions

x, y, ϵ	Scalar variables
$\mathbf{x}$, $\mathbf{y}$, $\boldsymbol{\epsilon}$	Vector variables
$\mathbf{x}^{(l)}$	Activity vector of layer l of a neural network
$\mathbf{x}(i)$	i-th vector sample in a dataset
$\mathbf{x}_k$	k-dimensional subvector of $\mathbf{x}$ or k-th sample in a sequence of vectors
x_i	i-th dimension of a vector
$\dot{\mathbf{x}}$	Dynamics of a variable $\mathbf{x}$
$\mathbf{A}$, $\boldsymbol{\Theta}$	Matrix
$\mathbf{A}_{mn}$	$m \times n$ submatrix of $\mathbf{A}$
A_{ij}	Matrix element at the i-th row and j-th column
$\mathcal{F}, \mathcal{L}$	Objective function
$\mathbb{R}^{d_1 \times d_2}$	Matrix of size $d_1 \times d_2$ of real numbers
$\mathbb{E}_p$	Expectation over a distribution p
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\frac{\partial \mathcal{F}}{\partial \mathbf{x}}$	Partial derivative of $\mathcal{F}$ with respect to $\mathbf{x}$

Chapter 1

Introduction

What is intelligence? From the early musings of Plato and Aristotle on minds and reasoning to current debates on what defines real artificial intelligence, humans have pondered this question for millennia. Arguably, the most compelling example—and the ultimate benchmark—of intelligence is our own brain: a biological network of remarkable complexity that seamlessly integrates perception, action, and learning. The past century has witnessed remarkable advancements in understanding the brain: from Pavlov's pioneering conditioning studies (Pavlov, 1906) to the advent of functional magnetic resonance imaging (fMRI) (Ogawa et al., 1990); from the concept of action potentials (Bernstein, 1902) to how they are transmitted between neurons (Hodgkin and Huxley, 1952); from the definition of synapses (Sherrington, 2023) to the foundational principles of synaptic transmission and learning (Dale, 1935; Hebb, 1949)—all contributing to a more thorough and detailed understanding of the brain's components and their harmonious interplay.

Despite these extraordinary discoveries, a central question remains elusive: how do these components collectively give rise to the diverse functionalities and representations observed in the brain? Historically, computational neuroscience has sought to address this question by constructing bottom-up models that simulate brain processes. These models range from abstract, high-level frameworks—often employing one or a few equations to capture general principles, such as the brain's tendency to represent sensory information with minimal activity (Barlow, 1961)—to detailed, low-level systems that demonstrate how complex functions emerge from networks of simple units, such as spiking neural networks (Vogels et al., 2011). However, a persistent challenge is that many high-level models lack detailed neural implementations of their computational principles, while many low-level models are not grounded in overarching theoretical frameworks. Predictive coding bridges this gap by integrating both perspectives. Rooted in the Bayesian Brain hypothesis, which posits that

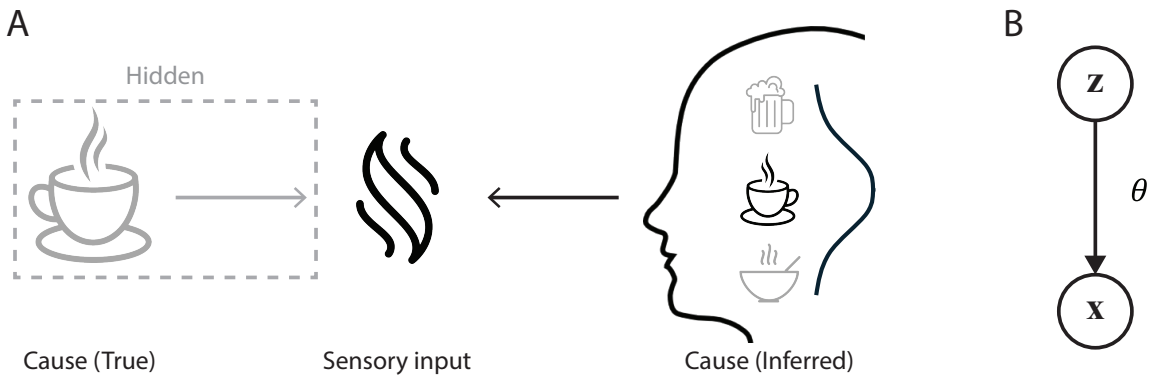

Figure 1.1: **The Bayesian Brain hypothesis.** A: Schematic of Bayesian Brain, where aroma (sensory input) is generated from a cup of coffee and the brain tries to reverse-engineer this process via Bayesian inference. B: Abstraction of Bayesian Brain using random variables $\mathbf{z}$ and $\mathbf{x}$.

the brain constructs and optimizes a generative model of sensory information (Doya et al., 2007), predictive coding not only provides a high-level computational principle but also offers biologically plausible neural circuit implementations (Bogacz, 2017). Building on this foundation, this thesis extends the predictive coding framework to address three fundamental challenges faced by the brain: space, time, and memory—all while preserving the integrity of both the high-level principles and the low-level neural plausibility.

1.1 Predictive coding

Before stepping into the specificity, two key ideas that lay the foundation of predictive coding need to be introduced. First, the Bayesian Brain hypothesis, and second, the idea of artificial neurons and networks.

1.1.1 Bayesian Brain

Imagine waking up in the morning and making your way into the kitchen. Gladly someone is making coffee. Before you enter the kitchen you can already smell the aroma of coffee and you think, "Ah, there's coffee in the kitchen!". How does this perception of coffee takes place, even though you did not witness the coffee itself with your own eyes? Could it be some other drinks emitting the aroma? All we know is that there exists some true, but hidden "cause" (the coffee) that generates the sensory observation (the aroma). One possibility is that the perception of the coffee happens

as the brain tries to reverse-engineer this generative process, by building up its own generative model to approximate the true one and infer the cause from the model (Fig. 1.1A). This is the essence of the Bayesian Brain hypothesis.

The concept of Bayesian Brain was first articulated by Helmholtz (1866) and then formalized in computational neuroscience in multiple studies in the late 90s and early 2000s (Dayan et al., 1995; Lee and Mumford, 2003; Knill and Pouget, 2004). Multiple lines of experimental research have provided evidence for this Bayesian inference perspective of perception, such as those on sensory cue integration (Jacobs, 1999; Knill and Saunders, 2003; Hillis et al., 2004) and those on perceptual biases from inappropriate priors (Weiss et al., 2002). This Bayesian approach to neural computation has also been applied to data fitting, from fMRI data (Behrens et al., 2014; Friston et al., 2003) to electrophysiological recordings (Beck et al., 2007).

Formally, Bayesian Brain can be described using a latent variable model depicted in Fig. 1.1B (Bishop and Nasrabadi, 2006), where $\mathbf{z}$ represents a hidden, or latent cause (coffee) and $\mathbf{x}$ represents the observed sensory input (aroma) of the brain. The generative model is parameterized by θ, which provides an abstraction of the exact model. Due to uncertainty, the model cannot be deterministic and thus probability distributions are needed to describe the variables $\mathbf{z}$ and $\mathbf{x}$. Specifically, the probability distribution of interest is:

$$p_\theta(\mathbf{z}|\mathbf{x}) \tag{1.1}$$

which defines the conditional distribution of the latent $\mathbf{z}$ given the observations $\mathbf{x}$. In essence, this distribution describes the underlying cause given the observations. However, in the setting of the Bayesian Brain hypothesis, the observable direction of dependence is the opposite i.e., the observation $\mathbf{x}$ is generated from the cause $\mathbf{z}$. We thus need to resort to Bayes theorem (Bayes, 1763) to find an alternative expression for the above distribution:

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})} \tag{1.2}$$

where $p_\theta(\mathbf{x}|\mathbf{z})$ represents the generative process, which is also referred to as the likelihood. $p_\theta(\mathbf{z})$ is a prior distribution on $\mathbf{z}$, and $p_\theta(\mathbf{x})$ is the marginal likelihood of the observed data, also known as "evidence". In this setting, the term $p_\theta(\mathbf{z}|\mathbf{x})$ is called the posterior distribution. Although the generative process and the prior are usually available under certain assumptions, the evidence is not, as it requires us to evaluate the intractable integral over all latent variables:

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z} \tag{1.3}$$

This conundrum thus requires us to seek an alternative method to obtain the posterior. Although methods such as those based on sampling exist (Van Ravenzwaaij et al., 2018), we focus on an approximation approach called variation inference here.

Variational inference One way of looking at variational inference is from an approximation perspective: if we cannot model $p_\theta(\mathbf{z}|\mathbf{x})$ exactly, can we find another distribution, say $q(\mathbf{z})$, to approximate it? To make the approximation as accurate as possible, we need to minimize a dissimilarity measure between $p_\theta(\mathbf{z}|\mathbf{x})$ and $q(\mathbf{z})$. The standard measure is the Kullback-Leibler (KL) divergence, D_{KL} (Kullback and Leibler, 1951), which is defined as follows:

$$D_{\mathrm{KL}}(q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})) = \mathbb{E}_q[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z}|\mathbf{x})] \tag{1.4}$$

It can be further written as (with Bayes Theorem):

$$\begin{aligned} D_{\mathrm{KL}}(q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_q\left[\log q(\mathbf{z}) - \log \frac{p_\theta(\mathbf{z},\mathbf{x})}{p_\theta(\mathbf{x})}\right] \\ &= \mathbb{E}_q[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z},\mathbf{x})] + \log p_\theta(\mathbf{x}) \\ &= \mathcal{F} + \log p_\theta(\mathbf{x}) \end{aligned} \tag{1.5}$$

where $\mathcal{F} := \mathbb{E}_q[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z},\mathbf{x})]$ is the variational free energy (Feynman, 1972; Friston, 2003, 2005). The term $-\mathcal{F}$ is also known as the evidence lower bound (ELBO) in machine learning, as the KL divergence is non-negative and thus $-\mathcal{F}$ serves as a lower bound for the (log) evidence $\log p_\theta(\mathbf{x})$.

Note that to minimize the KL divergence as a function of $q(\mathbf{z})$, we only need to minimize the free energy $\mathcal{F}$ as $\log p_\theta(\mathbf{x})$ is not a function of $q(\mathbf{z})$. The minimization of $\mathcal{F}$, or the maximization of $-\mathcal{F}$, is often performed through the Expectation-maximization (EM) algorithm (Dempster et al., 1977), where $\mathcal{F}$ is first minimized with respect to $q(\mathbf{z})$ and then minimized with respect to the parameters θ. As will be shown later, computations of predictive coding follows exactly this alternating process.

1.1.2 Artificial neurons and networks

The idea of artificial neurons and networks, as opposed to biological ones, is core to the low-level understanding of predictive coding. In particular, we focus on the neural network model where a neuron is assumed to receive a vector of inputs linearly weighted by synapses (weights) connecting each input neuron to the current neuron,

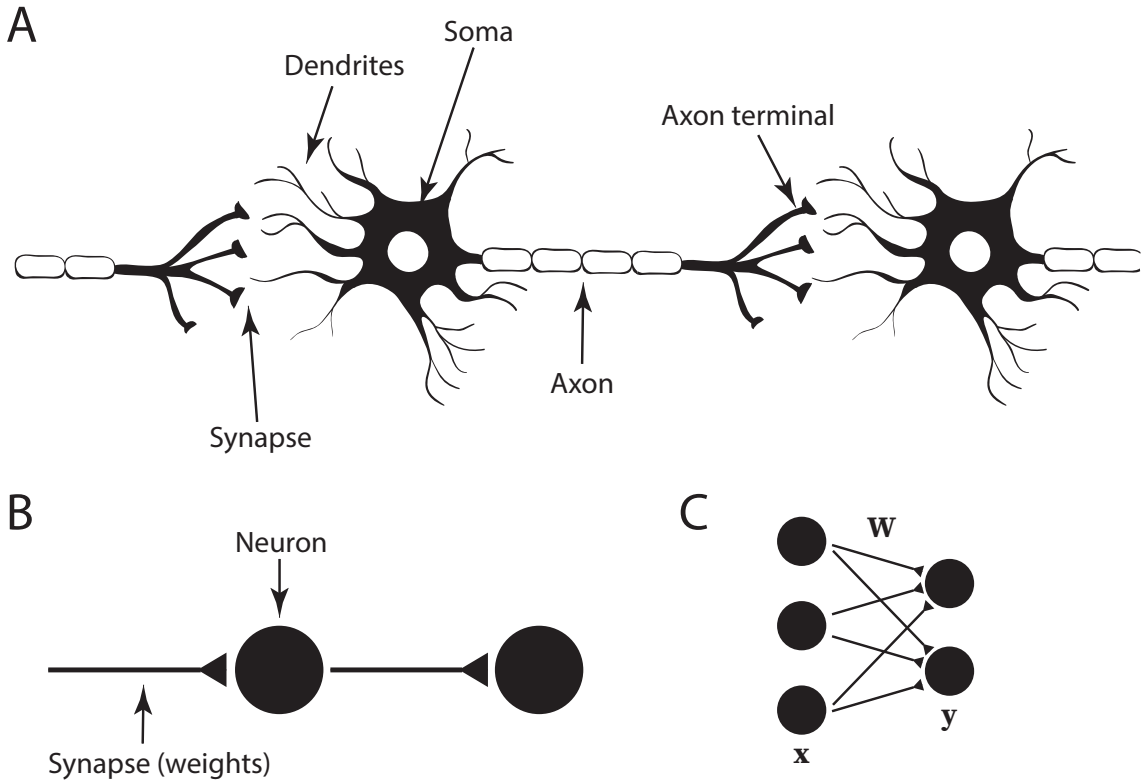

Figure 1.2: **Biological and artificial neurons.** A: Biological neurons and how they transmit signals. B: Artificial neurons and synapses as an abstraction of biological ones. C: A network of artificial neurons connected hierarchically.

perform a sum of the weighted inputs and then output the thresholded sum (Rosenblatt, 1958). This process can be mapped to a biological neuron in Fig 1.2A where the dendrites of a neuron receive inputs from a pre-synaptic neurons, and the soma performs the summation function. Once the soma reaches a certain potential, the axon will transmit the signal to its terminal connecting another neuron via synapses. Artificial neurons provide an abstraction of this process, where specific architectural components of a neuron, including dendrites, soma and axon, are included in a computing unit in Fig 1.2B.

Formally, assuming d_{in} pre-synaptic neurons providing an input vector $\mathbf{x} \in \mathbb{R}^{d_{in}}$, and d_{out} post-synaptic neurons receiving the input and performing the thresholded linear summation and outputing $\mathbf{y} \in \mathbb{R}^{d_{out}}$. The synaptic weight is a $d_{out} \times d_{in}$ matrix $\mathbf{W}$. The output follows:

$$\mathbf{y} = \mathbf{W} f(\mathbf{x}) \tag{1.6}$$

or if we look at individual neurons $i = 1, ..., d_{out}$:

$$y_i = \sum_{j=1}^{d_{in}} W_{ij} f(x_j) \tag{1.7}$$

where f is the threshold function, which essentially provides a non-linear transformation of the inputs. Depending on the connectivity pattern, this network can either be fully feedforward (Fig 1.2C), which has been shown to have the capability of approximating any functions (Hornik et al., 1989), or recurrent e.g., a Hopfield Network (Hopfield, 1982), which can perform pattern completion/associative memory tasks. Eq. 1.7 can also be described using a dynamical equation:

$$\dot{y}_i = -y_i + \sum_{j=1}^{d_{in}} W_{ij} f(x_j) \tag{1.8}$$

which has its equilibrium equating Eq. 1.7 and captures the dynamical nature of neurons.

1.1.3 Implementing Bayesian Brain in neural networks

We are now ready to move on to predictive coding, which implements the Bayesian Brain hypothesis into a network of artificial neurons. On the basis of Bayesian Brain and the free energy principle, predictive coding makes the following two assumptions:

1. The variational approximation $q(\mathbf{z})$ follows a delta distribution with density concentrated on $\boldsymbol{\phi}$ i.e., $q(\mathbf{z}) = \delta(\mathbf{z} - \boldsymbol{\phi})$. Furthermore, the center $\boldsymbol{\phi}$ has a Gaussian prior $\boldsymbol{\phi} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_\phi)$;

2. The generative process $p_\theta(\mathbf{x}|\mathbf{z})$ follows a Gaussian distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\Theta} f(\mathbf{z}), \boldsymbol{\Sigma}_\mathbf{x})$,

which enable us to re-write the free energy $\mathcal{F}$ as:

$$\begin{aligned}
\mathcal{F} &= \int_\mathbf{z} q(\mathbf{z}) \log q(\mathbf{z}) d\mathbf{z} - \int_\mathbf{z} q(\mathbf{z}) \log p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= -\log p_\theta(\mathbf{x}, \boldsymbol{\phi}) + C \\
&= -\log p_\theta(\mathbf{x}|\boldsymbol{\phi}) - \log p_\theta(\boldsymbol{\phi}) + C \\
&= \frac{1}{2}(\mathbf{x} - \boldsymbol{\Theta} f(\boldsymbol{\phi}))^\top \boldsymbol{\Sigma}_\mathbf{x}^{-1} (\mathbf{x} - \boldsymbol{\Theta} f(\boldsymbol{\phi})) + \frac{1}{2}(\boldsymbol{\phi} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}_\phi^{-1} (\boldsymbol{\phi} - \boldsymbol{\mu}) \\
&+ \frac{1}{2} \log |\boldsymbol{\Sigma}_x| + \frac{1}{2} \log |\boldsymbol{\Sigma}_\phi| + C
\end{aligned} \tag{1.9}$$

where from the first to the second line we use the fact that for a delta function $\delta(\mathbf{z} - \boldsymbol{\phi})$, the integral $\int_\mathbf{z} q(\mathbf{z}) \log p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$ is equal to $\log p_\theta(\mathbf{x}, \boldsymbol{\phi})$, and that $\int_\mathbf{z} q(\mathbf{z}) \log q(\mathbf{z}) d\mathbf{z}$ is

independent of $\boldsymbol{\phi}$ and can thus be absorbed in a constant C. This expression of the free energy can be regarded as the sum of the (weighted and squared) prediction errors $\mathbf{x} - \boldsymbol{\Theta} f(\boldsymbol{\phi})$ and $\boldsymbol{\phi} - \boldsymbol{\mu}$, thus the name "predictive coding". Intuitively, in the predictive coding framework, neurons in the brain generate predictions of the sensory inputs, resulting in prediction errors. The objective of the brain, which leads to its synaptic and neuronal dynamics, is to minimize the predictions errors. When the prediction errors are close to 0, $\boldsymbol{\phi}$ is predictive of the sensory observation $\mathbf{x}$, thus forming a "good" representation of the sensory input.

Using the EM algorithm to minimize the free energy (and thus prediction errors) consists of first minimizing $\mathcal{F}$ with respect to $\boldsymbol{\phi}$, which defines $q(\mathbf{z})$, via gradient descent:

$$\dot{\boldsymbol{\phi}} \propto -\frac{\partial \mathcal{F}}{\partial \boldsymbol{\phi}} = -\boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1}(\boldsymbol{\phi} - \boldsymbol{\mu}) + f'(\boldsymbol{\phi}) \odot \boldsymbol{\Theta}^{\top} \boldsymbol{\Sigma}_{\mathbf{x}}^{-1}(\mathbf{x} - \boldsymbol{\Theta} f(\boldsymbol{\phi})) \tag{1.10}$$

where $\odot$ is the element-wise product between vectors. If we define $\boldsymbol{\epsilon}_{\mathbf{x}} := \boldsymbol{\Sigma}_{\mathbf{x}}^{-1}(\mathbf{x} - \boldsymbol{\Theta} f(\boldsymbol{\phi}))$ and $\boldsymbol{\epsilon}_{\boldsymbol{\phi}} := \boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1}(\boldsymbol{\phi} - \boldsymbol{\mu})$, this becomes:

$$\dot{\boldsymbol{\phi}} \propto -\boldsymbol{\epsilon}_{\boldsymbol{\phi}} + f'(\boldsymbol{\phi}) \odot \boldsymbol{\Theta}^{\top} \boldsymbol{\epsilon}_{\mathbf{x}} \tag{1.11}$$

The second step is to minimize $\mathcal{F}$ with respect to the parameters $\theta = \{\boldsymbol{\Theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\boldsymbol{\phi}}\}$, also via gradient descent:

$$\Delta \boldsymbol{\Theta} \propto -\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Theta}} = \boldsymbol{\epsilon}_{\mathbf{x}} f(\boldsymbol{\phi})^{\top}; \quad \Delta \boldsymbol{\mu} \propto -\frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}} = \boldsymbol{\epsilon}_{\boldsymbol{\phi}} \tag{1.12}$$

$$\Delta \boldsymbol{\Sigma}_{\mathbf{x}} \propto -\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Sigma}_{\mathbf{x}}} = \frac{1}{2}(-\boldsymbol{\Sigma}_{\mathbf{x}}^{-1} + \boldsymbol{\epsilon}_{\mathbf{x}} \boldsymbol{\epsilon}_{\mathbf{x}}^{\top}); \quad \Delta \boldsymbol{\Sigma}_{\boldsymbol{\phi}} \propto -\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Sigma}_{\boldsymbol{\phi}}} = \frac{1}{2}(-\boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{-1} + \boldsymbol{\epsilon}_{\boldsymbol{\phi}} \boldsymbol{\epsilon}_{\boldsymbol{\phi}}^{\top}) \tag{1.13}$$

Neural implementation At this stage, the objective of free energy minimization has been translated into the dynamic equations above describing neural and synaptic activities. How may a neural network implement such computations? If we assume $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}$ and $\boldsymbol{\Sigma}_{\mathbf{x}}$ are both identity matrices for simplicity (this will be discussed in more detail at later parts of this thesis), the computations above can be performed by the network shown in Fig. 1.3A, where we implement $\mathbf{x}$, $\boldsymbol{\phi}$ and $\boldsymbol{\epsilon}$'s as neural activities, and $\boldsymbol{\Theta}$ and $\boldsymbol{\mu}$ as synaptic weights. Error neurons $\boldsymbol{\epsilon}$ (circles) will receive excitatory inputs (red arrows) from their corresponding value neurons (triangles)[1], and inhibitory

[1]Note that throughout this thesis, "value" neuron is a concept completely unrelated to "value" functions in reinforcement learning. Rather, it represents the value of neural activities, to compare to the error of neural activities.

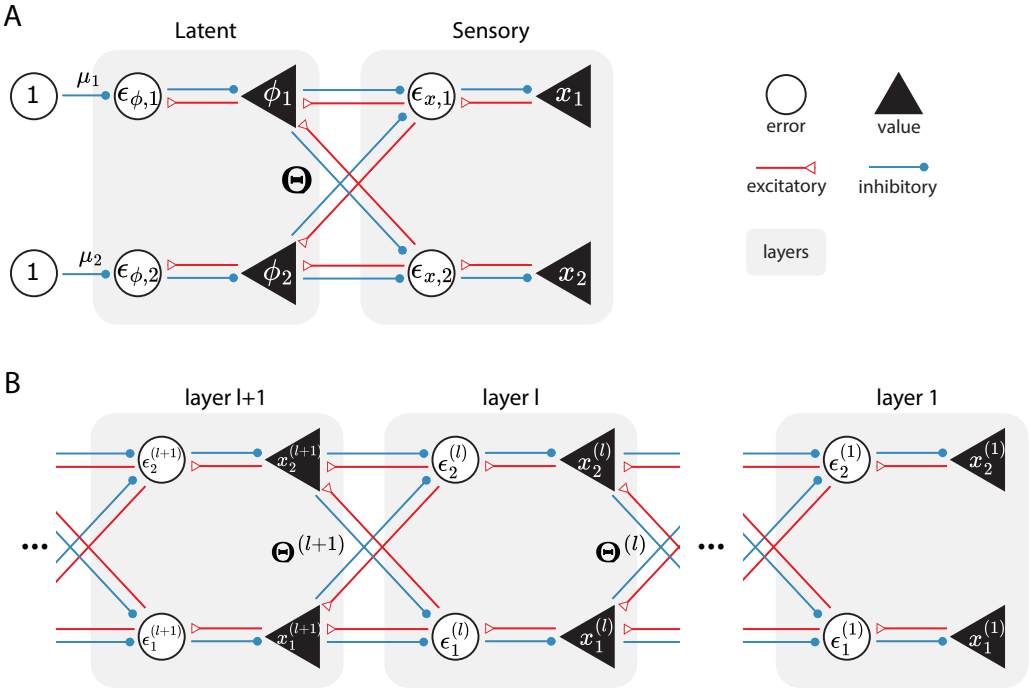

Figure 1.3: **Network implementation of predictive coding.** A: A simple, two-layer case. B: Extending predictive coding to a multi-layer architecture.

inputs from top-down paths (left-to-right in the schematic) to compute the prediction mismatch. The neural dynamics $\dot{\boldsymbol{\phi}}$ is driven by excitatory inputs from $\boldsymbol{\epsilon}_{\mathbf{x}}$ and inhibitory inputs from $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}$. Importantly, all neurons and synapses in this network use only locally available information, and in particular, the learning rules (Eq. 1.12) are Hebbian i.e., they are products of pre- and post-synaptic activities (Hebb, 1949). The nonlinear transformation f can also be naturally implemented in this network (see Whittington and Bogacz (2017) for details).

A characteristic feature of the network implementation above is the separation of value and error neurons, which enables the local computations. Value neurons $\boldsymbol{\phi}$ sends top-down prediction to the sensory level, while error neurons $\boldsymbol{\epsilon}_{\mathbf{x}}$ sends bottom-up error signals that corrects the value neurons to reduce predictions errors over time. There is abundant evidence for the existence of error signals in neocortex (Keller and Mrsic-Flogel, 2018) and other regions such as the hippocampus (Ku et al., 2021). Although the exact neuron-level encoding of prediction errors is less generic and restricted to areas such as dopaminergic reward prediction errors (Schultz, 1998), it is also possible to implement the error signals into dendrites of pyramidal neurons to avoid the error-encoding neurons in predictive coding (Mikulasch et al., 2023). This dendritic approach has been shown to match backpropagation in training feedforward

neural networks (Sacramento et al., 2018; Greedy et al., 2022; Whittington and Bogacz, 2019) will be adopted throughout this thesis. In addition, the error neurons in this network implementation can carry both positive and negative values, which can be considered as separate neuron populations signaling "more" and "less", as discussed in (Keller and Mrsic-Flogel, 2018).

Learning covariance matrices The above network implementation did not account for the learning of the covariance matrices $\mathbf{\Sigma}_{\phi}$ and $\mathbf{\Sigma}_{\mathbf{x}}$ (Eqs. 1.13). The matrix inverse poses challenges for implementing the plasticity computation within a neural network: Computationally, inverting a (large) matrix is inefficient and unstable; Biologically, computing one element of an inverted matrix requires accessing all the elements of the original matrix, making the computation non-local and non-Hebbian. Therefore, an external storage mechanism is needed to store all the synaptic weights to update one single synapse. Although Bogacz (2017) has proposed a network that can implement the learning of covariance matrices locally, the implementation requires additional interneurons complicating the circuit. In Chapter 2 of this thesis, I will propose an alternative way of implementation based on a reparameterization of the covariance matrices. For introductory purposes, the remainder of this chapter will focus on the case where both $\mathbf{\Sigma}_{\phi}$ and $\mathbf{\Sigma}_{\mathbf{x}}$ are identity matrices.

Multi-layer predictive coding In the description of predictive coding above, $\mathbf{x}$ denotes the lowest-level sensory input. However, the theory can also be extended to multiple levels $\mathbf{x}^{(l)}, l = 1, ..., L$, to account for the multi-layer structure of the neocortex. If we denote the lowest-level sensory input as $\mathbf{x}^{(1)}$, and the topmost latent cause as $\mathbf{x}^{(L)}$, the free energy of a multi-layer predictive coding model can be derived as:

$$\begin{aligned}\mathcal{F} &= -\log p_{\theta}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(L)}) \\ &= -\log \left[\prod_{l=1}^{L-1} p_{\theta}(\mathbf{x}^{(l)}|\mathbf{x}^{(l+1)})\right] p_{\theta}(\mathbf{x}^{(L)}) \\ &= -\sum_{l=1}^{L-1} \log p_{\theta}(\mathbf{x}^{(l)}|\mathbf{x}^{(l+1)}) - \log p_{\theta}(\mathbf{x}^{(L)})\end{aligned} \tag{1.14}$$

Assuming $\mathbf{x}^{(l)}|\mathbf{x}^{(l+1)} \sim \mathcal{N}(\mathbf{\Theta}^{(l+1)} f(\mathbf{x}^{(l+1)}), \mathbf{I})$ and $\mathbf{x}^{(L)} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I})$, where $\mathbf{I}$ denotes the identity covariance matrix, and $\mathbf{\Theta}^{(l+1)}$ denotes the weights connecting the $l+1$th

layer and the lth layer, the free energy can be further expressed as:

$$\begin{aligned}\mathcal{F} &= \sum_{l=1}^{L-1} \frac{1}{2}\|\mathbf{x}^{(l)} - \boldsymbol{\Theta}^{(l+1)} f(\mathbf{x}^{(l+1)})\|_2^2 + \frac{1}{2}\|\mathbf{x}^{(L)} - \boldsymbol{\mu}\|_2^2 + C \\ &= \frac{1}{2}\sum_{l=1}^{L} \|\boldsymbol{\epsilon}^{(l)}\|_2^2 + C\end{aligned} \tag{1.15}$$

where we define $\boldsymbol{\epsilon}^{(l)} := \mathbf{x}^{(l)} - \boldsymbol{\Theta}^{(l+1)} f(\mathbf{x}^{(l+1)}), \forall l \neq L$ and $\boldsymbol{\epsilon}^{(L)} := \mathbf{x}^{(L)} - \boldsymbol{\mu}$. In essence, the free energy can be viewed as the sum of squared prediction errors at each layer, where the predictions are parameterized by first applying a nonlinear transformation on the higher layer's activity, and then applying a linear transformation by $\boldsymbol{\Theta}^{(l+1)}$. The objective of a predictive coding model is to minimize the prediction errors at all layers. This objective is achieved by minimizing $\mathcal{F}$ with respect to both $\mathbf{x}^{(l)}$ (neural activities) and $\theta = \{\boldsymbol{\Theta}^{(l)}, \boldsymbol{\mu}\}$, following gradient descent:

$$\dot{\mathbf{x}}^{(l)} \propto -\frac{\partial \mathcal{F}}{\partial \mathbf{x}^{(l)}} = -\boldsymbol{\epsilon}^{(l)} + f'(\mathbf{x}^{(\boldsymbol{l})}) \odot (\boldsymbol{\Theta}^{(l)})^\top \boldsymbol{\epsilon}^{(l-1)} \tag{1.16}$$

$$\Delta\boldsymbol{\Theta}^{(l)} \propto -\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Theta}^{(l)}} = \boldsymbol{\epsilon}^{(l-1)} f(\mathbf{x}^{(l)})^\top; \quad \Delta\boldsymbol{\mu} \propto -\frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}} = \boldsymbol{\epsilon}^{(L)} \tag{1.17}$$

Similar to the two-layer case, these computations can also be implemented in a network with local dynamics and Hebbian plasticity, shown in Fig. 1.3B.

Modeling the brain with predictive coding Predictive coding has been primarily used to model the visual system. For example, Srinivasan et al. (1982) argued that predictive coding provides a natural explanation for inhibition in the retina. In their seminal work, Rao and Ballard (1999) modeled the extra-classical receptive-field effects in the visual cortex using predictive coding, a phenomenon postulated in the theoretical work by Mumford (1992). They also found that the model developed Gabor-like visual filters observed in the brain (Hubel and Wiesel, 1962) following a sparse constraint on the latent activities, similar to what was found with the sparse coding theory (Olshausen and Field, 1996). Using fMRI, Murray et al. (2002) found that grouping individual elements into shapes increased activities in higher visual areas while decreased activities in early areas, providing experimental evidence for the subtraction of top-down signals in lower areas in predictive coding. Predictive coding has also been successfully used to explain V4-prefontal cortex interaction in occluded visual scenes (Choi et al., 2018), repetition suppression (Auksztulewicz and Friston, 2016), bistable perception (Hohwy et al., 2008; Weilnhammer et al., 2017), illusory

motions (Watanabe et al., 2018) and retinal stabilization (Millidge and Shillcock, 2019).

Features from other brain regions may also be explained using the predictive coding theory. For example, in the auditory domain, predictive coding provides a mechanistic explanation for the mismatch negativity phenomenon, where the brain responds to a abnormal auditory event (Garrido et al., 2009). In the domain of motor control, predictive coding has been used to explain the mirror neurons i.e., neurons that fire not only when a motion is executed but also when it is observed (Kilner et al., 2007). The modeling of motor control with predictive coding can also be extended to the modeling of social interactions (Wolpert et al., 2003). A recent theory has suggested that the hippocampo-neocortical interactions can be explained under a predictive coding account, and in particular, predictive coding underlies both prediction and memory in the hippocampus (Barron et al., 2020). This theory was later verified in simulations, where a predictive coding model successfully performed an associative memory task (Salvatori et al., 2021). Many studies have also focused on matching computations of predictive to cortical microcircuits (Bastos et al., 2012; Keller and Mrsic-Flogel, 2018). Although these studies have established behavioral and functional modeling using predictive coding in these other regions, the representational modeling capability of predictive coding has been limited to the visual cortex. This gap will be addressed in this thesis by extending the representational modeling of predictive coding to the medial entorhinal cortex in Chapter 5.

Relationship to machine learning In machine learning and statistics, minimizing the free energy $\mathcal{F} = \mathbb{E}_q[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z}, \mathbf{x})]$ is an important problem as its negative value $-\mathcal{F}$ serves as a lower bound for the evidence $\log p_\theta(\mathbf{x})$, which is ultimately the value that any model θ of the data $\mathbf{x}$ wants to maximize. Historically, many algorithms have been proposed to efficiently perform free energy minimization, such as the incremental EM algorithm (Neal and Hinton, 1998) and the Helmholtz Machine (Dayan et al., 1995). The former algorithm has recently been incorporated into predictive coding for improved efficiency (Salvatori et al., 2022b), although the optimization process still involves solving a dynamical equation iteratively (Eq. 1.11). The latter model uses a wake-sleep algorithm to perform optimization, which, however, does not guarantee minimization of the exact free energy. Modern machine learning has focused on circumventing the computationally expensive inference of $q(\mathbf{z})$ i.e., the E step in the EM algorithm. For example, variational autoencoder (Kingma and Welling, 2013) amortized the inference with a parameterized model

(e.g., a neural network), essentially asking the model to "learn to infer". Such models have achieved great success in generating realistic but unseen images given a large set of training data. Although classical predictive coding lacks such capability due to the delta distribution assumption on $q(\mathbf{z})$, a recent study has successfully extended predictive coding for this task with Monte Carlo sampling (Oliviers et al., 2024).

As described above, predictive coding is a training algorithm for multi-layer neural networks. This connects it to backpropagation (Rumelhart et al., 1986), the training algorithm underlying almost all mainstream deep learning architectures (LeCun et al., 2015). The extraordinary ability of backpropagation to train neural networks once raised great interests in adopting it to model learning in the brain. However, many aspects of the algorithm are unrealistic from a biological perspective (Crick, 1989). Most notably, the weight updates in backpropagation are non-local i.e., to update a single synapse, backpropagation requires access to other weights in the network. On the other hand, predictive coding uses Hebbian plasticity for weight updates. Strikingly, Whittington and Bogacz (2017) showed that, under certain conditions, predictive coding can approximate backpropagation when training a multi-layer neural network. It is then shown that this approximation still holds along arbitrary computational graphs beyond a multi-layer structure (Millidge et al., 2020a), and a version of predictive coding that implements backpropagation exactly was also proposed (Song et al., 2020). These results have provided evidence for a possible implementation of backpropagation in the brain using dynamics of predictive coding. Interestingly, recent studies have also revealed certain machine learning tasks, which are also the ones usually faced by biological organisms, where predictive coding outperforms backpropagation e.g., learning with small training sets (Song et al., 2024).

1.2 Thesis research and overview

Building on the foundation of predictive coding, this thesis explores three relatively unexamined or underdeveloped aspects of the theory, thereby extending its application to broader functions and representations of the brain. These three aspects are *space, time, and memory*. Note that the thesis will not follow this specific order of the three aspects, as certain aspects may be revisited in the discussion of others.

1.2.1 Memory

Theoretical (Barron et al., 2020) and modeling (Salvatori et al., 2021) studies applying predictive coding to the hippocampus and its role in memory have provided initial evidence that predictive coding can effectively model this crucial brain function. However, a key anatomical feature of the hippocampus—its recurrent connectivity—is absent from the current model (Salvatori et al., 2021). Many computational models, such as the Hopfield Network (Hopfield, 1982), capture the recurrent nature of the hippocampal network and rely on these recurrent connections to perform memory tasks. The next part of this thesis (Chapter 2) will explore a recurrent formulation of predictive coding, in which a single layer of neurons performs associative memory by predicting one another.

A natural candidate for recurrent predictive coding is a model incorporating covariance matrices ($\mathbf{\Sigma}$), as proposed in the recurrent network implementations of predictive coding by Friston (2003) and Bogacz (2017), where recurrent connections encode $\mathbf{\Sigma}$. Furthermore, covariance and correlation naturally relate to associative memory (Kohonen, 1972). We will demonstrate that this formulation of recurrent predictive coding can indeed perform associative memory but suffers from biological implausibility and numerical instability due to the matrix inversion required in the covariance learning rule (as discussed earlier). To address these limitations, we will propose an alternative formulation of recurrent predictive coding that is both more biologically plausible and numerically stable. Notably, this revised model is mathematically equivalent to the classical recurrent predictive coding model in associative memory, effectively serving as a reparameterization. Finally, we will establish a connection between recurrent predictive coding and Hopfield Networks, bridging two of the most influential theories of neural computation.

1.2.2 Time

The word "recurrent" is naturally reminiscent of temporal processing. Although the aforementioned "recurrent" predictive coding concerns only a static layer of neurons recurrently connected with each other, Chapter 3 of the thesis will discuss another form of recurrence involving time. To avoid confusion, we will term it as "temporal" predictive coding. Although earlier studies have attempted to extend predictive coding to the temporal and dynamical domain (Rao and Ballard, 1997; Friston et al., 2003; Friston and Kiebel, 2009), none have inherited the simple and plausible implementation of hierarchical predictive coding (Fig. 1.3) while incorporating the neural

components that enable temporal processing. We will present a novel formulation of temporal predictive coding that retains the plausible network implementation. We will demonstrate that this formulation is rooted in the first principle of Bayesian filtering (Stengel, 1986) and can thus be regarded as an approximation of Kalman filters (Kalman, 1960). Mirroring the way that Rao and Ballard (1999) followed to apply predictive coding to natural images, we will also apply temporal predictive coding to natural *movies*. Intriguingly, we found not only Gabor-like edge detectors resembling V1 simple cells (Hubel and Wiesel, 1962), but also motion-detecting neurons (Ohzawa et al., 1996).

It is in Chapter 4 that we will return to memory, although this time we will explore *sequential* memory. In contrast to memories of static patterns, such as images, sequential memories memorize dynamic patterns, such as movies. We will show that temporal predictive coding is perfectly suitable for the sequential memory task. We will provide a mathematical analysis of how temporal predictive coding performs sequential memory, deriving an analytical expression for the retrieved sequential patterns. Importantly, this analytical expression will relate temporal predictive coding to a dynamical extension of Hopfield Networks: Asymmetric Hopfield Network (Sompolinsky and Kanter, 1986), thus further bridging these two influential theories. The expression will also reveal an implicit whitening operation happening in temporal predictive coding, which has been proposed to happen in the brain (Carandini and Heeger, 2012) and gives it advantages over Asymmetric Hopfield Networks in sequential memory. Furthermore, we will go beyond simple memory retrieval and demonstrate the context-learning and generalization capabilities of temporal predictive coding. We will show that temporal predictive coding will develop representations of time steps in its latent activities, essentially signaling "where I am" within a sequential input.

1.2.3 Space

This final part, Chapter 5, of the thesis will focus on an aspect often intertwined with time: space. In particular, it will focus on an intriguing representation of space in the medial entorhinal cortex of the brain: grid cells (Hafting et al., 2005), and try to explain the emergence of grid cells from the lens of predictive coding. With their unique periodic firing pattern over space that forms a hexagonal grid, grid cells serve as a metric system that provides contextual spatial information to the brain. This contrasts them to cells signaling individual locations i.e., place cells (O'Keefe, 1976). Following the earlier discussion of temporal predictive coding's context-learning capability, the

motivation for this line of research is straightforward: If temporal predictive coding can represent temporal context, can it also develop spatial contextual representations like grid cells?

We will show that if temporal predictive coding is trained to perform path integration i.e., estimating one's current location based on velocity inputs, its latent neurons will form hexagonal firing centers over space, resembling grid cells. We will also compare temporal predictive coding with another model of grid cells based on recurrent neural networks (Sorscher et al., 2023). Recurrent neural networks are trained with backpropagation through time (Werbos, 1990), which is implausible due to similar reasons for the implausibility of backpropagation discussed earlier. We will show that temporal predictive coding, which is biologically more plausible, is equivalent to a special case of backpropagation through time. This theory thus extends the relationship between predictive coding and backpropagation (Whittington and Bogacz, 2017) to the temporal learning domain.

Chapter 2

Recurrent Predictive Coding for Associative Memory

2.1 Introduction

Memory systems in the brain often store information about the relationships or associations between objects or concepts. This particular type of memory, referred to as *Associative Memory* (AM), is ubiquitous in our everyday lives. For example, we memorize the smell of a particular brand of perfume, the taste of a kind of coffee, or the voice of different singers we like. After a memory is formed, AM will support its retrieval when a related cue is presented to our senses using the learned association between the provided cues and missing components.

It has long been argued that the hippocampus and adjacent cortical areas, located in the medial temporal lobe of the brain, are crucial for AM (Scoville and Milner, 1957; Squire and Zola-Morgan, 1991; Squire, 1992). Historically, various theoretical and computational works have been developed in an effort to model the hippocampus in AM tasks (Hopfield, 1982; O'Reilly and McClelland, 1994; Treves and Rolls, 1994). In this chapter, we are interested in a particular approach to modelling, which assumes that the hippocampus employs its *predictive activities* to perform AM. Predictive processing is thought to be a key computational principle underlying various hippocampal activities: Experimentally, abundant evidence has suggested that the hippocampus is capable of predicting ongoing sensory inputs (Lisman and Redish, 2009). High-level theories and computational models have also been proposed to explain how predictive coding (PC) may support various properties of the hippocampus, including the formation of cognitive maps (Stachenfeld et al., 2017) and memory (Barron et al., 2020). In particular, Barron et al. (2020) proposed that the hippocampus sits at the top of a hierarchical generative model that generates predictions of

neocortical activities based on past experiences, thus enabling retrieval of activities from memory at lower neocortical levels. The ability of predictive coding networks (PCNs) to complete previously learned patterns have been already demonstrated in pioneering work by Rao (1997), although the model in this work was not formally introduced within the PC framework. More recently, the ability of the hierarchical PCNs introduced by Rao and Ballard (1999) in performing AM has been computationally analysed (Salvatori et al., 2021; Ororbia and Kifer, 2022). Specifically, Salvatori et al. (2021) showed that hierarchical PCNs can store training data points as memories, and can retrieve these memories given partial or noisy cues. Under this PC framework, the memorization of a sensory input is driven by the Hebbian learning dynamics (Hebb, 1949) that minimize the error between the input and a prediction generated by the network. The retrieval of this input is performed by the inferential neural dynamics, also to minimize the error between the internal predictions and the corrupted sensory input. The error signals are represented by prediction error neurons in the network implementation.

This hierarchical PC model provides a possible network mechanism that the hippocampo-neocortical entity employs to support AM, thus offering a possible implementation of the theory of Barron et al. (2020). However, this model only included feedforward and feedback connections between layers of neurons (representing different cortical areas), and thus failed to provide an account for how the *recurrent* hippocampal network performs AM. Furthermore, the absence of recurrent connections in the hierarchical PC models for AM dissociates them from earlier recurrent models of AM, such as the Hopfield Network (Hopfield, 1982). Hopfield Networks assume that the recurrent connections in the hippocampal network learn a covariance matrix representing the associations between individual neurons activated by a memory item. A missing link to Hopfield Networks thus hinders a unified understanding of the computational principles adopted by the hippocampus to support AM. We are thus brought to ask the following two questions: *1. How can the recurrent hippocampal network employ PC to perform AM, in a biologically plausible and computationally stable manner? 2. Can recurrent connections in such a network encode the covariance of neural activity?* Here, by computationally stable, we mean the ability of the model to steadily converge to a fixed point both during learning (memory) and inference (retrieval), and the criteria for biological plausibility include:

1. **Local computation**: A neuron's computation is only dependent on its input neurons and weights connecting itself to its input neurons.

2. **Local plasticity**: The plasticity rule of synapses in a model only depends on quantities encoded by pre- and post-synaptic neurons.

3. **Architectural similarity**: Components of a model resembles architectures of real neurons, such as the recurrent connections and the apical dendrites of the pyramidal neurons in the hippocampus.

The hierarchical PC models for AM are stable and satisfy the first two criteria of plausibility (Salvatori et al., 2021). However, as we pointed out above, they fail to meet the third criterion, architectural similarity, due to the missing recurrent connections. In this chapter, we propose a family of PCNs with **r**ecurrent connections between neurons, which we call rPCNs, as candidate models satisfying these criteria. In particular, we first identify that an earlier type of PCN has already incorporated recurrent synaptic connections encoding the covariance information of inputs (Friston, 2003, 2005; Bogacz, 2017), and can thus be considered as a PC model meeting the criterion of architectural similarity. We refer to it as the *explicit rPCN*, as it encodes the covariance matrix explicitly into its recurrent synapses. The explicit rPCN was originally proposed as a model for learning representations of sensory inputs, and we show in this chapter that its covariance-learning nature can be utilized to perform simple AM tasks. However, we note that the learning rule for the recurrent connections in this model is non-Hebbian, and poses significant computational issues, which makes it fail to satisfy the local plasticity and stability conditions. To address these issues, we propose in this chapter a novel recurrent PCN, which also encodes the covariance matrix via its recurrent connections, but in an implicit manner, thus referred to as *implicit rPCN*. We show that the new implicit model also performs AM via covariance learning, and it is equivalent to the explicit rPCN both theoretically and empirically in simple AM tasks, while only employing local Hebbian plasticity. We also show that the implicit model can be further modified to achieve biological resemblance to the hippocampal pyramidal cells by incorporating a dendritic structure while retaining the theoretical and empirical equivalence to the explicit rPCN at convergence. We name it the *dendritic rPCN* in this chapter. Importantly, we show that both the implicit and dendritic models can perform more complex AM tasks in which the explicit rPCN would fail due to its unstable dynamics. Finally, we propose a *hybrid PCN* that combines the implicit rPCN with a hierarchical PCN (Rao and Ballard, 1999; Salvatori et al., 2021) to model the whole hippocampo-neocortical region, and show that it performs challenging AM tasks efficiently.

In this chapter, we describe a series of more stable and more plausible implementations of recurrent PC to model AM in the hippocampus, to shed light on how the hippocampal *structure* (apical dendrite and recurrent network) may support its *computations* (PC and covariance learning) underlying its *functions* (AM). The key contribution of this chapter is a reparameterization of the (weighted) prediction errors in the explicit rPCN (Friston, 2003), which leads to simplified forms of a free energy. Crucially, the simplified forms of free energy that we consider all share the same minima or fixed points with that of the explicit rPCN, thereby leading to the same inference and learning. Practically, this allows us to drop certain terms from the gradients, leading to more robust convergence to free energy minima and, crucially, affording more biologically plausible implementation.

2.2 Models

In this section, we introduce single-layer covariance-learning PCNs, specifically the explicit, implicit, and dendritic rPCNs, arranged in order of increasing biological plausibility. To perform AM tasks, rPCNs first memorize a set of patterns by optimizing their model parameters through gradient descent to minimize an objective function—specifically, the (variational) free energy (Friston, 2003). After memorization, rPCNs are presented with cues, such as corrupted memories, and retrieve the stored patterns by performing inference (or relaxation) of the neurons to minimize the energy function again. Our focus is on the computational processes and corresponding neural implementations of these models, which are designed to simulate the recurrent networks in the CA3 subfield of the hippocampus. We then extend this framework to describe a complete model of the hippocampo-neocortical system. In this model, implicit or dendritic rPCNs represent the hippocampal recurrent network, while the hierarchical PCN (Salvatori et al., 2021) captures the structure of the neocortical hierarchy.

2.2.1 Explicit rPCN

The explicit rPCN (Friston, 2003, 2005) was proposed to extend the PC model for the visual system (Rao and Ballard, 1999) to a general model of how the brain performs representation learning. Following the probabilistic framework of PC, it introduced the covariance matrix by encoding it explicitly into the network's recurrent connections. We denote the activity of neurons in a single-layer explicit rPCN by a vector $\mathbf{x}$. The model assumes that the set of patterns to memorize $\{\mathbf{x}(i)\}_{i=1}^{N}$ of dimension d

is a set of samples from a Gaussian distribution with true mean $\boldsymbol{\mu}_{\text{true}}$ and covariance matrix $\boldsymbol{\Sigma}_{\text{true}}$, where i indicates the ith sample within the dataset. The dynamics of the network model, parameterized by mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, aim to minimize the following negative log likelihood (or free energy) of observed neural activity $\mathbf{x}$ given this Gaussian distribution:

$$\mathcal{F} = \frac{1}{2}\log|\boldsymbol{\Sigma}| + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \tag{2.1}$$

An explicit rPCN learns its parameters by iteratively setting neural activity to training patterns $\mathbf{x} = \mathbf{x}(i)$ and then modifying parameters by directly computing the derivative of $\mathcal{F}$ with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ (Friston, 2005):

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \Delta\boldsymbol{\mu} = \boldsymbol{\mu} - \alpha \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}}\right|_{\mathbf{x}=\mathbf{x}(i)}, \quad \boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} + \Delta\boldsymbol{\Sigma} = \boldsymbol{\Sigma} - \alpha \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Sigma}}\right|_{\mathbf{x}=\mathbf{x}(i)} \tag{2.2}$$

where α denotes the learning rate for parameters. Notice that this learning rule is fully online as it updates the parameters every time a single training pattern is received. This is closer to learning by biological systems. Here, to derive subsequent analytical properties of the rPCNs, we define the following full-batch learning rules, which can approximate the fully online learning if α is small enough:

$$\Delta\boldsymbol{\mu} = -\alpha \sum_{i=1}^{N} \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}}\right|_{\mathbf{x}=\mathbf{x}(i)} = \alpha \sum_{i=1}^{N} \boldsymbol{\Sigma}^{-1}(\mathbf{x}(i)-\boldsymbol{\mu}), \tag{2.3}$$

$$\Delta\boldsymbol{\Sigma} = -\alpha \sum_{i=1}^{N} \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Sigma}}\right|_{\mathbf{x}=\mathbf{x}(i)} = \alpha \left(-N\boldsymbol{\Sigma}^{-1} + \sum_{i=1}^{N} \boldsymbol{\Sigma}^{-1}(\mathbf{x}(i)-\boldsymbol{\mu})(\mathbf{x}(i)-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}\right) \tag{2.4}$$

With the above full-batch learning rules, both parameters will converge to the maximum likelihood estimate (MLE) of $\boldsymbol{\mu}_{\text{true}}$ and $\boldsymbol{\Sigma}_{\text{true}}$ based on the training data points, i.e., $\boldsymbol{\mu} \to \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}(i)$ and $\boldsymbol{\Sigma} \to \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}(i)-\boldsymbol{\mu})(\mathbf{x}(i)-\boldsymbol{\mu})^\top$. We denote these MLE estimates of mean and covariance by $\bar{\mathbf{x}}$ and S, respectively. Therefore, the parameter matrix $\boldsymbol{\Sigma}$ will *explicitly* encode the sample covariance of the data S, thus the name explicit rPCN. This can be shown by noting that at convergence $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ do not change, so setting $\Delta\boldsymbol{\mu} = 0$ and $\Delta\boldsymbol{\Sigma} = 0$ and solving Eqs 2.3 and 2.4 for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively, gives the above MLE estimates. It is worth noting that although we refer to these estimates MLE, in more general formulations with a latent variable $\mathbf{z}$ they would correspond to maximum a posteriori (MAP) estimates (Friston, 2003; Bogacz, 2017); see also Chapter 1. However, because we have not placed any prior constraints on the generative model implied by Eq 2.1, the MLE and MAP

become equivalent. This single-layer formulation of the free energy also circumvents the need for performing optimization over the neuron activities, as there is no hidden neurons. Therefore, unlike hierarchical predictive coding discussed in Chapter 1 that has a two-step optimization process during training, one for hidden neuron and one for weights, rPCNs discussed in this Chapter only optimize over the weights during training.

After learning, we fix the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and provide a single cue $\tilde{\mathbf{x}}$ to the network, i.e., we initialize the neural activity to $\mathbf{x} = \tilde{\mathbf{x}}$, and the explicit rPCN performs inference on the cue by updating it according to the derivative of the log likelihood:

$$\Delta\mathbf{x} = -\beta\frac{\partial\mathcal{F}}{\partial\mathbf{x}} = \beta\left(-\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.5}$$

where β defines step size along the gradient direction, which we refer to as "integration step". For example, if $\tilde{\mathbf{x}}$ is a corrupted or noisy data point, the equation above will drive it towards the mean in proportion to the precision or inverse variance.

To see how the above equations could be implemented in a neural circuit, it is useful to note that they greatly simplify if we define a vector of prediction errors:

$$\boldsymbol{\epsilon} := \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \tag{2.6}$$

Then, the dynamics of neurons (Eq 2.5) becomes:

$$\Delta\mathbf{x} = -\beta\boldsymbol{\epsilon} \tag{2.7}$$

Moreover, the learning rules for this model (Eqs 2.3 and 2.4) become:

$$\Delta\boldsymbol{\mu} = \alpha\sum_{i=1}^{N}\boldsymbol{\epsilon}(i) \tag{2.8}$$

$$\Delta\boldsymbol{\Sigma} = \alpha\left(-N\boldsymbol{\Sigma}^{-1} + \sum_{i=1}^{N}\boldsymbol{\epsilon}(i)\boldsymbol{\epsilon}(i)^{\top}\right) \tag{2.9}$$

The above neural dynamics (Eqs 2.6-2.9) can be implemented within the network shown in Fig 2.1A. In this network, $\boldsymbol{\epsilon}$ and $\mathbf{x}$ are encoded in activities of neurons, and parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in synaptic weights. The neurons $\mathbf{x}$, called the value neurons, receive inhibition from prediction error neurons (cf. Eq 2.7), while the neurons ϵ receive excitatory inputs from the value neurons, and top-down inhibitory inputs encoding the prior expectation, as well as lateral inhibitory inputs encoding the weight $\boldsymbol{\Sigma}$, to

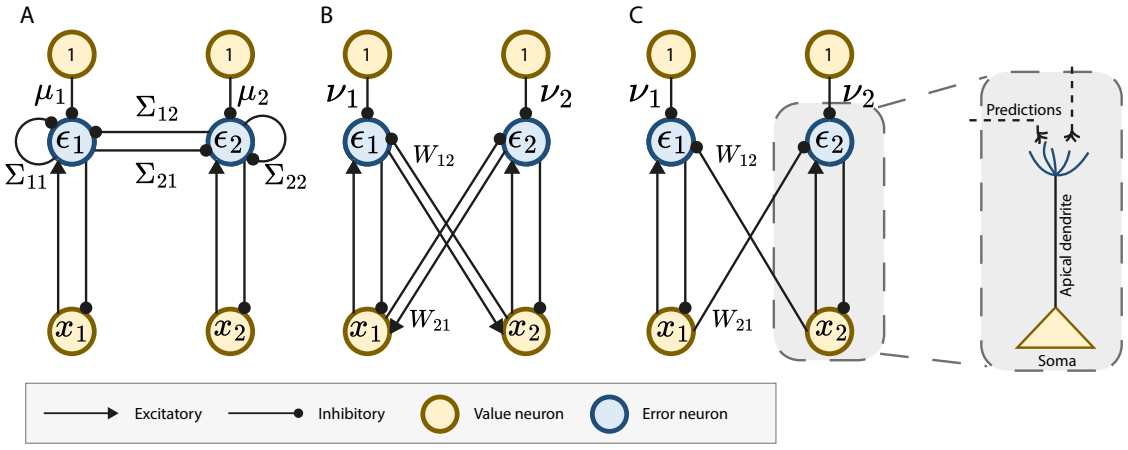

Figure 2.1: **Neural implementations of the single-layer rPCNs using error neurons.** x_a and ϵ_a denotes the a-th value/error neuron in the networks. For simplicity we show the case where the input patterns have only 2 dimensions, corresponding to 2 value and error neurons. A: explicit rPCN originally proposed in (Friston, 2003). B: Implicit rPCN. C: Dendritic rPCN with direct structural mapping to a pyramidal cell. The removal of excitatory connections between ϵ and x from panel B to C corresponds to the removal of the $\mathbf{W}^\top \boldsymbol{\epsilon}$ term from Eq. 2.15 to Eq. 2.18. Unlabeled connections have strengths 1.

compute the weighted prediction error of Eq 2.6 (for details of how this computation arises see (Bogacz, 2017)).

Notice that the learning for the top-down connections encoding the predictions $\boldsymbol{\mu}$ (Eq 2.8) follows the Hebbian rule, as it is the product of the pre-synaptic activity (1) and post-synaptic error activity (ϵ). However, the learning rule for the lateral connections $\boldsymbol{\Sigma}$ (Eq 2.9) is biologically implausible: to compute Σ^{-1}_{ab} needed for the update $\Delta\Sigma_{ab}$ connecting neuron a and b, non-local information from all other entries in $\boldsymbol{\Sigma}$ is needed, due to the nature of the inverse operation. That is, to compute the synaptic update between neuron a and b, synaptic weights from all over the neural circuit are needed. Moreover, as we will show in the Results section, this numerically unstable inverse term poses significant computational problems that make this model inapplicable to complex patterns, such as the image datasets MNIST (LeCun et al., 2010) and CIFAR10 (Krizhevsky et al., 2014), which are not generated by sampling from a Gaussian distribution.

2.2.2 Implicit rPCN

The biological implausibility of explicit rPCN raises the question of whether a more realistic PC model could perform AM through covariance learning. An alternative approach to encoding interactions between neurons, while preserving the predictive nature of the network, is to allow the neurons to predict each other. Guided by this intuition, we propose the implicit rPCN—a recurrently connected, single-layer network characterized by a weight matrix $\mathbf{W}$ and error neurons that encode prediction errors. The implicit model aims to minimize the following energy function:

$$\mathcal{F} = \frac{1}{2}\|\mathbf{x} - \mathbf{W}\mathbf{x} - \boldsymbol{\nu}\|_2^2, \quad s.t. \quad diag(\mathbf{W}) = 0 \tag{2.10}$$

where $\mathbf{x}$ and $\boldsymbol{\nu}$ are both d-dimensional vectors and W is a $d \times d$ 0-diagonal matrix. W_{ab} will be encoded in the synapse connecting the a-th and b-th neurons in the recurrent network. In the implicit model, we define the errors as $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{W}\mathbf{x} - \boldsymbol{\nu}$. The 0-diagonal property of $\mathbf{W}$ implies that the predictions $\mathbf{W}\mathbf{x}$ come from all value neurons in the vector of neurons $\mathbf{x}$ except each neuron itself. Like the explicit model, the implicit rPCN first updates its parameters $\mathbf{W}$ and $\boldsymbol{\nu}$ by computing the derivative of $\mathcal{F}$ given the dataset $\{\mathbf{x}(i)\}_{i=1}^N$ (again, we present the full-batch learning rules here for subsequent analytical derivations):

$$\Delta\boldsymbol{\nu} = -\alpha \sum_{i=1}^{N} \frac{\partial \mathcal{F}}{\partial \boldsymbol{\nu}}\bigg|_{\mathbf{x}=\mathbf{x}(i)} = \alpha \sum_{i=1}^{N} \boldsymbol{\epsilon}(i) \tag{2.11}$$

$$\Delta\mathbf{W} = -\alpha \sum_{i=1}^{N} \frac{\partial \mathcal{F}}{\partial \mathbf{W}}\bigg|_{\mathbf{x}=\mathbf{x}(i)} = \alpha \left(\sum_{i=1}^{N} \boldsymbol{\epsilon}(i)\mathbf{x}(i)^\top\right)_{diag=0} \tag{2.12}$$

where the $()_{diag=0}$ notation means "enforcing the diagonal elements to be 0" as we want to keep the 0 diagonal elements of $\mathbf{W}$ unchanged. Notice that by setting $\Delta\boldsymbol{\nu}$ and $\Delta\mathbf{W}$ to 0, we obtain the following relationships at the convergence of learning:

$$\boldsymbol{\nu} = (\mathbf{I} - \mathbf{W})\bar{\mathbf{x}} \tag{2.13}$$

$$[(\mathbf{I} - \mathbf{W})\mathbf{S}]_{diag=0} = 0 \tag{2.14}$$

Recall that $\bar{\mathbf{x}}$ and $\mathbf{S}$ denote the MLEs of the mean $\boldsymbol{\mu}_{\text{true}}$ and covariance matrix $\boldsymbol{\Sigma}_{\text{true}}$, and that the parameters of the explicit rPCN, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, converge to $\bar{\mathbf{x}}$ and $\mathbf{S}$. Therefore, the implicit rPCN also learns the mean and the elements of the covariance matrix, without encoding them explicitly in individual connections.

Like the explicit model, after learning, the implicit model performs inference after initializing the activity to a cue input $\tilde{\mathbf{x}}$ following the derivative of the objective function with respect to $\mathbf{x}$:

$$\Delta \mathbf{x} = -\beta \frac{\partial \mathcal{F}}{\partial \mathbf{x}} = \beta \left(-\boldsymbol{\epsilon} + \mathbf{W}^\top \boldsymbol{\epsilon} \right) \tag{2.15}$$

The above dynamics can be implemented in the network model in Fig 2.1B, by replacing the lateral connections $\boldsymbol{\Sigma}$ with $\mathbf{W}$ that projects the predictions from all other value neurons into each error neuron. Notice that the learning rule for the bias term $\boldsymbol{\nu}$ is Hebbian, and more importantly, the learning rule for the connections $\mathbf{W}$ is also Hebbian, as it is simply a product $\boldsymbol{\epsilon}\mathbf{x}^\top$ of the pre- and post-synaptic activities. We show in the Results section, that this biologically more plausible implicit model will converge to exactly the same retrieval of a memory as the explicit model, making it a perfect alternative to the implausible explicit model.

2.2.3 Dendritic rPCN

Inspired by the dendritic model in Sacramento et al. (2018) and Mikulasch et al. (2022), we push the biological plausibility of the implicit model further by imposing a "stop-gradient" (sg) operation on the objective function Eq 2.10:

$$\mathcal{F} = \frac{1}{2} \| \mathbf{x} - \mathbf{W} \mathrm{sg}(\mathbf{x}) - \boldsymbol{\nu} \|_2^2 \tag{2.16}$$

where

$$\mathrm{sg}(x) = x; \quad \frac{\partial \mathrm{sg}(x)}{\partial x} = 0 \tag{2.17}$$

Notice that the parameter learning rules in this model is exactly the same as Eqs 2.13 and 2.14. Only the dynamics of the value nodes during inference becomes:

$$\Delta \mathbf{x} = -\beta \boldsymbol{\epsilon} \tag{2.18}$$

The above dynamics differs from the implicit model (Eq 2.15) that it no longer includes $\mathbf{W}^\top \boldsymbol{\epsilon}$ term that corresponded to the backward excitatory connections from the error neurons to the value neurons in Fig 2.1B. This results in the neural network implementation shown in Fig 2.1C. This implementation is simpler, as the error neurons in this model only receive predictions from external sources, thus can be absorbed into the dendrites of the value neurons x, which is shown in the soma-dendrite demonstration in Fig 2.1C. This architecture could be viewed as more biologically plausible because it does not include the special one-to-one connections of strength

1 between value and corresponding error neurons present in Fig 2.1A and 2.1B, for which there is no evidence in cortical circuits. Such an error-encoding dendritic model relates to the model proposed in Sacramento et al. (2018) and Mikulasch et al. (2022). Since this stop-gradient operation does not affect the learning of parameters in the implicit rPCN, and the fixed points of dynamics for the implicit and dendritic models are the same, the equivalence to the explicit model in converged memory retrieval is retained.

2.2.4 Hybrid PCN

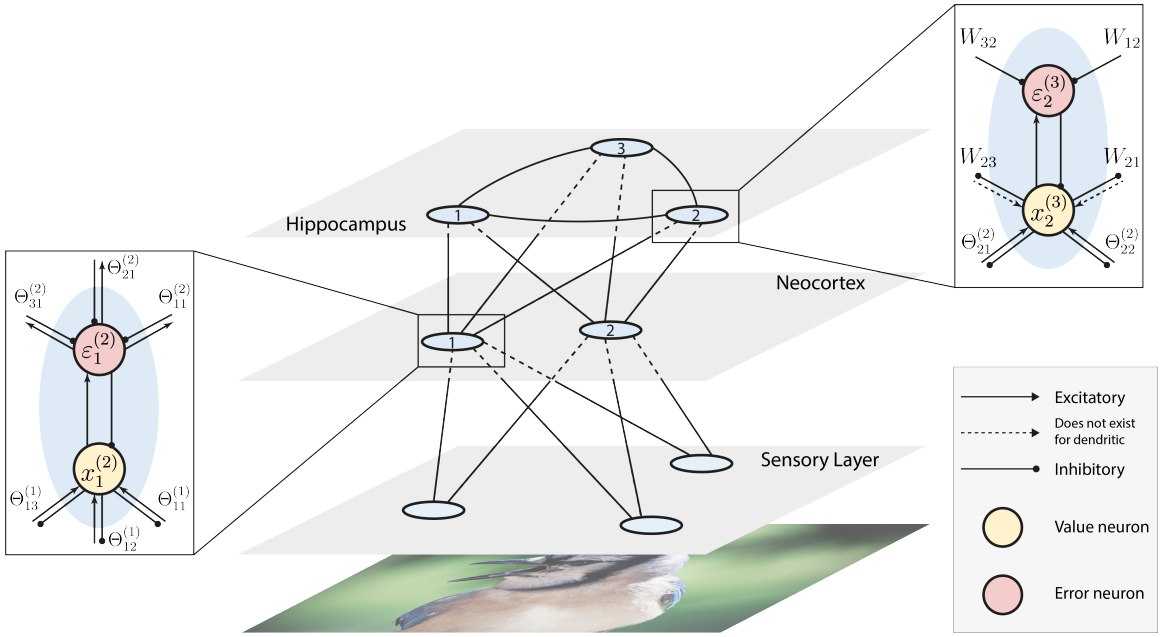

Figure 2.2: **Multilayer hybrid PCN.** We use the single-layer implicit/dendritic rPCN to model the hippocampus, and a hierarchical PCN from Salvatori et al. (2021) to model the sensory cortex and neocortex. For clarity of demonstration, only one layer of our neocortex model is shown. Expanded boxes show the detailed computations within individual neurons and related synapses specified in Eqs 2.20 and 2.22, where $x_a^{(l)}$ denotes the a-th neuron in the lth layer, and Θ_{ab} and W_{ab} denote the individual weights from the ath to the bth neurons. Animal image in this figure is obtained from Wikimedia Commons under a CC BY 4.0 license.

In this section, we propose an architecture that mimics the behavior of the hippocampus as a memory index and generative model. The recurrent, single-layer network models introduced above provide a model for the recurrent dynamics in the hippocampus, specifically the CA3 region that is known to perform pattern completion (O'Reilly and McClelland, 1994). However, raw sensory inputs are first processed

via a hierarchy of sensory and neocortical layers, allowing the hippocampus to memorize representations of raw signals, instead of the signal itself. Moreover, according to a recent theory, the hippocampus functions as a generative model that accumulates prediction errors from sensory and neocortical neurons lower in the hierarchy, and sends descending predictions to the neocortex, to correct the prediction errors in the neocortical neurons. The generative model of the hippocampus is updated until the hippocampal predictions correct the prediction errors by suppressing neocortical activity (Barron et al., 2020). Hierarchical PCNs have already been proposed as a generative model that learn representations of the sensory inputs (Friston, 2005). Particularly, it has been shown that a purely hierarchical PCN, without any recurrent structure, is able to perform associative memory tasks on highly complex datasets (Salvatori et al., 2021). Here, we combine the hierarchical PCN with our proposed recurrent architecture, obtaining an hierarchical model with recurrent dynamics at the topmost layer. What results is an *hybrid* network that models the whole pathway from sensory neurons to hippocampal neurons.

Consider a hierarchical PCN with L layers, with a recurrent implicit or dendritic network connected to the last layer. The first layer corresponds to the sensory layer, where sensory signals are presented to be processed, and the last layer corresponds to the hippocampal layer. Then, the intermediate layers $1 < l < L$ represent the hierarchical structure present in the neocortex that connects the sensory neurons to the hippocampus. We denote the synaptic weight matrix connecting the neurons in the lth layer and the neurons in the $(l+1)$th layer as $\mathbf{\Theta}^{(l)}$. Within the hippocampal layer L, we use $\mathbf{W}$ to denote the recurrent synaptic weight matrix. The vector of value nodes in each layer, denoted as $\mathbf{x}^{(l)}$, is coupled with the error node vector $\boldsymbol{\epsilon}^{(l)}$, computed as the following prediction error:

$$\boldsymbol{\epsilon}^{(l)} = \mathbf{x}^{(l)} - \boldsymbol{\rho}^{(l)} \tag{2.19}$$

where $\boldsymbol{\rho}^{(l)}$ denotes the descending prediction signals into the neurons, computed as:

$$\boldsymbol{\rho}^{(l)} = \begin{cases} \mathbf{W}\mathbf{x}^{(l)}, & \text{if } l = L \text{ and implicit} \\ \mathbf{W}\text{sg}(\mathbf{x}^{(l)}), & \text{if } l = L \text{ and dendritic} \\ \mathbf{\Theta}^{(l)} f\left(\mathbf{x}^{(l+1)}\right), & \text{otherwise} \end{cases} \tag{2.20}$$

where $f(\cdot)$ is a nonlinear function. For simplicity of illustration, we ignore the top-down connections $\boldsymbol{\nu}$ in rPCNs. The aim of the learning dynamics of this hybrid network is to minimize the sum of squared errors:

$$\mathcal{F} = \frac{1}{2} \sum_{l=1}^{L} \|\boldsymbol{\epsilon}^{(l)}\|_2^2 \tag{2.21}$$

Like purely hierarchical PCNs, the learning of the hybrid PCN consists of two stages: *inference* and *weight update*. During inference, the value neurons relax to maximize $\mathcal{F}$ following the gradient descent rule:

$$\Delta \mathbf{x}^{(l)} = \begin{cases} \beta \left(-\boldsymbol{\epsilon}^{(l)} + f'\left(\mathbf{x}^{(l)}\right) \odot \left(\boldsymbol{\Theta}^{(l-1)}\right)^{\top} \boldsymbol{\epsilon}^{(l-1)} + \gamma \mathbf{W}^{\top} \boldsymbol{\epsilon}^{(l)} \right), & \text{if } l = L \\ \beta \left(-\boldsymbol{\epsilon}^{(l)} + f'\left(\mathbf{x}^{(l)}\right) \odot \left(\boldsymbol{\Theta}^{(l-1)}\right)^{\top} \boldsymbol{\epsilon}^{(l-1)} \right), & \text{if } 1 < l < L \\ \mathbf{0}, & \text{if } l{=}1 \end{cases} \tag{2.22}$$

where $\odot$ denotes the element-wise product between two vectors, and $\gamma = 1$ if the topmost layer is an implicit rPCN, or 0 if it is a dendritic rPCN. The update at the sensory layer ($l = 1$) is 0 because during learning, the sensory neurons are fixed at the raw sensory inputs. The inference dynamics are carried out for a certain number of iterations until the value nodes reach an equilibrium. The weight update is then performed, also to maximize $\mathcal{F}$ using gradient descent:

$$\Delta \boldsymbol{\Theta}^{(l)} = \alpha \boldsymbol{\epsilon}^{(l)} f\left(\mathbf{x}^{(l+1)}\right)^{\top} \quad \text{and} \quad \Delta \mathbf{W} = \alpha \boldsymbol{\epsilon}^{(L)} \left(\mathbf{x}^{(L)}\right)^{\top} \tag{2.23}$$

In each iteration of learning, the inference will be performed for multiple iterations and then the weight update will take place for one step. The whole learning process will be iterated multiple times until $\mathcal{F}$ reaches the minimum.

The proposed architecture also models the reconstructions of past memories: as a memory index, the hippocampus sends top-down information to the neocortical neurons to reinstate activity patterns that replicate previous sensory experience (Barron et al., 2020). In the reconstruction phase of our generative model, the hippocampal layer provides descending inputs to the sensory neurons to generate stored data points. Assume we have a hybrid PCN already trained on a dataset of memories. To retrieve a stored datapoint, the synaptic weights $\boldsymbol{\Theta}$ and $\mathbf{W}$ are fixed, and the retrieval process is triggered by providing a partly corrupted version of a memorized pattern to a subset sensory neurons. During retrieval all other value nodes will relax following the same rule specified in Eq 2.22. The only difference is that, the value nodes in the sensory layer will also experience relaxation:

$$\Delta \mathbf{x}_c^{(1)} = -\beta \boldsymbol{\epsilon}_c^{(1)} \tag{2.24}$$

where the subscript c denote the corrupted dimensions of the input pattern, as we keep the intact part of the patterns unchanged during retrieval. Notice that all the dynamics above requires only local computations. Fig 2.2 shows the general structure of the hybrid PCN, as well as detailed implementations of neural computations.

2.3 Results

2.3.1 Explicit rPCN performs associative memory

The explicit rPCN was originally proposed as a generative model of how cortical responses are evoked given observations (Friston, 2003, 2005). To our knowledge, it has never been employed to perform AM tasks. To verify our hypothesis that, after training, the explicit rPCN can perform AM using its inferential dynamics and learned covariance, we designed a simple task where multiple 5×5 random Gaussian patterns are memorized. We then covered the bottom two rows of the 5×5 patterns with 0s and run inference on these corrupted entries only. Examples of the original patterns, along with the corrupted and retrieved patterns, can be found in Fig 2.3A. In this task, the models can be considered as memorizing the association between the top 3 rows and bottom 2 rows of the random patterns. As can be seen in the third column of panel A, the single-layered explicit rPCN can retrieve the covered part of the original pattern well: it performs AM. We then measured the MSEs between the retrieved and original 5×5 random patterns by these models, and plotted them as a function of corrupted/masked pixels in Fig 2.3B (Fig 2.3A shows examples when there are 10 masked pixels). As can be seen, the explicit rPCN has slightly higher retrieval MSEs on average, although this performance gap is not visually observable from the retrieved patterns (e.g., Fig 2.3A).

We now show that a theoretical result can be derived to analytically describe the pattern retrieved by an explicit rPCN. We will use this result later to establish equivalence between explicit and implicit rPCNs. Consider a corrupted pattern $\mathbf{x} \in \mathbb{R}^d$, which can be divided into the corrupted part $\mathbf{x}_k \in \mathbb{R}^k$ and "intact" part $\mathbf{x}_m \in \mathbb{R}^m$, where $k + m = d$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_m \end{bmatrix} \tag{2.25}$$

At the time of retrieval, the training of the rPCN has already finished, during which the network has memorized the association between $\mathbf{x}_k$ and $\mathbf{x}_m$. We assume that the training has converged, so that the connections $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ explicitly encode the MLE, $\bar{\mathbf{x}}$ and $\mathbf{S}$, of the mean and covariance respectively. In this case, the inference dynamics of the explicit model follows $\Delta\mathbf{x} = \beta\mathbf{S}^{-1}(\mathbf{x} - \bar{\mathbf{x}})$ (Eq 2.5). To correspond to the division of the corrupted and intact parts of $\mathbf{x}$, we also divide $\mathbf{S}$ and $\bar{\mathbf{x}}$ into blocks and set $\Delta\mathbf{x} = 0$ to study the convergence of the inference stage:

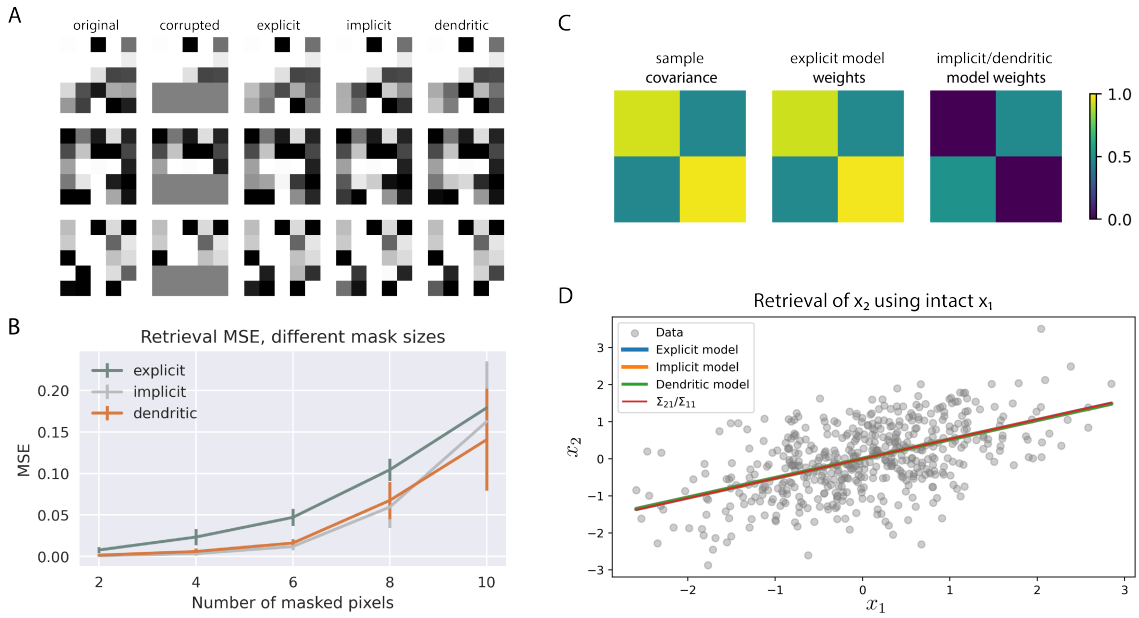

Figure 2.3: **Performance of rPCNs in AM of random patterns, and the equivalence between them.** A: A subset of 5×5 random patterns memorized by all 3 models. After training, we corrupted the bottom 2 rows (10 pixels) and let the networks run inference on the corrupted parts for retrieval. B: Retrieval MSEs of the models when corrupted with different mask sizes. Experiments in A and B are performed with networks with $d = 25$ neurons. C: Sample covariance of a random 2-dimensional dataset and the learned weight matrices of an explicit model and an implicit/dendritic model on this dataset. D: The random 2-dimensional dataset to memorize, and the linear retrieval obtained by masking the second dimension x_2 by all 3 models, as well as the theoretical retrieval line. All the lines overlap as they are equivalent in theory. Experiments in C and D are performed with networks with $d = 2$ neurons.

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{kk} & \mathbf{S}_{km} \\ \mathbf{S}_{mk} & \mathbf{S}_{mm} \end{bmatrix}, \quad \bar{\mathbf{x}} = \begin{bmatrix} \bar{\mathbf{x}}_k \\ \bar{\mathbf{x}}_m \end{bmatrix} \tag{2.26}$$

where $\mathbf{S}_{pq}$, $p, q \in \{k, m\}$ denotes the $p \times q$ submatrices of the covariance matrix $\mathbf{S}$. Using Eq 2.26 above, we can have the following theorem:

Theorem 1 *After training, an explicit rPCN retrieves the corrupted* $\mathbf{x}_k$ *using the intact* $\mathbf{x}_m$ *following the dynamics* $\Delta\mathbf{x} = \beta\mathbf{S}^{-1}(\mathbf{x} - \bar{\mathbf{x}})$*, and the retrieval dynamics on* $\mathbf{x}_k$ *converge to:*

$$\hat{\mathbf{x}}_k = \mathbf{S}_{km}\mathbf{S}_{mm}^{-1}(\mathbf{x}_m - \bar{\mathbf{x}}_m) + \bar{\mathbf{x}}_k \tag{2.27}$$

where $\hat{\mathbf{x}}$ denotes the retrieval of the corrupted part. This theorem describes the activities in the network after retrieval given by the explicit rPCN, using the learned parameters and the intact $\mathbf{x}_m$. Details of the derivation can be found in Appendix A.

2.3.2 Explicit and implicit rPCNs retrieve equivalent memories

Having established the theoretical foundation of the retrieval dynamics in the explicit rPCN, we next consider the theoretical aspects of the implicit and dendritic models: what is the retrieval of memorized patterns given by these models, based on learned associations? Again consider the case of a vector $\mathbf{x}$ consisting of the top k corrupted entries $\mathbf{x}_k$ and bottom m intact entries $\mathbf{x}_m$, and assume that the learning has converged when the corrupted $\mathbf{x}$ is presented to the network, which gives the conditions in Eq 2.13 and 2.14 for both implicit and dendritic models (notice that these two models do not differ in parameter learning). We now write the model parameters $\boldsymbol{\nu}$ and W into block matrices:

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\nu}_k \\ \boldsymbol{\nu}_m \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_{kk} & \mathbf{W}_{km} \\ \mathbf{W}_{mk} & \mathbf{W}_{mm} \end{bmatrix} \tag{2.28}$$

where $\mathbf{W}_{pq},\ p, q \in \{k, m\}$ denotes the $p \times q$ submatrices of the weight matrix $\mathbf{W}$. Notice that for both implicit and dendritic models, the retrieval dynamics converge if and only if all error nodes become zero, that is, $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{W}\mathbf{x} - \boldsymbol{\nu} = 0$. For the dendritic model this is obvious (Eq 2.18), whereas for the implicit model this comes from the fact that W has all its diagonal entries equal to 0 and thus cannot be an identity matrix (Eq 2.15). We can thus derive the following theorem on the converged retrieval dynamics of the implicit and dendritic rPCNs:

Theorem 2 *After training, both the implicit and dendritic models retrieve the corrupted* $\mathbf{x}_k$ *given intact* $\mathbf{x}_m$ *following the dynamics specified in Eqs 2.15 and 2.18, which converge to the following equilibrium:*

$$\mathbf{W}_{km}(\mathbf{x}_m - \bar{\mathbf{x}}_m) = (\mathbf{I}_k - \mathbf{W}_{kk})(\mathbf{x}_k - \bar{\mathbf{x}}_k) \tag{2.29}$$

where $\mathbf{I}_k$ *is the* $k \times k$ *identity matrix. Using Eq 2.14, this equation is equivalent to:*

$$\hat{\mathbf{x}}_k = \mathbf{S}_{km}\mathbf{S}_{mm}^{-1}(\mathbf{x}_m - \bar{\mathbf{x}}_m) + \bar{\mathbf{x}}_k \tag{2.30}$$

The details of the derivations can be found in Appendix A. Notice that the equilibrium condition for implicit and dendritic rPCNs is exactly the same as that for the explicit rPCN, specified in Eq 2.27. This equivalence was also verified empirically in a simple 2-dimensional example shown in Fig 2.3D. In this example both $\mathbf{x}_k$ and $\mathbf{x}_m$ are single-dimensional scalars, and all three models retrieved the corrupted x_2 following the same linear equation (Eq 2.27). Notice that this line is also the least squares regression line, where x_2 is the dependent variable. This also implies that the memory "attractors" learned by the model in 2-dimension is a line, instead of individual points, which is different from the attractors learned by classical models such as Hopfield Networks (Hopfield, 1982). In the final section of the Results, we provide an empirical investigation into this difference. Fig 2.3C also shows that the weight matrix $\mathbf{\Sigma}$ of the explicit rPCN directly encodes the sample covariance matrix, whereas the other two models encode it implicitly. Due to their equivalence, the implicit and dendritic models also perform AM on the random Gaussian patterns in Fig 2.3A.

At this point, we have established the complete equivalence between the explicit and implicit/dendritic rPCNs, showing that the latter models can serve as perfect substitutes for the explicit model. However, in contrast to the explicit rPCN, where the learning rule (Eq 2.4) for the connections $\mathbf{\Sigma}$ employs non-local information, the plasticity rules (Eq 2.12) for the implicit and dendritic models are entirely Hebbian. Moreover, as was shown above, the neural implementation of a dendritic rPCN can be mapped to the structure of hippocampal pyramidal cells. Therefore, our proposed models are equivalent to but biologically more plausible than the explicit rPCN.

2.3.3 Relationship between rPCNs and Hopfield Networks

The ability of explicit and implicit rPCNs to retrieve patterns, as shown in Fig. 2.3, is a well-known property of Hopfield Networks (Hopfield, 1982) when the patterns consist of binary entries. Moreover, we have also demonstrated that, like Hopfield Networks, the implicit rPCN also learns the covariance. What is the exact relationship between these models? The following theorem demonstrates that both types of models perform *metric learning* during the process of parameter optimization:

Theorem 3 *Consider a collection of memorized patterns* $\mathbf{X} = \{\mathbf{x}(i)\}_{i=1}^N$ *with zero mean. The energy functions of both rPCNs and Hopfield Networks can be expressed as a metric learning function with a linear embedding transformation:*

$$d^2_{\mathbf{L},\mathbf{X}}(\mathbf{q}) := c\|\mathbf{L}(\mathbf{X})\mathbf{q}\|_2^2 \tag{2.31}$$

where $\mathbf{q}$ *is a query presented to the model after training (e.g., a corrupted version of a memorized pattern),* c *is a model-specific scalar constant, and* $\mathbf{L}(\mathbf{X})$ *is a model specific matrix.*

The c and $\mathbf{L}$ parameters for rPCNs and Hopfield Network are summarized in the following table:

Model	c	$\mathbf{L}(\mathbf{X})$
Hopfield Network	$-1/2$	$\mathbf{S}^{\frac{1}{2}}$
Explicit rPCN	$1/2$	$\mathbf{S}^{-\frac{1}{2}}$
Implicit rPCN	$1/2$	$\text{diagMat}(\mathbf{1} \oslash \text{diag}(\mathbf{S}^{-1}))\mathbf{S}^{-1}$

Table 2.1: AM as metric learning.

where $\mathbf{S}$ is the covariance matrix of the patterns, $\oslash$ is the Hadamard (element-wise) division, $\mathbf{1}$ denotes the 1-vector, diag extracts the diagonal elements of a matrix and converts them into a d-dimensional vector, and diagMat converts a vector into a diagonal matrix. The proof of these results can be found in the Appendix A. In particular, for implicit rPCN, the individual entries of the parameter $\mathbf{L}(\mathbf{X})$ follows:

$$L_{ij}(\mathbf{X}) = (\text{diagMat}(\mathbf{1} \oslash \text{diag}(\mathbf{S}^{-1}))\mathbf{S}^{-1})_{ij} = \frac{(\mathbf{S}^{-1})_{ij}}{(\mathbf{S}^{-1})_{ii}}, \tag{2.32}$$

i.e. entries of the inverse of $\mathbf{S}$ weighted by its own diagonal entries.

This theorem establishes a formal relationship between Hopfield Networks and rPCNs, showing that both models use the covariance matrix to implement AM. Empirical tests further illustrate this shared mechanism. To evaluate the models' ability to memorize and retrieve binary patterns effectively, we generated randomly two 5-dimensional binary patterns $\{-1, 1\}^5$ for memorization. We then corrupted these patterns by masking the last few dimensions with zeros and examine the retrieval performance of the models. Fig 2.4A displays an example of an original pattern, a corrupted version with the last two dimensions masked, and the retrievals produced by a Hopfield Network, implicit rPCN, and explicit rPCN. All three models perform identically in this case. However, as the number of corrupted dimensions increases, Hopfield Networks exhibit higher retrieval mean squared errors (MSEs) compared to rPCNs (Fig 2.4B). Notably, the two rPCNs have identical performances across different numbers of masked dimensions, due to their theoretical equivalence and the simplicity of the patterns. We then repeat the same experiments on binary MNIST images of dimension 28×28, where we masked the bottom half of the images and query the models to retrieve the original images. Fig 2.4C presents two examples of

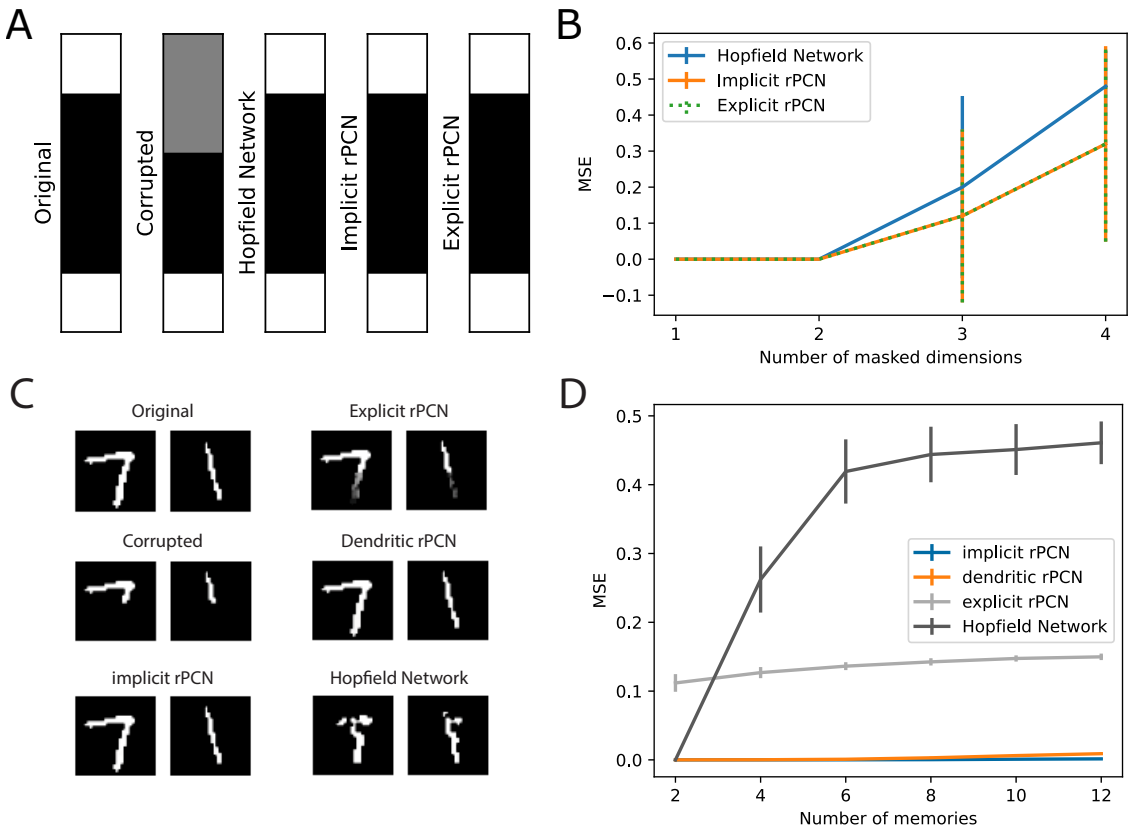

Figure 2.4: **Comparison of the Hopfield Network and rPCNs in binary AM task.** A: Example binary pattern and its corruption and retrievals. B: Retrieval MSE of Hopfield Network and rPCNs across different numbers of masked dimensions. C: Example binary MNIST images and retrievals by different models. D: Retrieval MSEs as a function of number of memorized binary MNIST images.

the retrievals when the total number of memories is 4. Implicit and dendritic rPCNs are successful in retrieving the binary memories, whereas the explicit rPCN can only retrieve blurry versions of the memory. On the other hand, Hopfield Network fails in this more challenging task, only able to retrieve noisy versions of the memories. We systematically studied the retrieval MSEs of these models in Fig 2.4D. As the number of memories is small, Hopfield Network can almost retrieve the memories perfectly, although the performance deteriorates quickly as the number of images increases. For rPCNs, the implicit and dendritic models have stable and good performance across all memory sizes, whereas the explicit rPCN have higher MSEs. As we will also demonstrate in the next section with continuous memories, this performance difference is due to the numerical instability in the dynamics of explicit rPCN. Also note that binary patterns are used here because the classical Hopfield Network can

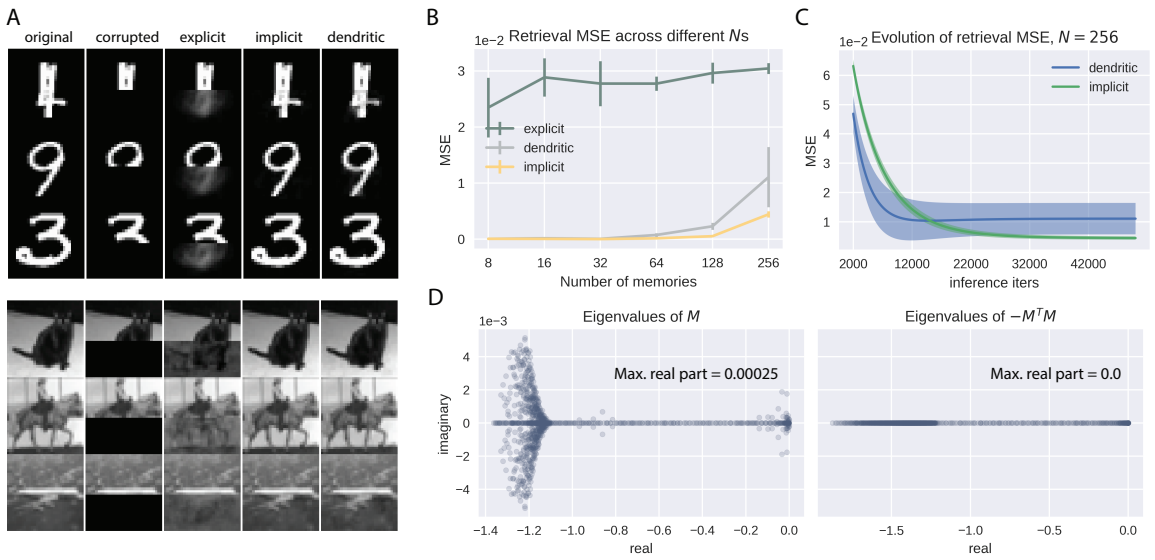

Figure 2.5: **Performance of the single-layer rPCNs in AM of structured images.** A: Examples of retrieved MNIST (top) (LeCun et al., 2010) and grayscale CIFAR10 (bottom)(Krizhevsky et al., 2014) images by explicit, implicit and dendritic models. All models here are trained to memorize 64 images. For MNIST, the networks have $d = 784$ neurons; for grayscale CIFAR10, $d = 1024$. B: Retrieval mean squared errors (MSEs) of the single-layer models across multiple numbers of training memories (N). C: Evolution of the retrieval MSEs of the implicit and dendritic models when $N = 256$. D: Example eigenspectra of the weight matrices defining the inferential dynamics for the dendritic (left) and implicit (right) rPCNs. Error bars obtained by 5 different seeds for image sampling. Please see main text for an explanation of the matrix $\mathbf{M}$.

only process such patterns. To memorize patterns with continuous values, modern variations of Hopfield Networks (Ramsauer et al., 2020) are needed. On the other hand, rPCNs can work with both continuously and discretely distributed patterns, without modifying any of their dynamics and architectures.

2.3.4 Model performance in AM with structured image data

In each learning iteration, the explicit rPCN needs to compute the inverse of the current weight matrix. In practice, this works well when the underlying dataset has some specific regularities, but becomes problematic when dealing with structured data, such as images of handwritten digits and natural objects. In this section, we show that the explicit model can no longer perform stable memorization and retrieval of more complex and structured data, whereas the implicit and dendritic models are able to perform AM on such data with a high level of precision, and

are hence interesting from an application perspective. To do that, we replicate the pattern completion experiment performed in Fig 2.3, but using images sampled from the MNIST (LeCun et al., 2010) and grayscale CIFAR10 (Krizhevsky et al., 2014) datasets. Fig 2.5A shows some retrieved examples from both datasets by all three types of single-layer rPCNs, which were trained to memorize 64 images. The visual results suggest that explicit rPCN can no longer retrieve clear images, whereas both implicit and dendritic models can obtain visually perfect retrievals of the memories. The failure of the explicit rPCN with these structured images is due to the need to compute the inverse of $\mathbf{\Sigma}$ in each iteration (Eq 2.4): for low-dimensional patterns with some specific regularities such as Gaussian distribution, the inverse can be precisely computed. However, when encountered with high-dimensional data such as structured images, the inverse computation can become imprecise, and the computational errors may accumulate when the model is trained iteratively, leading to blurry retrievals.

We next perform a quantitative analysis of the performance of these models, by measuring the mean squared errors (MSEs) between the original and retrieved grayscale CIFAR10 images across different numbers of images we train the models to memorize (N). Fig 2.5B shows that the explicit model indeed has much larger retrieval MSEs than the implicit and dendritic model, confirming our visual observations. The MSE gap is also larger than that in Fig 2.3A, where the random Gaussian patterns makes the learning of the explicit model more stable when inverting $\mathbf{\Sigma}$.

Interestingly, Fig 2.5B also shows that despite the equivalence between the implicit and dendritic models when their inferential dynamics (Eqs 2.15, 2.18) have converged, the dendritic model has larger retrieval MSEs than the implicit model, especially when N is large. We investigate this phenomenon by plotting the evolution of the retrieval MSEs of both implicit and dendritic models during inference, when $N = 256$. As can be seen in Fig 2.5C, the MSE of the dendritic model fails to decrease to the same (lower) value as that of the implicit model, and it also fails to converge. To understand the distinction between the implicit and the dendritic rPCNs in inference, recall that their inferential dynamics are described by two linear differential equations (Eqs 2.15, 2.18). We now define:

$$\mathbf{M} = \mathbf{W} - \mathbf{I} \tag{2.33}$$

where $\mathbf{W}$ is the learned weight matrix in the implicit/dendritic models (note that the learning of these two models are identical) and $\mathbf{I}$ is the identity matrix of the same size as $\mathbf{W}$. We can therefore rewrite the inferential dynamics as:

$$\Delta\mathbf{x}_{\text{im}} = \beta(-\mathbf{M}^\top\mathbf{M}\mathbf{x}), \quad \Delta\mathbf{x}_{\text{den}} = \beta(\mathbf{M}\mathbf{x}) \tag{2.34}$$

where "im" stands for implicit, and "den" stands for dendritic. Notice that in general, the stability of a linear dynamical system $\Delta \mathbf{x} = \mathbf{A}\mathbf{x}$ for some matrix $\mathbf{A}$ is determined by the eigenvalues of $\mathbf{A}$: the system is stable if and only if the real part of all eigenvalues of $\mathbf{A}$ are non-positive (Arnold, 1992). As we show in Fig 2.5D, the largest eigenvalue of $-\mathbf{M}^\top\mathbf{M}$ is 0, suggesting stable inferential dynamics for the implicit rPCN. This is also generally true as $-\mathbf{M}^\top\mathbf{M}$ defines a negative (semi-)definite matrix for any $\mathbf{M} \neq 0$, which has all eigenvalues smaller than or equal to 0. On the other hand, Fig 2.5D shows that the largest eigenvalue of $\mathbf{M}$ is greater than 0, indicating unstable dynamics for the dendritic model. Thus, although the implicit and dendritic models have the same fixed point of their dynamics during retrieval, the neural dynamics defined by the dendritic model may become unstable, preventing itself from the convergence. For the dendritic model there are no theoretical reasons that guarantee negative eigenvalues of $\mathbf{M}$, so they may become positive as the values in that matrix grow with learning more patterns. The experiments with MNIST (LeCun et al., 2010) yielded similar results, and we describe them in S1 Fig.

2.3.5 Performance of hybrid models with natural images

While performing well on small subsets of structured data, the purely recurrent implicit/dendritic models present some problems: first, the recurrent structure only takes into account the hippocampal structure, but does not correctly represent the hierarchical structure that connects it to the neocortex; second, the number of parameters is quadratic to the dimension of the inputs, which does not allow it to work on high-dimensional images, such as high-quality images. For example, consider a recurrent network trained on 224×224 pictures of the ImageNet dataset (Deng et al., 2009). Every image consists of ~ 150000 pixels, and hence needs a network with the same number of neurons. A recurrent network of this size would be impossible to train without an exceptional amount of computational power, as it has ~ 40 billion parameters. On the other hand, a hierarchical structure that precedes the hidden implicit layer guarantees the flexibility of choosing the number of parameters. For example, a network with 7 hidden layers, followed by a recurrent implicit layer, all of dimension 1024, would be more than 200 times smaller than the above implicit model for ImageNet, and hence feasible to be trained. In fact, we have successfully trained this model on high-dimensional ImageNet pictures. Particularly, we trained this 7-layer network with 100 samples from the ImageNet dataset. We then defined successful retrievals as those with retrieval MSEs smaller than $5e^{-3}$, and found that

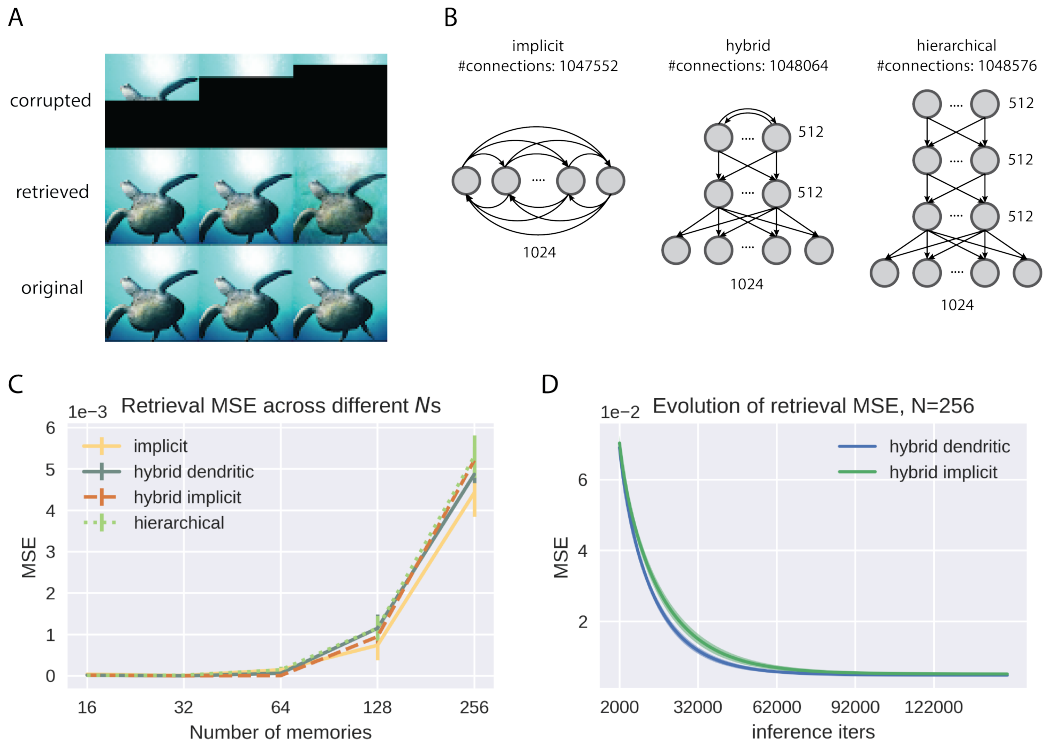

Figure 2.6: **Performance of the multi-layer models.** A: Example performance on 100 images of the ImageNet dataset by a 7-layer hybrid model with 1024 hidden neurons per layer, with an implicit rPCN as the topmost layer. B: Demonstration of how we keep the number of parameters across different models to be roughly the same. B: Retrieval mean squared errors (MSEs) of the multi-layer models across multiple numbers of training memories (N). The curve for the implicit model is the same as the one in Fig 2.5. C: Evolution of the retrieval MSEs of the implicit and dendritic hybrid models when $N = 256$. Error bars obtained by 5 different seeds for image sampling.

when presenting the network a partial cue consisting of $1/2, 1/4$ and $1/8$ of the original pixels, the model has successfully recovered, respectively and on average, 100, 97, and 44 of the original memories. Similar performance was also obtained with the colored CIFAR10 dataset (Krizhevsky et al., 2014) and examples of colored CIFAR10 images can be found in S2 Fig.

We then examine whether the hybrid structure leads to better retrieval performances compared to the single-layer models. To do that, we trained an implicit rPCN and a hybrid model with a topmost implicit rPCN to memorize the same subsets of grayscale CIFAR10 images (1024 pixels in total) of varying numbers of images N, and initialized the retrieval with half-covered partial cues. To compare with earlier works (Salvatori et al., 2021) fairly, we also trained a purely hierarchical PCN

to perform the same AM task. The sizes of the networks are chosen such that they have approximately the same number of parameters. The hybrid model we use has 3 layers: the sensory layer has 1024 neurons corresponding to the pixel space, with 512 neurons in the hidden layer, and 512 neurons in the topmost implicit layer. We construct the hierarchical model by replacing the topmost implicit recurrent layer with 2 hierarchical layers of size 512. We chose such a configuration of hidden sizes and number of layers because in general, an implicit rPCN with d neurons (for d-dimensional inputs) will have $d(d-1)$ parameters. A hybrid model with d sensory neurons, one feedforward hidden layer of size $d/2$ and a topmost implicit layer of size $d/2$ will have approximately the same number of parameters. Replacing the final implicit layer with two feedforward layers of the same size results in a hierarchical PCN with roughly the same number of parameters. This ensures the fairness of comparison across models, and is illustrated in Fig 2.6B, where we also included the number of neurons in each layer used in our experiments next to each layer, as well as the number of connections at the top. As Fig 2.6C shows, the performance of the implicit model is in fact slightly better than the hybrid implicit model and the purely hierarchical model when they have the same number of parameters. Indeed, the retrieval MSE of the implicit model should be the smallest among all linear retrieval functions: notice that the retrieval $\hat{\mathbf{x}}_k$ of the implicit model defined in Eq 2.27 is the least squares solution if we regress $\mathbf{x}_k$ on $\mathbf{x}_m$. Although the exact retrieval function of the hybrid and hierarchical models is unclear due to their nonlinearity, it has been shown that a two-layer linear PCN with 1 neuron in the hidden layer yields linear retrievals following the principal components of the sensory data (Whittington and Bogacz, 2017), shedding light on the slightly worse performance of the multi-layer models. However, we note that the similar performance in retrieval fidelity between the implicit and hybrid models does not make the hierarchical structure in the hybrid model redundant, as the functionality of the brain is far more heterogeneous than merely memorization, and many functions require the hierarchical structure. For example, the hierarchical structure can support the learning of meaningful representations of the sensory inputs, which can be utilized to perform other tasks such as classification, while paying only a negligible cost in memory retrieval tasks.

We further compared the implicit and dendritic models when they are "plugged in" as the topmost recurrent layer in the hybrid model. Fig 2.6C shows that the hybrid dendritic model performs identically to the hybrid implicit model, contrasting their performance difference in the single-layer case. During the inferential iterations, the retrieval MSEs of both models also converge stably to the same value (Fig 2.6D). In

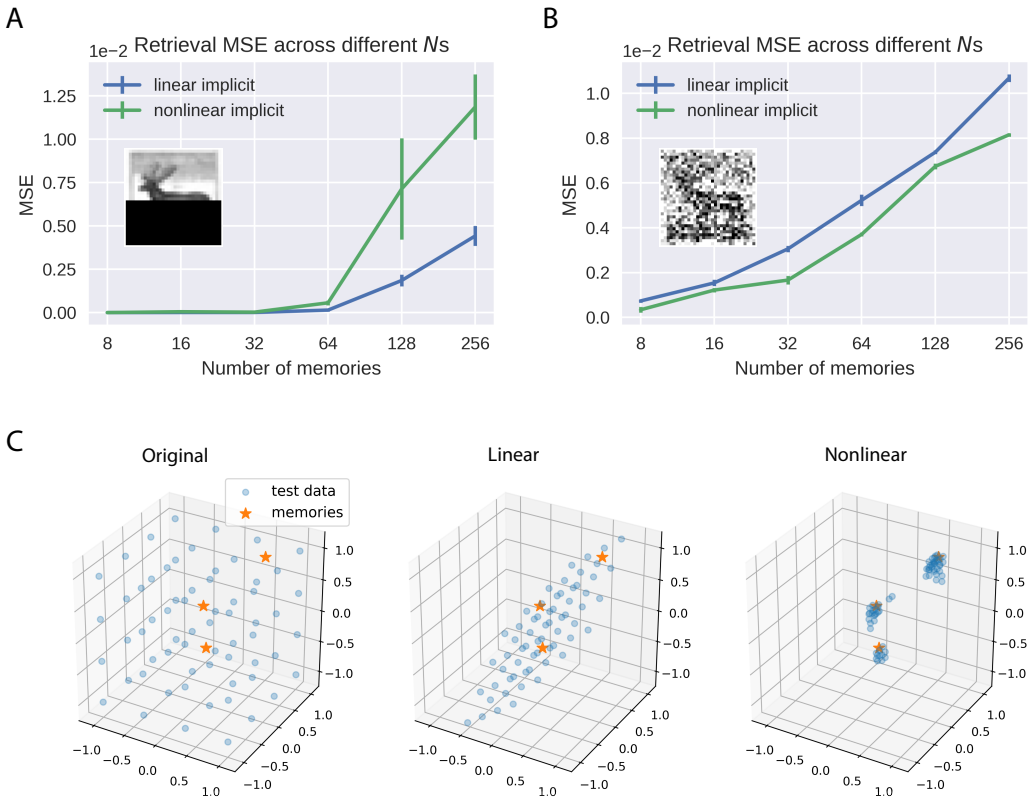

Figure 2.7: **Comparison of linear and nonlinear implicit rPCN.** A: Performance of linear and nonlinear implicit models in the completion task with varying Ns. B: Same as A, but with the denoising task, where cues are memories with Gaussian noise of variance 0.1. C: A simple 3-dimensional example, where stars are data points the networks were trained to memorize. After training we ran inference on both linear and nonlinear models, initialized with grid test data drawn from the range $[-1, 1]^3$. The position of the test data at convergence of inference indicates the shape of attractors. Images taken from the CIFAR10 dataset (Krizhevsky et al., 2014).

other words, the introduction of the neocortical layers remedies the unstable inferential dynamics due to the biologically more plausible dendritic structure in our models. We suspect that the hierarchical pre-processing of the sensory inputs regularizes the covariance matrix, such that the topmost recurrent weight matrix $\mathbf{W}$ defines stable inferential dynamics for the dendritic layer. However, since the top-layer inference is no longer a simple linear dynamical system (Eq 2.22), the connections between a regularized $\mathbf{W}$ and the stability of the dynamics may not be straightforward.

2.3.6 Nonlinear rPCN learns individual attractors

The implicit rPCN we have investigated so far assumes that the relationship between neurons in the recurrent network is linear, i.e., projections into neuron i is $\sum_{j \neq i} W_{ij} x_j$.

Here, we introduce an ad hoc nonlinearity into the definition of the free energy – as motivated by previous heuristics in neural networks and machine learning:

$$\mathcal{F} = \frac{1}{2}\|\mathbf{x} - \mathbf{W}f(\mathbf{x}) - \boldsymbol{\nu}\|_2^2 \tag{2.35}$$

where $f(\cdot)$ is a nonlinear function. One way to understand these heuristics more formally is to appreciate that the free energy is a statement about the underlying generative model that generates inputs (e.g., images) from some latent causes. Although linear rPCNs have provided nice analytical results discussed above, nonlinearities can speak to more expressive generative models similar to the hierarchical and hybrid schemes demonstrated above. The plasticity rules of this model will be the same as Eq 2.11 and Eq 2.12, with the error $\boldsymbol{\epsilon}$ changing to $\mathbf{x} - \mathbf{W}f(\mathbf{x}) - \boldsymbol{\nu}$. The neural dynamics during inference follows:

$$\Delta\mathbf{x} = \beta\frac{\partial\mathcal{F}}{\partial\mathbf{x}} = \beta(-\boldsymbol{\epsilon} + f'(\mathbf{x}) \odot \mathbf{W}^\top\boldsymbol{\epsilon}) \tag{2.36}$$

We first train this nonlinear implicit model to perform the same AM task as before, where the model is trained to memorize different numbers of grayscale CIFAR10 images, and then retrieve these memories from images with covered bottom half. For these experiments with CIFAR10 we used `tanh`() as the nonlinear function $f()$. Fig 2.7A shows that the performance of the nonlinear implicit rPCN is worse than that of the linear model in this completion task, especially when N is large. To examine the reasons for this performance difference, we then corrupt the memories differently by adding Gaussian noise with variance 0.1, and initialize the retrieval dynamics with these noisy memories. Fig 2.7B shows that with this denoising task, the linear and nonlinear implicit rPCNs perform similarly, although the retrieval MSE of the nonlinear model is overall lower. We hypothesize that the difference across different retrieval tasks results from the attractor structures learned by the linear and nonlinear models. Classical nonlinear models, such as the Hopfield Networks (Hopfield, 1982), usually learn individual attractors corresponding to each memorized item. On the other hand, as we have demonstrated in Fig 2.3D, the linear implicit rPCN appears to learn a line attractor in the 2-dimensional space. We generalize this observation by training both linear and nonlinear models to memorize 3 random data points in range $[-1, 1]$ in $\mathbb{R}^3$. After training, we initialize the inference with test data in a grid from the range $[-1, 1]$, and examine where these test data points are attracted to at the convergence of the inferential dynamics. Fig 2.7C shows that the attractor formed by the linear model is a hyperplane, where any point on this plane defines the lowest energy. On the other hand, the test data points all converge to the memories learned

by the nonlinear model, suggesting that the nonlinear model indeed learns individual attractors.

The observation above in turn helps explain the performance difference in completion and denoising tasks between the linear and nonlinear models. Notice that the completion task is inherently more challenging than the denoising task in terms of retrieval, as the corrupted pixels contain no information about the original images. Therefore, in the completion task, individual attractors formed by the nonlinear model will confound with each other more severely than in the denoising task, so that more retrievals will converge into the wrong attractors. The retrieval MSE of the nonlinear model thus comes from these incorrect attractor choices. With the linear model, the retrieved data points are always pulled towards the least squares regression hyperplane, such as the regression line in Fig 2.3. The MSE of the linear model therefore only comes from the least squares prediction, which is the minimum across all linear solutions. Thus, when we average across the whole dataset, a relatively large number of totally wrong retrievals will produce a higher MSE than the retrievals from the least squares predictions. On the other hand, in the less challenging denoising task, the confounding effect between attractors of the nonlinear model will be less severe, yielding a smaller retrieval MSE across the whole dataset than in the completion task. Moreover, the least squares prediction from the linear model is less reliable in the denoising task, as all entries of the data are corrupted, i.e., the independent variables in the regression problem are also changing. Therefore the retrieval MSE of the linear model will increase, resulting in the observation in Fig 2.7A and B.

2.4 Discussion

2.4.1 Summary

This chapter introduces a family of rPCNs for AM tasks, providing a possible mechanism of how the recurrent hippocampal network may perform AM via predictive processing, which also unifies two disparate modelling approaches to the hippocampal network, namely covariance learning and PC. First, we identified that a previously proposed PCN already considered the learning of parameters encoding covariance in an explicit manner (Friston, 2003, 2005), but was never tested for AM tasks. We showed that, both theoretically and empirically, this model is able to perform AM on random patterns. However, this *explicit rPCN* is neither biologically plausible nor numerically stable, due to the inverse term in its learning rule. We address both limitations by proposing a model we call *implicit rPCN*, which also learns the covariance

matrix, but in an implicit manner. More importantly, it adopts a Hebbian learning rule that is biologically more plausible than the learning rule of the explicit rPCN. We then showed that this model can be further simplified by introducing an apical dendritic architecture, which is even more biologically realistic. We named it the *dendritic rPCN*. In theory, we showed that both the implicit and the dendritic models are exactly equivalent to the explicit rPCN in AM tasks if their inference converges, although the dendritic model may suffer unstable inferential dynamics. Nevertheless, in practice, both implicit and dendritic models can be trained to memorize complex patterns such as handwritten digits and natural images. However, these models are computationally and memory expensive, a drawback that limits its applicability when confronted with more complex datasets. We solve this problem by combining our implicit rPCN with a hierarchical PCN (Rao and Ballard, 1999; Salvatori et al., 2021), to model the predictive interaction between the hippocampus and neocortex (Barron et al., 2020). We showed that this hybrid network can memorize and retrieve large-scale images in a parameter-flexible manner. We further conducted empirical analysis of the memory attractors learned by our linear implicit model, and showed that the memorized patterns are all stored on a hyperplane attractor of the inputs space. We found that nonlinearities are needed to obtain individual point attractors corresponding to each memorized pattern, similar to those observed in Hopfield Networks (Hopfield, 1982). Table 2.2 below summarizes the property of each model discussed in this chapter. Overall, our models benefit the theoretical understanding of the computational principles adopted by the hippocampal network, by providing a unitary account for two disparate theories of how the hippocampus performs AM tasks.

Table 2.2: **Summary of rPCNs discussed in this chapter.** "Hebbian" denotes whether the model employs the Hebbian learning rule; "Parameter flexibility" denotes whether we can freely control the number of parameters of the network, i.e., whether the number of parameters in the model is not constrained to scale quadratically with the input size; "Random/Strctured pattern" denotes whether the model can perform AM on random/strctured patterns.

	Hebbian	**Parameter flexibility**	**Random pattern**	**Structured pattern**
explicit	✗	✗	✓	✗
implicit	✓	✗	✓	✓
dendritic	✓	✗	✓	✓
hybrid	✓	✓	✓	✓

2.4.2 Predictive coding

The explicit rPCN described in this work are essentially predictive coding models with parameters encoding the precision (covariance) matrix (Friston, 2003, 2005; Bogacz, 2017) with a single layer architecture, which corresponds to a model where there is no latent variable $\mathbf{z}$. In this case, the MLE estimates of the model are equivalent to the MAP estimates. These models introduced the covariance matrix to represent the uncertainty of different input features, by differentially weighting error signals derived from different input sources. However, to our knowledge, they have never been tested in AM tasks. Earlier works have also identified the implausibility of the learning rule for the covariance parameters, and addressed this issue by introducing additional inhibitory interneurons on top of error neurons (Bogacz, 2017). On the other hand, our implicit/dendritic rPCN naturally circumvents the implausible learning rule by introducing a new set of recurrent weights $\mathbf{W}$. A natural follow-up of rPCNs is to investigate whether they can be added to all layers of hierarchical predictive coding networks, where the precision matrices were originally included (Friston, 2003, 2005). Furthermore, our nonlinear rPCN is reminiscent of the sparse coding schemes of the kind found in independent component analysis (Olshausen and Field, 1996). This suggests an interesting relationship between sparse coding, PC and the properties evinced by the numerical studies above.

2.4.3 Models of AM and the hippocampus

In this chapter, we have established a connection between rPCN and Hopfield Networks (Hopfield, 1982). This linkage further associates predictive coding with numerous classical memory models (Kanerva, 1988; Kohonen, 1972), which share similar computational principles i.e., learning a covariance or correlation matrix to memorize associations between memory features. This in turn relates rPCN to modern variants of these classical memory models (Krotov and Hopfield, 2016; Ramsauer et al., 2020; Bricken and Pehlevan, 2021). While our empirical findings highlight several advantages of rPCN over Hopfield Networks, particularly in terms of memory capacity, exploring the theoretical capacity limits of rPCN presents a compelling avenue for future research (Keeler, 1988; Hertz et al., 2018; Amit et al., 1987; Chaudhry et al., 2023).

Beyond these analytically tractable models, another noteworthy direction in hippocampal memory research emphasizes anatomical and neurophysiological fidelity. For instance, the models proposed by Treves and Rolls (1994) and O'Reilly and

McClelland (1994) incorporate specific hippocampal subregions such as the dentate gyrus, which is involved in orthogonalization before memory storage in the recurrent CA3 region. Additional studies have examined hippocampal-neocortical interactions (Gluck and Myers, 1993; McClelland et al., 1995). Recent studies indicate that predictive coding can effectively train networks of arbitrary structures (Salvatori et al., 2022a). Specifically, implicit rPCN can accommodate arbitrary constraints on the weight matrix $\mathbf{W}$ during training, which can reflect the connectivity structure of the hippocampal network more faithfully. This paves the way for future research employing rPCN variants to model the comprehensive hippocampal circuit more realistically.

Chapter 3

Temporal Predictive Coding

3.1 Introduction

This study is concerned with extending the theory of predictive coding to the problem of processing dynamically changing sequences of sensory inputs arriving over time. Predictive coding, which originated from an algorithm for compression in information theory (Spratling, 2017), was initially developed and applied to the analysis of the brain by Srinivasan et al. (1982) and Mumford (1992) and then formalized into a general computational model of the cortex by Rao and Ballard (1999). The core hypothesis behind predictive coding is that the brain computes predictions of its observed input, and compares these predictions to the actually received input. The difference between the two is called the prediction error and quantifies how incorrect the brain's prediction was. Predictive coding proposes that the brain then adjusts its neural activities and synaptic strengths to minimize prediction errors which ultimately results in more accurate predictions (Friston, 2005; Clark, 2015). Thus, solely by minimizing prediction errors, the brain is forced to learn a general *world model* which can generate accurate predictions of its incoming sensory input (Friston, 2003, 2005). Moreover, these prediction errors are computed *locally* between the local input to the neuron and the predictions it receives. This means that learning in predictive coding model requires only local information and can be accomplished in most cases with purely Hebbian synaptic plasticity (Buckley et al., 2017; Bogacz, 2017; Millidge et al., 2020b).

As has been introduced in Chapter 1, predictive coding has become an influential theoretical model for understanding cortical functions, and recent progresses have also established its capability in machine learning tasks and relationship to backpropagation. However, most of these works are concerned with inputs that are independently and identically distributed (i.i.d.) samples from some datasets and are presented in

batches to the predictive coding model in random order. The visual input to the brain, on the other hand, is a continually changing sensory stream conveying a sequence of individual images with much correlation and rich structure embedded in the timing of the sequence elements. Therefore, to better describe the information processing in the brain, predictive coding models must take into account one more crucial element: *time*.

There are several established algorithms in statistics and machine learning for sequence processing over time, but they typically require very complex computations that would be difficult for biological circuits to perform. Nevertheless, it is useful to consider them as a reference against which the performance of biologically plausible models can be assessed. When there is a linear relationship between the current and future states (and noise is assumed to be Gaussian), the optimal temporal predictions can be achieved by the Kalman filter (Kalman, 1960). For more complex nonlinear problems, one can employ recurrent neural networks (Jordan, 1990), which contain recurrent[1] connections that maintain and update an internal hidden state over time. While these recurrent networks, and more advanced successors such as the Long-Short-Term-Memory (LSTM) (Hochreiter and Schmidhuber, 1997) can be very expressive and powerful, they are typically trained with backpropagation through time algorithm (BPTT) (Williams and Zipser, 1989; Williams, 1992; Werbos, 1988) which requires storing a history of all computations through a sequence and then "unrolling" it sequentially backwards through time to make updates. This algorithm is biologically implausible, because the brain can only receive inputs in a sequential stream, and must be able to process them online, i.e. as the inputs are received, and seems unlikely to be able to unfold a precise sequence of computations in reverse. In this work, we focus our comparison to these statistical and machine learning models on the Kalman filter, an online algorithm that processes data sequentially, as biological systems appear to do.

Within the field of predictive coding, there are a few tracks of research that have considered incorporating the temporal dimension. Earlier works (Rao, 1997; Rao and Ballard, 1997) employed Kalman filtering to model the visual processing of dynamical sequences. Instead of using fixed transformation matrices which are assumed to be known as in Kalman filtering, these works introduced learning rules to model synaptic plasticity in neural circuits performing filtering. Friston (2008) have proposed the notion of extending predictive coding to use *generalized coordinates* (Friston et al.,

[1]Unlike recurrent PCNs in the previous chapter, the "recurrent" connections discussed in this chapter will involve time.

2008a, 2010) which model a dynamical state by including a set of temporal derivatives in the state vector and making predictions of these derivatives along with the current state. A few recent studies have also developed predictive coding models that perform well in various tasks involving temporal dependencies, such as those commonly examined in machine learning (Kutschireiter et al., 2017; Ororbia et al., 2020; Jiang and Rao, 2022). However, the works mentioned above have departed from the simple and flexible architecture of classical predictive coding for static inputs (Rao and Ballard, 1999) in order to take into account the temporal dimension. For example, the pioneering work that introduced Kalman filtering into predictive coding (Rao, 1997; Rao and Ballard, 1997) has not described how the computations of Kalman filtering could be implemented in biological networks. Mapping of models employing generalized coordinates (Friston, 2008) on a neural circuit would require explicit hard-coding of the expected temporal dependencies between different dynamical orders. Other models include specialised network features to aid the temporal processing, such as hypernetworks (Jiang and Rao, 2022), multiple sub-networks (Kutschireiter et al., 2017) or complex connectivity and neuron types (Ororbia et al., 2020).

In this study, we propose a simple predictive coding network that also incorporates the temporal dimension, which we call *temporal predictive coding* (tPC). This model generates predictions not only about the current inputs but also about its own *future* neural responses, which is achieved by recurrent connections between neurons to transmit the prediction of one time-step to the next. The study makes four main contributions: First, we propose a predictive coding model that addresses the problem of temporal prediction, while inheriting from static predictive coding (Rao and Ballard, 1999) the simple and biologically plausible neural network implementations employing only local connectivity and Hebbian plasticity rules. Second, when the model is linear, we show that our model is a close approximation of the Kalman filtering model analytically, and has empirical performance comparable to Kalman filtering in benchmark filtering tasks while being computationally cheaper and requiring only biologically plausible operations. Also, unlike the Kalman filter, our model can learn the parameters of the system online. Third, when trained with natural moving stimuli, we find that the model develops localized, Gabor-like receptive fields similar to those observed by Rao and Ballard (Rao and Ballard, 1999), and more importantly, the receptive fields are also motion-sensitive, a property unique to neurons responding to dynamic stimuli (Baker, 1990). Overall, our model provides a possible computational mechanism underlying the cortical processing of dynamic inputs based on predictive

coding, suggesting that the brain may learn to represent both static and continuous sensory observations using a single computational framework.

3.2 Models

The structure of the exposition in this section is as follows. Firstly, we present the underlying graphical structure of our proposed generative tPC network (tPCN) i.e., a Hidden Markov Model (HMM). Next, we show that, with Gaussian assumptions on the HMM and certain parametric assumptions on the nonlinear generative processes, we can derive an objective function that is similar to the original predictive coding network (Rao and Ballard, 1999) but taking into account the temporal dimension. We then demonstrate that neural dynamics and plasticity rules can be derived via the minimization of the objective function by gradient descent, and a corresponding training algorithm is presented. Finally, we show that the proposed computations and algorithms afford biologically plausible neural implementations in several different cases and discuss how they may be mapped to neural circuitry in the cortex. Since this study contains much mathematical notation, for ease of reading, we have collected it in Table 3.1 for quick reference [2].

3.2.1 Model foundations: HMM and free energy

The conceptual level of our model is grounded within the Bayesian Brain paradigm (Knill and Pouget, 2004; Pouget et al., 2013; Friston, 2005; Friston and Ao, 2012). Specifically, we assume that the problem of perception is fundamentally an *inference* problem, where there exists some real world "out there", from which we only receive noisy and distorted sensory input. We assume that the task of perception is to untangle and counteract the noise in order to reconstruct the real (but hidden) state of the world given only our sensory observations. Thus, mathematically, we can represent the problem of perception as trying to infer a series of latent states of the world $\mathbf{x}_k$ $(k = 1, 2, \dots)$[3] from their corresponding sensory observations $\mathbf{y}_k$ $(k = 1, 2, \dots)$. We assume that the underlying graphical structure of our tPC network is an HMM, where the hidden states $\mathbf{x}_k$ follow a Markov chain. That is, the current hidden state of the world only depends upon the previous hidden state. Also, the current

[2]Note that in this Chapter, we use $\mathbf{x}$ to denote hidden or latent state and $\mathbf{y}$ to denote observations. This is to echo the notational tradition in control theory as we will compare tPC with Kalman filtering. This notation is different from elsewhere in this thesis.

[3]Throughout this thesis, k will be used to denote time steps in the sensory domain, and t will be used for the rate of change of neural activities as in dx/dt.

Table 3.1: **Table of mathematical notation used in the study.** Vector and matrix variables are defined with their dimensions. Otherwise, the variables are scalars.

Notation	Meaning
$\mathbf{y}$	Observation ($\mathbb{R}^{d_y}$)
$\mathbf{x}$	Latent state ($\mathbb{R}^{d_x}$)
$\mathbf{u}$	Control input ($\mathbb{R}^{d_u}$)
k	Observation "frame"
t	Inference time
$\mathbf{W}$	Dynamics matrix ($\mathbb{R}^{d_x \times d_x}$)
$\mathbf{B}$	Control matrix ($\mathbb{R}^{d_x \times d_u}$)
$\mathbf{F}$	Observation matrix ($\mathbb{R}^{d_y \times d_x}$)
$\mathbf{\Sigma}_x$	Process noise ($\mathbb{R}^{d_x}$)
$\mathbf{\Sigma}_y$	Observation noise ($\mathbb{R}^{d_y}$)
f	Nonlinear function
$\hat{\mathbf{x}}$	Inferred latent state ($\mathbb{R}^{d_x}$)
$q(\cdot)$	Variational distribution
$\mathcal{F}$	Variational free energy
$\boldsymbol{\epsilon}^{\mathbf{x}}$	Dynamics prediction error ($\mathbb{R}^{d_x}$)
$\boldsymbol{\epsilon}^{\mathbf{y}}$	Observation prediction error ($\mathbb{R}^{d_y}$)
η	learning rate
Δt	Inference step size
$\mathbf{\Sigma}_k$	Posterior variance ($\mathbb{R}^{d_x \times d_x}$)
$\mathbf{K}$	Kalman Gain matrix ($\mathbb{R}^{d_x \times d_y}$)

observation is generated by the current state of the world only, with no dependence on past noisy observations. Fig 3.1 shows the generative process of the tPC where, for generality, we have also added "control" inputs $\mathbf{u}$ which can be thought of as known inputs to the system at every time step (such inputs are included in the Kalman filter that we will later use as a benchmark). This control input is useful to model systems where there are known external forces acting on the system (such as an agent's own actions) which we do not necessarily want to model simply as part of the environmental dynamics.

From the generative model in Fig 3.1, we can also write out specific equations for the dynamics of the states and observations in what is called the state-space representation:

$$\mathbf{x}_k = \mathbf{W} f(\mathbf{x}_{k-1}) + \mathbf{B}\mathbf{u}_k + \boldsymbol{\omega}_x, \tag{3.1}$$

$$\mathbf{y}_k = \mathbf{F} f(\mathbf{x}_k) + \boldsymbol{\omega}_y \tag{3.2}$$

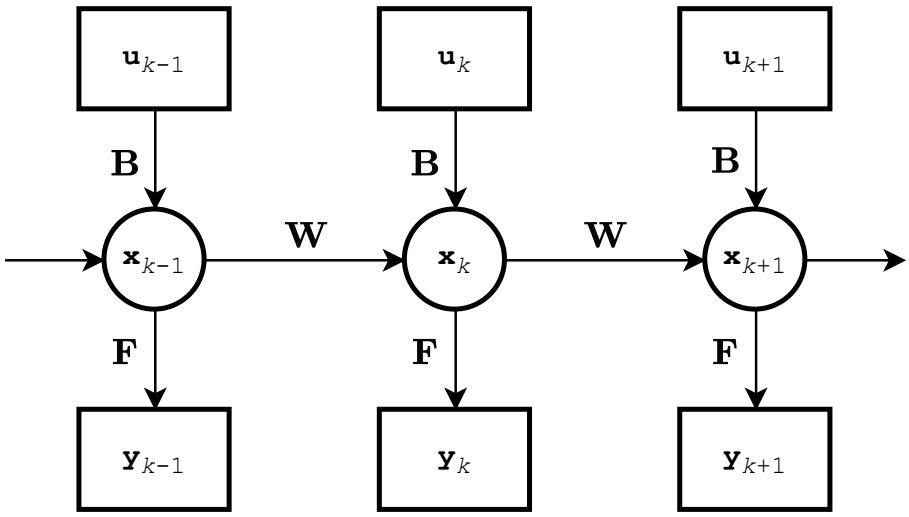

Figure 3.1: **Graphical model of the generative process assumed by temporal predictive coding.** $\mathbf{x}_k$ correspond to hidden states, $\mathbf{y}_k$ to observations, and $\mathbf{u}_k$ to control inputs. Circles denote latent variables, squares denote observations, and arrows denote conditional dependence of the variables (the absence of an arrow indicates conditional independence).

where $\mathbf{W}$ is the matrix that transitions the previous hidden state $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$, $\mathbf{B}$ is the matrix governing how the control input $\mathbf{u}_k$ affects the current hidden state, and $\mathbf{F}$ is the "emission" matrix that determines how the observation $\mathbf{y}_k$ is generated from the hidden state. f is a function transforming $\mathbf{x}_k$ that may be nonlinear. The above state-space representation also includes sources of noise $\boldsymbol{\omega}_x$ and $\boldsymbol{\omega}_y$. We will assume a white Gaussian noise model such that $\boldsymbol{\omega}_x \sim \mathcal{N}(0, \boldsymbol{\Sigma}_x)$ and $\boldsymbol{\omega}_y \sim \mathcal{N}(0, \boldsymbol{\Sigma}_y)$ are zero-mean Gaussian random variables with covariance matrices $\boldsymbol{\Sigma}_x$, $\boldsymbol{\Sigma}_y$ [4]. Therefore, $\mathbf{x}_k$ and $\mathbf{y}_k$ can be considered as random variables that follow the Gaussian distributions:

$$\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k \sim \mathcal{N}(\mathbf{W} f(\mathbf{x}_{k-1}) + \mathbf{B}\mathbf{u}_k, \boldsymbol{\Sigma}_x), \tag{3.3}$$

$$\mathbf{y}_k | \mathbf{x}_k \sim \mathcal{N}(\mathbf{F} f(\mathbf{x}_k), \boldsymbol{\Sigma}_y). \tag{3.4}$$

The objective of our model is to obtain an estimate $\hat{\mathbf{x}}_k$ of the current $\mathbf{x}_k$, given the previously estimated $\hat{\mathbf{x}}_{k-1}$, current noisy observation $\mathbf{y}_k$ and control input $\mathbf{u}_k$. This objective can be expressed as estimating the posterior distribution $p(\mathbf{x}_k | \mathbf{y}_k, \hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)$.

[4] In the control theory literature, the process noise $\boldsymbol{\Sigma}_x$ is often denoted as $\mathbf{Q}$ and the observation noise $\boldsymbol{\Sigma}_y$ as $\mathbf{R}$. We instead follow the convention that has arisen within the predictive coding literature, which we also believe is more straightforward, by calling them $\boldsymbol{\Sigma}_x$ and $\boldsymbol{\Sigma}_y$, since it makes explicit that these variances are simply the variances of the two distributions composing the generative model.

If function f is nonlinear, an analytic expression for the posterior cannot be found, and thus variational inference (Neal and Hinton, 1998; Beal et al., 2003; Ghahramani et al., 2000) is utilized to find an approximate of the posterior. Similar to what was introduced in Chapter 1 for the static case, we assume that there exists an approximate posterior $q(\mathbf{x}_k)$ and we seek an approximate posterior that is as close as possible to the "true" posterior $p(\mathbf{x}_k|\mathbf{y}_k, \hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)$, and variational inference finds such a $q(\mathbf{x}_k)$ by minimizing an upper bound on the divergence between the true and approximate posteriors called the variational free energy. If we make an additional simplifying assumption that $q(\mathbf{x}_k)$ follows a Gaussian distribution with its density highly concentrated at its mean, the free energy $\mathcal{F}_k$ becomes approximately equal to (Buckley et al., 2017; Bogacz, 2017):

$$\begin{aligned} \mathcal{F}_k &= -\log p(\mathbf{y}_k, \mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \\ &= -\log p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \end{aligned} \tag{3.5}$$

and it is also sufficient to estimate the mode of the approximate posterior (which is the same as its mean) instead of estimating the whole distribution with this assumption. Furthermore, with the Gaussian assumptions underlying Eqs. 3.3 and 3.4, the free energy becomes (we choose to omit the other terms in the multivariate Gaussian density as they do not affect the optimization over $\mathbf{x}_k$ and $\mathbf{W}$, $\mathbf{B}$, $\mathbf{F}$):

$$\begin{aligned} \mathcal{F}_k = &\frac{1}{2}(\mathbf{y}_k - \mathbf{F}f(\mathbf{x}_k))^\top \boldsymbol{\Sigma}_y^{-1}(\mathbf{y}_k - \mathbf{F}f(\mathbf{x}_k)) \\ &+ \frac{1}{2}(\mathbf{x}_k - \mathbf{W}f(\hat{\mathbf{x}}_{k-1}) - \mathbf{B}\mathbf{u}_k)^\top \boldsymbol{\Sigma}_x^{-1}(\mathbf{x}_k - \mathbf{W}f(\hat{\mathbf{x}}_{k-1}) - \mathbf{B}\mathbf{u}_k) + C \end{aligned} \tag{3.6}$$

Importantly, we can express this objective as the sum of squared *prediction errors* weighted by their inverse covariances (which are often called precisions in the predictive coding literature). In this model, there are two kinds of prediction errors – "sensory" prediction errors which are the difference between the observation and predicted observation $\mathbf{y}_k - \mathbf{F}f(\mathbf{x}_k)$ and "temporal" prediction errors which are the difference between the inferred current state and the current state predicted from the previous state $\mathbf{x}_k - \mathbf{W}f(\hat{\mathbf{x}}_{k-1}) - \mathbf{B}\mathbf{u}_k$. Thus, by finding an estimate $\hat{\mathbf{x}}_k = \text{argmin}_{\mathbf{x}_k} \mathcal{F}_k$, we effectively minimize the squared sum of these prediction errors while the precision matrices serve to weight the importance of the sensory and temporal prediction errors in accordance with their intrinsic variance (so highly variable prediction errors are weighted less). After the minimization finishes, the estimated $\hat{\mathbf{x}}_k$ can then be used to estimate the hidden state at the next step $k + 1$.

3.2.2 Inference and learning algorithm

With the objective function in Eq. 3.6, it is then possible to derive an iterative algorithm to perform its minimization via gradient descent. Similar to static predictive coding (Rao and Ballard, 1999), the gradient descent is performed on two sets of values: the hidden states $\mathbf{x}_k$ and the weight parameter matrices $\mathbf{W}$, $\mathbf{B}$ and $\mathbf{F}$. As we will show in the next subsection, the former can be implemented as neural responses and the latter can be implemented as synaptic connections in a neural circuit in a similar way to static predictive coding. For the hidden state $\mathbf{x}_k$, its dynamics follow:

$$\tau \frac{d\mathbf{x}_k}{dt} = -\frac{\partial \mathcal{F}_k}{\partial \mathbf{x}_k}, \tag{3.7}$$

where τ is the time constant of the neurons. At convergence, we can say that the equilibrium value $\hat{\mathbf{x}}_k$ represents the optimal inference about the mean of the true hidden state of the world. It is worth mentioning that we now introduced two different indices k and t for distinct time scales: for discrete steps k at which the observations arise and continuous real time t in which computations are made within the model, and we will discuss the relationship between them in detail in the next subsection.

To derive the exact expression for the inferential dynamics, we can define the precision-weighted state and observation prediction errors as $\boldsymbol{\epsilon}^{\mathbf{x}}$ and $\boldsymbol{\epsilon}^{\mathbf{y}}$ respectively as follows:

$$\boldsymbol{\epsilon}^{\mathbf{y}} = \boldsymbol{\Sigma}_y^{-1}(\mathbf{y}_k - \mathbf{F} f(\mathbf{x}_k)), \tag{3.8}$$

$$\boldsymbol{\epsilon}^{\mathbf{x}} = \boldsymbol{\Sigma}_x^{-1}(\mathbf{x}_k - \mathbf{W} f(\hat{\mathbf{x}}_{k-1}) - \mathbf{B}\mathbf{u}_k). \tag{3.9}$$

We can then write Eq. 3.7 as:

$$\tau \frac{d\mathbf{x}_k}{dt} = -\frac{\partial \mathcal{F}_k}{\partial \mathbf{x}_k} = -\boldsymbol{\epsilon}^{\mathbf{x}} + f'(\mathbf{x}_k) \odot \mathbf{F}^\top \boldsymbol{\epsilon}^{\mathbf{y}}, \tag{3.10}$$

where $\odot$ denotes the element-wise product between vectors. The dynamics have contributions from both the sensory and the temporal prediction errors, and the contribution of the sensory prediction error is "mapped backwards" through the transpose matrix $\mathbf{F}^\top$.

Similarly, if we assume that the $\mathbf{W}$, $\mathbf{B}$ and $\mathbf{F}$ parameter matrices are learnable, we can derive update rules also following gradient descent on $\mathcal{F}_k$:

$$\begin{aligned}
\Delta \mathbf{W} &= -\eta \frac{\partial \mathcal{F}_k}{\partial \mathbf{W}} = \eta \boldsymbol{\epsilon}^{\mathbf{x}} f(\hat{\mathbf{x}}_{k-1})^\top, \\
\Delta \mathbf{B} &= -\eta \frac{\partial \mathcal{F}_k}{\partial \mathbf{B}} = \eta \boldsymbol{\epsilon}^{\mathbf{x}} \mathbf{u}_k^\top, \\
\Delta \mathbf{F} &= -\eta \frac{\partial \mathcal{F}_k}{\partial \mathbf{F}} = \eta \boldsymbol{\epsilon}^{\mathbf{y}} f(\mathbf{x}_k)^\top.
\end{aligned} \tag{3.11}$$

In the above equations, η denotes a scalar learning rate. As we will see below, the iterative updates will correspond to local Hebbian plasticity in the neural implementation of the model.

Typically, if the $\mathbf{W}$, $\mathbf{B}$, and $\mathbf{F}$ matrices are learned, then the matrices are updated by a single step according to Eqs. 3.11 after the $\mathbf{x}_k$ has already converged and using the equilibrium values $\hat{\mathbf{x}}_k$. This is because it is often assumed that these variables represent more slowly changing variables in the real world. Moreover, in the neural implementation, these matrices are often assumed to be implemented by synaptic strengths which typically change slowly while the $\hat{\mathbf{x}}$ variables are typically mapped to neural firing rates, which change quickly.

The process of learning a tPC model is shown in Algorithm 1. The algorithm assumes that sensory observations $\mathbf{y}_k$ arrive at discrete steps k. At each step k, we first initialize the $\mathbf{x}_k$ values with the previous estimated $\hat{\mathbf{x}}_{k-1}$ value from the last step. Then, we iterate Eq.3.10 until convergence for a given observation $\mathbf{y}_k$. Upon convergence our $\hat{\mathbf{x}}_k$ becomes our best estimate of the true state of the world. Given this $\hat{\mathbf{x}}_k$, we can update the $\mathbf{W}$, $\mathbf{B}$, and $\mathbf{F}$ matrices using Eqs.3.11.

Algorithm 1 Single training epoch for temporal predictive coding

```
K: Discrete steps of observations
for k = 1, ..., K do
    Initialize x_k with previously inferred x̂_{k-1}
    while x_k not converged do
        Perform inference by updating x_k (Eq 3.10)
    end while
    Update weight matrices ΔW, ΔB, ΔF (Eq 3.11) using inferred x̂_k
end for
```

3.2.3 Neural circuit implementation

The update rules and dynamics we have derived in Eqs.3.10 and 3.11 can be mapped to a recurrent predictive coding network architecture with biologically plausible Hebbian learning rules. In this section, we present two examples of networks implementing the algorithm exactly, and then a simplified network that approximates the algorithm.

Fig 3.2A presents a potential example of how the update rules we have derived can be implemented in a neural network with only local and Hebbian plasticity, which is similar to the standard predictive coding network (Rao and Ballard, 1999). In this network, observations $\mathbf{y}_k$ enter at the lowest level, and cause sensory prediction errors $\boldsymbol{\epsilon}^{\mathbf{y}}$ as the observations are met by top-down predictions. These prediction errors are

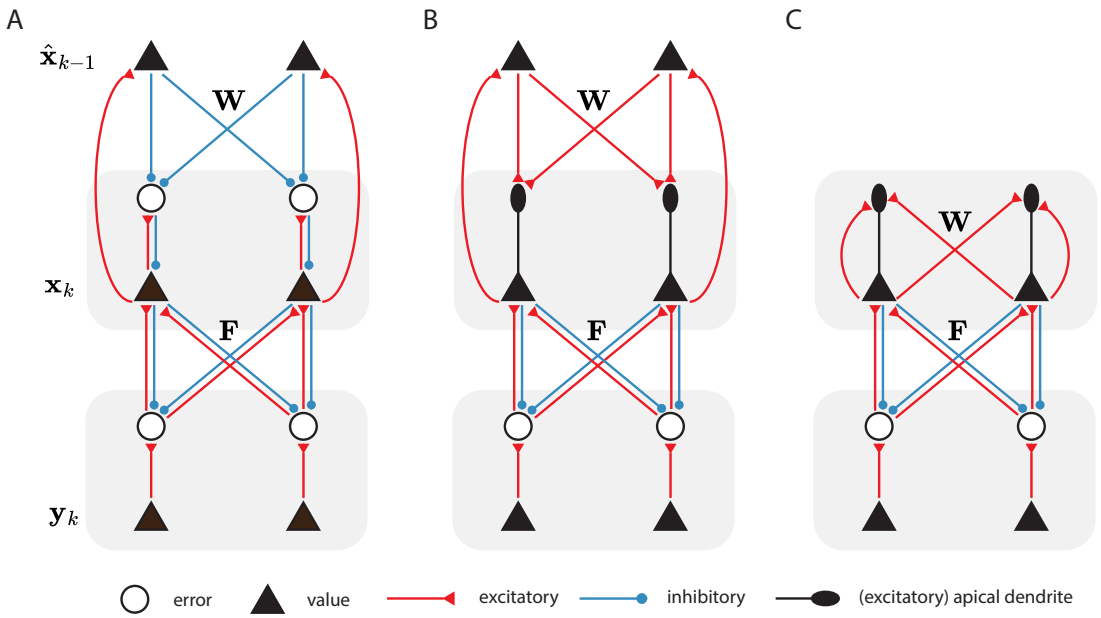

Figure 3.2: **Possible neural implementations of temporal predictive coding.** A: Potential neural circuit implementing the iterative recurrent predictive coding algorithm. For simplicity, we have depicted each neural layer as possessing only two neurons. B: Version of the model where the prediction errors are represented by the difference in membrane potential in soma and at apical dendrites (depicted as ellipses). C: Neural circuitry required to implement the single-iteration predictive coding algorithms. This model no longer includes a separate set of neurons explicitly storing the estimate of the previous timestep, but instead, the temporal prediction errors are computed naturally through recurrent connections. For simplicity, we omitted the control inputs $\mathbf{B}\mathbf{u}_k$, which can be implemented in a similar way to the recurrent inputs $\mathbf{W}\hat{\mathbf{x}}_{k-1}$ to the error neurons or apical dendrites.

explicitly represented by the activities of "prediction error neurons". These prediction error neurons receive top-down inhibitory connections from "value neurons" in the layer above which, through their firing rates, represent the inferred posterior values $\hat{\mathbf{x}}_k$. Similarly, at the layer above there are additional prediction error neurons that represent the difference between the current activity and that predicted based on the previous inference mapped through the dynamics function (Eq.3.9), and we assume that there are dedicated neurons that "memorize" latent activities $\hat{\mathbf{x}}_{k-1}$ inferred at the previous discrete time step, which is used to make a prediction of the current latent activities and reloaded at the end of inference at each time step.

There are several important aspects to note about this model. First, all the required update rules can be implemented using purely local information. The dy-

namics of the hidden state estimates $\mathbf{x}_k$ (Eq.3.10) can be reproduced locally since the value neurons receive inhibitory connections from the prediction error neurons $\boldsymbol{\epsilon}^{\mathbf{x}}$ at their layer as well as excitatory connections from the prediction error neurons $\boldsymbol{\epsilon}^{\mathbf{y}}$ at the layer below. Similarly, the prediction errors $\boldsymbol{\epsilon}^{\mathbf{y}}$ can be computed according to Eq.3.8 as the corresponding prediction error neurons receive excitatory input $\mathbf{y}_k$ and inhibition from neurons encoding $\mathbf{x}_k$. The prediction error $\boldsymbol{\epsilon}^{\mathbf{x}}$ can be computed analogously. The update rules for the $\mathbf{W}$, $\mathbf{B}$, and $\mathbf{F}$ weight matrices (Eqs.3.11) are also precisely Hebbian, since they are outer products between the prediction errors and the value neurons of the layer above which, crucially, are also precisely the pre and post-synaptic activities of the neurons where the synapses implementing these weight matrices are located. Moreover, we empirically demonstrate in the Results sections that scaling by the inverse covariance matrices $\boldsymbol{\Sigma}_x^{-1}$ and $\boldsymbol{\Sigma}_y^{-1}$ could be encoded in the learned $\mathbf{W}$ and $\mathbf{F}$ matrices, similarly as it has been shown in static predictive coding models (Chapter 2; Tang et al. (2023)). Thus, the tPC model can represent the covariance matrices implicitly in its synaptic weights, without needing to implement them in explicit synaptic weights. The nonlinear function $f(\cdot)$ can be implemented in this circuit following the way specified in Whittington and Bogacz (2017), through local inhibitory neurons.

The network shown in Fig 3.2A follows a standard predictive coding architecture, but it could be simplified because the prediction error neurons encoding $\boldsymbol{\epsilon}^{\mathbf{x}}$ only project to corresponding neurons encoding $\mathbf{x}$, and we could thus borrow the idea of dendritic computing similar to the model of Sacramento et al. (2018). In particular, substituting Eq.3.9 into Eq. 3.10, we obtain:

$$\tau \frac{d\mathbf{x}_k}{dt} = -\mathbf{x}_k + \mathbf{W} f(\hat{\mathbf{x}}_{k-1}) + \mathbf{B}\mathbf{u}_k + f'(\mathbf{x}_k) \odot \mathbf{F}^\top \boldsymbol{\epsilon}^{\mathbf{y}}, \tag{3.12}$$

where, for clarity of explanation, we omitted the precision $\boldsymbol{\Sigma}_x^{-1}$ that can be encoded in $\mathbf{W}$. By writing the dynamical equation in this way, we assume that there is no building block within the model that encodes the error explicitly; rather, the apical dendrite will encode the inputs $\mathbf{W} f(\hat{\mathbf{x}}_{k-1}) + \mathbf{B}\mathbf{u}_k$ and send the signal to the soma of the pyramidal neuron. This dendritic signal excites the soma and drives the inferential dynamics (Eq. 3.12), together with the decay $-\mathbf{x}_k$ that is intrinsic to the soma and feedback signals from the observation layer. The corresponding neural implementation is shown in Fig 3.2B. Although the architecture of the network becomes simpler, learning parameters $\mathbf{W}$ and $\mathbf{B}$ is less straightforward because the prediction error $\boldsymbol{\epsilon}^{\mathbf{x}}$ is not explicitly represented in the activity of any neurons in the network. Nevertheless, the prediction error $\boldsymbol{\epsilon}^{\mathbf{x}}$ is equal to the difference between the neural

activity and the membrane potential in the apical dendrite, and it has been proposed that such difference drives plasticity of synapses on the apical dendrite (Urbanczik and Senn, 2014). Since both the neural activity and the membrane potential are encoded within the same neuron, it is plausible that their difference could be computed within the neuron (e.g. the information on the neural activity could be brought to the synapses on apical dendrites via backpropagating action potentials). Such a signal could then drive local synaptic plasticity to learn the $\mathbf{W}$ and $\mathbf{B}$ matrices.

While simulating the model, we update the state estimates by numerically solving Eq. 3.12 using the Euler method, i.e. we calculate the state estimates for every interval Δt:

$$\mathbf{x}_k(t+\Delta t) = \mathbf{x}_k(t) + \frac{\Delta t}{\tau}\left[-\mathbf{x}_k(t) + \mathbf{W}f(\hat{\mathbf{x}}_{k-1}) + \mathbf{B}\mathbf{u}_k + f'(\mathbf{x}_k(t)) \odot \mathbf{F}^\top \boldsymbol{\epsilon}^{\mathbf{y}}\right]. \quad (3.13)$$

where $\frac{\Delta t}{\tau}$ is effectively the step size of the discretized inference that we tune in our simulations. The above expression highlights that the algorithm has two nested timescales – firstly there is the "external" timescale which is where sensory inputs $\mathbf{y}_k$ are received in a sequence of steps we index by subscript k. Then, for each external step, there is an internal inference of the hidden state that is numerically implemented as a set of recurrent iterations within that external step, which we denote as t. In such a nested framework, the implementations in Figs 3.2A and 3.2B need to store and hold fixed the state estimate of the previous step while the iterative inference of the state estimate for the current step is ongoing. Specifically, the estimate from the previous step $\hat{\mathbf{x}}_{k-1}$ needs to be held fixed throughout the iterative procedure while the actual current values $\mathbf{x}_k(t)$ vary in order to find a balance $\hat{\mathbf{x}}_k$ between the demands of matching the prediction from the last step and also the current observation (Algorithm 1). Once the iterations are complete for a step, the new value of $\hat{\mathbf{x}}_k$ needs to be loaded into the memory and stored as the last step for the next set of iterations. In situations where the observations are separated in time, it is known that neurons are able to store the representation of stimuli presented a few seconds earlier in their activity (Rainer et al., 1998). In the case where sensory input arrives continuously, the two timescales coincide, and there may be no time for inference between steps of sensory input. In that latter case, the algorithm can be adapted to remove the issue of nested timescales without unduly harming filtering performance by simply using a *single* iteration of the internal inference for each external step. This means that there is effectively no inner loop of the algorithm anymore, since the inner loop consists of just a single iteration. This makes the algorithm fully online in the sense that it

receives a new sensory input for every step. In particular, note that by equating time indices $\Delta k = \Delta t$, we obtain:

$$\hat{\mathbf{x}}_k = \mathbf{x}_k + \frac{\Delta t}{\tau}\left[-\mathbf{x}_k + \mathbf{W} f(\hat{\mathbf{x}}_{k-1}) + \mathbf{B}\mathbf{u}_k + f'(\mathbf{x}_k) \odot \mathbf{F}^\top \boldsymbol{\epsilon}^{\mathbf{y}}\right]. \tag{3.14}$$

A diagram of a potential neural circuit implementing this single-step algorithm is presented in Fig 3.2C. This network no longer includes neurons storing past inferences. Instead, the temporal prediction errors are computed solely using recurrent connections labelled $\mathbf{W}$, which are now assumed to introduce a temporal delay of one step.

The advantage of this approach is that it eschews the challenge of storing and loading the memory of the last step; instead, this memory can be dynamically maintained across a single external step simply through recurrent connections via their intrinsic synaptic delays. The disadvantage of this approach is that the update rules of the algorithm were derived as gradient descent on an objective function, and this approach is equivalent to taking only a single gradient step for each example. Clearly, in many cases, such an algorithm simply would not work because a single step is nowhere near enough to approach the optimum. However, there are two features of the problem that ameliorate much of this difficulty in practice. The first is that, when f is a linear function, the objective is actually convex, and thus the loss landscape is extremely well-behaved. This allows for the use of relatively high integration steps to move large distances in a single, or a few steps, without fear of overshooting the optimum or running into divergences. The other factor is due to the nature of the external world: typically, visual scenes change relatively slowly on a microsecond-by-microsecond level, and thus the optimal estimated hidden state in a single step is likely close to the optimum hidden state for the next. In this case, since we initialize the inference of each step with the optimum of the last step, this will usually be close to the optimum for the current step as well, thus meaning that the algorithm simply does not have to make many iterations to achieve the optimum since it already starts close by. In the next section, we show that, in practice, on standard tracking and filtering tasks, these two factors can often simplify the inference problem enough that this single iteration approach often works successfully, although it usually does not perform quite as well as multi-step methods.

3.3 Results

The results of this work are partitioned into a theoretical section and experimental sections. In the theoretical section, we examine the relationship between the tPC model and Kalman filtering and demonstrate that the tPC network, under certain assumptions, is equivalent to a Kalman filter with a fixed posterior variance. In simulations, we demonstrate that, despite not correctly representing the posterior variance, the tPC network nevertheless exhibits strong and robust tracking performance on filtering tasks while also being capable of online system identification through the learning of the $\mathbf{W}$ and $\mathbf{F}$ matrices, unlike the Kalman filter. Moreover, we show that when tPC is trained with natural movies, the simulated neurons in the latent layer develop motion-sensitive receptive fields resembling those of neurons in the primary visual cortex.

3.3.1 Relationship to Kalman filtering

Here we show that both Kalman filtering (Kalman, 1960) and tPC can be derived as special cases of the *Bayesian filtering* problem. Bayesian filtering concerns the problem of inferring the *sequence* of hidden causes $\mathbf{x}_1, ..., \mathbf{x}_K$ of the observations $\mathbf{y}_1, ..., \mathbf{y}_K$. This problem can be effectively factorised into a sequence of online inference problems i.e., inferring the hidden state $\mathbf{x}_k$ at time step k, given the whole history of observations $\mathbf{y}_1, ..., \mathbf{y}_k$ (Jazwinski, 2007; Stengel, 1986). For simplicity of notation, we denote $\mathbf{x}_{1:k} = \mathbf{x}_1, ..., \mathbf{x}_k$ and $\mathbf{y}_{1:k} = \mathbf{y}_1, ..., \mathbf{y}_k$. The Bayesian filtering problem can thus be formulated as inferring the following posterior distribution:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}). \tag{3.15}$$

We show in Appendix B that this posterior distribution is proportional to:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto p(\mathbf{y}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) d\mathbf{x}_{k-1} \tag{3.16}$$

where the integral is effectively the marginal distribution $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ of the joint $p(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ and can be considered as the prior on $\mathbf{x}_k$. Notice that the term $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ is exactly the posterior inferred from the previous time step $k-1$, making Bayesian filtering a recursive method (Chen et al., 2003). As a special case of Bayesian filtering, Kalman filtering assumes that the conditional distributions $p(\mathbf{y}_k|\mathbf{x}_k)$ and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ can be parameterized *linearly* as follows:

$$\mathbf{y}_k|\mathbf{x}_k \sim \mathcal{N}(\mathbf{F}\mathbf{x}_k, \mathbf{\Sigma}_y); \quad \mathbf{x}_k|\mathbf{x}_{k-1} \sim \mathcal{N}(\mathbf{W}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k, \mathbf{\Sigma}_x). \tag{3.17}$$

Further, it assumes that the posterior estimated at the previous step $k-1$ follows:

$$\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1} \sim \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \boldsymbol{\Sigma}_{k-1}) \tag{3.18}$$

where $\hat{\mathbf{x}}_{k-1}$ is the MAP estimate from the previous step $k-1$ and is the mode (or mean) of the Gaussian posterior with covariance $\boldsymbol{\Sigma}_{k-1}$ at step $k-1$. Under the above Gaussian assumptions, the prior $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ on $\mathbf{x}_k$ (i.e., the integral in Eq. 3.16) can be written as (Bishop and Nasrabadi, 2006):

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \mathcal{N}(\mathbf{W}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k, \mathbf{W}\boldsymbol{\Sigma}_{k-1}\mathbf{W}^\top + \boldsymbol{\Sigma}_x). \tag{3.19}$$

Kalman filtering then performs maximum a posteriori (MAP) to find $\hat{\mathbf{x}}_k$:

$$\begin{aligned}\hat{\mathbf{x}}_k &= \underset{\mathbf{x}_k}{\text{argmax}} \log p(\mathbf{x}_k|\mathbf{y}_{1:k}) \\ &= \underset{\mathbf{x}_k}{\text{argmin}} (\mathbf{y}_k - \mathbf{F}\mathbf{x}_k)^\top \boldsymbol{\Sigma}_y^{-1} (\mathbf{y}_k - \mathbf{F}x_k) \\ &+ (\mathbf{x}_k - \mathbf{W}\hat{\mathbf{x}}_{k-1} - \mathbf{B}\mathbf{u}_k)^\top (\mathbf{W}\boldsymbol{\Sigma}_{k-1}\mathbf{W}^\top + \boldsymbol{\Sigma}_x)^{-1} (\mathbf{x}_k - \mathbf{W}\hat{\mathbf{x}}_{k-1} - \mathbf{B}\mathbf{u}_k).\end{aligned} \tag{3.20}$$

Since all the transformation functions in this optimization problem are linear, an analytical expression for $\hat{\mathbf{x}}_k$ can be derived, which will result in the well-known algorithm for Kalman filtering i.e., the "projection":

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= \mathbf{W}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \\ \boldsymbol{\Sigma}_k^- &= \mathbf{W}\boldsymbol{\Sigma}_{k-1}\mathbf{W}^\top + \boldsymbol{\Sigma}_x\end{aligned} \tag{3.21}$$

and the "correction" step where we then incorporate the new information we have received from the environment to correct our estimates:

$$\begin{aligned}\hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{y}_k - \mathbf{F}\hat{\mathbf{x}}_k^-) \\ \boldsymbol{\Sigma}_k &= (\mathbf{I} - \mathbf{K}\mathbf{F})\boldsymbol{\Sigma}_k^- \\ \mathbf{K} &= \boldsymbol{\Sigma}_k^-\mathbf{F}^\top \Big[\mathbf{F}\boldsymbol{\Sigma}_k^-\mathbf{F}^\top + \boldsymbol{\Sigma}_y\Big]^{-1}\end{aligned} \tag{3.22}$$

where $\mathbf{K}$ is known as the Kalman Gain matrix and is central to the Kalman filter update rules for the estimated mean and variance in the correction step (Welch et al., 1995). $\hat{\mathbf{x}}_k$ and $\boldsymbol{\Sigma}_k$ are then our estimated mean and covariance of the posterior Gaussian distribution. The derivation of the projection and correction rules can be found in prior works (Chen et al., 2003; Millidge et al., 2021), while we also provide the derivation in Appendix B as to compare to the update rules of our tPC model, which is demonstrated below.

Our tPC model also aims to solve the Bayesian filtering problem in Eq. 3.16, and it can also make the linear and Gaussian assumptions underlying Eq. 3.17. Notice that

Eq. 3.17 is identical to Eqs. 3.3 and 3.4, but with a linear f. tPC differs from Kalman filtering by making a different assumption on the distribution on the previous-step posterior $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$. Instead of assuming it as a Gaussian distribution in Eq. 3.18, it assumes:

$$\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1} \sim \delta(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) \tag{3.23}$$

where $\delta(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1})$ denotes a Dirac distribution with its density concentrated at $\hat{\mathbf{x}}_{k-1}$. The prior on $\mathbf{x}_k$ for predictive coding thus becomes:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = p(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}) = \mathcal{N}(\mathbf{W}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k, \mathbf{\Sigma}_x) \tag{3.24}$$

since the density of $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ is concentrated at $\hat{\mathbf{x}}_{k-1}$. The MAP estimation of $\hat{\mathbf{x}}_k$ performed by predictive coding is thus:

$$\begin{aligned}
\hat{\mathbf{x}}_k &= \underset{\mathbf{x}_k}{\operatorname{argmax}} \log p(\mathbf{x}_k|\mathbf{y}_{1:k}) \\
&= \underset{\mathbf{x}_k}{\operatorname{argmin}} (\mathbf{y}_k - \mathbf{F}\mathbf{x}_k)^\top \mathbf{\Sigma}_y^{-1} (\mathbf{y}_k - \mathbf{F}\mathbf{x}_k) \\
&+ (\mathbf{x}_k - \mathbf{W}\hat{\mathbf{x}}_{k-1} - \mathbf{B}\mathbf{u}_k)^\top \mathbf{\Sigma}_x^{-1} (\mathbf{x}_k - \mathbf{W}\hat{\mathbf{x}}_{k-1} - \mathbf{B}\mathbf{u}_k)
\end{aligned} \tag{3.25}$$

which is the free energy in Eq. 3.5 with a linear f. The above derivation thus provides another interpretation of the tPC model, i.e. a special case of Bayesian filtering that assumes observation and state noise are the same—and known. Again, as all transformations in the optimization objective are linear, we can derive an analytical expression for $\hat{\mathbf{x}}_k$:

$$\begin{aligned}
\hat{\mathbf{x}}_k^- &= \mathbf{W}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \\
\hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{y}_k - \mathbf{F}\hat{\mathbf{x}}_k^-) \\
\mathbf{K} &= \mathbf{\Sigma}_x \mathbf{F}^\top \left[\mathbf{F}\mathbf{\Sigma}_x\mathbf{F}^\top + \mathbf{\Sigma}_y\right]^{-1}
\end{aligned} \tag{3.26}$$

which is similar to the projection and correction steps for Kalman filtering (Eqs. 3.21 and 3.22), but with $\mathbf{\Sigma}_{k-1}$ assumed to be 0, i.e., tPC does not propagate the uncertainty estimations. The derivation of these equations can be found in Appendix B.

It is also worth mentioning that in the tPC model, although we did not specify the estimation of uncertainty $\mathbf{\Sigma}_k$ at each step, not *propagating* the uncertainty $\mathbf{\Sigma}_{k-1}$ from the previous step is not equivalent to not *estimating* it. In fact, as was shown in (Baltieri and Isomura, 2021), even when we choose to estimate $\mathbf{\Sigma}_k$ in the tPC model, it is still not propagated. Therefore, our MAP estimation $\hat{\mathbf{x}}_k$ will not be affected.

To summarize, by assuming a linear f, here we have shown that both Kalman filtering and tPC are special cases of the Bayesian filtering problem, while the key difference is that tPC ignores the uncertainty estimated at the previous step when estimating the most likely hidden cause at the current time step, whereas Kalman filtering always estimates this uncertainty. However, as we will show in the experimental results section, in the benchmark tracking tasks, tPC performs on par with Kalman filtering, albeit not estimating the posterior uncertainty. Importantly, there are several advantages of tPC over Kalman filtering as a model of dynamical processing in the brain. Firstly, the projection and correction steps of Kalman filtering require complicated matrix algebra and are challenging to compute in neural circuitry, especially the Kalman Gain matrix $\mathbf{K}$. On the other hand, although we can derive analytical results for the estimates of tPC as well, these estimates can be obtained via the iterative methods mentioned above, which afford plausible circuitry implementations (Fig 3.2). Secondly, the iterative nature of our tPC model also makes it adaptable to nonlinear f, where there are no analytical solutions to the Bayesian filtering problem. In contrast, extending the Kalman Filter to nonlinear systems is challenging and standard methods such as the extended Kalman filter (Ruck et al., 1992) work by linearizing around the nonlinearity and thus require knowledge of the Jacobian of the nonlinearity at every state, which is also challenging to implement in neural circuits. Finally, the Kalman filter assumes knowledge of the correct $\mathbf{W}$, $\mathbf{B}$, and $\mathbf{F}$ matrices while these must presumably be learned from sensory observations in the brain.

3.3.2 Performance in linear filtering problems

Here we first present results for the linear tPC model on a simple tracking task of the kind to which the Kalman filter is commonly applied in industry. Here, the goal is simply to infer the unknown hidden state (position, velocity, acceleration) of an object that is undergoing an unknown acceleration. We receive noisy observations of the position, state, and acceleration of the object which are mapped through a random $\mathbf{F}$ matrix with additional observation noise. We use a random $\mathbf{F}$ matrix for the observation mapping to simulate and test the most difficult scenario where the observations are entirely scrambled.

Mathematically, the generative process of this task can be represented according to a linear state space model. We assume that the position and velocity of the object follow the usual laws of Newtonian physics, and there is a persistent acceleration

which is affected by the control input, giving us the following $\mathbf{W}$ and $\mathbf{B}$ matrices,

$$\mathbf{W} = \begin{bmatrix} 1 & \Delta k & \frac{1}{2}\Delta k^2 \\ 0 & 1 & \Delta k \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}. \tag{3.27}$$

In Eq. 3.27, Δk denotes the duration of the interval between successive sensory observations (we used $\Delta k = 0.001$). Additionally, we draw a fixed $\mathbf{F}$ matrix from a random Gaussian distribution $\mathbf{F} \sim \mathcal{N}(0, 1)$, and modeled the control inputs as $\mathbf{u}_k = e^{-0.01k}$. The process and observation noise $\mathbf{\Sigma}_x$ and $\mathbf{\Sigma}_y$ were set to identity matrices. We then generate the true latent states $\mathbf{x}_k$ and the noisy observations $\mathbf{y}_k$ using Eqs. 3.1 and 3.2, initialized with a zero vector when $k = 0$. The performance of the models is then measured as the mean squared error (MSE) between the estimated $\hat{\mathbf{x}}_k$ and true $\mathbf{x}_k$ across all observation time steps. Fig 3.3A shows an example of the true system state that we generated across 1000 time steps, and Fig 3.3B shows its corresponding noisy observations. As can be seen, the projected observations are completely scrambled by matrix $\mathbf{F}$, making it a challenging task for the models to retrieve the true system states.

We then investigate tracking performance using tPC compared to the Kalman filter for both 5 steps of inference between observations ($\Delta k = 5\Delta t$) and 1 step ($\Delta k = \Delta t$). Since the problem is linear, we also investigate the performance of a tPC when its inference dynamics have reached the equilibrium, using the equilibrium condition derived in Eq. 3.26. Since tracking performance is visually indistinguishable when zoomed out over 1000 timesteps, in Fig 3.3C, we plot the estimates of acceleration (x_3) on the 40 timesteps between 560 and 600 steps in. It can be seen from Fig 3.3C that both the tPC model with 5 inference steps and that with fully converged inferential dynamics could achieve comparable performance to the Kalman filter. Interestingly, the estimates of tPC tend to be closer to those of the Kalman filtering, rather than the true values. Although the tPC model with a single inference step (corresponding to the neural implementation in Fig 3.2C) has worse tracking performance, it is able to estimate a smoothed version of the trajectory of the system state. We hypothesize that the smoothed estimate with a single inference step is likely due to the fact that the tPC model does not completely converge in 1 iteration, and so does not completely optimize its estimate on every timestep, with the effect that the estimate is less sensitive to new information and effectively averages over recent experiences rather than optimally solving each one independently. A similar performance comparison is obtained on the position (x_1) and velocity (x_2) and is shown in Appendix B.

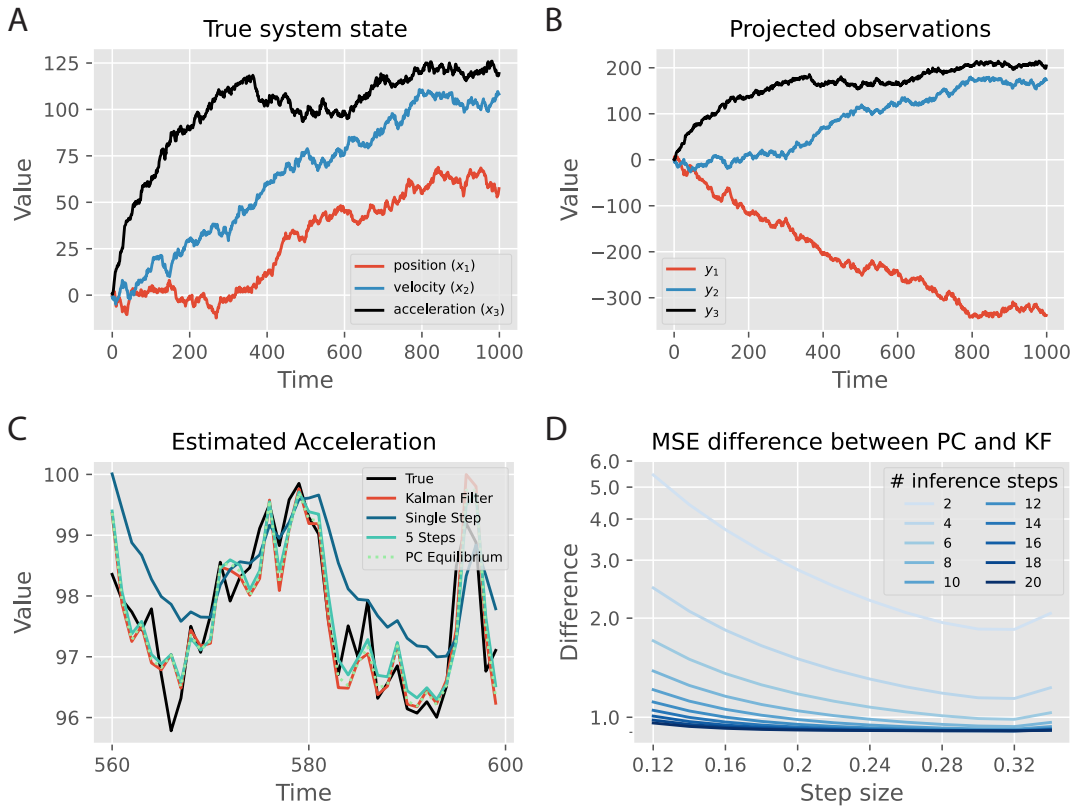

Figure 3.3: **The tracking task and the impact of inference step size and the number of inference steps on performance.** A. The dynamics of the true hidden state are represented as a 3-dimensional vector at each time step, with entries corresponding to position (x_1), velocity (x_2) and acceleration (x_3). B. The projected noisy observations from the true system state in A. C: Estimates of the acceleration with different models, zoomed in at the interval between 560 and 600 time steps. D: MSE difference between tPC and Kalman filter, with varying numbers of inference steps and step sizes for predictive coding. PC stands for temporal predictive coding and KF stands for Kalman filter. All values are with arbitrary units (a.u.).

To quantify the effect of the number of inference iterations and integration step size $\frac{\Delta t}{\tau}$ upon performance, in Fig 3.3D we plotted the MSE difference between predictive coding models with various inference iterations and inference step sizes and the Kalman filter, which is the optimal solution to the tracking problem. The MSE is calculated as the mean squared difference between the estimated system state $\hat{\mathbf{x}}_k$ and true state $\mathbf{x}_k$, averaged across time steps and trials. We find that with a small number of inference steps, the performance of the tPC model is worse, indicating that additional steps of inference aid the tracking performance of the algorithm. Moreover, although increasing the step size will initially improve the performance, the MSE will start to increase if the step size is too large. It can also be seen that with more

inference steps and appropriate step sizes, the performance of tPC will be able to approximate that of the (optimal) Kalman filter.

3.3.3 Learning the synaptic weights

In the previous investigations, we fixed the parameters $\mathbf{W}$, $\mathbf{B}$ and $\mathbf{F}$ to the true values and only performed the inference dynamics. However, in many cases, simply inferring the hidden states of the world is not enough because we cannot assume that we know a-priori the structure of the dynamics or observation functions of the world. That is, in most real-world situations, the $\mathbf{W}$, $\mathbf{B}$, and $\mathbf{F}$ matrices are unknown. Instead, we must *learn* these matrices from observations. In the tPC model, we have a natural Hebbian plasticity-based learning rule which we can use to learn these matrices directly (Eq. 3.11). Here, we investigate how learning these parameters affects the performance of the tPC model. Specifically, we compare three different ways of setting the values for $\mathbf{W}$ and $\mathbf{F}$: 1) fixing them to the true values used for generating the data; 2) learning them using Eq. 3.11 and Algorithm 1; 3) fixing them to random values. We then examine the performance of these models on two levels, the latent state level ($\mathbf{x}$) and the observation level ($\mathbf{y}$), by measuring how well the model estimates the activities on both levels. It is worth noting that the observation estimates are calculated by performing a forward pass at each time step i.e., $\hat{\mathbf{y}}_k = \mathbf{F}(\mathbf{W}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k)$, where the values of $\mathbf{W}$ and $\mathbf{F}$ are obtained at each time step k via the aforementioned three approaches. The results are shown in Fig 3.4.

For this set of results, we use 20 inference steps with a step size 0.2 to get the optimal performance for the tPC models, based on Fig 3.3D. Fig 3.4A shows that, while our tPC estimates the latent state well with true $\mathbf{W}$ and $\mathbf{F}$, when asked to learn the parameters, the model fails to accurately estimate the latent state. Quantitatively, as shown in Fig 3.4C, the estimation MSE of the learning model on the state level is similar to tPC with totally random parameters, which is much higher than those of the Kalman filter and tPC with true parameters. On the other hand, however, we find that the model learning $\mathbf{W}$ and $\mathbf{F}$ can accurately estimate the observations even with the incorrectly estimated latent state (Fig 3.4B). This effect arises because the problem of inferring the true hidden state from the data is fundamentally underdetermined. There are many possible hidden states that, given a flexible learned mapping, could result in an identical predicted observation. Importantly, despite inferring a different representation of the hidden state, the network is able to learn $\mathbf{W}$ and $\mathbf{F}$ that correctly predicts the incoming observations (Fig 3.4D).

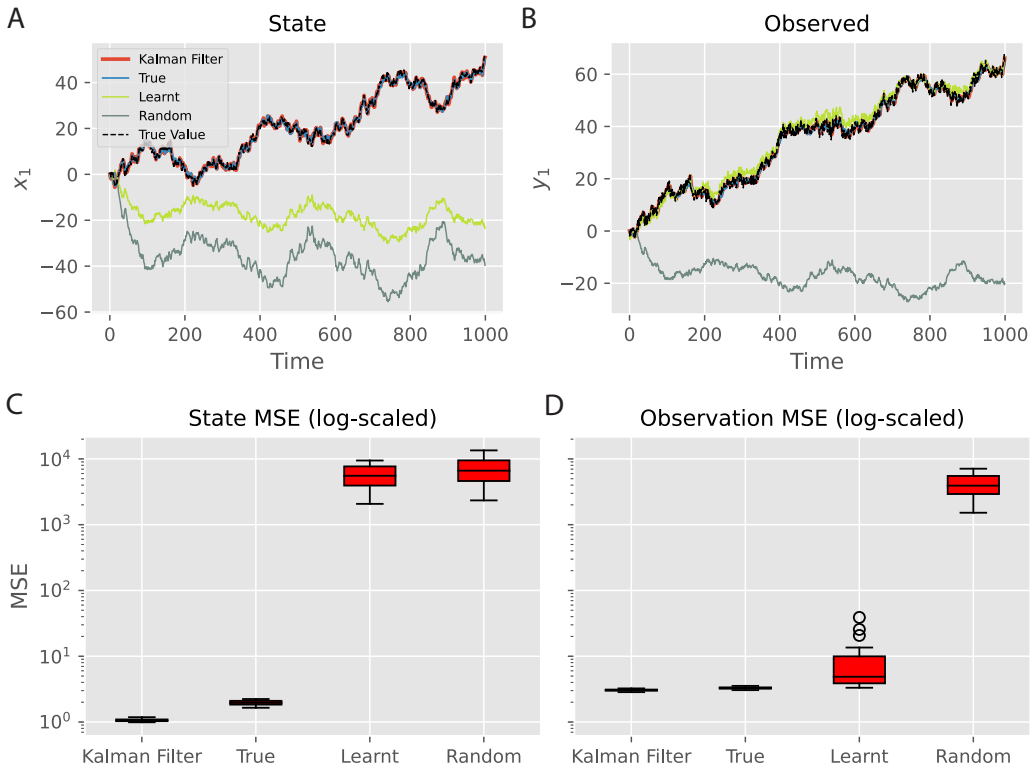

Figure 3.4: **Effects of learning parameters W and F.** A, B: Estimation of the state and observation trajectories respectively by different models. 'True', 'learned' and 'Random' denote the predictive coding model with true, learned and random $\mathbf{W}$ and $\mathbf{F}$ respectively. Only the first dimension of the latent and observation is shown for simplicity. The other two dimensions have similar performance. C, D: MSE of the predictions on the hidden and observation levels respectively. Boxplots were obtained with 40 trials for each model. Both x and y are with arbitrary units (a.u.).

3.3.4 Learning the noise covariance matrices

For all our experiments above, we have used $\mathbf{\Sigma}_x = \mathbf{\Sigma}_y = \mathbf{I}$ when generating the training data, where $\mathbf{I}$ is the identity matrix. However, the noise covariance underlying a natural dynamic process may not always be identity. Although earlier works have proposed to encode the noise precision matrices $\mathbf{\Sigma}_x^{-1}$ and $\mathbf{\Sigma}_y^{-1}$ into additional connections explicitly Bogacz (2017), this approach would introduce extra complexity into the neural implementation of tPC. On the other hand, it has been shown that in the static case, the noise precision matrix can be implicitly encoded in recurrent connections similar to the $\mathbf{W}$ matrix in our tPC model (Tang et al., 2023), without needing to represent the precision matrix explicitly as in (Bogacz, 2017). Therefore, here we investigate whether $\mathbf{W}$ and $\mathbf{F}$ can encode the precision matrices of the process and observation noise respectively after learning. We used the same $\mathbf{W}$, $\mathbf{B}$ and $\mathbf{F}$ matrices

for data generation as before, but set the noise covariance matrices as:

$$\boldsymbol{\Sigma}_x = \boldsymbol{\Sigma}_y = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ or } \boldsymbol{\Sigma}_x = \boldsymbol{\Sigma}_y = \begin{bmatrix} 10 & 2 & 0.5 \\ 2 & 1 & 0.4 \\ 0.5 & 0.4 & 1 \end{bmatrix}. \tag{3.28}$$

We refer to the first case as "non-identity diagonal" and the second case as "positive definite". The choice of these covariance matrices is arbitrary, although we intentionally make one diagonal entry larger than the others to examine tPC's capability of learning such a unique structure. The training hyper-parameters for this experiment are identical to the ones used for Fig 3.4, with which the learning for $\mathbf{W}$ and $\mathbf{F}$ could also converge. As Fig 3.5A shows, when the noise covariance matrices are non-identity diagonal and positive definite, the recurrent matrix $\mathbf{W}$ is able to encode the large diagonal entry 10 in $\boldsymbol{\Sigma}_x$ into its diagonal entries. Likewise, the $\mathbf{F}$ matrix develops stronger weights to account for the larger variance of the inputs, suggesting that our tPC model can learn the noise covariance $\boldsymbol{\Sigma}_x$ and $\boldsymbol{\Sigma}_y$ into its recurrent and feedforward weights, without accounting for them in the energy function and learning/inference dynamics explicitly.

We then conducted a quantitative analysis of the impact of non-identity noise covariance in Fig 3.5B and 3.5C, similar to Fig 3.4. Here, both "Kalman Filter" and "True" use the correct $\mathbf{W}$, $\mathbf{F}$, $\boldsymbol{\Sigma}_x$ and $\boldsymbol{\Sigma}_y$, although the "True" tPC model performs inference to get the hidden states using Eqs. 3.8,3.9 and 3.10. The "learned" model has no access to all the matrices and has to learn both the generative process and the noise covariance with its weights. The results are similar to Fig 3.4: the model that learns $\mathbf{W}$ and $\mathbf{F}$ will fail to learn the correct hidden states but its observation estimates are on par with Kalman filtering, even when the noise covariance matrices are non-identity. Interestingly, we also observed slightly degraded latent estimation performance of the "True" model with non-identity noise covariance. We hypothesize that this is due to the large diagonal entry 10 affecting the inference dynamics, which can be solved using coordinate-wise inference step sizes.

Overall, these results suggest that the synaptic plasticity of tPC can encode the noise covariance in the generative process, without representing them explicitly in the dynamics and circuit implementations. It is also interesting to investigate the exact theoretical relationship between the weights and noise covariance, similar to the analytical relationship in Tang et al. (2023), and we intend to investigate it in future explorations.

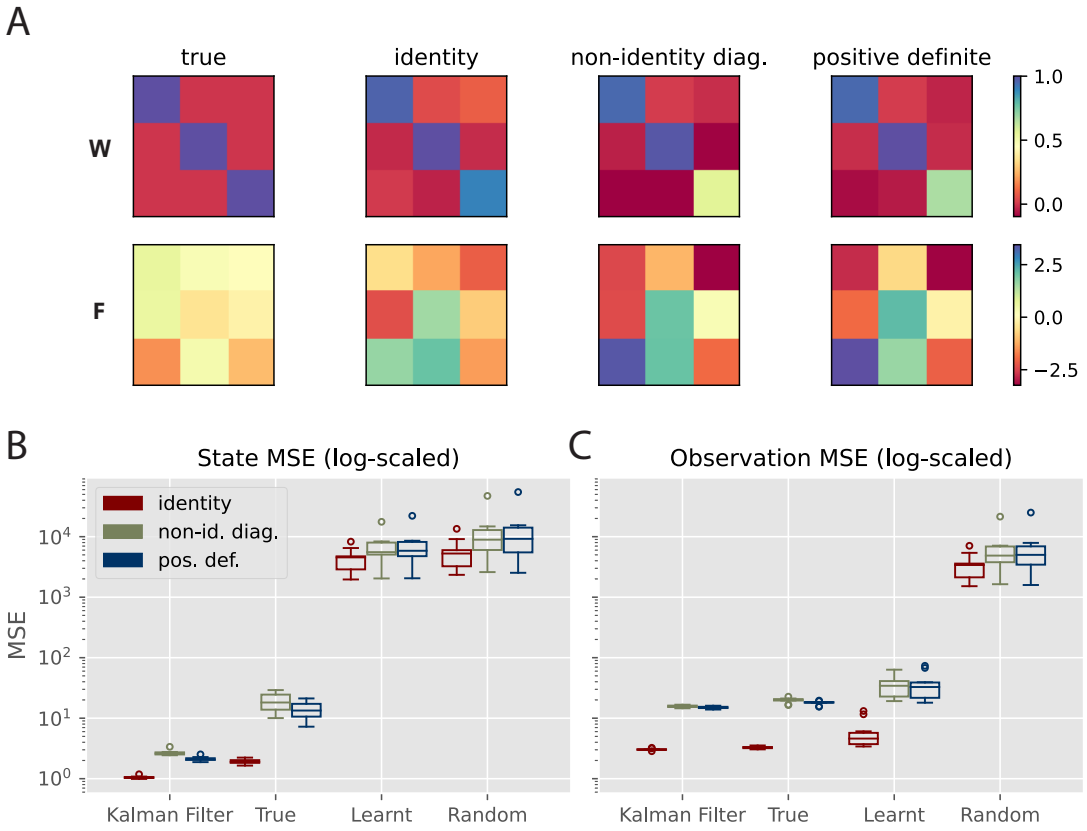

Figure 3.5: **Performance with non-identity noise covariance.** A: True and learned **W** and **F** matrices with different underlying noise covariance matrices. B, C: MSE of the predictions on the hidden and observation levels with different noise covariance matrices. Error bars obtained with 40 trials.

3.3.5 Training tPC with natural movies

Thus far, we have examined tPC in low-dimensional examples to understand its computational properties. In this section, we demonstrate that this model can also be applied to high-dimensional stimuli and provide a plausible account of how the biological visual system develops representations of dynamical inputs. To do so, we trained the tPC model on a dataset of patches extracted from movies of natural scenes. Each of the movies consisted of 50 frames of 200×200 pixels that we spatially bandpass filtered by a retina-like centre-surround filter (Olshausen and Field, 1997). We extracted 16×16 patches from the movies with the highest motion to form a dataset of 2000 moving patches of 50 frames. We also augmented the dataset with a left-right flipped version of itself to ensure a rich variety of motion directions. Two examples of the first 10 frames of the movies can be seen in Fig 3.6A. We then trained a tPC

model with 320 hidden neurons with this dataset. Specifically, due to the redundancy of information in natural scenes and neural connectivity and energy constraints, we trained a tPC model with $l1$ sparsity-constraint latent activities and forward weights $\mathbf{F}$. Therefore, the objective function we used in this set of experiments is:

$$\mathcal{F}_k = \|\mathbf{y}_k - \mathbf{F}f(\mathbf{x}_k)\|_2^2 + \|\mathbf{x}_k - \mathbf{W}f(\hat{\mathbf{x}}_{k-1})\|_2^2 + \lambda_x|\mathbf{x}_k|_1 + \lambda_F \sum_{i,j} |F_{ij}| \tag{3.29}$$

where $|\cdot|_1$ denotes vector 1-norm, and we used a linear $f(\cdot)$ in these experiments. We also assume there is no control input to the model, so $\mathbf{B} = 0$. The sparsity constraints are similar to the classical sparse coding model (Olshausen and Field, 1997) and recent temporal prediction neural network models of dynamical inputs (Singer et al., 2018, 2023). To describe the cortical processing of natural scenes more accurately, for the experiments with these natural movies, we also used a time constant $\tau = 10ms$ estimated for visual cortical neurons (Nowak et al., 2003). The natural movies used in this experiment were recorded with frame rates in the range of 20Hz to 30Hz. In our simulations, we set Δt to $0.1ms$ (and therefore the inference step size $\frac{\Delta t}{\tau} = 0.01$) and perform inference for 330 iterations to (approximately) match the 30Hz frame rate, although we did not observe any qualitative difference when we used more inference iterations to match the lower frame rates.

We then trained our adjusted model with the natural movies. In Fig 3.6B we show the (sparsity-constraint) forward weight $\mathbf{F}$ (of size 256×320), which develops Gabor-like and localized projective fields similar to those observed in mammalian primary visual cortex (Hubel and Wiesel, 1962). This finding is unsurprising as such a sparsity-constraint generative model is similar to the original predictive coding (Rao and Ballard, 1999) and sparse coding models (Olshausen and Field, 1997), which also reproduced localized Gabor filters.

To study the dynamical properties of tPC, we then performed a reverse correlation analysis (Dayan and Abbott, 2005) on the hidden neurons of the trained tPC. Specifically, after training, we supply a sequence of white noise stimuli to the model, let the hidden neurons relax according to Eq. 3.13 to develop latent representations of the stimuli, and average the stimuli giving more weights to those producing the highest response of a neuron. In particular, for each time step k and each neuron, we multiply the hidden activity $\mathbf{x}_k$ with each of the white noise stimuli from $\mathbf{y}_{k-4}$ to $\mathbf{y}_k$ and sum up the products. The weighted sum is the spatio-temporal receptive field (STRF) of this neuron (Dayan and Abbott, 2005). Formally, the STRF of the ith

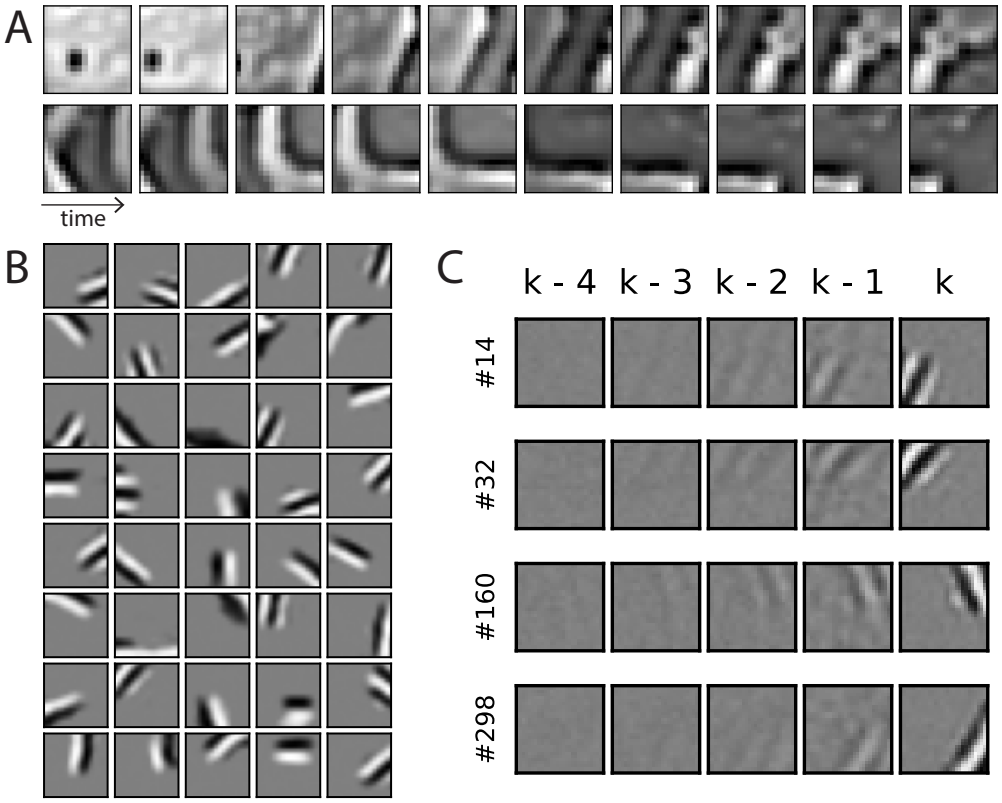

Figure 3.6: **Representations developed by the model when trained with patches from movies of dynamic natural scenes.** A: First 10 frames of 2 example training movies used in our experiments. Patches extracted from movies obtained at websites pexels.com, pixabay.com and commons.wikimedia.org. B: The projective fields **F** developed Gabor-like filters after training. C: Space-time receptive fields developed by hidden neurons of the tPC model.

hidden neuron is defined as:

$$\text{STRF}_i = \frac{1}{T-5} \sum_{k=5}^{T} x_{i,k} y_{k-4:k}, \tag{3.30}$$

where $x_{i,k}$ denotes and activity of the hidden neuron i at time step k and $y_{k-4:k}$ denotes the 5 frames of white noise preceding and including time step k. Some examples of the STRFs developed by the hidden neurons of the tPC are shown in Fig 3.6C, where each row denotes a neuron and each column a time step. Two important properties emerge: 1) The STRFs are temporally asymmetric i.e., the power of the receptive fields decays back in time. This is consistent with observations in real neurons (Ohzawa et al., 1996) and earlier computational models (Singer et al., 2018, 2023); 2) Importantly, many neurons also develop motion-sensitive STRFs, i.e., the spatial regions each neuron is responsive to shift location in the visual field over history

steps, moving in a fixed direction (right-ward for neurons 160 and 298, left-ward for neurons 14 and 32 in the shown examples). Such motion sensitivity has been observed in real neurons in early visual areas (Baker, 1990; Livingstone and Hubel, 1987) and demonstrated in earlier sparse coding models (Bogacz et al., 2000; Olshausen, 2002). Overall, these results demonstrate that the tPC model with realistic time constants matching cortical neurons can reproduce neural representations observed in the visual areas, providing a possible computational mechanism underlying the learning of such representations.

3.4 Discussion

3.4.1 Summary

In this chapter, we have analysed the recurrent predictive coding architecture for temporal prediction and filtering. This task is important because processing time-varying sequences of inputs and using them to infer dynamically changing hidden states of the world is often considered a core task of the sensory regions of the brain (Doya et al., 2007). As such, it is likely that these regions have an architecture heavily specialised for performing such filtering tasks. Here we have shown that the filtering problem can be tackled with a simple and biologically plausible algorithm with a straightforward implementation in neural circuitry.

We have derived our tPC model from first principles as a variational filtering algorithm, providing a clear algorithmic derivation in terms of gradient descents on the resulting free energy functionals. We have also proposed a direct implementation in neural circuitry which relies on only local information transmission as well as purely Hebbian plasticity that inherits the simple neural implementation of static predictive coding networks. Furthermore, we have also demonstrated that in the linear case, the algorithm is closely related to Kalman filtering, and is capable of robustly solving filtering and tracking tasks at a level close to the optimal linear solution. Moreover, and unlike the Kalman filter, we have demonstrated that our model can perform online *learning* of the parameters using Hebbian plasticity, which works rapidly and effectively in predicting the correct observations. Importantly, when trained on natural stimuli and constrained by sparse weights and activities, the tPC model develops motion-sensitive Gabor-filters of the visual scene, which is consistent with representations developed in the visual cortex.

3.4.2 Neural implementation of time

There are several interesting questions regarding the biological plausibility of the multi-step neural implementation in tPC. Initially these schemes, while they arise directly from the gradient descent derivation, appear biologically implausible for two reasons. The first is the issue of storage. Iterative schemes require the initial conditions (state estimate $\hat{\mathbf{x}}_{k-1}$) to be held fixed throughout multiple iterations, and this means that this state must be stored somewhere accessible to be utilized multiple times during the iterative inference phase. It is not clear where or how this information could be stored in the brain, especially in low-level sensory systems. This storage must be local and ubiquitous as a naive implementation of the multi-step algorithms proposed in this work would require separate storage for every single value neuron.

The second issue relates to the time it takes for an iterative algorithm to converge. Specifically, if we model the brain as receiving visual input as a continuous stream, then multiple iterations based upon a single stimulus would necessitate ignoring the data arriving in the intervals while the iterations are taking place. Moreover, an iterative approach would also take more time to update upon information newly received, which could be crucial for survival in some cases.

In the modeling results presented in this thesis, these two issues are addressed using the single-step tPC algorithm (3.2C). However, the performance of this algorithm could not perfectly match that of the iterative tPC algorithm. To achieve superior performance in temporal processing tasks, it is therefore likely that the brain may still employ some strategies to achieve a more "complete" inference. One potential solution to this issue is that the cortex may implement both an iterative and an amortized feedforward pass solution simultaneously (Tscshantz et al., 2023), similar to what has been demonstrated in machine learning (Kingma and Welling, 2013). There is also evidence for precisely this: Core visual functions can often occur within 100-200 ms (Thorpe et al., 1996; Keysers et al., 2001; Carlson et al., 2013; Thunell and Thorpe, 2019) which is too short to allow multiple steps of iterative optimization. Alternatively, neurons with different inferential speed may serve different purposes: In lower levels of the cortex, neurons operate at a lower frequency to process more refined features, whereas in higher levels such as the hippocampus, neurons perform faster inference, which is sufficient to form some high-level representation of stimuli. This is reflected in our Fig. 3.3C, where the fitted curve of the single-step algorithm is essentially a smoothed version of the True curve. Biologically, this hierarchically distributed inference speed may be achieved through different neural oscillation frequencies (Buzsaki and Draguhn, 2004; Buzsaki, 2006). In particular, the fast inference

can be achieved through hippocampal sharp wave-ripples, which serves as a compression of sensory information (Buzsáki, 2015).

3.4.3 Other temporal predictive coding models

Before our proposal of tPCN, some earlier studies have also explored predictive coding for temporal predictions. For instance, early studies (Rao, 1997; Rao and Ballard, 1997; Rao, 1999) utilized a Kalman filter combined with sparse image representations to make future predictions of visual stimuli. However, these works did not describe how the Kalman filter can be implemented in biological circuits, and how their Kalman filter-based models can be extended to the nonlinear case. More recently, Jiang and Rao (2022) trained a temporal version of predictive coding on sequences of natural video (filmed by a person walking through a forest) and observed that neurons have spatiotemporal receptive fields resembling those in the primary visual cortex. However, unlike our model, their model relies on an external hyper-network to perform temporal predictions. Temporal versions of predictive coding networks have also been extended to include multiple levels of hierarchy (Jiang and Rao, 2022; Ororbia et al., 2020). Analysis of these networks revealed that neurons on higher levels change their activity with a slower time scale than the neurons at the lower levels of the hierarchy (Jiang and Rao, 2022). It has been also demonstrated that hierarchical temporal predictive coding networks can achieve performance comparable to BPTT in standard machine learning benchmarks (Ororbia et al., 2020). However, these models require complex neural network implementations to perform these temporal tasks. It is thus an interesting future direction to see whether our tPC model, which inherits the simple neural implementation of the static predictive coding network, can present similar performance and neural responses.

A number of studies have used normative models to generate spatiotemporal receptive fields resembling those of direction-selective V1 simple cells (Bogacz et al., 2000; Olshausen, 2002; van Hateren and Ruderman, 1998; Pachitariu and Sahani, 2012, 2017; Berkes and Wiskott, 2005; Hyvärinen et al., 2003; Palm, 2012; Singer et al., 2018, 2023). Some models involved mechanisms related to predictive coding, such as sparse coding (Bogacz et al., 2000; Olshausen, 2002) and independent component analysis (van Hateren and Ruderman, 1998) and applied them to spatiotemporal stimuli. However, while the resulting receptive fields were sometimes direction-selective, they did not have the asymmetric temporal profile seen in real STRFs. Application of slowness principles to spatiotemporal stimuli can also produce direction-selective STRFs (Berkes and Wiskott, 2005; Hyvärinen et al., 2003),

but they again lack the asymmetric temporal profile. Other models have trained single-hidden-layer neural networks to perform temporal prediction on movies of natural scenes (Palm, 2012; Singer et al., 2018, 2023). This has been shown to reproduce direction-selective STRFs with an appropriate asymmetric temporal profile (Singer et al., 2018), and indeed when stacked hierarchically reproduces units resembling motion-sensitive simple, complex and pattern-motion cells (Singer et al., 2023). However, these models were trained by backpropagation, hence while they are informative about whether the cortex might have temporal prediction as a normative objective, they are agnostic about the potential learning mechanism. This thesis helps bridge this gap, demonstrating how temporal prediction can be combined with a more plausible learning mechanism to produce units with spectrotemporal receptive fields resembling those of direction-selective V1 simple cells.

3.4.4 Kalman filtering

Several earlier works have tried to approach the problem of Kalman filtering in the brain. Wilson and Finkel (2009) repurpose a line attractor network and show that it recapitulates the dynamics of a Kalman filter in the regime of low prediction error. However their model only works for a single-dimensional stimulus, does not encode uncertainty, and also only works when a linearisation around zero prediction error holds. Deneve et al. (2007) encoded Kalman filter dynamics in a recurrent attractor network. Their approach however encodes stimuli by means of basis functions, which leads to an exponentially growing number of basis functions required to tile the space as the dimensionality of the input grows. In the predictive coding approach, neurons directly encode the mean of the estimated posterior distribution, which means that the network size scales linearly with the number of dimensions. Our gradient method also completely eschews the direct computation of the Kalman gain, which simplifies the required computations significantly. Additionally, Beck et al. (2007) show that probabilistic population coding approaches can compute posteriors for the exponential family of distributions of which the Gaussian distribution is a member. However, no explicitly worked-out application of the population coding approach to Kalman filtering exists, to our knowledge. Rao and Ballard (1997) have used Kalman filtering as a model of dynamical processing in the brain, and recent works have also been interested in Kalman filtering in a biologically plausible setting (Friedrich et al., 2021). We have found that the crucial distinction between tPCN and Kalman filtering is in the representation of the model's uncertainty, mathematically represented in the posterior variance $\mathbf{\Sigma}_k$. Specifically, in the tPC model, although it represents the

two "objective" uncertainties $\mathbf{\Sigma}_x$ and $\mathbf{\Sigma}_y$ of the dynamical system it is inferring, it does not represent the uncertainty in the estimated posterior distribution (unlike the extended but inefficient model of Kutschireiter et al. (2017), which uses multiple copies of the network to create samples from the posterior distribution). Crucially, it is the assumption of the prior at each time step that prevents the tPC model from successfully propagating the posterior uncertainty through time.

Although tPCN does not optimally update and represent the dynamically changing posterior uncertainty of the agent, it has some computational advantages over Kalman filtering, which may render it more suitable for implementation in neural circuitry. The key advantage of the predictive coding approach is its computational simplicity compared to the Kalman filtering update rules which require complex matrix algebra and especially matrix inversions to compute the Kalman Gain matrix which are unlikely to be implementable in neural circuitry directly, while the predictive coding equations are simple and only require local and Hebbian updates and can be directly translated into relatively straightforward neural circuits. Moreover, as seen in Fig 3.3, the predictive coding estimate and the Kalman filter estimates of a given dynamical system end up closely converging anyway, which means that tPC networks could provide the brain an efficient and cheap way to approximate the highly effective Kalman filter using only simple circuitry.

One reason why predictive coding networks achieve performance similar to the Kalman filter is that the posterior uncertainty decays rapidly over trials (for an illustration see Fig 5A in Moeller et al. (2022)). Furthermore, for deterministic transition processes (with $\mathbf{\Sigma}_x = 0$), the posterior uncertainty decays to $\mathbf{\Sigma}_k = 0$ (Moeller et al., 2022). Therefore, as the learning progresses, the Kalman filter becomes more similar or even identical (for deterministic processes) to tPC.

Our work thus raises an interesting empirical question as to what kinds of uncertainties various agents – such as humans or animals – actually appear to represent in dynamical inference tasks. For instance, it is not clear in the literature that subjective confidence ratings are highly correlated with the true dynamical uncertainty of the decisions in a task (Navajas et al., 2017). Although the Kalman filter has been used to describe reinforcement learning (Daw et al., 2006), direct comparison with simpler reinforcement learning models did not favour the Kalman filter (Howard-Jones et al., 2010). It may be interesting, therefore, to compare predictions of the Kalman filter (and the extended model representing posterior uncertainty (Kutschireiter et al., 2017)) and the original tPC models to experimental data directly. For example, one could compare if the learning rate in reinforcement learning tasks is better described

by the Kalman Gain or the value from tPC. Another interesting line related to Kalman filtering is to compare nonlinear tPC with extended (Ribeiro, 2004) and unscented (Julier and Uhlmann, 2004) Kalman filters in nonlinear tasks. Since this work focuses on linear tPC and its properties, we leave the exploration of these directions for future investigations.

3.4.5 Generalized coordinates

Generalized coordinates, introduced into the predictive coding literature by Friston et al. (2008a), and well known in engineering practice, involve representing the n'th order derivatives of a state as additional coordinate dimensions. In effect, a point in generalized coordinates of motion to n'th order reflects the approximate trajectory of the state given by the n'th order Taylor expansion around that point. Original predictive coding models involving generalized coordinates typically required hard-coding the relationships between the coordinates, and the relative precisions between different dynamical orders (Friston, 2008; Friston et al., 2008b; Parr and Friston, 2018) which results in intricate and complex hard-coded connectivity, reducing the ultimate biological plausibility of such models. On the other hand, our tPC model can learn the $\mathbf{W}$ and $\mathbf{F}$ matrices directly from data, following Hebbian rules. It is therefore an intriguing future direction to explore whether these matrices encode the derivatives through learning.

3.4.6 Machine learning models

A further avenue for future work relates to the challenge of learning long-term dependencies which span over many time steps. This has long been a central challenge with these recurrent models, and emerges essentially due to the fact that information is permuted or lost at every step of the recurrent pass, and thus tracking dependencies across many recurrent loops becomes increasingly difficult (Hochreiter and Schmidhuber, 1997; Hochreiter, 1998; Tallec and Ollivier, 2018). Numerous solutions to this have been suggested in the literature, ranging from specially designed cell architectures that can explicitly store or pass along information unaltered (Hochreiter and Schmidhuber, 1997; Tallec and Ollivier, 2018) to having a nested hierarchy of recurrent models which allow for the propagation of information over longer and longer timescales (Koutnik et al., 2014). Recent work has also addressed the question of how biological recurrent neural networks can be trained for temporal prediction (Bellec et al., 2020). They rearranged learning dynamics of BPTT and proposed

that synapses maintain eligibility traces encoding to what extent they contributed to neural activity over time, and when combined with error signals, such traces enable effective credit assignment. Work by Lotter et al. (2016) adapted deep recurrent neural networks to perform a kind of predictive coding whereby the network was trained to predict future *prediction errors* of each layer. They demonstrated that the resulting network was capable of correctly predicting sequences of video frames. While substantially scaling up predictive coding architectures to challenging machine learning tasks, the networks of Lotter et al. (2016) diverged in many ways from classical predictive coding architectures, and also utilized many non-biologically plausible components from machine learning such as convolutional and LSTM layers as well as training their network with BPTT.

Chapter 4

Sequential Memory with Temporal Predictive Coding

4.1 Introduction

The ability to memorize and recall sequences of events with temporal dependencies is crucial for biological memory systems that also underpins many other neural processes (Tulving, 1972; Eichenbaum, 2013, 2014). For example, forming correct memories of words requires humans to memorize not only individual letters but also the sequential order following which the letters appear (e.g., "cat" and "act"). However, despite extensive research into models of *static*, temporally unrelated memories from both neuroscience and machine learning (Hopfield, 1982; Krotov and Hopfield, 2016; Ramsauer et al., 2020; Millidge et al., 2022b; Salvatori et al., 2021; Iatropoulos et al., 2022; Tang et al., 2023), computational modeling of *sequential* memory is not as developed. Existing models for sequential memory are either analytically intractable or have not yet been systematically evaluated in challenging sequential memory tasks (Wallenstein et al., 1998; Rolls, 2010; Hawkins et al., 2009; Sompolinsky and Kanter, 1986; Chaudhry et al., 2023), which hinders a comprehensive understanding of the computational mechanism underlying sequential memory, arguably a more general form of memory in the natural world than static memories.

In this chapter, we propose a novel approach to modeling sequential memory based on predictive coding (Rao and Ballard, 1999; Friston, 2003; Bogacz, 2017), a biologically plausible neural network model able to reproduce many phenomena observed in the brain (Walsh et al., 2020), which has also shown to have a close relationship to backpropagation in artificial neural networks (Whittington and Bogacz, 2017; Song et al., 2020). Using predictive coding to model sequential memory is motivated by two key factors: Firstly, neuroscience experiments and theories have suggested that

(temporal) predictive processing and memory are two highly related computations in the hippocampus, the brain region crucial for memory (Barron et al., 2020; Lisman and Redish, 2009). Secondly, the modeling of static memories (e.g., a single image) using predictive coding has recently demonstrated significant success (Salvatori et al., 2021; Yoo and Wood, 2022; Tang et al., 2023; Salvatori et al., 2022a), raising the question of whether predictive coding can also be employed to model sequential memory. To take into account the temporal dimension in sequential memory, in this chapter we adopt the temporal predictive coding (tPC) model introduced earlier in this thesis (Chapter 3), although we also introduce a simpler, single-layer version of tPC in this chapter. The contributions of this chapter can be summarized as follows:

- We show that tPC is capable of sequential memory tasks with a biologically plausible neural network implementation;
- We present an analytical result showing that the single-layer tPC can be viewed as the classical Asymmetric Hopfield Network (AHN) performing an implicit statistical whitening step during memory recall, providing a possible mechanism of statistical whitening in the brain (Duong et al., 2023; Pehlevan and Chklovskii, 2019; Golkar et al., 2022);
- Experimentally, we show that the whitening step in single-layer tPC models results in more stable performance than the AHN and its modern variants (Chaudhry et al., 2023) in sequential memory, due to the highly variable and correlated structure of natural sequential inputs;
- We show that tPC can successfully reproduce several behavioral observations in humans, including the impact of sequence length in word memories and the primacy/recency effect;
- Beyond memory, we show that our tPC model can also develop context-dependent representations (Eichenbaum, 2013; Wallenstein et al., 1998) and generalize learned dynamics to unseen sequences, suggesting a potential connection to cognitive maps in the brain (Tolman, 1948; Behrens et al., 2018).

4.2 Background: Asymmetric Hopfield Networks

Although there exist other models of sequential memory (Wallenstein et al., 1998; Rolls, 2010; Hawkins et al., 2009), these models are mostly on the conceptual level, used to provide theoretical accounts for physiological observations from the brain, and

are thus hard to analyze mathematically. Therefore, here we focus our discussion on the AHN (Sompolinsky and Kanter, 1986) and its modern variants (Chaudhry et al., 2023), which extended the HN (Hopfield, 1982) to account for sequential memory, and have a more explicit mathematical formulation. Denoting a sequence of $K+1$ patterns $\mathbf{x}_k$ $(k = 1, ..., K+1)$, $\mathbf{x}^k \in \{-1, 1\}^N$, the $N \times N$ weight matrix of an AHN is set as follows:

$$\mathbf{W}^{AHN} = \sum_{k=1}^{K} \mathbf{x}_{k+1}\mathbf{x}_k^\top \tag{4.1}$$

Notice that it differs from the static HN only by encoding the asymmetric autocovariance rather than the symmetric covariance $\sum_{k=1}^{K} \mathbf{x}_k\mathbf{x}_k^\top$, thus the name. The single-shot retrieval, which we define as R, is then triggered by a query, $\mathbf{q} \in \{-1, 1\}^N$:

$$R^{AHN}(\mathbf{q}) = \text{sgn}(\mathbf{W}^{AHN}\mathbf{q}) = \text{sgn}\left(\sum_{k=1}^{K} \mathbf{x}_{k+1}\mathbf{x}_k^\top\mathbf{q}\right) \tag{4.2}$$

The retrieval process in Eq 4.2 can be viewed as follows: the query $\mathbf{q}$ is first compared with each $\mathbf{x}_k$ using a dot product function $\mathbf{x}_k^\top\mathbf{q}$ that outputs a similarity score, then the retrieval is a weighted sum of all $\mathbf{x}_{k+1}$ $(k = 1, ..., K)$ using these scores as weights. Thus, if $\mathbf{q}$ is identical to a certain $\mathbf{x}_k$, the next pattern $\mathbf{x}_{k+1}$ will be given the largest weight in the output retrieval. Following the Universal Hopfield Network (UHN) framework (Millidge et al., 2022b), we can generalize this process to define a retrieval function of a general sequential memory model with real-valued patterns $\mathbf{x}_k \in \mathbb{R}^N$:

$$R^{UHN}(\mathbf{q}) = \sum_{k=1}^{K} \mathbf{x}_{k+1}\,\text{sep}\left(\text{sim}(\mathbf{x}_k, \mathbf{q})\right) \tag{4.3}$$

where sim is a similarity function such as dot product or cosine similarity, and sep is a separation function that separates the similarity scores i.e., emphasize large scores and de-emphasize smaller ones (Millidge et al., 2022b). When sim is dot product and sep is an identity function, we get the retrieval function of the original AHN (for binary patterns an ad-hoc sgn function can be applied to the retrievals). Chaudhry et al. (2023) have shown that it is possible to extend AHN to a model with polynomial sep function with a degree d, and a model with softmax sep function, which we call "Modern Continuous AHN" (MCAHN):

$$R^{AHN}(\mathbf{q}, d) = \sum_{k=1}^{K} \mathbf{x}_{k+1}\left(\mathbf{x}_k^\top\mathbf{q}\right)^d \tag{4.4}$$

$$R^{MCAHN}(\mathbf{q}, \beta) = \sum_{k=1}^{K} \mathbf{x}_{k+1} \operatorname{softmax}\left(\beta \mathbf{x}_k^\top \mathbf{q}\right) \tag{4.5}$$

where β is the temperature parameter that controls the separation strength of the MCAHN. Note that these two models can be respectively viewed as sequential versions of the Modern Hopfield Network (Krotov and Hopfield, 2016) and the Modern Continuous Hopfield Network (Ramsauer et al., 2020), which is closely related to the self-attention mechanism in Transformers (Vaswani et al., 2017). However, the family of AHNs has not yet been investigated in sequential memory tasks with structured and complex inputs such as natural movies.

4.3 Models

In this section, we introduce the tPC models by describing their computations during **memorization** and **recall** respectively, as well as the **neural implementations** of these computations. We describe the single-layer tPC first, and then move to the 2-layer tPC.

4.3.1 Single-layer tPC

Memorization The most straightforward intuition behind "good" sequential memory models is that they should learn the transition between every pair of consecutive patterns in the sequence, so that accurate recall of the full sequence can be achieved recursively by recalling the next pattern based on the current one. Given a sequence of real-valued patterns $\mathbf{x}_k$, $k = 1, ..., K+1$, this intuition can be formalized as the model minimizing the following loss at each time-step:

$$\mathcal{F}_k(\mathbf{W}) = \|\mathbf{x}_k - \mathbf{W} f(\mathbf{x}_{k-1})\|_2^2 \tag{4.6}$$

which is simply the squared temporal prediction error. $\mathbf{W}$ is the weight parameter of the model, and $f(\cdot)$ is a nonlinear function. Similar to static PC models (Salvatori et al., 2021; Tang et al., 2023), we assume that the model has two populations of neurons: value neurons that are loaded with the inputs $\mathbf{x}_k$ at step k, and error neurons representing the temporal prediction error $\boldsymbol{\epsilon} := \mathbf{x}_k - \mathbf{W} f(\mathbf{x}_{k-1})$. To memorize the sequence, the weight parameter $\mathbf{W}$ is updated at each step following gradient descent:

$$\Delta \mathbf{W} \propto -\frac{\partial \mathcal{F}_k(\mathbf{W})}{\partial \mathbf{W}} = \boldsymbol{\epsilon} f(\mathbf{x}_{k-1})^\top \tag{4.7}$$

and the model can be presented with the sequence for multiple epochs until $\mathbf{W}$ converges. Note that the model only has one weight parameter for the whole sequence, rather than K weight parameters for a sequence of length $K+1$.

Recall During recall, the weight matrix $\mathbf{W}$ is fixed to the learned values, and the value neurons no longer receive the correct patterns $\mathbf{x}_k$. Instead, while trying to recall the pattern $\mathbf{x}_k$ based on the query $\mathbf{q}$, the value neurons are updated to minimize the squared temporal prediction error based on the query $\mathbf{q}$ and the learned $\mathbf{W}$:

$$\mathcal{F}_k(\hat{\mathbf{x}}_k) = \|\hat{\mathbf{x}}_k - \mathbf{W}f(\mathbf{q})\|_2^2 \tag{4.8}$$

where we denote the value neurons' activities during recall as $\hat{\mathbf{x}}_k$ to differentiate it from the memorized patterns $\mathbf{x}_k$. The value neurons then perform the following inferential dynamics to minimize the loss $\mathcal{F}_k(\hat{\mathbf{x}}_k)$:

$$\dot{\hat{\mathbf{x}}}_k \propto -\frac{\partial \mathcal{F}_k(\hat{\mathbf{x}}_k)}{\partial \hat{\mathbf{x}}_k} = -\boldsymbol{\epsilon} \tag{4.9}$$

and the converged $\hat{\mathbf{x}}_k$ is the final retrieval. Note that the error neurons' activities during recall are defined as $\boldsymbol{\epsilon} := \hat{\mathbf{x}}_k - \mathbf{W}f(\mathbf{q})$, which is also different from their activities during memorization.

In the case of sequential memory, there are two types of recall. We define the first type as "online" recall, where the query $\mathbf{q}$ at each step k is the ground-truth pattern at the previous step $\mathbf{x}_{k-1}$. It is called online as these ground-truth queries can be viewed as real-time online feedback during the sequence recall. In this case, the original $\mathbf{x}_k$ will define a memory attractor as it defines an optimum of the loss, or energy in Eqs 4.6 and 4.8. The second type is referred to as "offline" recall, where $\mathbf{q}$ is the recall from the previous step i.e., $\hat{\mathbf{x}}_{k-1}$, except at the first step, where a ground-truth $\mathbf{x}_1$ is supplied to elicit recall of the whole sequence. This is called offline as there is no real-time feedback. In this case, errors from earlier steps may accumulate through time and $\mathbf{x}_k$ is no longer an ascertained attractor unless $\hat{\mathbf{x}}_{k-1} = \mathbf{x}_{k-1}$, which makes it more challenging and analogous to the replay of memories during sleep (Ólafsdóttir et al., 2018).

Neural implementation A possible neural network implementation of these computations is shown in Fig. 4.1A, which is similar to that of static PC models (Bogacz, 2017) characterized by separate populations of value and error neurons. The difference from static models is that the predictions are now from the previous time-step $k-1$. To achieve this, we assume that the value neurons are connected to the error neurons via two pathways: the direct pathway (the straight arrows between value and error neurons) and the indirect pathway through an additional population of inhibitory interneurons, which provides inhibitory inputs to the error neurons via $\mathbf{W}$. These interneurons naturally introduce a synaptic delay, such that when the inputs from

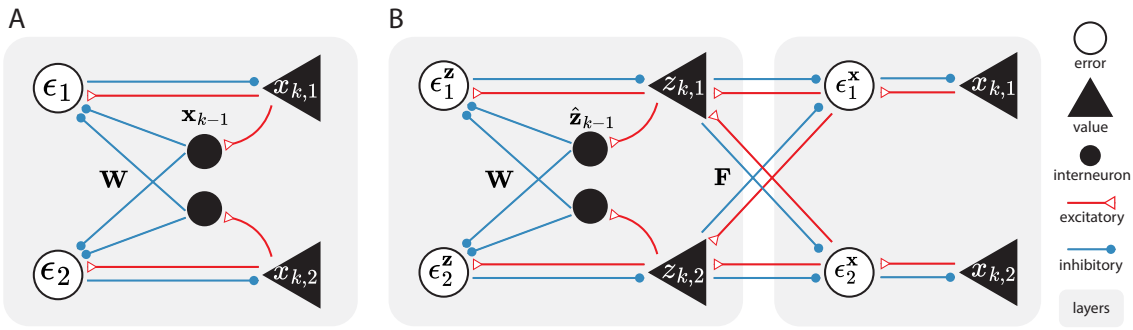

Figure 4.1: **Neural implementations of the tPC models.** A: single-layer tPC. A single layer of value neurons x makes predictions of their own future activities via a population of interneurons. Error neurons represent the temporal prediction errors. B: 2-layer tPC, where the hidden layer z performs similar temporal prediction as in single-layer tPC, but also makes hierarchical, top-down predictions on x, similar to hierarchical predictive coding.

step $k-1$ reach the error neurons through the indirect pathway, the error neurons are already receiving inputs from step k via the direct pathway, resulting in the temporal error. In our simulation, since $\mathbf{x}_k$ are modeled as discrete and frame-by-frame inputs, the temporal delay is exactly 1 time step. However, for real-world continuous dynamic inputs, this temporal difference will depend on the synaptic transmission delay, which is usually between 0.4 to 4ms (Katz and Miledi, 1965). Moreover, we assume that memory recall is a much faster process than the time-steps k so that the interneurons can hold a short working memory of $\mathbf{q}$ during the (iterative) inferential dynamics in Eq 4.9 at step k, which can be achieved by the mechanisms described in Barak and Tsodyks (2014). Notice that in this implementation, the learning rule (Eq 4.7) is Hebbian and the inference rule (Eq 4.9) is also local, inheriting the plausibility of static PC implementations (Bogacz, 2017).

4.3.2 2-layer tPC

Similar to multi-layer static PC models for memory, we can have multiple layers in tPC to model the hierarchical processing of raw sensory inputs by the neocortex, before they enter the memory system (Salvatori et al., 2021; Tang et al., 2023). We focus on the 2-layer tPC model, introduced in Chapter 3. In this model, we assume a set of *hidden* value neurons $\mathbf{z}_k$ to model the brain's internal neural responses to the sequential sensory inputs $\mathbf{x}_k$[1]. The hidden neurons make not only hierarchical, top-

[1]As mentioned in Chapter 3, here we changed the notation of hidden states to $\mathbf{z}$ and observations to $\mathbf{x}$, to reflect the original latent variable formulation of predictive coding in Chapter 1.

down predictions of the current activities in the sensory layer like in static PC models (Salvatori et al., 2021), but also temporal predictions like in the single-layer tPC. We also assume that the sensory layer $\mathbf{x}_k$ does not make any temporal predictions in this case. Thus, this 2-layer tPC can be viewed as an instantiation of the hidden Markov model (Roweis and Ghahramani, 1999).

Memorization During memorization, the 2-layer tPC tries to minimize the sum of squared errors at step k, with respect to the model parameters and the hidden activities:

$$\mathcal{F}_k(\mathbf{z}_k, \mathbf{W}, \mathbf{F}) = \|\mathbf{z}_k - \mathbf{W} f(\hat{\mathbf{z}}_{k-1})\|_2^2 + \|\mathbf{x}_k - \mathbf{F} f(\mathbf{z}_k)\|_2^2 \tag{4.10}$$

where $\mathbf{W}$ governs the temporal prediction in the hidden state, $\mathbf{F}$ is the **F**orward weight for the top-down predictions, and $\hat{\mathbf{z}}_{k-1}$ is the hidden state inferred at the previous time-step. Note that for sequential memory, no control input $\mathbf{u}$ as in Chapter 3 is needed.

During memorization, the 2-layer tPC follows a similar optimization processing to that of static hierarchical PC (Salvatori et al., 2021). It first infers the hidden representation of the current sensory input $\mathbf{x}_k$ by:

$$\dot{\mathbf{z}}_k \propto -\frac{\partial \mathcal{F}_k(\mathbf{z}_k, \mathbf{W}, \mathbf{F})}{\partial \mathbf{z}_k} = -\boldsymbol{\epsilon}^{\mathbf{z}} + f'(\mathbf{z}_k) \odot \mathbf{F}^\top \boldsymbol{\epsilon}^{\mathbf{x}} \tag{4.11}$$

where $\odot$ denotes the element-wise product between two vectors, and $\boldsymbol{\epsilon}^{\mathbf{z}}$ and $\boldsymbol{\epsilon}^{\mathbf{x}}$ are defined as the hidden temporal prediction error $\mathbf{z}_k - \mathbf{W} f(\hat{\mathbf{z}}_{k-1})$ and the top-down error $\mathbf{x}_k - \mathbf{F} f(\mathbf{z}_k)$ respectively. After $\mathbf{z}_k$ converges, $\mathbf{W}$ and $\mathbf{F}$ are updated following gradient descent on $\mathcal{F}_k$:

$$\begin{aligned} \Delta \mathbf{W} &\propto -\frac{\partial \mathcal{F}_k(\mathbf{z}_k, \mathbf{W}, \mathbf{F})}{\partial \mathbf{W}} = \boldsymbol{\epsilon}^{\mathbf{z}} f(\hat{\mathbf{z}}_{k-1})^\top; \\ \Delta \mathbf{F} &\propto -\frac{\partial \mathcal{F}_k(\mathbf{z}_k, \mathbf{W}, \mathbf{F})}{\partial \mathbf{F}} = \boldsymbol{\epsilon}^{\mathbf{x}} f(\mathbf{z}_k)^\top \end{aligned} \tag{4.12}$$

which are performed once for every presentation of the full sequence. The converged $\mathbf{z}_k$ is then used as $\hat{\mathbf{z}}_k$ for the memorization at time-step $k+1$.

Recall After learning/memorization, $\mathbf{W}$ and $\mathbf{F}$ are fixed. We also assume that the hidden activities $\mathbf{z}_k$ are unable to store memories, by resetting their values to randomly initialized ones. Thus, the sequential memories can only be recalled through the weights $\mathbf{W}$ and $\mathbf{F}$. Again, the sensory layer has no access to the correct patterns

during recall and thus needs to dynamically change its value to retrieve the memories. The loss thus becomes:

$$\mathcal{F}_k(\mathbf{z}_k, \hat{\mathbf{x}}_k) = \|\mathbf{z}_k - \mathbf{W} f(\hat{\mathbf{z}}_{k-1})\|_2^2 + \|\hat{\mathbf{x}}_k - \mathbf{F} f(\mathbf{z}_k)\|_2^2 \tag{4.13}$$

where $\hat{\mathbf{x}}_k$ denotes the activities of value neurons in the sensory layer during recall. Both the hidden and sensory value neurons are updated to minimize the loss. The hidden neurons will follow similar dynamics specified in Eq. 4.11, with the top-down error $\boldsymbol{\epsilon}^{\mathbf{x}}$ defined as $\hat{\mathbf{x}}_k - \mathbf{F} f(\mathbf{z}_k)$, whereas the sensory neurons are updated according to:

$$\dot{\hat{\mathbf{x}}}_k \propto -\frac{\partial \mathcal{F}_k(\mathbf{z}_k, \hat{\mathbf{x}}_k)}{\partial \hat{\mathbf{x}}_k} = -\boldsymbol{\epsilon}^{\mathbf{x}} \tag{4.14}$$

and the converged $\hat{\mathbf{x}}_k$ is the final retrieval. Similar to the single-layer case, if the converged $\hat{\mathbf{z}}_k$ is used for the recall at the next step directly, the recall is offline; on the other hand, if we query the model with $\mathbf{q} = \mathbf{x}_k$ i.e., the ground-truth and use the query to infer $\hat{\mathbf{z}}_k$, and then use $\hat{\mathbf{z}}_k$ for the recall at the next step, the recall is online.

Neural implementation A neural implementation of the computations above is shown in Fig. 4.1B. The hidden layer follows the same mechanism in the single-layer tPC, with interneurons introducing a synaptic delay for temporal errors and short-term memory to perform the inferential dynamics (Eq. 4.11). The connection between the hidden layer and the sensory layer $\mathbf{F}$ is modeled in the same way as in static PC models, which requires only Hebbian learning and local computations (Bogacz, 2017). Note that this implementation is essentially the same as Fig. 3.2A, although the neurons storing the inference result $\hat{\mathbf{z}}_{k-1}$ have a more specific neuron type i.e., interneurons. The memorization and recall pseudocode for the tPC models is provided in Appendix C.

4.4 Results

4.4.1 Theoretical relationship to AHNs

We first develop a theoretical understanding of single-layer tPC by relating it to AHNs:

Theorem 4 *Assume, without loss of generality, a sequence of memories* $\mathbf{x}_k \in \mathbb{R}^N$ *(*$k = 1, ..., K+1$*) with zero mean. With an identity nonlinear function* $f(x) = x$ *in*

Eq. 4.6, the retrieval of the single-layer tPC with query $\mathbf{q}$*, defined as* $R^{tPC}(\mathbf{q})$*, can be written as:*

$$R^{tPC}(\mathbf{q}) = \sum_{k=1}^{K} \mathbf{x}_{k+1}(\mathbf{M}\mathbf{x}_k)^\top \mathbf{M}\mathbf{q} \tag{4.15}$$

where M *is an empirical whitening matrix such that:*

$$\langle \mathbf{M}\mathbf{x}_k(\mathbf{M}\mathbf{x}_k)^\top \rangle_k = \mathbf{I}_N \tag{4.16}$$

where $\langle\cdot\rangle_k$ *is the expectation operation over* $\mathbf{x}_k$*'s.*

Proof of this property is provided in Appendix C. Essentially, this property implies that our single-layer tPC, in its linear form, can be regarded as a special case of the UHN for sequential memories (Eq. 4.3), with a "whitened dot product" similarity function where the two vectors $\mathbf{x}_k$ and $\mathbf{q}$ are first normalized and decorrelated (to have identity covariance $\mathbf{I}_N$) before the dot product. AHNs, on the other hand, calculate the dot product directly (due to its different energy function structure to tPC, as shown in Appendix C). Biologically, this property provides a possible mechanism of statistical whitening in the brain that, unlike earlier models of biological whitening with explicit objectives (Pehlevan and Chklovskii, 2019; Golkar et al., 2022; Duong et al., 2023), performs this computation *implicitly* via the circuit shown in Fig. 4.1 that minimizes the temporal prediction errors.

4.4.2 Experimental comparison to AHNs

To understand how the whitening step affects the performance in sequential memory tasks, we compare our single-layer tPC with the family of AHNs exeperimentally. To ensure consistency to Property 4 above, we use an identity nonlinearity $f(x) = x$ for all these experiments. Empirically, we found that using a `tanh` nonlinearity makes subtle differences irrelevant to our main discussion in this chapter, and we discuss it in Appendix C.

Polynomial AHNs We first compare tPC with polynomial AHNs (Eq. 4.4) in sequences of uncorrelated binary patterns, where AHNs are known to work well (Sompolinsky and Kanter, 1986; Chaudhry et al., 2023). We plot their sequence capacity P_{max} against the number of value neurons of the models i.e., the pattern dimension N. Here, P_{max} is defined as the maximum length of a memorized sequence, for which the probability of incorrectly recalled bits is less than or equal to 0.01. Fig. 4.2A shows that the capacity of our single-layer tPC is greater than that of the original

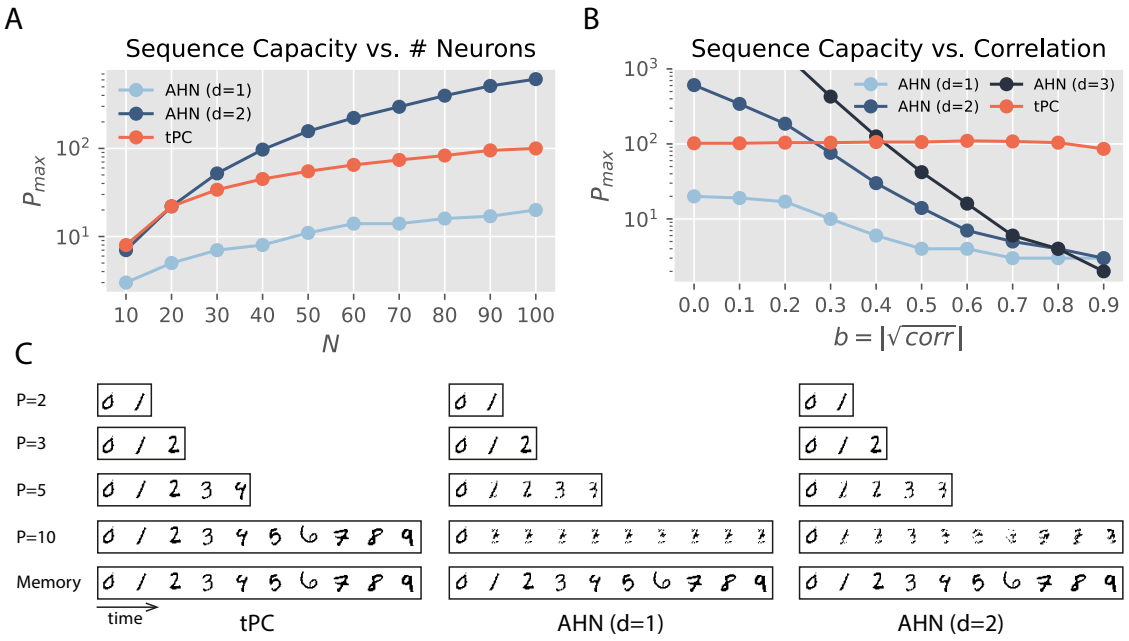

Figure 4.2: **Comparison between single-layer tPC and AHNs.** A: Capacity of models with uncorrelated binary patterns. B: Capacity of models with binary patterns with increasing feature correlations. C: Recall performance with sequences of binary MNIST digits.

AHN ($d = 1$) but smaller than that of a quadratic AHN ($d = 2$). Notice that the single-layer tPC has an identity sep function like the original AHN. Therefore the whitening operation has indeed improved the performance of the dot product sim function. Inspired by the decorrelation effect of statistical whitening, we then generated binary patterns with $N = 100$ *correlated* features, with a parameter b controlling the level of correlation ($b = |\sqrt{\text{correlation}}|$). The approach that we followed to generate the correlated patterns is provided in Appendix C. As shown in Fig. 4.2B, as the correlation increases, all AHNs up to $d = 3$ suffer from a quick decrease of capacity P_{max}, whereas the capacity of single-layer tPC almost remains constant. This observation is consistent with the theoretical property that the whitening transformation essentially decorrelates features such that patterns with any level of correlation are regarded as uncorrelated in tPC recall (Eq. 4.15). This result also explains the comparison in Fig. 4.2A: although the patterns generated in this panel are theoretically uncorrelated, the small correlation introduced due to experimental randomness will result in the performance gap between AHN and single-layer tPC.

We then investigate the performance of these models with sequences of binarized MNIST images (LeCun et al., 2010) in Fig. 4.2C. It can be seen that the AHNs with

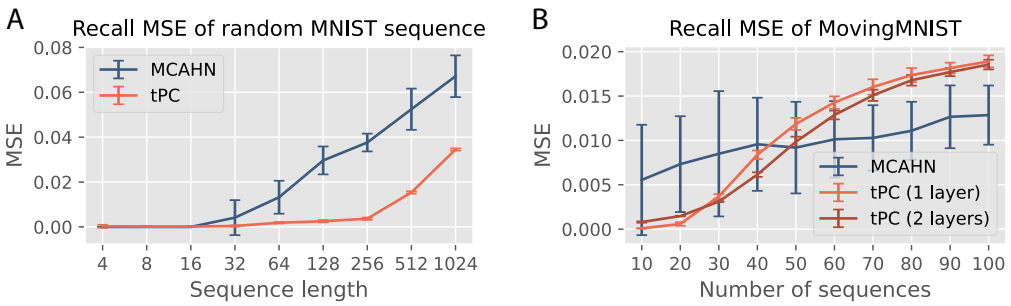

Figure 4.3: A: Recall MSE of MNIST sequences with increasing length; B: Recall MSE of MovingMNIST sequences of a fixed length 10 but with an increasing number of sequences. Error bars obtained with 5 seeds.

$d = 1$ and $d = 2$ quickly fail as the sequence length P reaches 3, whereas the single-layer tPC performs well. These results suggest that our tPC with a whitening step is superior to simple AHNs due to the inherently correlated structure of natural inputs such as handwritten digits. For all the experiments with binary patterns, we used the online recall mentioned above, and polynomial AHNs already fail in this simpler recall scenario.

MCAHN Due to the quick failure of AHNs with a polynomial sep function, we now compare our single-layer tPC with the MCAHN (Eq. 4.5). In static memory tasks, it is known that the softmax separation leads to exponentially high capacity, especially when β is high (Demircigil et al., 2017; Millidge et al., 2022b). In this chapter we use $\beta = 5$ for all MCAHNs. We first compare the performances of our single-layer tPC model and an MCAHN on random sequences of MNIST digits with varying lengths. Here we trigger recalls with online queries. Fig. 4.3A shows that the performance of our single-layer tPC, measured as the mean squared error (MSE) between the recalled sequence and the ground truth, is better than that of the MCAHN, further demonstrating the usefulness of the implicit whitening in our model.

Despite the superior performance of our model in this task, we note that random sequences of MNIST images are not naturally sequential inputs i.e., there are no sequential dynamics underlying them. We thus examine the models on the MovingMNIST dataset (Srivastava et al., 2015). Each video in this dataset consists of 20 frames of 2 MNIST digits moving inside a 64×64 patch. Due to the fixed sequence length, for experiments with MovingMNIST we vary the total number of sequences to memorize and fix the sequence length to the first 10 frames of the videos. The performance of the models is shown in Fig. 4.3B. On average, the recall MSE of MC-

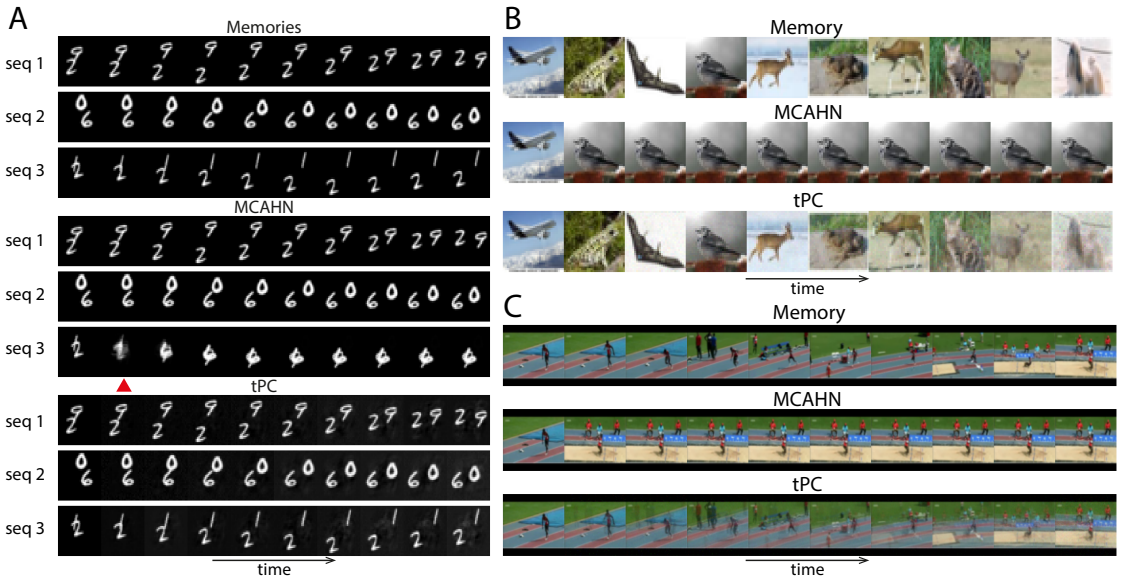

Figure 4.4: **Visual results of offline memory recall with 3 datasets.** A: MovingMNIST. B: CIFAR10. C: UCF101.

AHN has a slower increase than that of the single-layer tPC as the total number of sequences increases. However, the performance of MCAHN has very large variations across all sequence numbers. To probe into this observation, we visually examined 3 examples of the MovingMNIST movies recalled by MCAHN and our single-layer tPC in Fig. 4.4A, when the total number of sequences to memorize is 40. MCAHN produces very sharp recalls for the first 2 example sequences, but totally fails to recall the third one by converging to a different memory sequence after the red triangle in Fig. 4.4A. On the other hand, the recall by single-layer tPC is less sharp but stably produces the correct sequence. This phenomenon can be understood using the UHN framework (Millidge et al., 2022b): in Eq. 4.5, when β is large, the softmax separation function used by MCAHN will assign a weight close to 1 to the memory whose preceding frame is most similar to $\mathbf{q}$ (measure by dot product), and weights close to 0 to all other memories, which results in the sharp recall. In contrast, our single-layer tPC model uses an identity separation function that fails to suppress the incorrect memories, resulting in blurry recalls. Importantly, however, the failure of MCAHN in sequence 3 in Fig. 4.4A suggests that there are "strong attractors" in the memories with an undesirable advantage in dot product similarity which resulted in the large variance in the numerical results, and is addressed by the whitening step in single-layer tPC as it effectively normalized the patterns before dot product.

The importance of whitening in tPC is further demonstrated in our experiments

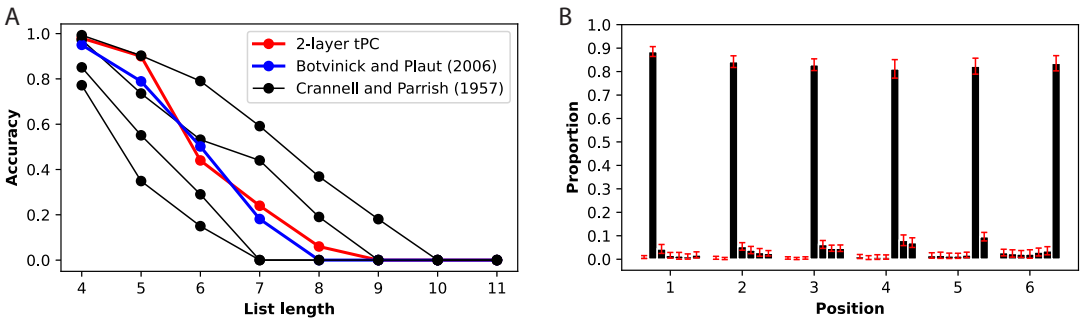

Figure 4.5: **Replicating behavioral data with tPC.** A: Experimental data from Crannell and Parrish (1957) that studies the impact of sequence length on serial recall of English words and the replications by Botvinick and Plaut (2006) and tPC. B: tPC replicates the primacy/recency effects in serial recall experiments (Henson, 1998).

with random sequences of CIFAR10 (Krizhevsky et al., 2014) images and movies from the UCF101 (Soomro et al., 2012) dataset, shown in Figs. 4.4B and C. When recalling these colored sequences, MCAHN can easily converge to strong attractors preceded by frames with many large pixel values that lead to large dot products e.g., the third image in the CIFAR10 example with bright backgrounds, and the penultimate frame in the UCF101 example with a large proportion of sand, which give their subsequent frames large similarity scores. This problem is consistent with earlier findings with static memories (Salvatori et al., 2021), and is circumvented in our single-layer tPC model with the whitening matrix $\mathbf{M}$ normalizing the pixel values across the sequence, yielding the correct and more stable memory recalls. However, they are less sharp due to the identity separation function. For all the experiments in Fig. 4.3B and Fig. 4.4, we used offline recall to make the tasks more challenging and more consistent with reality. Results with online recalls are shown in Appendix C.

4.4.3 Comparison to behavioral data in sequential memory experiments

We further demonstrated the biological relevance of tPC by comparing it to data from behavioral experiments in sequential memory. In Fig 4.5A, our 2-layer tPC is compared with Crannell and Parrish (1957) study on sequence length's impact on serial recall of English words. Using one-hot vectors to represent letters (i.e., each "word" in our experiment is an ordered combination of one-hot vectors of "letters", a minimal example of a word with 3 letters being: $[0, 1, 0], [1, 0, 0], [0, 0, 1]$), we demonstrate accuracy as the proportion of perfectly recalled sequences across varying lengths. Our

model aligns consistently with experimental data in Crannell and Parrish (1957) as well as the recurrent network model by Botvinick and Plaut (2006), displaying a sigmoidal accuracy drop with increasing sequence length.

Fig 4.5B introduces a qualitative comparison to Henson (1998) experimental data, examining primacy/recency effects in serial recall of letters. These effects involve higher accuracy in recalling early (primacy) and late (recency) entries in a sequence, with the recency effect slightly weaker than the primacy effect. Using one-hot vectors and a fixed sequence length 7 (6 positions are shown as the first position is given as the cue to initiate recall in our experiment), we visualize recall frequency at different positions across simulated sequences (100 repetitions, multiple seeds for error bars). Each bar in Fig 4.5B indicates the frequency of an entry at a particular position being recalled at each position. Our 2-layer tPC reproduces primacy/recency effects, albeit weaker than the model in Henson (1998) and previous models (Botvinick and Plaut, 2006). Additionally, the model tends to recall neighboring entries upon errors, echoing Henson's data. We attribute the weaker effects to tPC's memory storage in weights, which leads to overall improved performance across positions. Details for these experiments are shown in Appendix C.

4.4.4 tPC develops context-dependent representations

In Fig. 4.3, we plotted the recall MSE of the 2-layer tPC model in MovingMNIST, which is similar to that of the single-layer tPC. The close performance of these models raises the question of whether and when the hidden layer and hierarchical processing of sequential inputs are necessary. Inspired by earlier neuroscience theories that the hippocampus develops neuron populations signaling the sensory inputs and the context of the inputs separately (Eichenbaum, 2014; Wallenstein et al., 1998), we hypothesize that the hidden neurons in our model represent *when* an element occurs in a memory sequence i.e., its context. We thus designed a sequential memory task with aliased, or repeating inputs at different time-steps (Whitehead and Ballard, 1991; Rao, 1999). An example of such an aliased sequence can be seen in Fig. 4.6A, where the second and the fourth frames of a short MNIST sequence are exactly the same ("2"). Recalling such a sequence is inherently more challenging as the models have to determine, when queried with the "2" during recall (either online or offline), whether they are at the second or the fourth step to give the correct recall at the next step ("1" or "3"). As can be seen in Fig. 4.6A, both MCAHN (which can be regarded as a single-layer network (Millidge et al., 2022b)) and single-layer tPC fail in this task, recalling an average frame of "1" and "3" after the aliased steps, whereas the 2-layer

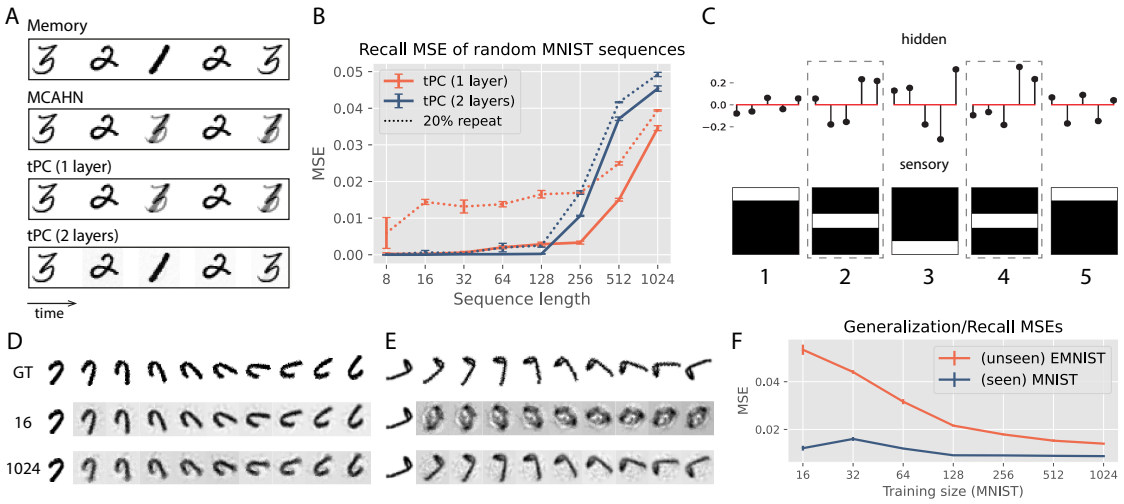

Figure 4.6: **Context representation and generalization of tPC.** A: A simple aliased example with MNIST. B: Numerical investigation into the impact of aliased or repeating elements on model performance. C: Different latent representations of aliased inputs by the 2-layer tPC. D/E: Recall/generalization of sequences with rotational dynamics. "GT" stands for ground truth and 16 and 1024 are the numbers of training sequences. F: Recall and generalization MSE of seen and unseen rotating MNIST images. Error bars obtained with 5 seeds.

tPC can recall the correct sequence. We then conducted a numerical investigation into this problem. We first plotted the (online) recall MSEs of the single- and 2-layer tPC models in random MNIST sequences (sampled from the training set) of varying lengths, which are shown as the solid lines in Fig. 4.6B. We then randomly replaced 20% (rounded up to the closest integer) of the elements in these sequences with a single digit from the test set of MNIST so that each sequence now has 20% repeating i.e., aliased elements, and plotted the recall MSEs as the dotted lines in Fig. 4.6B. The result suggests that sequences with aliased inputs affect the single-layer model much more than it affects the 2-layer one, producing significantly larger MSEs than recalls without aliased inputs. It is worth noting that aliased inputs are ubiquitous in natural sequential memories. In Appendix C, we provide a natural example from the UCF101 dataset where the 2-layer tPC successfully de-aliased repeating inputs whereas the MCAHN and single-layer tPC failed.

To further understand the mechanism underlying the 2-layer tPC with aliased memories, we used a simpler synthetic sequential memory shown in Fig. 4.6C bottom, where a white bar moves first down and then up in a 5×5 frame so that the steps 2 and 4 are aliased (Rao, 1999). We then trained a 2-layer tPC with a hidden size 5 to

memorize this sequence and queried it offline. The smaller hidden size allows us to plot, when the recall dynamics (Eq. 4.11) have converged, the exact hidden activities in Fig. 4.6C top, where each vertical line represents the activity of a hidden value neuron $\hat{\mathbf{z}}_k$. As can be seen in the circled time-steps, the 2-layer model represents the aliased inputs differentially in its hidden states, which helps it recall the next frame correctly. This property is consistent with early observations in neuroscience that when memorizing sequences, the hippocampus develops a conjunction of neurons representing individual inputs, as well as neurons signaling the temporal context within which an individual appears (Eichenbaum, 2014; Wallenstein et al., 1998). In our simple 2-layer model, the "sensory" layer represents individual inputs, whereas the hidden layer plays the role of indexing time and context.

4.4.5 tPC generalizes learned dynamics to unseen sequences

After memorizing a large number of training sequences sharing underlying dynamics, can tPC generalize the dynamics to unseen sequences? In this experiment, we train the 2-layer tPC with sequences of rotating MNIST digits of identical rotating dynamics (i.e., the digits are rotating towards a fixed direction with a fixed angle at each time step) and vary the number of training sequences ("training size"). An example of these rotating MNIST digits can be seen in Fig 4.6D, row "ground truth". The model's performance is assessed by its ability to rotate both seen MNIST digits and unseen EMNIST letters. For small training sizes (16), tPC can recall seen rotating digits but struggles with generalizing to unseen letters (Fig 4.6D and E, second row). Increasing the training size to 1024 improves generalization, evident in clearer rotating sequences (Fig 4.6D and E, bottom row). Panel F quantitatively confirms this trend: the generalization MSE on unseen EMNIST drops as MNIST training size increases, indicating the model learns the underlying dynamics. Interestingly, the recall MSEs for seen MNIST sequences also decrease due to the model extracting rotational dynamics from the larger training set, differing from the behavior observed in random MNIST sequences (Fig 4.3A). Generalization and the capability of developing contextual representations that disambiguate aliased inputs are two critical functions underlying the flexible behavior of animals, thus connecting our tPC to models of cognitive maps in the hippocampus and related brain regions (Whittington et al., 2020).

4.5 Discussion

4.5.1 Summary

Inspired by experimental and theoretical discoveries in neuroscience, in this chapter we have examined tPC models in sequential memory tasks. We have shown that our tPC model can memorize and recall sequential inputs such as natural movies, and performs more stably than earlier models based on Asymmetric Hopfield Networks. We have also provided a theoretical understanding of the stable performance of the tPC model, showing that it is achieved by an additional statistical whitening operation that is missing in AHNs. Importantly, this whitening step is achieved implicitly by a plausible neural circuit performing local error minimization. Moreover, our 2-layer tPC has exhibited representational and behavioral properties consistent with biological observations, including contextual representations and generalization. Overall, our model has not only provided a possible neural mechanism underlying sequential memory in the brain but also suggested a close relationship between PC and HN, two influential computational models of biological memories. Future directions include systematic investigations into tPC with more than 2 layers and modelling cognitive maps with tPC.

4.5.2 Relationship to AHNs

In this chapter we have established a formal relationship between the retrieval functions of single-layer tPCN and AHN. We also demonstrated that tPC outperforms AHN with a linear separation function but under-performs AHN with a polynomial separation function. An interesting question is thus whether there exists an analytical expression for the capacity of tPCN in sequential memory, similar to what was derived for AHNs in Chaudhry et al. (2023).

4.5.3 Relationship to other models of sequential memory

In addition to using Hopfield Networks for sequential memory, other models are rooted in anatomical and physiological observations. Models such as those proposed by Jensen et al. (1996) and Mehta et al. (2000) suggest hippocampal sequential memory encoding via neuronal firing chains. Other frameworks highlight the role of contextual representations in sequential memory (Wallenstein et al., 1998; Levy, 1989), successfully reproducing phenomena such as the recency and contiguity effects observed in free recall experiments (Howard and Kahana, 2002). Recurrent spiking network

models also exhibit sequential memory capabilities (Eliasmith et al., 2012). However, unlike predictive coding, these models do not operate under a unified computational framework—namely the Bayesian Brain hypothesis—that could be generalized across different brain regions. Nevertheless, it would be interesting to investigate if properties replicated in these physiologically more realistic models can naturally arise as a result of performing Bayesian inference via predictive coing. Broadly, aligning predictive coding models more closely with anatomical and neurophysiological structures supporting sequential memory remains a promising direction for future exploration.

Chapter 5

Learning Grid Cells by Predictive Coding

5.1 Introduction

Our brain contains a rich set of neural representations of space that help us navigate in an ever-changing world. These include hippocampal place cells (O'Keefe, 1976), which fire when an animal is at a specific spatial position, and grid cells observed in the medial entorhinal cortex (MEC) (Hafting et al., 2005), which fire when an animal occupies multiple positions on a hexagonal or triangular grid. Grid cells have been observed across various species (Fyhn et al., 2008; Yartsev et al., 2011; Doeller et al., 2010), and their remarkable regularity has raised extensive interest in the computational mechanism underlying their emergence. Earlier models have focused on how mechanisms, such as membrane potential oscillation (O'Keefe and Burgess, 2005; Hasselmo et al., 2007) and specialized recurrent connectivity, can generate grid-like firing patterns (Fuhs and Touretzky, 2006; Burak and Fiete, 2009). More recently, research has shown that grid cells can emerge in recurrent neural networks (RNNs) trained using backpropagation through time (BPTT) for path integration tasks. The models are trained to predict their current location by integrating velocity inputs (Cueva and Wei, 2018; Banino et al., 2018; Whittington et al., 2020; Sorscher et al., 2023), providing a normative, task-driven account of the computational problem that the MEC grid cells address. However, the process by which the MEC circuit acquires, or *learns* the grid cells in a biologically plausible way has been largely neglected, despite the fact that grid cells are known to be learned, rather than hardwired at birth (Langston et al., 2010; Wills et al., 2010). Although models for synaptic plasticity underlying grid cells do exist (e.g. Weber and Sprekeler (2018)), they are highly

specialized for learning grid cells, and do not capture of dynamic nature of the learning process through navigation.

In this chapter, we directly tackle the learning problem underlying the emergence of grid cells using predictive coding. Our approach to modeling grid cell emergence through predictive coding is motivated by three key factors: Firstly, the predictive coding algorithm can be implemented in predictive coding networks (PCNs) with local computations and Hebbian plasticity (Bogacz, 2017), making it more biologically plausible than learning rules such as backpropagation. Secondly, PCNs have been successful in replicating representations in other regions of the brain, such as the visual cortex (Rao and Ballard, 1999; Olshausen and Field, 1996; Millidge et al., 2024). Thirdly, PCNs have demonstrated the ability to perform hippocampus-related functions, such as associative and sequential memories (as discussed in previous chapters and in Salvatori et al. (2021), Tang et al. (2023) and Tang et al. (2024)).

The primary contribution of this chapter is to demonstrate for the first time that grid cells naturally emerge in PCNs trained to represent spatial inputs with biologically plausible plasticity rules. In this chapter we:

- show that hexagonal grid cells develop as the latent representations of place cells in classical PCNs (Rao and Ballard, 1999; Olshausen and Field, 1996) with sparse and non-negative constraints;
- train temporal predictive coding network (tPCN) (Chapter 3 and Millidge et al. (2024)) in path integration tasks and observe that the latent activities of the tPCN develop hexagonal, grid-like representations, similar to what has been discovered in RNNs;
- develop an understanding of grid cell emergence in tPCN, by showing analytically that the Hebbian learning rule of tPCN implicitly approximates truncated BPTT (Williams and Peng, 1990);
- show that tPCN can robustly develop grid cells under different architectural choices, and even without velocity inputs in path integration.

Overall, our results present an effective and plausible learning rule for hexagonal grid cells in the MEC based on predictive coding. We offer a novel extension of predictive coding theory, which has traditionally been used to model visual representations (Rao and Ballard, 1999; Olshausen and Field, 1996), to encompass spatial representations in the MEC. Our findings therefore offer a novel understanding of

how a single, unified learning algorithm can be employed by different brain regions to represent inputs of various levels of abstraction.

5.2 Background: Computational Models of Grid Cells

The periodicity of grid cells inspired early models of grid cells based on membrane potential oscillations, where the periodic firing of grid cells results naturally from the interference between somatic and dendritic oscillators in MEC pyramidal neurons (O'Keefe and Burgess, 2005; Hasselmo et al., 2007). These models were subsequently extended to incorporate multiple networks of oscillatory neurons (Zilli and Hasselmo, 2010). However, these models lack biological plausibility as they require an unrealistically large number of networks (Giocomo et al., 2011). Another major family of models leverages the recurrent attractor networks and obtains grid firing patterns (Fuhs and Touretzky, 2006; Burak and Fiete, 2009; Ocko et al., 2018) by hand-tuning the recurrent connectivity to form a center-surround structure. These networks perform robust and accurate path integration (Burak and Fiete, 2009) and can explain experimental observations such as the deformation of grid cells in irregular environments (Ocko et al., 2018). However, as pointed out by Sorscher et al. (2023), these models lack an explanation for the underlying spatial task that gives rise to the specific recurrent connectivity.

To address this gap, recent studies have explored the question *'If grid cell is the answer, what is the question?'*. Dordek et al. (2016) showed that grid cells emerge as the non-negative principal components of place cells, while Stachenfeld et al. (2017) proposed that grid cells form a basis for predicting future observations. Other studies have focused on the multi-modularity of grid cells by optimizing biologically constrained objective functions (Dorrell et al., 2022; Schaeffer et al., 2024). Notably, multiple research tracks have found that RNNs trained to perform path integration tasks will develop hexagonal grid representations in their latent states (Cueva and Wei, 2018; Banino et al., 2018; Whittington et al., 2020), suggesting that grid cells emerge as a result of successful navigation. These findings were further reinforced by Sorscher et al. (2023), who analytically demonstrated that path integration with certain implementation choices, such as non-negativity, is a sufficient condition for the emergence of grid cells, clarifying earlier controversies (Schaeffer et al., 2022). However, none of these works have addressed how the MEC/hippocampal network learns the grid cells. The RNN models are trained by BPTT, a learning rule unlikely to be

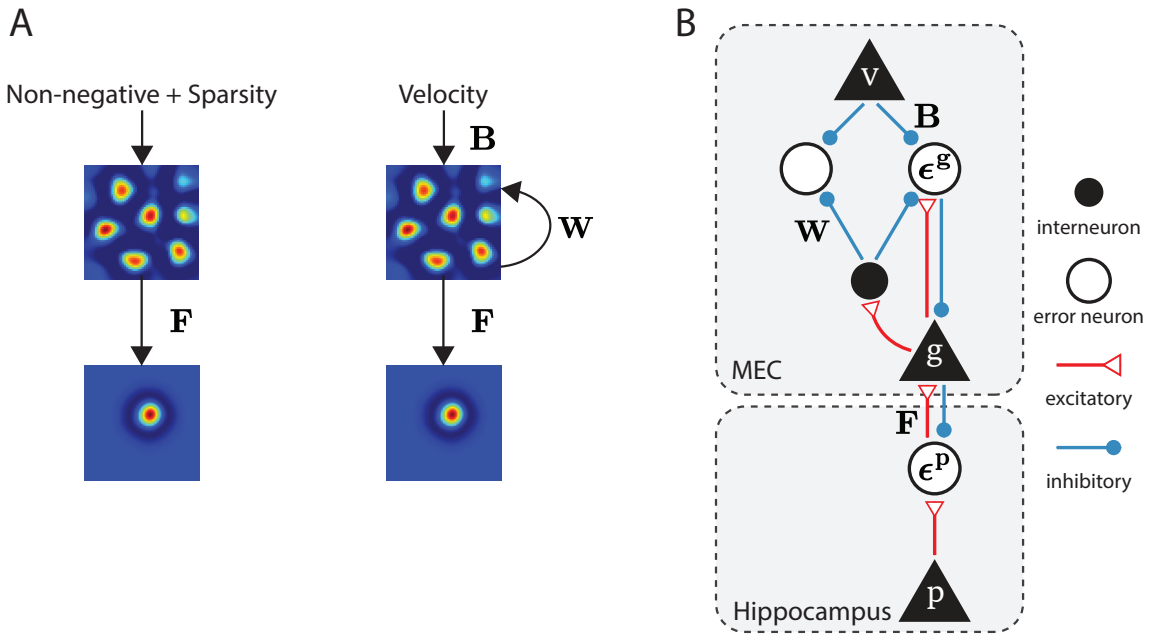

Figure 5.1: **Architecture and circuit implementation of PCNs.** A: Two ways of obtaining grid cells within PCNs. Left: Sparse, non-negative PCN as a generative model. Right: tPCN performing path integration. B: Circuit implementation of path-integrating tPCN with a mapping to MEC and hippocampus.

employed by the brain (Lillicrap and Santoro, 2019). Even though the principal component model by Dordek et al. (2016) can be learned with the plausible Sanger's rule (Sanger, 1989), it has been shown that principal component analysis (PCA) cannot be applied to other brain regions such as the visual areas (Olshausen and Field, 1996), and Sanger's rule cannot be generalized to dynamical tasks such as path integration. Earlier models of the learning process of grid cells have explored plausible learning rules such as spike time-dependent plasticity (Widloski and Fiete, 2014) and variants of Hebbian learning rules (Kropff and Treves, 2008) within networks of excitatory and inhibitory neurons (Weber and Sprekeler, 2018). However, these learning rules are highly specialized, and have not been shown to reproduce representations from other brain regions with non-spatial tasks. Recent works have also modeled the hippocampal formation using generative models with plausible learning rules similar to predictive coding (George et al., 2024; Bredenberg et al., 2021), though these studies did not address 2D spatial learning.

5.3 Models

5.3.1 Non-negative Sparse PCN

We first investigate the classical PCN (Rao and Ballard, 1999) for its ability to form grid representations. Assuming a place cell input $\mathbf{p} \in \mathbb{R}^{N_p}$ that represents a location in 2D space as an N_p-dimensional vector, a simple 2-layer PCN generates predictions of $\mathbf{p}$ using its latent activities $\mathbf{g} \in \mathbb{R}^{N_g}$ (which will develop grid-like representations) and a weight matrix $\mathbf{W}$ (Fig 5.1A). The generative model minimizes the following loss function subject to two constraints:

$$\mathcal{F}_{\text{PCN}} = \|\mathbf{p} - \mathbf{W}\mathbf{g}\|_2^2 + \|\mathbf{g}\|_2^2 + 2\lambda\|\mathbf{g}\|_1 \tag{5.1}$$

where $\|\mathbf{g}\|_2^2$ constrains the $l2$ norm of the latent $\mathbf{g}$ and $\lambda\|\mathbf{g}\|_1$ enforces sparsity, similar to the sparse coding model (Olshausen and Field, 1996). This loss function is minimized via an expectation-maximization (EM) algorithm, alternating between the optimization over $\mathbf{g}$ (inference) and $\mathbf{W}$ (learning) (see Appendix D.1 for the training algorithm):

$$\Delta\mathbf{g} \propto -\frac{\partial\mathcal{F}_{\text{PCN}}}{\partial\mathbf{g}} = -\mathbf{g} - \lambda\text{sgn}(\mathbf{g}) + \mathbf{W}^\top\boldsymbol{\epsilon}^{\mathbf{p}}; \quad \mathbf{g} \leftarrow \texttt{ReLU}(\mathbf{g} + \Delta\mathbf{g}) \tag{5.2}$$

$$\Delta\mathbf{W} \propto -\frac{\partial\mathcal{F}_{\text{PCN}}}{\partial\mathbf{W}} = \boldsymbol{\epsilon}^{\mathbf{p}}\mathbf{g}^\top \tag{5.3}$$

where $\boldsymbol{\epsilon}^{\mathbf{p}} := \mathbf{p} - \mathbf{W}\mathbf{g}$ and we apply a $\texttt{ReLU}$ to the inference dynamics to constrain the latent activities to be non-negative. The inference and learning dynamics can be implemented in a plausible circuit (Bogacz, 2017). After convergence, we examine the firing fields of the latent activities $\mathbf{g}$.

5.3.2 Path Integrating tPCN

To account for the learning of spatial representations in moving animals, we also investigate tPCN that extends the classical PCNs to the temporal domain (Millidge et al., 2024; Tang et al., 2024) in path integration tasks (Fig. 5.1B). The model receives a series of place cell activities $\mathbf{p}_1, ..., \mathbf{p}_T$ and velocity inputs $\mathbf{v}_1, ..., \mathbf{v}_T$ that represent the trajectory of an agent moving in a 2D space, and minimizes the following loss function at each time step t[1]:

$$\mathcal{F}_{\text{tPCN},t} = \|\mathbf{p}_t - f(\mathbf{F}\mathbf{g}_t)\|_2^2 + \|\mathbf{g}_t - h(\mathbf{W}\hat{\mathbf{g}}_{t-1} + \mathbf{B}\mathbf{v}_t)\|_2^2 \tag{5.4}$$

[1]Note that the position of the nonlinear functions h and f is different from previous chapters. In this chapter, they are applied after the activities have been linearly weighted. This is to ensure the non-negativity of the latent, grid activities via $h = \texttt{ReLU}$. The latent and observed variables are also denoted by $\mathbf{g}$ and $\mathbf{p}$ in this Chapter to reflect grid and place cells.

where f and h are both nonlinear activation functions, and $\mathbf{B}$, $\mathbf{W}$ and $\mathbf{F}$ are weight matrices projecting the predictions. We define $\boldsymbol{\epsilon}_t^{\mathbf{p}} := \mathbf{p}_t - f(\mathbf{F}\mathbf{g}_t)$, $\boldsymbol{\epsilon}_t^{\mathbf{g}} := \mathbf{g}_t - h(\mathbf{W}\hat{\mathbf{g}}_{t-1} + \mathbf{B}\mathbf{v}_t)$. The model learns by first optimizing the loss function with respect to $\mathbf{g}_t$ via gradient descent:

$$\Delta \mathbf{g}_t \propto -\frac{\partial \mathcal{F}_{\text{tPCN},t}}{\partial \mathbf{g}_t} = -\boldsymbol{\epsilon}_t^{\mathbf{g}} + \mathbf{F}^\top f'(\mathbf{F}\mathbf{g}_t)\boldsymbol{\epsilon}_t^{\mathbf{p}} \tag{5.5}$$

and then optimizing weights by:

$$\begin{aligned} \{\Delta \mathbf{F}, \Delta \mathbf{W}, \Delta \mathbf{B}\} &\propto \frac{\partial \mathcal{F}_{\text{tPCN},t}}{\partial \{\mathbf{F}, \mathbf{W}, \mathbf{B}\}} \\ &= \{f'(\mathbf{F}\mathbf{g}_t)\boldsymbol{\epsilon}_t^{\mathbf{p}}\mathbf{g}_t^\top, h'(\tilde{\mathbf{g}}_t)\boldsymbol{\epsilon}_t^{\mathbf{g}}\hat{\mathbf{g}}_{t-1}^\top, h'(\tilde{\mathbf{g}}_t)\boldsymbol{\epsilon}_t^{\mathbf{g}}\mathbf{v}_t^\top\} \end{aligned} \tag{5.6}$$

where f' and h' are Jacobians of the nonlinear functions f and h, and $\tilde{\mathbf{g}}_t := \mathbf{W}\hat{\mathbf{g}}_{t-1} + \mathbf{B}\mathbf{v}_t$. After the inference (Eq. 5.5) converges, we set $\hat{\mathbf{g}}_t$ to the converged value of $\mathbf{g}_t$, which will be used for optimizing the objective function at the next time step i.e., $\mathcal{F}_{\text{tPCN},t+1}$. The model is trained on a large number of trajectories $\{\mathbf{v}_t, \mathbf{p}_t\}$ and after training, a set of velocity inputs from unseen trajectories is presented to the model. The model then performs a forward pass through time and layers to predict the positions encoded by place cells (see Appendix D.1 for the training and testing algorithms of tPCN):

$$\mathbf{g}_t = h(\mathbf{W}\mathbf{g}_{t-1} + \mathbf{B}\mathbf{v}_t), \quad \hat{\mathbf{p}}_t = f(\mathbf{F}\mathbf{g}_t) \tag{5.7}$$

The model is evaluated on 1) the accuracy of path integration position prediction $\hat{\mathbf{p}}_t$ and 2) the firing fields of the latent $\mathbf{g}$. When both f and h are linear, these computations can be plausibly implemented in a neural circuit shown in Fig. 5.1C, with local inference computations (Eq. 5.5) and Hebbian learning rules (Eq. 5.6) (see Chapters 3 and 4 for detailed discussions of this implementation). When the activation functions involve only local nonlinearity, such as `tanh` or `ReLU`, the Jacobians are diagonal and the inference and learning rules remain local and Hebbian (Millidge et al., 2022a), and additional circuitry components can be included to plausibly implement the nonlinearities (Whittington and Bogacz, 2017). Within the context of spatial representation learning, this circuit implementation can be naturally mapped to the circuitry of the hippocampal formation, the the latent layer represents the MEC and the observation layer represents the hippocampus.

5.3.3 Input of the Model

In models discussed in this chapter, we assume that grid cells are inferred as latent representations of place cells. Although previous models have followed the opposite

direction of the relationship, several strands of experimental evidence have suggested the emergence of grid cells as a result of place cells, including the earlier development of place cells (Bush et al., 2014; Langston et al., 2010; Wills et al., 2010). In both PCN and tPCN models, the place cell inputs are constructed as 2D difference-of-softmaxed-Gaussian (DoS) curves flattened into 1D vectors, which have been shown to yield hexagonal grid representations in RNNs (Schaeffer et al., 2022; Sorscher et al., 2023). The firing centers of the place cells are uniformly distributed across a 2D environment. For PCN, the inputs are N_x evenly distributed locations in the environment (N_x large enough to cover the whole environment) represented by the N_p place cells. For tPCN, the trajectories for the path integration task are obtained by simulating an agent performing a smooth random walk in the square environment. At each point in time, the N_p place cells will be uniquely activated, representing the agent's current location. The velocity inputs $\mathbf{v}_t$ are 2D vectors representing the speed of the simulated agent on the x and y coordinates at time step t. The effect of boundaries is simulated by slowing down the agent and reverting its moving direction near the borders of the environment. We sample a large number of trajectories to cover the whole simulated environment for training.

5.4 Results

5.4.1 Sparse non-negative PCN develops latent grid cells

Here we examine whether the sparse non-negative PCN can develop hexagonal, grid-like latent representations of the space after training, by plotting each latent neuron's responses to the $N_x = 900$ locations in the 2D space. We use $N_p = 512$ and $N_g = 256$. The "gridness" of the 2D latent representations is evaluated using the *grid score* metric, commonly employed in both experimental and computational studies (Sargolini et al., 2006; Banino et al., 2018) (see Appendix D.4 for grid score calculation). We found that this simple, 2-layer PCN can develop hexagonal grid cells similar to those observed in the MEC (Fig. 5.2A). For comparison, we reproduce the results from Dordek et al. (2016) and Sorscher et al. (2023), which show theoretically that performing non-negative PCA on the place cell inputs is guaranteed to produce hexagonal grid representations as the principal components of the $N_x \times N_p$ place cell input matrix. The visual results of the reproduction are shown in Fig. 5.2B, and we compare the distribution of grid scores of the PCN's latent neuron firing fields with those of the non-negative principal components in Fig. 5.2E. The grid scores between our sparse non-negative PCN and non-negative PCA are similarly distributed.

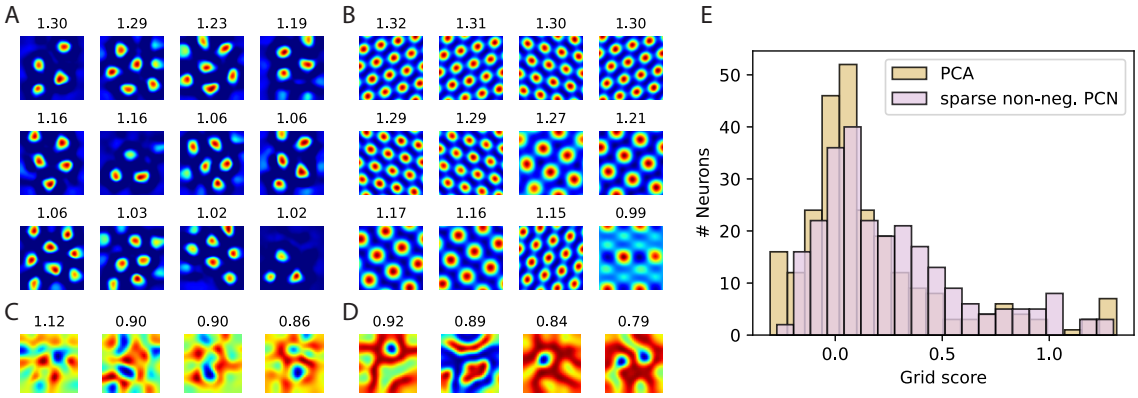

Figure 5.2: **Grid cells developed in PCN.** A: Latent representations of a sparse, non-negative PCN, resembling hexagonal grid cells in the MEC. Numbers in the title reflect the grid scores. B: Grid cells obtained via the pattern formation theory/non-negative PCA discussed in Sorscher et al. (2023); Dordek et al. (2016). C, D: Latent representations without sparsity or non-negativity, respectively. E: Distribution of grid scores of the representations in A and B.

Why does the sparse, non-negative PCN develop hexagonal grid cells? While a precise analytical explanation is left for future work, we offer an intuitive hypothesis here. When presented with a batch N_x of place cell inputs, the objective of PCN (Eq. 5.1) can be written compactly as:

$$\mathcal{F}_{\text{PCN}} = \|\mathbf{P} - \mathbf{G}\mathbf{W}^\top\|_F^2 + \sum_{i=1}^{N_x} \|\mathbf{g}_i\|_2^2 + 2\lambda\|\mathbf{g}_i\|_1 \tag{5.8}$$

where $\mathbf{P} \in \mathbb{R}^{N_x \times N_p}$ is the place cell activities across N_x locations, and $\mathbf{G} \in \mathbb{R}^{N_x \times N_g}$ represents grid cell responses. On the other hand, the objective function of PCA is (see Appendix D.2 for how this squared error minimization formulation of PCA is related to the more well-known maximum variance formulation):

$$\mathcal{F}_{\text{PCA}} = \|\mathbf{P} - \mathbf{G}\mathbf{M}\|_F^2 \quad \text{s.t.} \quad \mathbf{G}^\top\mathbf{G} = \mathbf{I}_{N_g} \tag{5.9}$$

where $\mathbf{M}$ is the $N_g \times N_p$ readout matrix[2]. The constraint $\mathbf{G}^\top\mathbf{G} = \mathbf{I}_{N_g}$ in Eq. 5.9 enforces orthonormality of the grid cell matrix $\mathbf{G}$ columns, meaning they are orthogonal and have unit norm. We hypothesize that the constraint $\|\mathbf{g}_i\|_2^2 + 2\lambda\|\mathbf{g}_i\|_1$ for our sparse PCN achieves this orthonormality implicitly: while the constraints are imposed on the rows of $\mathbf{G}$, the overall sparsity of entries in $\mathbf{G}$ could induce orthogonality among

[2]Note that although in PCA, $\mathbf{G}$ is usually framed as the matrix that projects $\mathbf{P}$ into its lower (N_g)-dimensional *column* representations $\mathbf{M}$, itself can also be understood as lower (N_g)-dimensional *row* representations of $\mathbf{P}$.

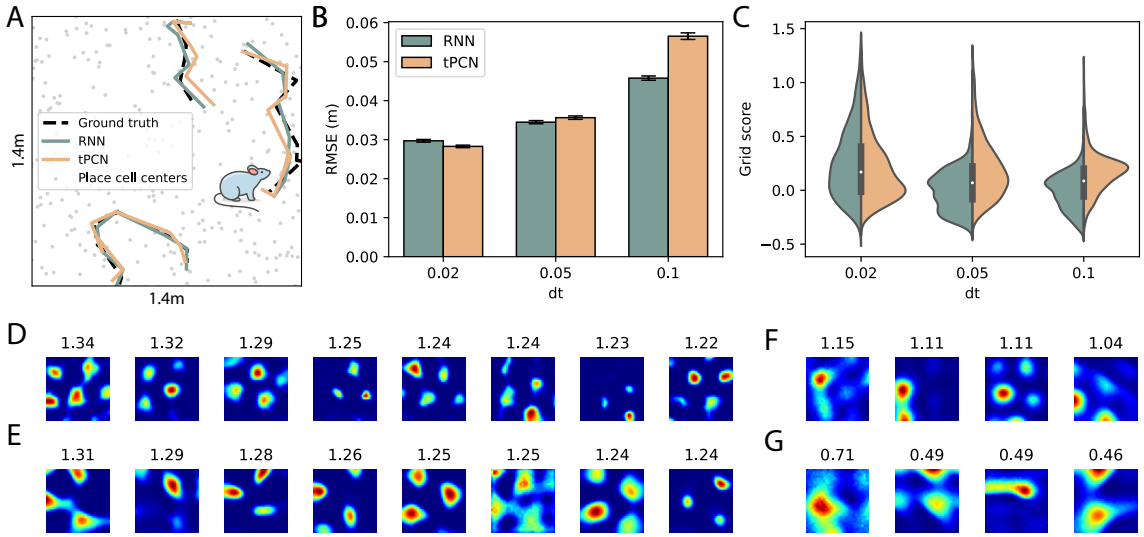

Figure 5.3: **tPCN in path integration**. A: Visual demonstration of the performance of tPCN and RNN in path integration. B: RMSEs between the decoded and ground-truth 2D positions by tPCN and RNN with different agent moving speed. C: Grid score distributions of tPCN and RNN with different agent moving speed. D, E: Firing fields of latent neurons in a tPCN and an RNN respectively, when $dt = 0.02$. F, G: Firing fields of latent neurons in a tPCN and an RNN respectively, when $dt = 0.1$.

its columns, with the l_2 term constraining the norm of the columns to achieve normality implicitly. Indeed, Fig. 5.2C shows that if we remove the sparsity constraint, the latent neurons' firing fields will no longer be hexagonal. Similarly, without `ReLU` i.e., non-negativity applied to the inference dynamics, we also could not obtain hexagonal grid cells (Fig. 5.2D). It is worth noting that although (non-negative) PCA can be learned with the biologically plausible Sanger's rule (Sanger, 1989), it lacks PCN's generalizability to different architectures (Salvatori et al., 2022a) and to other brain regions such as the visual cortex (Olshausen and Field, 1996; Rao and Ballard, 1999). However, it can be noticed that the grid cells by PCN lack the multi-modularity of the grid cells by non-negative PCA i.e., grid cells with different firing periods. We suspect that although sparse PCNs can approximate the orthonormality of latent variables, they lack PCA's ability to extract latent variables ordered by the amount of explained variance in data, with higher variance naturally corresponding to larger spatial scales and vice versa.

5.4.2 tPCN develops grid cells by path integration

Although training a static PCN with a large number of place cell activations can already give rise to brain-like hexagonal grid cells, the emergence of grid cells is known to rely on dynamic motion of animals (McNaughton et al., 2006; Winter et al., 2015). Therefore, we investigate tPCN in a path integration task, where the simulated agent uses dynamic velocity inputs to determine its current position. As a reference, we compare tPCN with RNNs trained in path integration, which have been shown to develop hexagonal grid cells (Cueva and Wei, 2018; Banino et al., 2018; Sorscher et al., 2023) and share the same graphical structure as tPCN (Fig. 5.1B). However, it is important to note that RNNs are trained with the biologically implausible backpropagation-though-time (BPTT) algorithm, which requires "unrolling" of the network through time, a process unlikely to occur in the brain (Lillicrap and Santoro, 2019).

We first evaluate whether tPCN can learn to perform the path integration task using local and Hebbian learning rules. We trained a tPCN model with $N_g = 2048$ latent neurons on trajectories within a $1.4\text{m} \times 1.4\text{m}$ environment represented by $N_p = 512$ place cells. After training, we tested the model on a set of unseen trajectories with velocity input $\mathbf{v}_t$, and assessed whether the tPCN and RNN models could predict the correct positions using Eq. 5.7. As the output of the networks is the N_p-dimensional population activity of the place cells, we calculate the predicted 2D positions by averaging the center positions of the 3 most active place cells in the output $\hat{\mathbf{p}}_t$, and calculate the root mean square error (RMSE) between the decoded and ground-truth 2D positions. The visual and numerical results are shown in Fig. 5.3A and B, where we also varied a scaling factor dt of the simulated agent's speed, sampled from a Rayleigh distribution with mean 1, to test the robustness of the results. Note that we do not intend to model physiologically realistic speed of animals with these values. The performance of tPCN is comparable to that of the RNN, though it slightly deteriorates when the agent moves at higher speeds.

Next, we examine whether the tPCN model develops grid-like representations in its latent layer during path integration. We plot the firing fields of the 2048 latent neurons given an unseen set of trajectories covering the entire space. The neurons with the highest grid scores are shown in Fig. 5.3C, which reveals a grid-like, hexagonal firing pattern with high grid scores. Visually, these grid cells are similar to those in a trained RNN with the same architecture shown in Fig. 5.3E, replicating the results from (Sorscher et al., 2023). To systematically compare the grid cells in tPCN and RNN, we plot the distribution of grid scores in both models

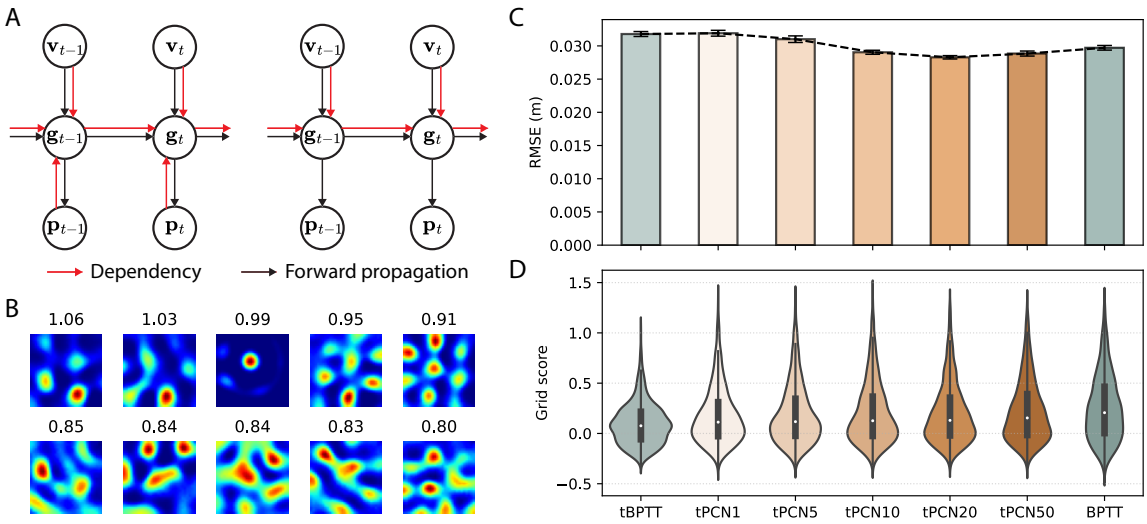

Figure 5.4: **Comparing tPCN and tBPTT.** A: Dependencies of latent grid cells in tPCN and RNN trained with 1-step tBPTT. Black arrows indicate the flow of computations during a forward pass and red arrows indicate the dependency of latent variables. B: Firing fields of the latent neurons of an RNN trained by 1-step tBPTT. C, D: Path integration RMSE and grid score distributions of 1-step tBPTT, BPTT and tPCNs with different inference iterations. "tPCNk" indicates tPCN trained with k inference iterations.

as a function of the movement speed of the agent in the environment in Fig. 5.3C. When the movement is slow, the grid score distributions are similar between tPCN and RNN. However, as the dt increases to 0.05 and 0.1, tPCN tends to have higher grid scores than RNN. This is visually reflected in Fig. 5.3F (tPCN) and G (RNN), which shows the latent representations developed by tPCN largely retain the grid-like pattern whereas firing centers of many of the RNN neurons no longer form a grid when $dt = 0.1$. Interestingly, the band-like representations present in both models in this case are observed in MEC (Krupic et al., 2012), although their existence is controversial (Navratilova et al., 2016).

5.4.3 tPCN approximates truncated BPTT

Next, we asked why hexagonal grid representations emerge both when training a tPCN using a BPTT-free Hebbian learning rule and when training an RNN using BPTT. We provide an analytical comparison between the learning rules of tPCN and RNN. Assuming a vanilla, sequence-to-sequence RNN with exactly the same graphical structure as in Fig. 5.1A, its dynamics can be recursively described as:

$$\mathbf{g}_t = h(\mathbf{W}\mathbf{g}_{t-1} + \mathbf{B}\mathbf{v}_t); \quad \hat{\mathbf{p}}_t = f(\mathbf{F}\mathbf{g}_t) \tag{5.10}$$

The loss that this RNN is trained to minimize is the cumulative prediction error:

$$\mathcal{F}_{\text{RNN}} = \sum_{t=1}^{T} \mathcal{F}_{\text{RNN},t} = \sum_{t=1}^{T} \|\mathbf{p}_t - \hat{\mathbf{p}}_t\|_2^2 \tag{5.11}$$

Suppose BPTT is performed at every step t to update weights in this RNN, the learning rule for $\mathbf{W}$ at step t can be expressed as (see Appendix D.3 for derivations):

$$\Delta \mathbf{W}^{\text{RNN}} = \sum_{k=1}^{t} \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_k} h'(\tilde{\mathbf{g}}_t) \mathbf{F}^\top f'(\mathbf{F}\mathbf{g}_t) \boldsymbol{\epsilon}_t^{\mathbf{p}} \underline{\mathbf{g}_{k-1}^\top} \tag{5.12}$$

where $\boldsymbol{\epsilon}_t^{\mathbf{p}}$ denotes the prediction error $\mathbf{p}_t - \hat{\mathbf{p}}_t$ and the $\frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_k}$ terms correspond to the unrolling in BPTT, which can be factorized into a chain of partial derivatives (Bellec et al., 2020). On the other hand, for tPCN, if we assume that the inference dynamics in Eq. 5.5 has fully converged ($\Delta \mathbf{g}_t = 0$) at the time of weight update, the learning rule of tPCN can be written as (see Appendix D.3 for derivations):

$$\Delta \mathbf{W}^{\text{tPCN}} = h'(\tilde{\mathbf{g}}_t) \mathbf{F}^\top f'(\mathbf{F}\mathbf{g}_t) \boldsymbol{\epsilon}_t^{\mathbf{p}} \underline{\hat{\mathbf{g}}_{t-1}^\top} \tag{5.13}$$

Two key differences between these learning rules stand out. First, tPCN does not involve the recursive unrolling term, thereby avoiding the need to maintain a perfect memory of all preceding hidden states. Second, instead of using the forward-propagated $\mathbf{g}_{t-1}$ as in Eq. 5.10, tPCN employs the inferred $\hat{\mathbf{g}}_{t-1}$ from Eq. 5.5 (underlined). The first difference suggests an equivalence between tPCN and RNN trained with truncated BPTT (tBPTT) with a truncation window of size 1 (1-step tBPTT) (Williams and Peng, 1990), where the RNN does not backpropagate *any* hidden states through time when updating the weights. This characteristic could potentially harm the RNN's performance as it cannot effectively perform temporal credit assignment. However, the second difference partially solves this problem, as $\hat{\mathbf{g}}_{t-1}$ is inferred following Eq. 5.5, which includes the term $\boldsymbol{\epsilon}_{t-1}^{\mathbf{p}}$ that communicates the place cell prediction error at step $t-1$. Therefore, when $\mathbf{W}$ is updated at step t, the $\hat{\mathbf{g}}_{t-1}^\top$ term in $\Delta \mathbf{W}^{\text{tPCN}}$ will effectively form an eligibility trace (Bellec et al., 2020) that allows the model to access historical prediction errors on the place cell level. Fig. 5.4A illustrates this difference between tPCN and RNN trained by 1-step tBPTT, highlighting the dependency of tPCN hidden states on past place cell activations. In Appendix D.3 we also discuss the relationship between the update rules for $\mathbf{B}$ and $\mathbf{F}$ in these two models.

To verify this theoretical difference, we compare tPCN with RNNs trained by tBPTT in the path integration task. Since $\hat{\mathbf{g}}_t$ in tPCN is initialized by a forward pass $f(\mathbf{W}\hat{\mathbf{g}}_{t-1})$ and then updated by the iterative inference (Appendix D.1), the behavior of 1-step tBPTT, which computes its latent states via a forward pass at each time step,

should be closer to tPCN with fewer inference iterations. Therefore, we evaluate tPCN with various inference iterations. Fig. 5.4B shows the grid cells learned by an RNN trained with 1-step tBPTT, which still exhibit hexagonal grid firing fields, though with lower grid scores than those from full BPTT. This suggests that backpropagating the error through all time steps is not entirely necessary for RNNs to generate grid cell-like representations. In Fig. 5.4C we show the path integration performance of RNN by 1-step tBPTT and BPTT, as well as tPCNs with different inference iterations from 1 to 50. As can be seen, tPCN with a single inference iteration has identical performance to RNN trained by tBPTT, and its performance will improve as we increase the number of inference iterations but will saturate around 20 iterations. Overall, this graph suggests that tPCN with 5 or more inference iterations can effectively perform temporal credit assignment that improves upon tPCN1 or 1-step tBPTT, potentially due to the eligibility trace. However, this eligibility trace arises from local inference dynamics (Eq. 5.5) rather than from unrolling the RNN graph as in Bellec et al. (2020). This improvement is also reflected in the grid scores (Fig. 5.4D), although increasing the inference iterations does not necessarily result in better grid score representations. We suspect that although the gridness of latent representations is somewhat related to path integration performance, their relationship is not linear. It is also worth noting that to fully evaluate the similarities and differences between BPTT and tPCN, an in-depth comparison is needed across different tasks and versions of tBPTT. We aim to investigate this question in future works as it is beyond the scope of this chapter.

5.4.4 Robustness of grid cell representations in tPCN

Inspired by Schaeffer et al. (2022), we examine the robustness of our results against different architectural choices of the tPCN model, to understand what contributes to the emergence of grid cells within tPCN. Specifically, we vary the following components of the model: 1) Encoding of the place cell activities; 2) Output nonlinearity f; 3) Recurrent nonlinearity h; 4) Environment sizes; 5) Latent sizes and 6) Velocity input to the model. The baseline model has DoS place cell encodings, $h = \texttt{ReLU}$, $f = \texttt{softmax}$, 1.4m $\times$ 1.4m environment and latent size 2048 with velocity inputs.

We first examine whether replacing the place encoding with Gaussian curves affects the model's performance. As shown in Fig. 5.5A, B and C, the Gaussian place cells do not affect the path integration performance, but the latent representations are no longer hexagonal. This is consistent with earlier findings that the DoS place

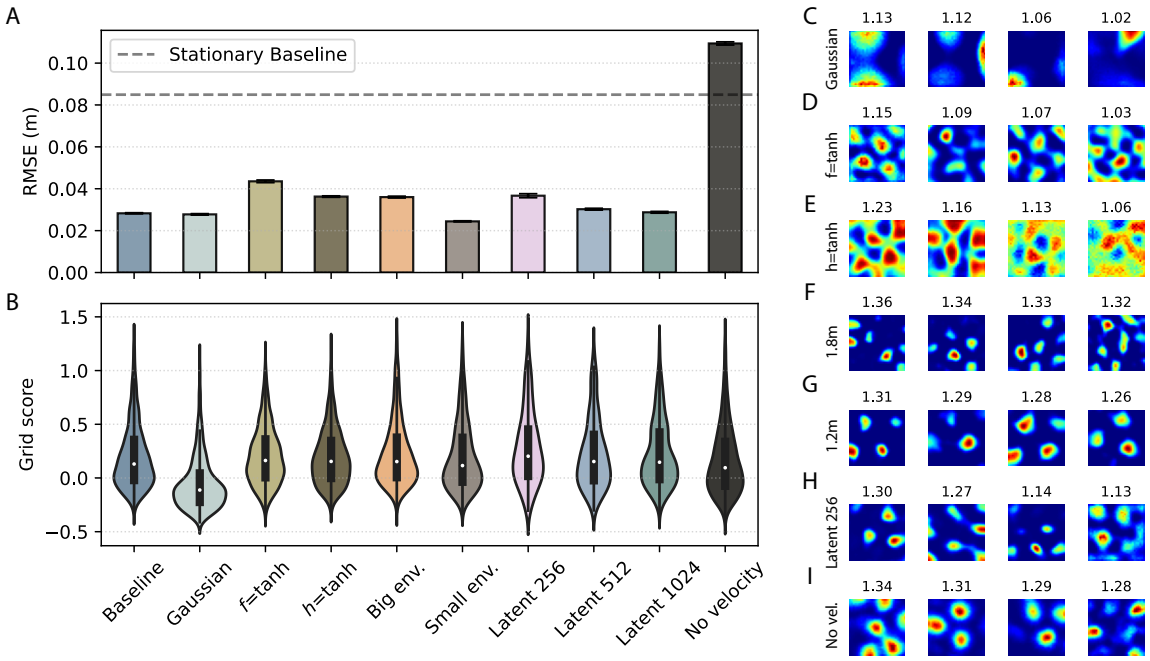

Figure 5.5: **Robust emergence of grid cells in tPCN.** A, B: Path integration RMSE and grid scores of tPCN in different setups. "Stationary baseline" refers to a model that always predicts the initial position regardless of movement. C-I: Firing fields of latent neurons in tPCNs with C: Gaussian place cells; D: f =`tanh`; E: h =`tanh`; F: 1.8m $\times$ 1.8m environment; G: 1.2m $\times$ 1.2m environment; H: 256 latent neurons; I: tPCN without velocity input.

cell encoding is necessary for hexagonal grid cells (Dordek et al., 2016; Sorscher et al., 2023; Schaeffer et al., 2022).

The choices of f and h are particularly interesting: as discovered by earlier works (Dordek et al., 2016; Sorscher et al., 2023), a choice of h that imposes non-negativity constraint on the latent activities, such as `ReLU`, is necessary for the emergence of hexagonal grid cells. In our tPCN model, the activation functions are also important for biological plausibility: in both Eq. 5.5 and Eq. 5.6, the multiplication with the Jacobians h' and f' can be reduced to local, element-wise multiplications if h and f are element-wise nonlinearities such as `ReLU` and `tanh`. Although it is possible to design a circuit to perform the computations in `softmax` (Snow and Orchard, 2022), it is unclear how the Jacobian matrix of `softmax` can be computed in a biological circuit. Therefore, we first replace f with a `tanh` function in our tPCN model and evaluate the model's performance in both path integration and its latent representations. As shown in Fig. 5.5A, replacing f with `tanh` results in slightly worse path integration performance and lower grid scores than the `softmax` baseline. However, visually, the latent representations are hexagonal and grid-like (Fig. 5.5D), suggesting that using

a biologically more plausible f would not significantly affect the emergence of grid cells within tPCN. On the other hand, replacing the non-negative constraint (`ReLU`) on the latent activities with $h = \texttt{tanh}$ results in the amorphous latent representations (Fig. 5.5E), which is consistent with Sorscher et al. (2023).

We next investigate the impact of the size of the environment, by training tPCN within a square environment of size 1.8m $\times$ 1.8m (big) and an environment of size 1.2m $\times$ 1.2m (small). Changing environment sizes does not affect the path integration performance, and does not affect tPCN's capability of developing grid cells either (Fig. 5.5F for big environment and G for small environment). We also vary the number of latent neurons in the model from 256 to 512 and 1024, which does not affect the grid cell representations (Fig. 5.5H shows the latent representations learned by a tPCN with 256 latent neurons). However, with fewer latent neurons, the performance in path integration becomes worse as the model has fewer number of parameters to perform the task (Fig. 5.5A).

Earlier studies using PCNs to model visual representations have mostly used unsupervised PCNs (Rao and Ballard, 1999; Olshausen and Field, 1996; Millidge et al., 2024), which corresponds to blocking the velocity input $\mathbf{v}_t$ into tPCN in Fig. 5.1B. Here we asked how removing velocity input would affect the path integration performance and grid cell emergence of tPCN. Mathematically, this is achieved simply by re-defining $\tilde{\mathbf{g}}_t := \mathbf{W}_{\text{r}}\hat{\mathbf{g}}_{t-1}$ without changing any inference or learning dynamics. It can be seen from Fig. 5.5A that the path integration performance is significantly affected by the absence of velocity input, with an RMSE even higher than the stationary baseline, where the model does not predict any movement at all. Intriguingly, the latent representations developed by this unsupervised tPCN are still grid cell-like (Fig. 5.5I) with a similar grid score distribution to the baseline model. This result demonstrates that grid cells can still emerge even in a model unable to perform path integration at all. Therefore, our model predicts that path integration is not a sufficient condition for the emergence of grid cells, which resonates with Schaeffer et al. (2022). In other words, it predicts that animals unable to navigate due to impaired velocity encoding may still develop grid cells as a result of self-motion.

5.5 Discussion

5.5.1 Summary

In this chapter, we have demonstrated a biologically plausible learning rule for grid cells based on predictive coding. We have shown that with sparsity and non-negative

constraints, classical PCNs can develop grid cell-like representations of batched place cell inputs. With inputs representing trajectories of moving agents, tPCN can also develop grid cell activations while performing path integration. We have developed a theoretical understanding of this property of tPCN by deriving and comparing its learning dynamics with that of BPTT, showing that unrolling a recurrent network is unnecessary for it to learn grid cells, and a more plausible approach with recursive inference dynamics should suffice. Furthermore, we have examined the robustness of our results by varying hyper-parameters of the model, and found that grid cells can be learned even without velocity inputs. Overall, our work demonstrates that predictive coding can serve as an effective and biologically plausible plasticity rule for neural networks to learn grid cells observed in the MEC. Importantly, compared with earlier learning rules specialized for grid cells, predictive coding is a general learning rule able to reproduce many other cortical functions and representations. Thus, our findings suggest that a single, unified plausible learning rule can be employed by the brain to find the most appropriate representation of cortical inputs in different regions.

5.5.2 Relationship to RNNs

In this chapter we have demonstrated the relationship between tPCN and single-step truncated BPTT, which partly resolves the question left in Chapter 3 on whether tPCN can effectively learn long-term dependencies. Our results here suggest that in highly Markovian tasks, such as path integration, where long-range dependency does not play an important role, tPCN can match BPTT. However, it may under-perform in tasks that requires long-term credit assignment, such as those involving language (Graves et al., 2013). An interesting future line of research can thus explore tPCN's performance in these more challenging machine learning tasks. A possible solution to the long-term learning issue with tPCN can be multiple layers of recurrently connected neurons, where higher layers may account for long-term dependencies, similar to what was discussed in another temporal version of predictive coding (Jiang and Rao, 2022). Another route is to apply tPCN to neuroscience-related tasks that have been studied using BPTT-trained RNNs, such as those studied in Yang et al. (2019), which could provide a more plausible account of how the brain performs these tasks.

5.5.3 Relationship to biological data

This chapter is mainly focused on demonstrating that grid cells can emerge in predictive coding models. The abundant experimental findings, on the other hand, provide

many future directions for this research to follow. For example, it is known that the development of grid cells is abrupt (Wills et al., 2010). It would be interesting to investigate if predictive coding can train networks to abruptly learn grid cells, and if the model could provide a mechanistic account for the abrupt learning. Additionally, grid cells emerged in predictive coding models does not present the multi-scale property observed in rats (Hafting et al., 2005). In particular, it is known that grid cell scales have a multi-peak, discretized distribution (Stensola et al., 2012). One possible approach to reproduce this observation is to use place cell inputs with distributed place field sizes, which has a known continuous distribution (Dabaghian et al., 2012), instead of the fixed field size used in experiments in this chapter. The hexagonal firing pattern of grid cells is also known to distort in polarized environments such as trapezoidal ones (Krupic et al., 2015). This property has been replicated in reinforcement learning models also employing the concept of predictive coding (Stachenfeld et al., 2017), and it would be intriguing to explore whether predictive coding networks can also reproduce this phenomenon.

Chapter 6

Discussion

6.1 Summary of results

This thesis has extended the theory of predictive coding to three crucial aspects of computations in the brain: space, time and memory. It has been shown that the classical, hierarchically organized predictive coding models can be extended to account for associative memory through a recurrent formulation of the free energy (Chapter 2), and can also be extended to describe temporally varying stimuli by incorporating temporal predictions (Chapter 3). This temporal version of predictive coding can then be applied to model sequential memory (Chapter 4) and to learn grid cells (Chapter 5), two important properties of the hippocampal formation. This chapter discusses topics relevant to multiple chapters of this thesis.

6.2 Biological plausibility

6.2.1 Error neurons

The biological plausibility of predictive coding, which mainly refers to its local inferential computations and Hebbian plasticity, is based on the inclusion of error neurons in its network implementation. Despite the convenience, the general existence of error neurons per se in the brain is controversial. Unlike the dopaminergic reward prediction error neurons that are well-documented (Schultz et al., 1997; Schultz, 1998), there is limited evidence for the general presence of error neurons in the cortex (Walsh et al., 2020). In the neocortex, recent theories and computational models have suggested that layer L5 could serve as a temporal mismatch detector as it receives direct inputs from thalamus while indirect and delayed inputs from L2/3 (Kermani Nejad et al., 2024). This could provide a possible mechanism for the temporal error in the

modeling of the visual cortex using tPC (Chapter 3). Similarly, for the hippocampal formation, the CA1 region receives inputs directly from the entorhinal cortex and also indirect inputs from the dentate-gyrus and CA3. This anatomical property has led to theories of CA1 being a (temporal) error detector (Lisman, 1999) and recent models for hippocampal place cells leveraging this theory (Chen et al., 2022). This property could provide a possible mechanism of error computation in the hippocampal formation modelled in Chapters 2, 4 and 5. However, error neurons in all models discussed in this thesis have the special one-to-one connections to their corresponding value neurons, which is unlikely to exist in the brain. Future works could focus on how this constraint can be relaxed to better match the connectivities in the neocortex and the hippocampal formation.

In the dendritic rPCN proposed in Chapter 2 and tPCN in Chapter 3, we circumvented this problem by computing error signals from recurrent connections within apical dendrites, rather than within explicit error nodes in the original implementation. Such an architecture can reconcile predictive coding networks with the lack of strong evidence for explicit prediction error neurons in the cortex, and aligns well with experimental findings. For example, a recent study in rat cortex found little evidence for prediction-related signals in spikes, but found strong evidence for it in local field potentials, which are thought to be driven by somatic and dendritic potentials (Auksztulewicz et al., 2023). Although in Fig 3.2B and 3.2C we still use error neurons in the hierarchical part of the network, they can also be circumvented following the dendritic implementations in Mikulasch et al. (2023) or Sacramento et al. (2018). In both studies, the dendritic computations naturally arise by rearranging the dynamical equations of predictive coding (Whittington and Bogacz, 2019), which are then used to extend predictive coding in spiking networks (Mikulasch et al., 2022) and to approximate backpropagation (Sacramento et al., 2018). Broadly speaking, recent studies have shown that incorporating dendritic architectures into artificial neural networks may benefit transfer and continual learning, and help the design of neuromorphic hardware with lower energy consumption (Chavlis and Poirazi, 2021; Richards and Lillicrap, 2019; Guerguiev et al., 2017). It is thus an interesting future avenue to explore fully dendritic implementations of models studied in this thesis, and whether they can match neurophysiological data.

6.2.2 Weight symmetry

All predictive coding models presented in this thesis still inherit the weight symmetry problem i.e., the same weight that was used to propagate top-down predictions

is also used to propagate bottom-up error signals (see for example $\mathbf{F}$ in Eq. 3.10 and Eq. 3.8). Such symmetric synaptic connections does not exist in the brain. Multiple studies have investigated this issue with backpropagation (Lillicrap et al., 2016; Akrout et al., 2019; Amit, 2019), and preliminary results with hierarchical predictive coding have demonstrated that this symmetry is not necessary for successful performance in various tasks (Millidge et al., 2020b). It is therefore important to investigate whether relaxing the weight symmetry on rPCN and tPCN can equally retain their performance, while achieving a higher level of biological plausibility.

6.3 Scaling up predictive coding

In this thesis, the predictive coding models have mostly been trained to perform biologically relevant tasks, such as memory and navigation. It is known that hierarchical predictive coding can perform machine learning tasks as well as, or even better than backpropagation-trained neural networks (Whittington and Bogacz, 2017; Song et al., 2024), and can also be scaled up to have complex architectures (e.g., convolutional layers) and to process large-scale datasets that are usually encountered in machine learning scenarios (Pinchetti et al., 2025). However, all these studies focused on the classical, hierarchical predictive coding models. It is thus an important future avenue to investigate whether models discussed in this thesis, especially tPCN, can also be scaled up to tasks for RNNs, such as language (Graves et al., 2013) or sequence generation (Graves, 2013).

6.4 Modeling the hippocampal formation

A main theme of this thesis is to model the hippocampal formation with predictive coding. Experimental evidences have suggested mechanisms of general predictive processing within the hippocampus and related areas. It has long been postulated that the hippocampus predicts upcoming sensory inputs, in both spatial and non-spatial experimental setups (Lisman and Redish, 2009; Mehta et al., 2000; Stachenfeld et al., 2017; Gray and McNaughton, 2003). The special feedforward circuitry of the hippocampus may also be an evidence for predictive processing: Lisman (1999) suggested that the CA1 region of the hippocampus serves as a "mismatch" detector, encoding the error between sensory reality and the internally generated prediction based on past events communicated through the feedforward pathway from CA3 to CA1. Our

findings in Chapters 4 and 5 suggest that this mismatch signal can be used to memorize sequences or learn grid cells. However, although hippocampal cell populations that fire exclusively following erroneous trials in associative learning tasks have been found in animals (Wirth et al., 2009; Ku et al., 2021), it is unclear, at the single-cell level, whether prediction error representations exist in the hippocampus, and in what form they exist: are there neurons encoding them, or are they encoded in specific neuronal structures such as apical dendrites? Our models inform future experimental works to answer these questions by making the prediction that hippocampal neurons can representation prediction errors, via somatic or dendritic activities. In Chapter 5, grid cells in the MEC layer are recurrently connected through a specialized circuit involving interneurons $\hat{\mathbf{g}}_{t-1}$ that inhibit the output signal from the grid cells, allowing the error neurons $\boldsymbol{\epsilon}_t^{\mathbf{g}}$ to compute the temporal prediction errors. Experimental evidence suggests that lateral interactions in layer II of the MEC are predominantly inhibitory (Witter and Moser, 2006) and are mediated by interneurons such as basket cells (Jones and Bühl, 1993). Our model also predicts that these interneurons encode an eligibility trace $\hat{\mathbf{g}}_{t-1}$ from the immediate past. While recent studies have reported grid cells representing prospective locations (Ouchi and Fujisawa, 2024), it remains to be verified whether these cells are mechanistically supported by such "past" cells. Additionally, neurons in the entorhinal cortex are known to encode errors (Ku et al., 2021), suggesting a possible error-driven learning mechanism similar to that in tPCN.

Two important aspects of hippocampal formation have not been extensively addressed within this thesis. Firstly, although predictive coding frameworks have been employed to model hippocampal networks, the predictive nature of hippocampal neuronal activity itself affords deeper exploration. For example, hippocampal place cells are known to encode an animal's anticipated future positions during movement along linear tracks or in two-dimensional environments (Lisman and Redish, 2009), through phenomena like phase precession (O'Keefe and Recce, 1993) and temporally skewed place fields (Huxter et al., 2008). Investigating this predictive encoding of future events, potentially by training a tPCN model with visual movement cues (akin to Banino et al. (2018)), or integrating these insights into the path integration tasks discussed in Chapter 5, provides a valuable research opportunity.

Secondly, the concept of cognitive maps within the hippocampal formation (Tolman, 1948; Behrens et al., 2018) remains largely unexplored in predictive coding frameworks. The cognitive map hypothesis posits that spatial representations, such as place cells, grid cells, and border cells, can generalize beyond physical space to encode relationships among various objects and events (e.g., social relationships). Such

a generalized cognitive map can facilitate adaptive behaviors, including generalization. Findings presented in this thesis suggest several ways predictive coding models could extend to represent cognitive maps. Firstly, tPCN naturally develops grid-cell representations derived from place-cell activity, which can be extended to encode generalized events. Secondly, tPCN can disambiguate aliased observations via latent representations of different time steps, effectively encoding temporal relationships between events. Thirdly, tPCN generalizes dynamics from linear 1D sequences, suggesting the potential to generalize similarly across two-dimensional graphs of events. While Whittington et al. (2020) have proposed a computational model addressing cognitive maps, their framework depends heavily on ad-hoc components and is trained through backpropagation. Thus, applying predictive coding, rooted in Bayesian inference principles and implemented via local plasticity mechanisms, to cognitive maps presents an intriguing and promising research trajectory.

6.5 Relationship to self-supervised models of the brain

Predictive coding-based models discussed in this thesis are closely to self-supervised learning models i.e., the learning scheme where a model updates its synaptic weights according to self-generated teaching signals rather than external ones, to learn hidden representations of sensory inputs (Becker and Hinton, 1992; Chen et al., 2020; Zbontar et al., 2021). Predictive coding can be considered as a type of self-supervised learning as it also relies on self-generated predictions to perform synaptic updates, which relates it to studies investigating the biological plausibility of self-supervised learning (Tang et al., 2022; Siddiqui et al., 2023). An interesting future avenue for predictive coding is to compare its hidden representation to those learned by self-supervised learning methods.

Self-supervised learning has also been used to model the brain. For example, multiple lines of research have used self-supervised objective functions to train neural networks that develop grid cells (Xu et al., 2022; Schaeffer et al., 2024; Pettersen et al., 2024), relating it to Chapter 5 of this thesis. In Chapters 3 and 4, the tPC model learns from temporal prediction errors that arise due to two distinct pathways entering the error neuron at different time points. Self-supervised learning through mismatch or error from temporal delays is also a widely explored scheme in computational neuroscience. For example, Kermani Nejad et al. (2024) proposed a temporally

self-supervised model where neocortex L5 receives delayed inputs from L2/3 and direct inputs from the thalamus, and then computes the temporal difference between them to update synaptic weights. This view has also been adopted in models of the hippocampal formation, where the CA1 subregion is postulated to be a mismatch detector (Lisman, 1999) and can thus guide error-driven learning. Chen et al. (2022) used this principle to train a temporal self-supervised model that explains the place cell dynamics in CA1 and CA3 subregions. Levenstein et al. (2024) also employed temporal self-supervision to account for both spatial representations and replay in the hippocampus, providing a unifying view. This (temporally) predictive view of the hippocampal formation has also been discussed in a reinforcement learning framework (Stachenfeld et al., 2017), which explained hippocampal phenomena such as place cell prospective encoding and grid cell deformation. Despite these extensive studies adopting the temporal self-supervised view of the brain, a common missing theme in them is a mechanistic view of how a circuit of neurons can support the self-supervised learning of brain functions and representations. On the other hand, predictive coding models discussed in this thesis provide both a normative (Bayesian inference) and a mechanistic (local computations) view of the cortex. It would thus be interesting to investigate how predictive coding models, for example tPC, can be adapted to similar experimental set-ups of the studies above.

Appendix A

Supplementary: Recurrent Predictive Coding for Associative Memory

A.1 Expression for retrieved patterns in the rPCNs (Eq 2.27)

Here we provide the details of derivation of Eq. 2.27, showing that all three models perform the same retrieval at convergence. At convergence, the retrieval dynamics of the explicit PCN satisfy:

$$\mathbf{S}^{-1}(\mathbf{x} - \bar{\mathbf{x}}) = 0 \tag{A.1}$$

This equation can be written into its block matrix form:

$$\begin{bmatrix} \mathbf{S}_{kk} & \mathbf{S}_{km} \\ \mathbf{S}_{mk} & \mathbf{S}_{mm} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}_k - \bar{\mathbf{x}}_k \\ \mathbf{x}_m - \bar{\mathbf{x}}_m \end{bmatrix} = 0 \tag{A.2}$$

The inverse of $\mathbf{S}$ can thus be written as:

$$\begin{bmatrix} \mathbf{S}_{kk} & \mathbf{S}_{km} \\ \mathbf{S}_{mk} & \mathbf{S}_{mm} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{Q}^{-1} & -\mathbf{Q}^{-1}\mathbf{S}_{km}\mathbf{S}_{mm}^{-1} \\ -\mathbf{S}_{mm}^{-1}\mathbf{S}_{km}^{\top}\mathbf{Q}^{-1} & \mathbf{S}_{mm}^{-1} + \mathbf{S}_{mm}^{-1}\mathbf{S}_{km}^{\top}\mathbf{Q}^{-1}\mathbf{S}_{km}\mathbf{S}_{mm}^{-1} \end{bmatrix} \tag{A.3}$$

where $\mathbf{Q}$ is called the Schur complement of $\mathbf{S}_{mm}$ in $\mathbf{S}$ (Haynsworth, 1968; Boyd et al., 2004) and $\mathbf{Q} = \mathbf{S}_{kk} - \mathbf{S}_{km}\mathbf{S}_{mm}^{-1}\mathbf{S}_{km}^{\top}$. Since only $\mathbf{x}_k$ is relaxed during retrieval, we get:

$$\mathbf{Q}^{-1}(\mathbf{x}_k - \bar{\mathbf{x}}_k) - \mathbf{Q}^{-1}\mathbf{S}_{km}\mathbf{S}_{mm}^{-1}(\mathbf{x}_m - \bar{\mathbf{x}}_m) = 0 \tag{A.4}$$

which immediately gives us the retrieval dynamics in Eq. 2.27.

Now we show that the retrieval of the implicit/dendritic models also follows Eq. 2.27 at the convergence of inference. Notice that the sufficient and necessary

condition for the convergence is $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{W}\mathbf{x} - \boldsymbol{\nu} = 0$. Splitting it into blocks corresponding to $\mathbf{x}_k$ and $\mathbf{x}_m$ this becomes:

$$\begin{bmatrix} \mathbf{W}_{kk} & \mathbf{W}_{km} \\ \mathbf{W}_{mk} & \mathbf{W}_{mm} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_m \end{bmatrix} + \begin{bmatrix} \boldsymbol{\nu}_k \\ \boldsymbol{\nu}_m \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_m \end{bmatrix} \tag{A.5}$$

Since we only relax the top k corrupted entries in $\mathbf{x}$, we have:

$$\mathbf{W}_{kk}\mathbf{x}_k + \mathbf{W}_{km}\mathbf{x}_m + \boldsymbol{\nu}_k = \mathbf{x}_k \Rightarrow (\mathbf{I}_{kk} - \mathbf{W}_{kk})\mathbf{x}_k = \mathbf{W}_{km}\mathbf{x}_m + \boldsymbol{\nu}_k \tag{A.6}$$

We now investigate the parameter values in the above equation. Notice that we assumed the convergence of learning at the time of retrieval, which gives us Eqs. 2.13 and 2.14. By splitting Eq. 2.13 into blocks we have:

$$\boldsymbol{\nu}_k = (\mathbf{I}_{kk} - \mathbf{W}_{kk})\bar{\mathbf{x}}_k - \mathbf{W}_{km}\bar{\mathbf{x}}_m \tag{A.7}$$

Substituting the $\boldsymbol{\nu}_k$ in Eq. A.6 with this relationship we get:

$$\mathbf{W}_{km}(\mathbf{x}_m - \bar{\mathbf{x}}_m) = (\mathbf{I}_{kk} - \mathbf{W}_{kk})(\mathbf{x}_k - \bar{\mathbf{x}}_k) \tag{A.8}$$

Notice that Eq. 2.14 connects the value of W and S, which can also be written into the block-matrix form:

$$\left(\begin{bmatrix} \mathbf{W}_{kk} & \mathbf{W}_{km} \\ \mathbf{W}_{mk} & \mathbf{W}_{mm} \end{bmatrix} \begin{bmatrix} \mathbf{S}_{kk} & \mathbf{S}_{km} \\ \mathbf{S}_{mk} & \mathbf{S}_{mm} \end{bmatrix} - \begin{bmatrix} \mathbf{S}_{kk} & \mathbf{S}_{km} \\ \mathbf{S}_{mk} & \mathbf{S}_{mm} \end{bmatrix} \right) \Big|_{diag=0} = 0 \tag{A.9}$$

which gives us two useful relationships:

$$\mathbf{W}_{kk}\mathbf{S}_{km} + \mathbf{W}_{km}\mathbf{S}_{mm} = \mathbf{S}_{km} \Rightarrow \mathbf{W}_{km} = (\mathbf{I}_{kk} - \mathbf{W}_{kk})\mathbf{S}_{km}\mathbf{S}_{mm}^{-1} \tag{A.10}$$

$$(\mathbf{W}_{kk}\mathbf{S}_{kk} + \mathbf{W}_{km}\mathbf{S}_{mk} - \mathbf{S}_{kk})|_{diag=0} = 0 \tag{A.11}$$

Substituting the expression of $\mathbf{W}_{km}$ in terms of $\mathbf{W}_{kk}$ into Eq. A.8 we have:

$$(\mathbf{I}_{kk} - \mathbf{W}_{kk})\mathbf{S}_{km}\mathbf{S}_{mm}^{-1}(\mathbf{x}_m - \bar{\mathbf{x}}_m) = (\mathbf{I}_{kk} - \mathbf{W}_{kk})(\mathbf{x}_k - \bar{\mathbf{x}}_k) \tag{A.12}$$

The above expression is already close to Eq. 2.27 which we seek to prove, i.e., we could obtain Eq. 2.27 by cancelling $\mathbf{I}_{kk} - \mathbf{W}_{kk}$ on both sides of Eq. A.12, hence we will now show that $\mathbf{I}_{kk} - \mathbf{W}_{kk}$ is invertible. To do it we first observe that according to Eq. A.11, $\mathbf{W}_{kk}\mathbf{S}_{kk} + \mathbf{W}_{km}\mathbf{S}_{mk} - \mathbf{S}_{kk}$ is a diagonal matrix, which we call D. Therefore we have:

$$\begin{aligned} \mathbf{D} &= (\mathbf{I}_{kk} - \mathbf{W}_{kk})\mathbf{S}_{kk} - \mathbf{W}_{km}\mathbf{S}_{mk} \\ &= (\mathbf{I}_{kk} - \mathbf{W}_{kk})\mathbf{S}_{kk} - (\mathbf{I}_{kk} - \mathbf{W}_{kk})\mathbf{S}_{km}\mathbf{S}_{mm}^{-1}\mathbf{S}_{mk} \\ &= (\mathbf{I}_{kk} - \mathbf{W}_{kk})(\mathbf{S}_{kk} - \mathbf{S}_{km}\mathbf{S}_{mm}^{-1}\mathbf{S}_{mk}) \\ &= (\mathbf{I}_{kk} - \mathbf{W}_{kk})\mathbf{Q} \end{aligned} \tag{A.13}$$

Since $\mathbf{Q}$ is the Schur complement of $\mathbf{S}_{mm}$ in $\mathbf{S}$, a sample covariance matrix that we assume to be positive definite, it is also positive definite and thus invertible Boyd et al. (2004). Therefore $\mathbf{I}_{kk} - \mathbf{W}_{kk} = \mathbf{D}\mathbf{Q}^{-1}$. Notice that since $\mathbf{W}_{kk}$ has all its diagonal elements equal to 0, the matrix $\mathbf{D}\mathbf{Q}^{-1}$ must have 1's on its diagonal, which prevents the diagonal matrix $\mathbf{D}$ from having 0's on its diagonal (for $(\mathbf{D}\mathbf{Q}^{-1})_{ii} = \mathbf{D}_{ii}\,(\mathbf{Q}^{-1})_{ii} = 1$, $\mathbf{D}_{ii}$ cannot be 0). Therefore, $\mathbf{D}$ is also invertible, which makes $\mathbf{I}_{kk} - \mathbf{W}_{kk}$ an invertible matrix as well. This enables us to "cancel out" the $\mathbf{I}_{kk} - \mathbf{W}_{kk}$ on both sides of Eq. A.12 and establish the equivalence to the retrieval of explicit rPCN (Eq. 2.27).

A.2 Derivation of rPCNs and Hopfield Networks as metric learning

Hopfield Networks Given a set of memorized patterns $\{\mathbf{x}(i)\}_{i=1}^{N}$ and a query $\mathbf{q}$, the energy function of a (vanilla) Hopfield Network can be written as:

$$\mathcal{F}_{HN} = -\sum_{i=1}^{N} \left(\mathbf{q}^\top \mathbf{x}(i)\right)^2 \tag{A.14}$$

If we denote the collection of all patterns as a matrix $X \in \mathbb{R}^{N\times d}$ where d denotes the dimension of each pattern, this energy can also be rewritten in terms of the sample covariance matrix of patterns $\mathbf{S}$:

$$\mathcal{F}_{HN} = -\frac{1}{2}\mathbf{q}^\top \mathbf{X}^\top \mathbf{X}\mathbf{q} \propto -\frac{1}{2}\mathbf{q}^\top \mathbf{S}\mathbf{q} = -\frac{1}{2}\mathbf{q}^\top (\mathbf{S}^{\frac{1}{2}})^\top \mathbf{S}^{\frac{1}{2}}\mathbf{q} = -\frac{1}{2}\|\mathbf{S}^{\frac{1}{2}}\mathbf{q}\|_2^2 \tag{A.15}$$

which follows the general expression in Eq. 2.31 with $c = -\frac{1}{2}$ and $\mathbf{L} = \mathbf{S}^{\frac{1}{2}}$.

Explicit rPCN The parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of the energy of the explicit rPCN will converge to the sample statistic $\bar{\mathbf{x}}$ and $\mathbf{S}$ after learning. Thus, the energy given a query $\mathbf{q}$ would become:

$$\mathcal{F}_{\text{exp}} = \frac{1}{2}\log|\mathbf{S}| + \frac{1}{2}(\mathbf{q}-\bar{\mathbf{x}})^\top \mathbf{S}^{-1}(\mathbf{q}-\bar{\mathbf{x}}) \tag{A.16}$$

Assuming zero-mean memories ($\bar{\mathbf{x}}$=0) and dropping the constant term $\frac{1}{2}\log|\mathbf{S}|$ unrelated to $\mathbf{q}$ after training, this energy can be simplified as:

$$\mathcal{F}_{\text{exp}} = \frac{1}{2}\mathbf{q}^\top \mathbf{S}^{-1}\mathbf{q} = \frac{1}{2}\mathbf{q}^\top (\mathbf{S}^{-\frac{1}{2}})^\top \mathbf{S}^{-\frac{1}{2}}\mathbf{q} = \frac{1}{2}\|\mathbf{S}^{-\frac{1}{2}}\mathbf{q}\|_2^2 \tag{A.17}$$

which follows the general expression in Eq. 2.31 with $c = \frac{1}{2}$ and $\mathbf{L} = \mathbf{S}^{-\frac{1}{2}}$.

Implicit rPCN To start, note that the training phase of implicit rPCN can be seen as a constrained optimization problem. By Eq. 2.10, assuming the entire set of patterns $\mathbf{X}$ and zero bias ($\boldsymbol{\nu} = 0$), we have

$$\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{X} - \mathbf{X}\mathbf{W}\|_F^2 \quad \text{s.t.} \quad \text{diag}(\mathbf{W}) = \mathbf{0} \tag{A.18}$$

where for simplicity of notation, $\mathbf{W}$ in the appendix is the transpose of W used in the main text. Then, we can equivalently write the constraints into the Lagrangian

$$\mathcal{F}_{\text{imp}} = \frac{1}{2}\|\mathbf{X} - \mathbf{X}\mathbf{W}\|_F^2 + \boldsymbol{\lambda}^\top \text{diag}(\mathbf{W}) \tag{A.19}$$

where $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_d)$ is a vector of Lagrangian multipliers. Taking gradient with respect to $\mathbf{W}$ yields that

$$\begin{aligned}
\frac{\partial \mathcal{F}_{\text{imp}}}{\partial \mathbf{W}} &= \frac{\partial}{\partial \mathbf{W}} \left(\frac{1}{2}\text{Tr}(\mathbf{X}^\top\mathbf{X} - \mathbf{X}^\top\mathbf{X}\mathbf{W} - \mathbf{W}^\top\mathbf{X}^\top\mathbf{X} + \mathbf{W}^\top\mathbf{X}^\top\mathbf{X}\mathbf{W}) + \boldsymbol{\lambda}^\top \text{diag}(\mathbf{W}) \right) \\
&= -\frac{\partial}{\partial \mathbf{W}}\text{Tr}(\mathbf{W}\mathbf{X}^\top X) + \frac{\partial}{\partial \mathbf{W}}\text{Tr}(\mathbf{W}^\top\mathbf{X}^\top\mathbf{X}\mathbf{W}) + \text{diagMat}(\boldsymbol{\lambda}) \\
&= \mathbf{X}^\top\mathbf{X}(\mathbf{W} - \mathbf{I}) + \text{diagMat}(\boldsymbol{\lambda})
\end{aligned} \tag{A.20}$$

Similarly, taking gradient with respect to $\boldsymbol{\lambda}$ yields:

$$\frac{\partial \mathcal{F}_{\text{imp}}}{\partial \boldsymbol{\lambda}} = \text{diag}(\mathbf{W}) \tag{A.21}$$

Setting the gradient $\frac{\partial \mathcal{F}_{\text{imp}}}{\partial \mathbf{W}}$ to $\mathbf{0}$ yields:

$$\hat{\mathbf{W}} = I - \mathbf{S}^{-1}\text{diagMat}(\boldsymbol{\lambda}) \tag{A.22}$$

By substituting $\hat{\mathbf{W}}$ into Eq. A.21 and setting it to $\mathbf{0}$ we get:

$$\hat{\boldsymbol{\lambda}} = \mathbf{1} \oslash \text{diag}(\mathbf{S}^{-1}) \tag{A.23}$$

where $\oslash$ is the element-wise division. Finally, by substituting $\hat{\boldsymbol{\lambda}}$ back into Eq. A.22 we get the expression of the optimal $\mathbf{W}$:

$$\hat{\mathbf{W}} = \mathbf{I} - \mathbf{S}^{-1}\text{diagMat}(\mathbf{1} \oslash \text{diag}(\mathbf{S}^{-1})) \tag{A.24}$$

It can also be verified that $(\hat{\mathbf{W}}, \hat{\boldsymbol{\lambda}})$ is indeed the global minimum by substituting it in Eq. A.18.

Now, to express rPCN as performing metric learning in the form of Eq. 2.31 with a query $\mathbf{q}$, note that by replacing W with the optimal $\hat{\mathbf{W}}$:

$$\begin{aligned}
\mathcal{F}_{\text{imp}} &= \frac{1}{2}\|\mathbf{q} - \mathbf{W}^\top\mathbf{q}\| \\
&= \frac{1}{2}\|(\mathbf{I} - \mathbf{W})^\top\mathbf{q}\|_2^2 \\
&= \frac{1}{2}\|\text{diagMat}(\mathbf{1} \oslash \text{diag}(\mathbf{S}^{-1}))\mathbf{S}^{-1}\mathbf{q}\|_2^2
\end{aligned} \tag{A.25}$$

which follows the general expression of Eq. 2.31.

A.3 Experimental results with MNIST

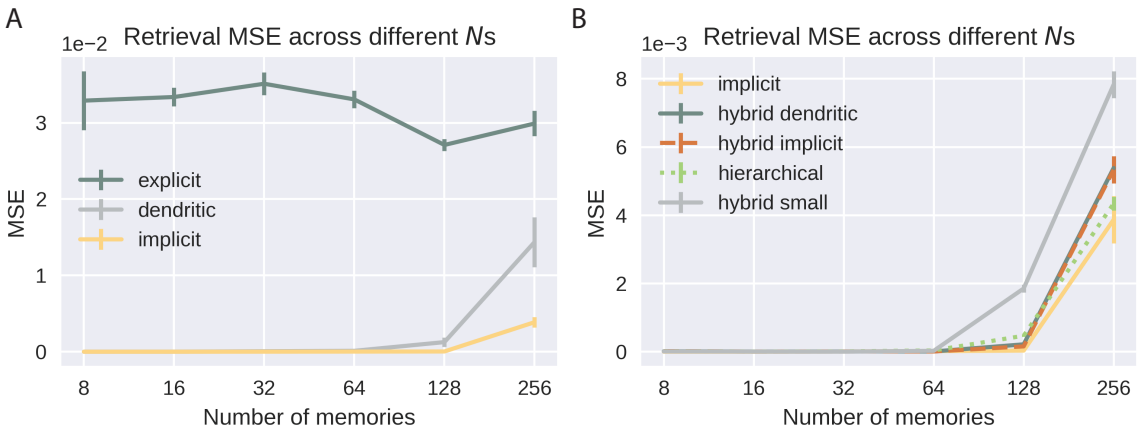

Figure A.1: A: Retrieval MSEs of the single-layer models across multiple Ns. B: Retrieval MSEs of the multi-layer models across multiple Ns.

In this section of the Appendix we show the quantitative results obtained with the MNIST dataset. Fig A.1A shows the memory retrieval MSEs when the explicit, implicit and dendritic models were trained to memorize various numbers of MNIST images, and were given half-covered memories to retrieve from. The performances of these models are similar to those with grayscale CIFAR10 (Fig 2.5), where the implicit model has the lowest retrieval MSE, while the dendritic model's retrieval performance will start to deteriorate when N becomes large, due to its unstable neural dynamics. The performance of the explicit model is the worst due to the need to compute the inverse term. In fact, we noticed that the learning of the explicit models with MNIST can hardly converge with some subsets of data, which resulted in the non-monotonically increasing retrieval MSEs as we increase N. We suspect that this is due to the innate dark backgrounds of the MNIST images: the black pixels in the MNIST images have perfect collinearity (e.g., the top-right and top-left black pixels), which makes the covariance matrix rank-deficient and thus non-invertible. This will prevent the learning of Σ from converging.

Fig A.1B shows the performance of the multi-layer models in the same retrieval task from half-covered MNIST images. For comparison the same line from Fig A.1A for the implicit model is also added. The number of parameters in these models were also controlled to be the same. The results are mostly similar to those with the grayscale CIFAR10 images, where the performance different between the implicit and dendritic models are remedied by the hierarchical pre-processing, and the implicit

model performs the best as it represents the linear solution with the lowest MSE. Interestingly though, the purely hierarchical model from Salvatori et al. (2021) performs slightly better than the hybrid models. We leave the investigation into this phenomenon for future works.

A.4 Examples of retrieved color CIFAR10 images with different levels of corruptions.

We also performed preliminary experiments with the dendritic model on colored CIFAR10 images, and show some examples here. Particularly, we found that when presenting the network a cue consisting of 1/2 (panel A), 1/4 (panel B) and 1/8 (panel C) of the original pixels, the model has successfully recovered, respectively and on average, 89, 79, and 49 of the original memories in the experiments using colored CIFAR10.

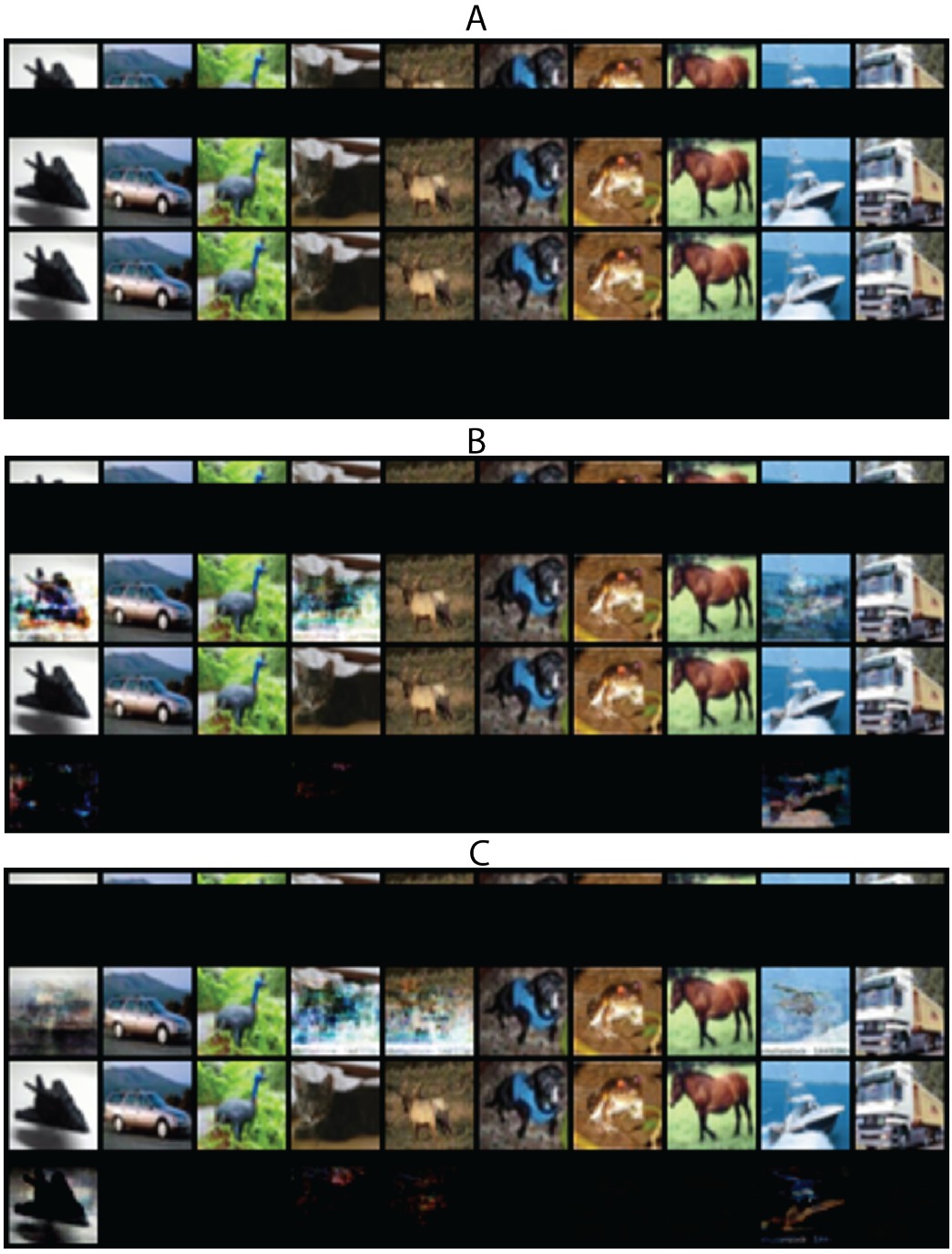

Figure A.2: Experiments with colored CIFAR10 imags. See main text above for details.

A.5 Implementation details of experiments with grayscale CIFAR10

For all sample size of memories N, we use a batch size of $N/8$. For the inference iterations with the multi-layer models, the first number 400 is the number of inference iterations during training and within each training iteration. The second number 100000 is the number of inference iterations used to retrieve the original patterns.

model	corruption	α	learning iters	β	inference iters
explicit	cover half	1.00E-05	800	1.00E-01	5000
implicit/dendritic	cover half	1.00E-04	800	1.00E-01	50000
implicit/nonlinear implicit	noise (var=0.1)	1.00E-04	800	1.00E-03	20000
nonlinear implicit	cover half	1.00E-04	800	1.00E-01	50000
hybrid/hierarchical	cover half	1.00E-03	200	1.00E-02	400/100000

A.6 Implementation details of experiments with MNIST

For all sample sizes of memories N, we use a batch size of $N/8$. For the inference iterations with the multi-layer models, the first number 400 is the number of inference iterations during training and within each training iteration. The second number 100000 is the number of inference iterations used to retrieve the original patterns.

model	corruption	α	learning iters	β	inference iters
explicit	cover half	1.00E-05	1000	1.00E-01	5000
implicit/dendritic	cover half	2.00E-04	1000	2.00E-02	50000
hybrid/hierarchical	cover half	1.00E-03	200	1.00E-02	400/100000

Appendix B

Supplementary: Temporal Predictive Coding

B.1 Derivation of recursive Bayesian estimation

Here we provide detailed derivations of Equation 3.16, showing that the original Bayesian filtering problem can be simplified as a recursive estimation process that finally leads to Kalman filtering and our own temporal predictive coding:

$$
\begin{aligned}
p(\mathbf{x}_k|\mathbf{y}_{1:k}) &= \int p(\mathbf{x}_{1:k}|\mathbf{y}_{1:k}) d\mathbf{x}_1, \ldots, d\mathbf{x}_{k-1} \\
&\overset{(a)}{\propto} \int p(\mathbf{y}_{1:k}|\mathbf{x}_{1:k}) p(\mathbf{x}_{1:k}) d\mathbf{x}_1, \ldots, d\mathbf{x}_{k-1} \\
&\overset{(b)}{=} \int \prod_{i=1}^{k} p(\mathbf{y}_i|\mathbf{x}_i) p(\mathbf{x}_i|\mathbf{x}_{i-1}) d\mathbf{x}_1, \ldots, d\mathbf{x}_{k-1} \\
&= p(\mathbf{y}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) \prod_{i=1}^{k-1} p(\mathbf{y}_i|\mathbf{x}_i) p(\mathbf{x}_i|\mathbf{x}_{i-1}) d\mathbf{x}_1, \ldots, d\mathbf{x}_{k-1} \\
&= p(\mathbf{y}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) \left[\int \prod_{i=1}^{k-1} p(\mathbf{y}_i|\mathbf{x}_i) p(\mathbf{x}_i|\mathbf{x}_{i-1}) d\mathbf{x}_1, \ldots, d\mathbf{x}_{k-2} \right] d\mathbf{x}_{k-1} \\
&\overset{(c)}{\propto} p(\mathbf{y}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) d\mathbf{x}_{k-1}
\end{aligned}
\tag{B.1}
$$

where step (a) is achieved by Bayes' rule, step (b) is achieved by the Markov assumption we made on the hidden states, and step (c) is achieved by observing that the integral in the square bracket in the penultimate line is proportional to the Bayesian filtering posterior at the previous step $k-1$. This can be observed by comparing it with the expression on the third line of the above equations. Importantly, this expression provides a recursive solution to the problem i.e., knowing the posterior at step $k-1$, we could estimate the posterior at step k.

B.2 Derivation of the update rules for the models

Here we derive the update rule for $\hat{\mathbf{x}}_k$ that underlies both predictive coding and Kalman filtering (Equations 3.22 and 3.26). Notice that the objective functions for these two models follow a unified form:

$$\mathcal{F}_k = \frac{1}{2}(\mathbf{y}_k - \mathbf{F}\mathbf{x}_k)^\top \boldsymbol{\Sigma}_y^{-1}(\mathbf{y}_k - \mathbf{F}\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^\top \mathbf{M}^{-1}(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) \tag{B.2}$$

where $\mathbf{M} = \boldsymbol{\Sigma}_x$ for predictive coding and $\mathbf{M} = \mathbf{W}\boldsymbol{\Sigma}_{k-1}\mathbf{W}^\top + \boldsymbol{\Sigma}_x$ for Kalman filtering. To obtain $\hat{\mathbf{x}}_k$ that minimizes $\mathcal{F}_k$, we first take the derivative of $\mathcal{F}_k$ with respect to $\mathbf{x}_k$:

$$\frac{\partial \mathcal{F}_k}{\partial \mathbf{x}_k} = (\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}\mathbf{F} + \mathbf{M}^{-1})\mathbf{x}_k - (\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}\mathbf{y}_k + \mathbf{M}^{-1}\hat{\mathbf{x}}_k^-). \tag{B.3}$$

Then, by setting the derivative to 0 we have the optimal $\hat{\mathbf{x}}_k$:

$$\begin{aligned}
\hat{\mathbf{x}}_k &= (\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}\mathbf{F} + \mathbf{M}^{-1})^{-1}(\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}\mathbf{y}_k + \mathbf{M}^{-1}\hat{\mathbf{x}}_k^-) \\
&\overset{(a)}{=} [\mathbf{M} - \mathbf{M}\mathbf{F}^\top(\boldsymbol{\Sigma}_y + \mathbf{F}\mathbf{M}\mathbf{F}^\top)^{-1}\mathbf{F}\mathbf{M}](\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}\mathbf{y}_k + \mathbf{M}^{-1}\hat{\mathbf{x}}_k^-) \\
&\overset{(b)}{=} [\mathbf{M} - \mathbf{K}\mathbf{F}\mathbf{M}](\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}\mathbf{y}_k + \mathbf{M}^{-1}\hat{\mathbf{x}}_k^-) \\
&= \hat{\mathbf{x}}_k^- - \mathbf{K}\mathbf{F}\hat{\mathbf{x}}_k^- + [\mathbf{M}\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1} - \mathbf{K}\mathbf{F}\mathbf{M}\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}]\mathbf{y}_k \\
&= \hat{\mathbf{x}}_k^- - \mathbf{K}\mathbf{F}\hat{\mathbf{x}}_k^- + [\mathbf{K}\mathbf{K}^{-1}\mathbf{M}\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1} - \mathbf{K}\mathbf{F}\mathbf{M}\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}]\mathbf{y}_k \\
&= \hat{\mathbf{x}}_k^- - \mathbf{K}\mathbf{F}\hat{\mathbf{x}}_k^- + \mathbf{K}[(\boldsymbol{\Sigma}_y + \mathbf{F}\mathbf{M}\mathbf{F}^\top)\mathbf{F}^{-T}\mathbf{M}^{-1}\mathbf{M}\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1} - \mathbf{F}\mathbf{M}\mathbf{F}^\top \boldsymbol{\Sigma}_y^{-1}]\mathbf{y}_k \\
&= \hat{\mathbf{x}}_k^- - \mathbf{K}\mathbf{F}\hat{\mathbf{x}}_k^- + \mathbf{K}\mathbf{y}_k \\
&= \hat{\mathbf{x}}_k^- + \mathbf{M}\mathbf{F}^\top(\boldsymbol{\Sigma}_y + \mathbf{F}\mathbf{M}\mathbf{F}^\top)^{-1}(\mathbf{y}_k - \mathbf{F}\hat{\mathbf{x}}_k^-).
\end{aligned} \tag{B.4}$$

In step (a) we use the Woodbury matrix inversion identity and in step (b) we replace $\mathbf{M}\mathbf{F}^\top(\boldsymbol{\Sigma}_y + \mathbf{F}\mathbf{M}\mathbf{F}^\top)^{-1}$ with $\mathbf{K}$. Substituting $\mathbf{M} = \boldsymbol{\Sigma}_x$ for predictive coding and $\mathbf{M} = \mathbf{W}\boldsymbol{\Sigma}_{k-1}\mathbf{W}^\top + \boldsymbol{\Sigma}_x$ for Kalman filtering we get Equations 3.22 and 3.26 respectively.

B.3 Estimated position and velocity in the tracking task

Fig. B.1 shows the impact of the number of inference steps on the velocity and position dimensions in the tracking experiment. All values are with arbitrary units (a.u.).

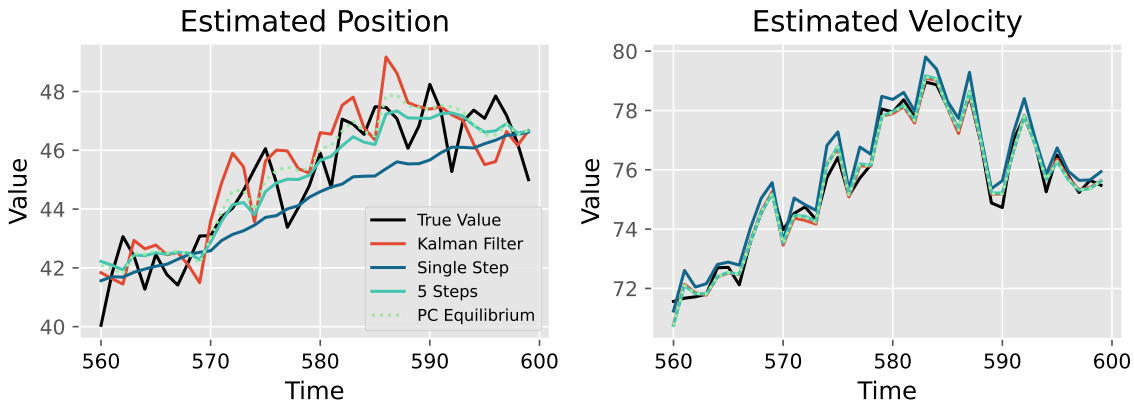

Figure B.1: **Estimated position and velocity (a.u.) in the tracking experiments with tPC and Kalman filtering.**

B.4 Tracking performance on the other two dimensions

The following figure shows the performance on the other two dimensions in both hidden and observation levels in the tracking experiment, where **W** and **F** are learned. All values are with arbitrary units (a.u.).

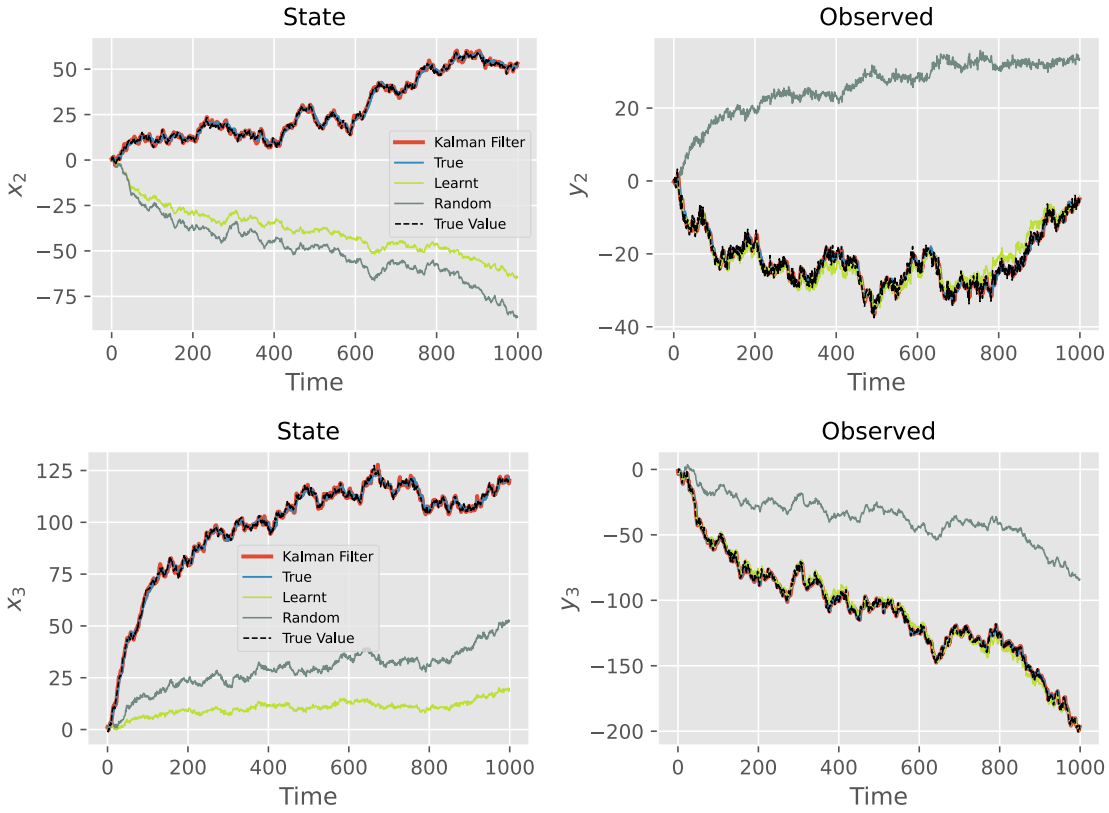

Figure B.2: **Estimated position and velocity (a.u.) in the tracking experiments with tPC and Kalman filtering, where W and F are learned.**

Appendix C

Supplementary: Sequential Memory with Temporal Predictive Coding

C.1 Algorithms

In Algorithm 2 we present the memorizing and recalling procedures of the single-layer tPC. In Algorithm 3 we present the memorizing and recalling procedures of the 2-layer tPC.

Algorithm 2 Memorizing and recalling with single-layer tPC

1: ▷ *Training/memorization*
2: **while** $\mathbf{W}$ not converged **do**
3: | **for** $k = 1, ..., K$ **do**
4: | | Input: $\mathbf{x}_k$, $\mathbf{x}_{k-1}$
5: | | Update $\mathbf{W}$ (Eqs. 4.7)
6: | **end for**
7: **end while**
8:
9: ▷ *Cued recall*
10: **for** $\mu = 1, ..., P$ **do**
11: | Input: $\mathbf{x}_{k-1}$ (online) or $\hat{\mathbf{x}}_{k-1}$ (offline)
12: | **while** $\hat{\mathbf{x}}_k$ not converged **do**
13: | | Infer $\hat{\mathbf{x}}_k$ (Eq. 4.9)
14: | **end while**
15: | $R^{tPC}(\mathbf{q}) \leftarrow \hat{\mathbf{x}}_k$
16: **end for**

It is worth noting that, although in both algorithms we used iterative inference (line 14-16 in Algorithm 2 and line 17-19 in Algorithm 3), these inferential dynamics can be replaced by forward passes in simulation. For the single-layer model the retrieval $R^{tPC}(\mathbf{q})$ can be directly obtained by $R^{tPC}(\mathbf{q}) = \mathbf{W}f(\mathbf{q})$ with the learned $\mathbf{W}$, while for the 2-layer model the retrieval $\mathbf{x}_{k-1}$ can be obtained by first forward passing the latent $\hat{\mathbf{z}}_k = \mathbf{W}f(\hat{\mathbf{z}}_{k-1})$ and then set the retrieval as $R^{tPC}(\mathbf{q}) = \mathbf{F}f(\hat{\mathbf{z}}_k)$. Effectively, setting the retrieval directly by forward passes will result in the same

Algorithm 3 Memorizing and recalling with 2-layer tPC

▷ *Training/memorization*
while $\mathbf{W}$, $\mathbf{F}$ not converged **do**
  randomly initialize $\hat{\mathbf{z}}^0$
  for $k = 1, ..., K$ **do**
    Input: $\mathbf{x}_k$, $\hat{\mathbf{z}}_{k-1}$
    while $\mathbf{z}_k$ not converged **do**
      Infer $\mathbf{z}_k$ (Eq. 4.11)
    end while
    Update $\mathbf{W}$, $\mathbf{F}$ (Eqs. 4.12)
    $\hat{\mathbf{z}}_k \leftarrow \mathbf{z}_k$
  end for
end while
▷ *Cued recall*
randomly initialize $\hat{\mathbf{z}}^0$
for $k = 1, ..., K$ **do**
  Input: $\hat{\mathbf{z}}_{k-1}$
  while $\mathbf{z}_k$ and $\hat{\mathbf{x}}_k$ not converged **do**
    Infer $\mathbf{z}_k$, $\hat{\mathbf{x}}_k$ (Eqs. 4.11,4.14)
  end while
  $\hat{\mathbf{z}}_k \leftarrow \mathbf{z}_k$
  $R^{tPC}(\mathbf{q}) \leftarrow \hat{\mathbf{x}}_k$
end for

retrieval as performing the inferential iterations as they are the fixed points of the inferential dynamics. However, obtaining the retrievals via iterative methods allows us to implement the computations in the plausible neural circuits in Fig. 4.1 whereas forward passes cannot.

C.2 Proof of Property 4

Here we present the proof for Property 4 in the main text, that the single-layer tPC can be viewed as a "whitened" version of the AHN. Without loss of generality, assume a sequence of zero-mean, real-valued patterns $\mathbf{x}_k$, $k = 1, ..., K$ is given to the model to memorize. The step-wise objective with an identity non-linearity $f(\cdot)$ for single-layer tPC is:

$$\mathcal{F}_k(\mathbf{W}) = \|\mathbf{x}_k - \mathbf{W}\mathbf{x}_{k-1}\|_2^2 \tag{C.1}$$

The weight $\mathbf{W}$ is then updated at each time-step once, and the whole sequence is presented for multiple iterations until $\mathbf{W}$ converges. We now consider the case where we presented the sequence at once, i.e., the model now minimizes the following objective at each learning iteration:

$$\mathcal{F} = \sum_{k=1}^{K} \|\mathbf{x}_{k+1} - \mathbf{W}\mathbf{x}_k\|_2^2 \tag{C.2}$$

which can be viewed as a "batched" version of Eq. C.1. Since the objective is now convex (with identity $f(\cdot)$), the fixed point obtained by these two objectives will be the same. The gradient descent update of $\mathbf{W}$ on $\mathcal{F}$ is then:

$$\Delta\mathbf{W} = -\frac{\partial \mathcal{F}}{\partial \mathbf{W}} = \sum_{k=1}^{K} \mathbf{x}_{k+1}(\mathbf{x}_k)^\top - \mathbf{W}\sum_{k=1}^{K} \mathbf{x}_k(\mathbf{x}_k)^\top \tag{C.3}$$

By setting $\Delta\mathbf{W}$ to 0, we could obtain the optimal $\mathbf{W}$, which we call $\mathbf{W}^{tPC}$:

$$\mathbf{W}^{tPC} = \left(\sum_{k=1}^{K} \mathbf{x}_{k+1}(\mathbf{x}_k)^\top\right)\left(\sum_{k=1}^{K} \mathbf{x}_k(\mathbf{x}_k)^\top\right)^{-1} = \sum_{k=1}^{K} \mathbf{x}_{k+1}(\mathbf{x}_k)^\top \left(\mathbf{X}^\top\mathbf{X}\right)^{-1} \tag{C.4}$$

where we define $\mathbf{X} = \left[\mathbf{x}^1, ..., \mathbf{x}^K\right]^\top$, the $K \times N$ data matrix. Recall that when presented with a query $\mathbf{q}$, the single-layer tPC updates its value nodes to minimize:

$$\mathcal{F}_k(\hat{\mathbf{x}}_k) = \|\hat{\mathbf{x}}_k - \mathbf{W}^{tPC}\mathbf{q}\|_2^2 \tag{C.5}$$

which will converge to $\mathbf{R}^{tPC}(\mathbf{q}) = \mathbf{W}^{tPC}\mathbf{q}$ due to convexity. We can now substitute $\mathbf{W}^{tPC}$ with the expression from Eq. C.4 to obtain the retrieval:

$$\mathbf{R}^{tPC}(\mathbf{q}) = \mathbf{W}^{tPC}\mathbf{q} = \sum_{k=1}^{K} \mathbf{x}_{k+1}(\mathbf{x}_k)^\top \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{q} \tag{C.6}$$

It can be immediately seen that the retrieval function of tPC is a special case of the UHN framework, where the similarity function is defined as $\text{sim}(\mathbf{x}_k, \mathbf{q}) = (\mathbf{x}_k)^\top \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{q}$ and the separation function is identity. Notice that since we assumed a zero-mean sequence (sequences with non-zero mean can be accounted for with a bias term in the objective function Eq. C.2), the term $\mathbf{X}^\top\mathbf{X}$ is exactly the covariance matrix of the sequence. Defining it as $\mathbf{\Sigma}$, the retrieval can be written as:

$$\mathbf{R}^{tPC}(\mathbf{q}) = \mathbf{W}^{tPC}\mathbf{q} = \sum_{k=1}^{K} \mathbf{x}_{k+1}(\mathbf{x}_k)^\top\mathbf{\Sigma}^{-1}\mathbf{q} \tag{C.7}$$

Assume a positive definite covariance $\mathbf{\Sigma}$, it is possible to decompose $\mathbf{\Sigma}^{-1}$ as follows:

$$\mathbf{\Sigma}^{-1} = \mathbf{M}^\top\mathbf{M} \tag{C.8}$$

The matrix $\mathbf{M}$ is called the whitening matrix, which does not hold a unique value, e.g., $\mathbf{M} = \mathbf{\Sigma}^{-\frac{1}{2}}$ or $\mathbf{M} = \mathbf{L}^\top$ where $\mathbf{L}$ is the Cholesky decomposition of $\mathbf{\Sigma}^{-1}$ (Kessy et al., 2018). Here, we are agnostic about its exact value. When applied to the data sequence, it whitens the data such that (i.e., Eq.16 in the main text):

$$\langle \mathbf{M}\mathbf{x}_k(\mathbf{M}\mathbf{x}_k)^\top\rangle_k = \mathbf{I}_N \tag{C.9}$$

Therefore, the retrieval of our single-layer tPC with an identity $f(\cdot)$ can be written as:

$$\mathbf{R}^{tPC}(\mathbf{q}) = \sum_{k=1}^{K} \mathbf{x}_{k+1}(\mathbf{M}\mathbf{x}_k)^\top\mathbf{M}\mathbf{q} \tag{C.10}$$

by decomposing $\mathbf{\Sigma}^{-1}$ in Eq. C.7 into $\mathbf{M}^\top\mathbf{M}$, which is Eq.15 in the main text and concludes the proof.

C.3 Energy comparison between tPC and AHN

The whitening effect of single-layer tPC can be traced back to it energy function as compared to that of AHN. For simplicity, we limit our discussion here to the minimal case where a single pair of consecutive patterns, $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ are being memorized, although it can be easily generalized to arbitrarily long sequences. Recall that, in this case, the energy function of a single-layer tPC is:

$$\begin{aligned}\mathcal{F}^{tPC}(\mathbf{W}) &= \|\mathbf{x}_k - \mathbf{W}\mathbf{x}_{k-1}\|_2^2 \\ &= \mathbf{x}_k^\top \mathbf{x}_k + \mathbf{x}_{k-1}^\top \mathbf{W}^\top \mathbf{W}\mathbf{x}_{k-1} - 2\mathbf{x}_k^\top \mathbf{W}\mathbf{x}_{k-1}\end{aligned} \tag{C.11}$$

On the other hand, the energy function of a vanilla AHN is (Hopfield, 1982; Sompolinsky and Kanter, 1986):

$$\mathcal{F}^{AHN}(\mathbf{W}) = -\mathbf{x}_k^\top \mathbf{W}\mathbf{x}_{k-1} \tag{C.12}$$

It can be seen that both energy functions have the quadratic term $\mathbf{x}_k^\top \mathbf{W}\mathbf{x}_{k-1}$, which tells both models to learn a *cross*-covariance matrix between the two patterns ($\partial \mathbf{x}_k^\top \mathbf{W}\mathbf{x}_{k-1}/\partial \mathbf{W} = \mathbf{x}_k \mathbf{x}_{k-1}^\top$). However, tPC also has an extra term $\mathbf{x}_{k-1}^\top \mathbf{W}^\top \mathbf{W}\mathbf{x}_{k-1}$. This term essentially regularizes the energy function, so that tPC also considers the *auto*-covariance while minimizing this energy ($\partial \mathbf{x}_{k-1}^\top \mathbf{W}^\top \mathbf{W}\mathbf{x}_{k-1}/\partial \mathbf{W} = 2\mathbf{W}(\mathbf{x}_{k-1}\mathbf{x}_{k-1}^\top)$), which is the reason for the whitening effect in single-layer tPC.

C.4 Generation of binary patterns

For the experimental results in Fig. 4.2A and B, the correlated binary patterns are generated following the approach mentioned in Bogacz and Brown (2003). For a particular pattern dimension N, a template $\mathbf{x}^{temp} \in \{-1, 1\}^N$ is first generated. Then, for each of the $k = 1, ..., K$ patterns, the ith entry of $\mathbf{x}_k$ is equal to the ith entry of $\mathbf{x}^{temp}$ with probability $0.5 + 0.5b$ where b is the parameter controlling bias. We also invert the sign of each pattern by chance to keep the level of activity constant for each neuron. This whole process is then repeated for multiple trials to average out randomness. The capacity P_{max} is calculated as the maximum K such that the percentage of erroneous entries across these trials is below 0.01. It can be shown that the correlation between two features of the generated patterns $r(\mathbf{x}_{k,i}, \mathbf{x}_{k,j})$ is $-b^2$ or b^2, by using the identity $r(\mathbf{x}_{k,i}, \mathbf{x}_{k,j}) = \langle \mathbf{x}_{k,i}\mathbf{x}_{k,j}\rangle_k - \langle \mathbf{x}_{k,i}\rangle_k \langle \mathbf{x}_{k,j}\rangle_k$.

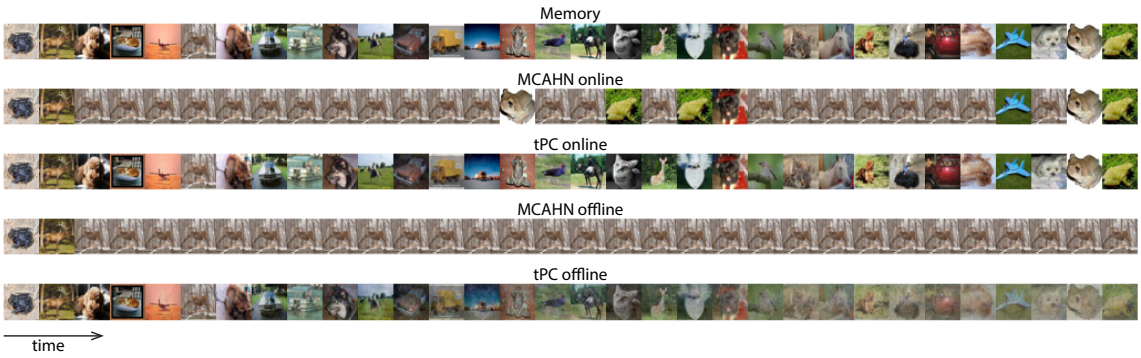

Figure C.1: **Visual results with CIFAR10 sequences.**

C.5 Additional results with CIFAR10 and MovingMNIST

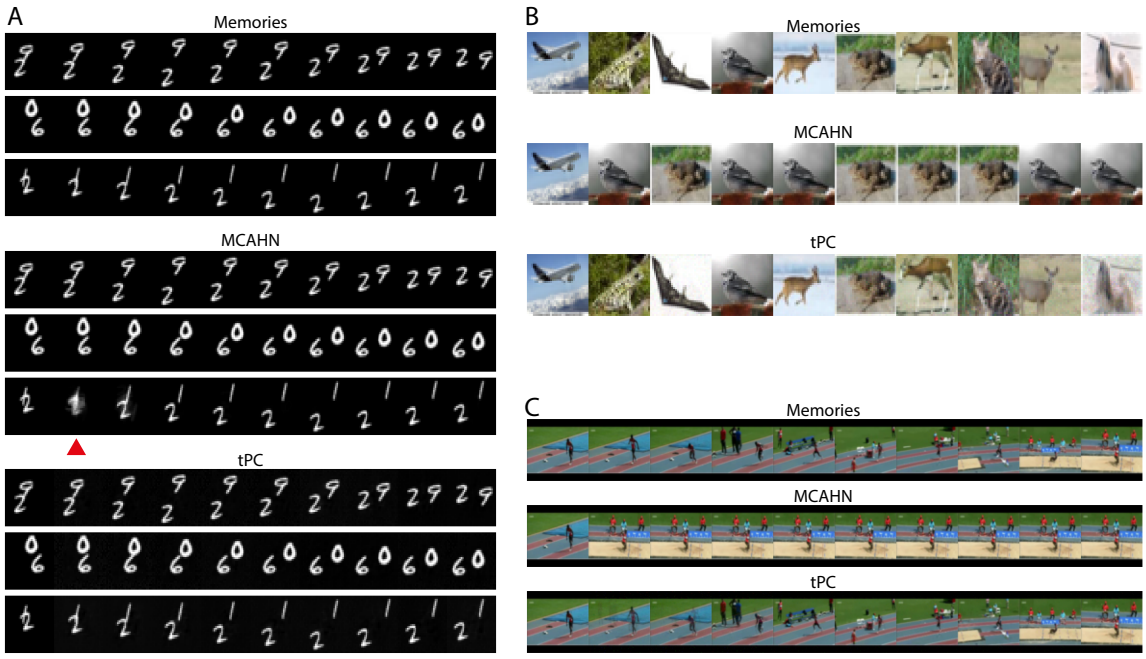

Figure C.2: **Online recalls with** A: MovingMNIST; B: CIFAR10; C: UCF101.

In Fig. C.1 we present additional visual results when MCAHN and single-layer tPC are trained to memorize a random CIFAR10 sequence of 32 images and are queried both online and offline during recall. It can be seen that MCAHN, like what we have shown in the main text, recalls memories preceded with images with large pixel values (images are presented to the models as $32 \times 32 \times 3 = 3072$-dimensional vectors where

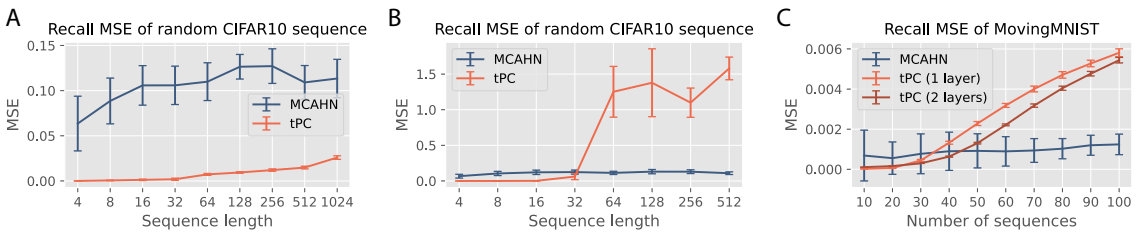

Figure C.3: **Numerical results with CIFAR10 and MovingMNIST.** A: *Online* recall MSE of random CIFAR10 sequences; B: *Offline* recall MSE of random CIFAR10 sequences, with a `tanh` nonlinearity; C: *Offline* recall MSE of movingMNIST dataset, with a `tanh` nonlinearity.

3 represents the RGB channels) in both online and offline recall regimes, whereas tPC does not suffer from this problem because of the whitening procedure. However, it can be seen that the recall by tPC will gradually become more blurry and noisier when queried offline because the recall errors will accumulate temporally.

These observations are consistent with our numerical results shown in Fig. C.3. In Fig. C.3A we show the online recall MSE of CIFAR10 sequences by single-layer tPC (with linear $f(\cdot)$) and MCAHN. MCAHN has a much larger MSE than that of the tPC because of the entirely wrong recalls. In offline recalls in Fig. C.3B however, tPC will have exploding MSE as soon as P reaches 64 because of the accumulating recall errors. It is worth mentioning that for these experiments, we used a `tanh` nonlinearity, as the recall error will accumulate to infinity with an identity $f(\cdot)$. This is the only case where identity and `tanh` are different in our experiments.

In Fig. C.2 we also present the online recall results of the models in MovingMNIST, CIFAR10 and UCF101. The results with CIFAR10 are consistent with the discussions above, and the results with UCF101 is clearer with online queries than those with offline queries. Moreover, although in MovingMNIST MCAHN still suffers from the wrong attractor problem (red triangle in Fig. C.2A), the online query can prevent it from staying in the wrong attractor. This is consistent with our numerical obervation in Fig. C.3C, where the performance of MCAHN in online recall of MovingMNIST is better than that of the tPC models.

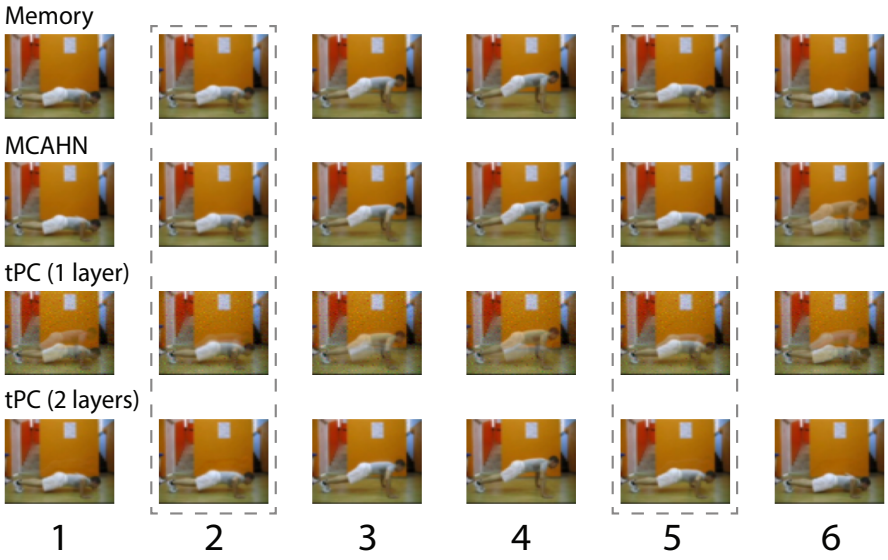

Figure C.4: **A natural aliased example from the UCF101 dataset, showing a human doing push-ups.**

C.6 A natural example of aliased sequences from UCF101

In Fig C.4 we show a natural example of aliased sequences where a movie of a human doing push-ups is memorized and recalled by the model. The frames at the second and the fifth steps are almost identical, leading to inaccurate predictions of the single-layer models (including MCAHN and single-layer tPC) at the next time steps. On the other hand, the 2-layer tPC performs well and produces sharp and correct recalls.

Appendix D

Supplementary: Learning Grid Cells by Predictive Coding

D.1 Algorithms

Below is the training algorithm for a sparse, non-negative PCN given spatial inputs $\mathbf{p}$. We obtain the grid cells shown in the main text directly by taking the converged latent activities $\mathbf{g}$ after training.

Algorithm 4 Learning latent representations of space with a PCN

▷ *Training*
while $\mathbf{W}$ not converged **do**
    Initialize $\mathbf{g}$ randomly;
    Input: $\mathbf{p}$
    while $\mathbf{g}$ not converged **do**
        $\mathbf{g} \leftarrow \texttt{ReLU}(\mathbf{g}_t + \Delta\mathbf{g}_t)$ (Eq. 5.2)
    end while
    Update $\mathbf{W}$ (Eqs. 5.3)
end while

Below is the training algorithm for tPCN in path integration tasks. The testing performance and grid cells shown in the main text are obtained by performing a forward pass through the model after training, given an unseen trajectory $\{\mathbf{v}_t, \mathbf{p}_t\}$.

D.2 Formulation of principal component analysis (PCA) within the context of spatial learning

Here we provide an intuition for the PCA formulation presented in Eq. 5.9 as a constrained least square problem, by relating it to the more well-known "maximum

Algorithm 5 Path integration with tPCN

▷ *Training*
while $\mathbf{W}$, $\mathbf{F}$, $\mathbf{B}$ not converged **do**
  Initialize $\hat{\mathbf{g}}_0$ randomly or from $\mathbf{p}_0$ via a PCN;
  for $t = 1, ..., T$ **do**
    Input: $\mathbf{p}_t$, $\hat{\mathbf{g}}_{t-1}$ and optionally $\mathbf{v}_t$
    Initialize $\mathbf{g}_t = f(\mathbf{W}\hat{\mathbf{g}}_{t-1})$
    for $k = 1, ..., K$ **do**
      $\mathbf{g}_t \leftarrow \mathbf{g}_t + \Delta\mathbf{g}_t$ (Eq. 5.5)
    end for
    Update $\mathbf{W}$, $\mathbf{F}$, $\mathbf{B}$ (Eqs. 5.6)
    $\hat{\mathbf{g}}_t \leftarrow \mathbf{g}_t$
  end for
end while

▷ *Testing*
Initialize $\mathbf{g}_0$ randomly or from $\mathbf{p}_0$ via a PCN;
for $t = 1, ..., T$ **do**
  Input: $\mathbf{g}_{t-1}$ and optionally $\mathbf{v}_t$
  Obtain $\mathbf{g}_t$, $\hat{\mathbf{p}}_t$ via Eq. 5.7
end for

variance" formulation. Recall that the goal of PCA is to find a lower-dimensional projection of the data ($\mathbf{P}$ in our case), such that the projected data has the maximized variance. Suppose the ith place cell has its activity across N_x locations $\mathbf{p}(i) \in \mathbb{R}^{N_x}$ and we (tentatively) want to find a 1-dimensional projection of it through a projection vector $\mathbf{g}(1) \in \mathbb{R}^{N_x}$. The scalar, 1-dimensional projection is performed by $\mathbf{g}(1)^\top \mathbf{p}(i)$. For all $i = 1, ..., N_p$ place cells, assume the mean of their activities across N_x locations is 0 (without loss of generality), the variance of the projected dataset is then:

$$\frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{g}(1)^\top \mathbf{p}(i))^2 = \mathbf{g}(1)^\top \left(\frac{1}{N_p}\sum_{i=1}^{N_p}\mathbf{p}(i)\mathbf{p}(i)^\top\right)\mathbf{g}(1) = \mathbf{g}(1)^\top \mathbf{S}_p \mathbf{g}(1) \tag{D.1}$$

where $\mathbf{S}_p$ is the $N_x \times N_x$ covariance of matrix of the place cells, which represents "how similar are two positions, meausred as the similarity between their place cell representations or codes?". To achieve the maximum-variance objective of PCA, we want to maximize $\mathbf{g}(1)^\top \mathbf{S}_p \mathbf{g}(1)$ with respect to $\mathbf{g}(1)$. An obvious but trivial solution is $\mathbf{g}(1) = \infty$. Therefore, to ensure our solution is sensible, we should constrain the norm $\|\mathbf{g}(1)\|_2$ to some finite value, which, without loss of generality, can simply be 1. The optimization problem is then turned into a constrained one:

$$\max_{\mathbf{g}(1)} \mathbf{g}(1)^\top \mathbf{S}_p \mathbf{g}(1) \quad \text{s.t.} \quad \mathbf{g}(1)^\top \mathbf{g}(1) = 1 \tag{D.2}$$

We can solve this problem using a Lagrangian multiplier λ_1 and reformulate it as:

$$\max_{\mathbf{g}(1)} \mathbf{g}(1)^\top \mathbf{S}_p \mathbf{g}(1) + \lambda_1(1 - \mathbf{g}(1)^\top \mathbf{g}(1)) \tag{D.3}$$

By setting the derivative of the above objective with respect to $\mathbf{g}(1)$ to 0 we get

$$\mathbf{S}\mathbf{g}(1) = \lambda_1 \mathbf{g}(1) \tag{D.4}$$

which says that $\mathbf{g}(1)$ must be the eigenvector of $\mathbf{S}_p$. If we now multiply both sides of the equation by $\mathbf{g}(1)^\top$, we see that the variance of the projected data becomes:

$$\mathbf{g}(1)^\top \mathbf{S}\mathbf{g}(1) = \lambda_1 \tag{D.5}$$

which means that if we want to obtain the maximum variance of projected data $\mathbf{g}(1)^\top \mathbf{S}\mathbf{g}(1)$, λ_1 has to be the largest eigenvalue. In other words, $\mathbf{g}(1)$ has to be the first eigenvector of $\mathbf{S}_p$.

To relate this view of PCA to the least square formulation in Eq. 5.9, we first rewrite the Frobenius norm to the sum of vector $l2$ norms:

$$\mathcal{F}_{PCA} = \sum_{i=1}^{N_p} \|\mathbf{p}(i) - \mathbf{G}\mathbf{m}(i)\|_2^2 \quad \text{s.t.} \quad \mathbf{G}^\top\mathbf{G} = \mathbf{I} \tag{D.6}$$

where $\mathbf{G}$ is an $N_x \times N_g$ matrix, whose columns are $\mathbf{g}(1), ..., \mathbf{g}(N_g)$ and $\mathbf{m}(i)$ is a N_g dimensional column vector. Taking derivative of $\mathcal{F}_{PCA}$ with respect to $\mathbf{m}(i)$ an set it to 0 we get:

$$\mathbf{G}^\top\mathbf{p}(i) - \mathbf{G}^\top\mathbf{G}\mathbf{m}(i) = 0 \Rightarrow \mathbf{m}(i) = \mathbf{G}^\top\mathbf{p}(i) \tag{D.7}$$

with the constraint that $\mathbf{G}^\top\mathbf{G} = \mathbf{I}$. We can thus rewrite the objective as:

$$\begin{aligned}
\mathcal{F}_{PCA} &= \sum_{i=1}^{N_p} \|\mathbf{p}(i) - \mathbf{G}\mathbf{G}^\top\mathbf{p}(i)\|_2^2 \\
&= \sum_{i=1}^{N_p} \mathbf{p}(i)^\top\mathbf{p}(i) - 2\mathbf{p}(i)^\top\mathbf{G}\mathbf{G}^\top\mathbf{p}(i) + \mathbf{p}(i)^\top\mathbf{G}\mathbf{G}^\top\mathbf{G}\mathbf{G}^\top\mathbf{p}(i) \\
&= \sum_{i=1}^{N_p} \mathbf{p}(i)^\top\mathbf{p}(i) - \mathbf{p}(i)^\top\mathbf{G}\mathbf{G}^\top\mathbf{p}(i)
\end{aligned} \tag{D.8}$$

If we minimize the above expression with respect to $\mathbf{G}$, it becomes equivalent to maximize:

$$\begin{aligned}
\sum_{i=1}^{N_p} \mathbf{p}(i)^\top\mathbf{G}\mathbf{G}^\top\mathbf{p}(i) &= tr(\mathbf{P}^\top\mathbf{G}\mathbf{G}^\top\mathbf{P}) \\
&= tr(\mathbf{G}^\top\mathbf{P}\mathbf{P}^\top\mathbf{G}) \\
&= tr(\mathbf{G}^\top\mathbf{S}_p\mathbf{G}) \\
&= \sum_{j=1}^{N_g} \mathbf{g}(j)^\top\mathbf{S}_p\mathbf{g}(j)
\end{aligned} \tag{D.9}$$

where tr denotes the trace of a matrix. To maximize for the case $N_g = 1$, we have shown above that $\mathbf{g}(1)$ has to be the first eigenvector of $\mathbf{S}_g$. Since $\mathbf{S}_g$ is a positive semi-definite matrix, we know that for any j, $\mathbf{g}(j)^\top\mathbf{S}_p\mathbf{g}(j) \geq 0$, and thus to maximize

the sum $\sum_{j=1}^{N_g} \mathbf{g}(j)^\top \mathbf{S}_p \mathbf{g}(j)$, $\mathbf{g}(j), j = 1, ..., N_g$ has to be the first N_g eigenvectors of $\mathbf{S}_g$ so that the sum is the sum of the biggest N_g eigenvalues. We have now formally linked the least-square formulation of PCA to the maximum variance interpretation. Bringing it back to place/grid cells, we have established that grid cells are, in essence, the top-N_g eigenvectors of the spatial covariance matrix $\mathbf{S}_p$ (Dordek et al., 2016; Sorscher et al., 2023).

D.3 Derivations of learning dynamics

Here we derive the recurrent weight update rules for $\mathbf{W}^{\text{RNN}}$ (Equation 5.12) and $\mathbf{W}^{\text{tPCN}}$ (Equation 5.13). For RNN, we assume that the weights are updated at each time step and therefore $\mathbf{W}^{\text{RNN}}$ is updated following the chain rule:

$$\Delta \mathbf{W}^{\text{RNN}} = -\frac{d\mathcal{F}_{\text{RNN},t}}{d\mathbf{W}} = -\frac{d\mathcal{F}_{\text{RNN},t}}{d\mathbf{g}_t}\frac{d\mathbf{g}_t}{d\mathbf{W}} \tag{D.10}$$

We first look at the term $\frac{d\mathbf{g}_t}{d\mathbf{W}}$, which, following the rule of partial derivatives, can be written as:

$$\begin{aligned}
\frac{d\mathbf{g}_t}{d\mathbf{W}} &= \frac{\partial \mathbf{g}_t}{\partial \mathbf{W}} + \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_{t-1}}\frac{d\mathbf{g}_{t-1}}{d\mathbf{W}} \\
&= \frac{\partial \mathbf{g}_t}{\partial \mathbf{W}} + \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_{t-1}}\left(\frac{\partial \mathbf{g}_{t-1}}{\partial \mathbf{W}} + \frac{\partial \mathbf{g}_{t-1}}{\partial \mathbf{g}_{t-2}}\frac{d\mathbf{g}_{t-2}}{d\mathbf{W}}\right) \\
&= ... \\
&= \sum_{k=1}^{t} \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_k}\frac{\partial \mathbf{g}_k}{\partial \mathbf{W}}
\end{aligned} \tag{D.11}$$

due to the recursive and implicit dependency of $\mathbf{g}_t$ on $\mathbf{g}_{t-1}$ and $\mathbf{g}_{t-1}$ on $\mathbf{W}^{\text{RNN}}$ for all t. Thus, the update rule can be written as:

$$\Delta \mathbf{W}^{\text{RNN}} = -\sum_{k=1}^{t} \frac{d\mathcal{F}_{\text{RNN},t}}{d\mathbf{g}_t}\frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_k}\frac{\partial \mathbf{g}_k}{\partial \mathbf{W}} \tag{D.12}$$

Since $\mathbf{g}_k = h(\tilde{\mathbf{g}}_k) = h(\mathbf{W}\mathbf{g}_{k-1} + \mathbf{B}\mathbf{v}_k)$, and $\mathcal{F}_{\text{RNN},t} = \|\mathbf{p}_t - f(\mathbf{F}\mathbf{g}_t)\|_2^2$ the update rule can be written as:

$$\Delta \mathbf{W}^{\text{RNN}} = \sum_{k=1}^{t} \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_k} h'(\tilde{\mathbf{g}}_t)\mathbf{F}^\top f'(\mathbf{F}\mathbf{g}_t)\boldsymbol{\epsilon}_t^{\text{p}} \mathbf{g}_{k-1}^\top, \tag{D.13}$$

concluding our proof for Equation 5.12. The derivation for $\mathbf{B}^{\text{RNN}}$ is similar:

$$\Delta \mathbf{B}^{\text{RNN}} = -\frac{d\mathcal{F}_{\text{RNN},t}}{d\mathbf{B}} = -\frac{d\mathcal{F}_{\text{RNN},t}}{d\mathbf{g}_t}\frac{d\mathbf{g}_t}{d\mathbf{B}}, \tag{D.14}$$

and

$$\begin{aligned}\frac{d\mathbf{g}_t}{d\mathbf{B}} &= \frac{\partial \mathbf{g}_t}{\partial \mathbf{B}} + \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_{t-1}} \frac{d\mathbf{g}_{t-1}}{d\mathbf{B}} \\ &= \frac{\partial \mathbf{g}_t}{\partial \mathbf{B}} + \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_{t-1}} \left(\frac{\partial \mathbf{g}_{t-1}}{\partial \mathbf{B}} + \frac{\partial \mathbf{g}_{t-1}}{\partial \mathbf{g}_{t-2}} \frac{d\mathbf{g}_{t-2}}{d\mathbf{B}} \right) \\ &= ... \\ &= \sum_{k=1}^{t} \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_k} \frac{\partial \mathbf{g}_k}{\partial \mathbf{B}}\end{aligned} \tag{D.15}$$

Therefore, the update rule for $\mathbf{B}$ in an RNN can be written as:

$$\Delta \mathbf{B}^{\text{RNN}} = \sum_{k=1}^{t} \frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_k} h'(\tilde{\mathbf{g}}_t) \mathbf{F}^\top f'(\mathbf{F}\mathbf{g}_t) \boldsymbol{\epsilon}_t^{\mathbf{p}} \mathbf{v}_k^\top \tag{D.16}$$

Finally, the update rule for $\mathbf{F}^{\text{RNN}}$ can be straightforwardly expressed as:

$$\begin{aligned}\Delta \mathbf{F}^{\text{RNN}} &= -\frac{d\mathcal{F}_{\text{RNN},t}}{d\mathbf{F}} \\ &= -\frac{d\mathcal{F}_{\text{RNN},t}}{d\hat{\mathbf{p}}_t} \frac{d\hat{\mathbf{p}}_t}{d\mathbf{F}} \\ &= f'(\mathbf{F}\mathbf{g}_t) \boldsymbol{\epsilon}_t^{\mathbf{p}} \mathbf{g}_t^\top\end{aligned} \tag{D.17}$$

as there is no recursive dependency.

For tPCN, at each time step t the following loss is minimized with respect to $\mathbf{W}$:

$$\mathcal{F}_{\text{tPCN},t} = \|\mathbf{p}_t - f(\mathbf{F}\mathbf{g}_t)\|_2^2 + \|\mathbf{g}_t - h(\mathbf{W}\hat{\mathbf{g}}_{t-1} + \mathbf{B}\mathbf{v}_t)\|_2^2 \tag{D.18}$$

Since $\hat{\mathbf{g}}_{t-1}$ is inferred through Equation 5.5, rather than forward-propagated by $\mathbf{W}$, the recursive dependency on $\mathbf{W}$ disappears, and thus the update rule for $\mathbf{W}$ can be locally derived as:

$$\Delta \mathbf{W}^{\text{tPCN}} = -\frac{d\mathcal{F}_{\text{tPCN},t}}{d\mathbf{W}} = h'(\tilde{\mathbf{g}}_t) \boldsymbol{\epsilon}_t^{\mathbf{g}} \hat{\mathbf{g}}_{t-1}^\top \tag{D.19}$$

If we also assume that the inference dynamics in Eq. 5.5 have converged when the weights are updated, namely:

$$\Delta \mathbf{g}_t = 0 \Rightarrow \boldsymbol{\epsilon}_t^{\mathbf{g}} = \mathbf{F}^\top f'(\mathbf{F}\mathbf{g}_t) \boldsymbol{\epsilon}_t^{\mathbf{p}}, \tag{D.20}$$

the update rule can be written as:

$$\Delta \mathbf{W}^{\text{tPCN}} = h'(\tilde{\mathbf{g}}_t) \mathbf{F}^\top f'(\mathbf{F}\mathbf{g}_t) \boldsymbol{\epsilon}_t^{\mathbf{p}} \hat{\mathbf{g}}_{t-1}^\top, \tag{D.21}$$

which concludes our proof for Equation 5.13. Similarly, following the same assumption of converged inference and Eq. 5.6, the update rule $\Delta \mathbf{B}^{\text{tPCN}}$ can be written as:

$$\Delta \mathbf{B}^{\text{tPCN}} = h'(\tilde{\mathbf{g}}_t)\mathbf{F}^\top f'(\mathbf{F}\mathbf{g}_t)\boldsymbol{\epsilon}_t^{\mathbf{p}}\mathbf{v}_t^\top \tag{D.22}$$

It can be seen that it differs from $\Delta \mathbf{B}^{\text{RNN}}$ only in the absence of the unrolling term $\frac{\partial \mathbf{g}_t}{\partial \mathbf{g}_k}$. On the other hand, the update rules $\Delta \mathbf{F}^{\text{RNN}}$ and $\Delta \mathbf{F}^{\text{tPCN}}$ are exactly the same.

D.4 Experimental setups and hyperparameters

Place cell and trajectory parameters We use DoS place cell encodings throughout most of our experiments. Formally, the activity of the ith place cell with this encoding, given a particular location x can be written as:

$$K(x, C, \tau) := \exp\left(-\frac{(x-C)^2}{\tau \xi^2}\right) \tag{D.23}$$

$$p_i = \frac{K(x, C_i, 2)}{\sum_{j=1}^{N_p} K(x, C_j, 2)} - \frac{K(x, C_i, 4)}{\sum_{j=1}^{N_p} K(x, C_j, 4)} \tag{D.24}$$

where C_i is the center of the place cell's firing field, and τ and ξ define the width of the firing field's center and surround. The table below specifies parameters defining the place cells and trajectories:

ξ	**Path length**	**Average agent speed**	**Environment size**
0.12m	10 steps	$\{0.02, 0.05, 0.1\}$m/s	$\{1.4^2, 1.8^2, 2.0^2\}$m^2

Specifically, at time step $t = 0$, a 2D position and a head direction scalar in $[0, 2\pi]$ are randomly initialized. At each of the subsequent time steps, a random turn angle is sampled from a normal distribution and a random speed is sampled from a Rayleigh distribution. Both values are then multiplied by dt mentioned in the main text. If the simulated agent hits a border wall at this time step, its speed is slowed and its turn angle is inverted. The position of the agent is updated according to the speed and turn angle at this time step. The trajectories are simulated using parameters adapted from the code provided in Sorscher et al. (2023).

Hyperparameter	sparse, non-neg. PCN	tPCN	RNN
N_p	512	512	512
N_g	256	{256, 512, 1024, 2048}	2048
h	N/A	{`ReLU`, `tanh`}	`ReLU`
f	N/A	{`softmax`, `tanh`}	`softmax`

Model and training hyperparameters In our experiments, we have used three models: sparsity and non-negativity constrained PCN, RNN and tPCN. The table below specifies parameters of model architectures:

The table below specifies hyperparameters used in training RNN and tPCN. We use `Adam` optimizer for all weight updates, and plain `SGD` for inference dynamics in tPCN. We found that in general, RNNs take more epochs to converge in the path integration task.

Hyperparameter	tPCN	RNN
N_x	50000	50000
batch size	500	500
learning rate	10^{-4}	10^{-4}
inference step size	10^{-2}	N/A
epochs	150	200
inference iters	20	N/A
weight decay	10^{-4}	10^{-4}

The table below specifies hyperparameters used in training the sparse, non-negative PCN. We use `Adam` optimizer for all weight updates, and plain `SGD` for inference dynamics.

Hyperparameter	Value
N_x	900
batch size	100
learning rate	2×10^{-3}
inference step size	10^{-2}
epochs	600
inference iters	20
weight decay	10^{-5}
λ	0.05

Calculation of grid scores The following grid score calculation process is adapted from Sargolini et al. (2006) and the code of Sorscher et al. (2023). It is summarized below for completeness and clarity:

- Get the rate map of latent neurons (potentially hexagonal grid cells);
- Place one copy of the rate map on top of the other, and start moving the top copy by $\delta \in \mathbb{R}^2$. If the rate maps are hexagonal grids, for particular δ's that make the firing peaks overlap, the autocorrelation between the stationary and moved maps will be 1; otherwise, the autocorrelation will be 0. We will then have a hexagonal autocorrelation map if the rate map itself is hexagonal;
- We then rotate the autocorrelation map and compute the correlation between each rotated map and the original map. If the rate maps are hexagonal, the correlation as a function of rotated degrees will be sinusoidal, with 60 and 120 degrees as peaks and 30, 90 and 150 degrees as troughs.
- Grid score is calculated as the minimum difference between the peak and trough correlation, which in theory is a real value in range $[-2, 2]$.

All experiments were performed on a single Tesla V100 GPU.

Bibliography

Akrout, M., Wilson, C., Humphreys, P., Lillicrap, T., and Tweed, D. B. (2019). Deep learning without weight transport. In *Advances in Neural Information Processing Systems*, pages 974–982.

Amit, D. J., Gutfreund, H., and Sompolinsky, H. (1987). Information storage in neural networks with low levels of activity. *Physical Review A*, 35(5):2293.

Amit, Y. (2019). Deep learning with asymmetric connections and hebbian updates. *Frontiers in computational neuroscience*, 13:18.

Arnold, V. I. (1992). *Ordinary differential equations*. Springer Science & Business Media.

Auksztulewicz, R. and Friston, K. (2016). Repetition suppression and its contextual determinants in predictive coding. *cortex*, 80:125–140.

Auksztulewicz, R., Rajendran, V. G., Peng, F., Schnupp, J. W. H., and Harper, N. S. (2023). Omission responses in local field potentials in rat auditory cortex. *BMC biology*, 21(1):130.

Baker, C. L. (1990). Spatial-and temporal-frequency selectivity as a basis for velocity preference in cat striate cortex neurons. *Visual neuroscience*, 4(2):101–113.

Baltieri, M. and Isomura, T. (2021). Kalman filters as the steady-state solution of gradient descent on variational free energy. *arXiv preprint arXiv:2111.10530*.

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., et al. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433.

Barak, O. and Tsodyks, M. (2014). Working models of working memory. *Current opinion in neurobiology*, 25:20–24.

Barlow, H. B. (1961). The coding of sensory messages. *Current problems in animal behavior*.

Barron, H. C., Auksztulewicz, R., and Friston, K. (2020). Prediction and memory: A predictive coding account. *Progress in neurobiology*, 192:101821.

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., and Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370–418.

Beal, M. J. et al. (2003). *Variational algorithms for approximate Bayesian inference*. university of London London.

Beck, J., Ma, W., Latham, P., and Pouget, A. (2007). Probabilistic population codes and the exponential family of distributions. *Progress in brain research*, 165:509–519.

Becker, S. and Hinton, G. E. (1992). Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163.

Behrens, T. E., Muller, T. H., Whittington, J. C., Mark, S., Baram, A. B., Stachenfeld, K. L., and Kurth-Nelson, Z. (2018). What is a cognitive map? organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509.

Behrens, T. E., Sotiropoulos, S. N., and Jbabdi, S. (2014). Mr diffusion tractography. In *Diffusion MRI*, pages 429–451. Elsevier.

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *bioRxiv*, page 738385.

Berkes, P. and Wiskott, L. (2005). Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of vision*, 5(6):9–9.

Bernstein, J. (1902). Untersuchungen zur thermodynamik der bioelektrischen ströme. *Pflügers Archiv European Journal of Physiology*, 92(10):521–562.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, 76:198–211.

Bogacz, R., Brown, M., and Giraud-Carrier, C. (2000). Emergence of movement sensitive neurons' properties by learning a sparse code for natural moving images. *Advances in neural information processing systems*, 13.

Bogacz, R. and Brown, M. W. (2003). Comparison of computational models of familiarity discrimination in the perirhinal cortex. *Hippocampus*, 13(4):494–524.

Botvinick, M. M. and Plaut, D. C. (2006). Short-term memory for serial order: a recurrent neural network model. *Psychological review*, 113(2):201.

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Bredenberg, C., Lyo, B., Simoncelli, E., and Savin, C. (2021). Impression learning: Online representation learning with synaptic plasticity. *Advances in Neural Information Processing Systems*, 34:11717–11729.

Bricken, T. and Pehlevan, C. (2021). Attention approximates sparse distributed memory. *Advances in Neural Information Processing Systems*, 34.

Buckley, C. L., Kim, C. S., McGregor, S., and Seth, A. K. (2017). The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81:55–79.

Burak, Y. and Fiete, I. R. (2009). Accurate path integration in continuous attractor network models of grid cells. *PLoS computational biology*, 5(2):e1000291.

Bush, D., Barry, C., and Burgess, N. (2014). What do grid cells contribute to place cell firing? *Trends in neurosciences*, 37(3):136–145.

Buzsaki, G. (2006). *Rhythms of the Brain*. Oxford University Press.

Buzsáki, G. (2015). Hippocampal sharp wave-ripple: A cognitive biomarker for episodic memory and planning. *Hippocampus*, 25(10):1073–1188.

Buzsaki, G. and Draguhn, A. (2004). Neuronal oscillations in cortical networks. *science*, 304(5679):1926–1929.

Carandini, M. and Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1):51–62.

Carlson, T., Tovar, D. A., Alink, A., and Kriegeskorte, N. (2013). Representational dynamics of object vision: the first 1000 ms. *Journal of vision*, 13(10):1–1.

Chaudhry, H. T., Zavatone-Veth, J. A., Krotov, D., and Pehlevan, C. (2023). Long sequence hopfield memory. *arXiv preprint arXiv:2306.04532*.

Chavlis, S. and Poirazi, P. (2021). Drawing inspiration from biological dendrites to empower artificial neural networks. *Current Opinion in Neurobiology*, 70:1–10.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR.

Chen, Y., Zhang, H., and Sejnowski, T. J. (2022). Hippocampus as a generative circuit for predictive coding of future sequences. *bioRxiv*, pages 2022–05.

Chen, Z. et al. (2003). Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69.

Choi, H., Pasupathy, A., and Shea-Brown, E. (2018). Predictive coding in area v4: dynamic shape discrimination under partial occlusion. *Neural computation*, 30(5):1209–1257.

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

Crannell, C. and Parrish, J. (1957). A comparison of immediate memory span for digits, letters, and words. *The Journal of Psychology*, 44(2):319–327.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337(6203):129–132.

Cueva, C. J. and Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*.

Dabaghian, Y., Mémoli, F., Frank, L., and Carlsson, G. (2012). A topological paradigm for hippocampal spatial map formation using persistent homology.

Dale, H. (1935). Pharmacology and nerve-endings.

Daw, N. D., O'Doherty, J. P., Dayan, P., Seymour, B., and Dolan, R. J. (2006). Cortical substrates for exploratory decisions in humans. *Nature*, 441(7095):876–879.

Dayan, P. and Abbott, L. F. (2005). *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The helmholtz machine. *Neural computation*, 7(5):889–904.

Demircigil, M., Heusel, J., Löwe, M., Upgang, S., and Vermet, F. (2017). On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168:288–299.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Deneve, S., Duhamel, J.-R., and Pouget, A. (2007). Optimal sensorimotor integration in recurrent cortical networks: a neural implementation of kalman filters. *Journal of Neuroscience*, 27(21):5744–5756.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Doeller, C. F., Barry, C., and Burgess, N. (2010). Evidence for grid cells in a human memory network. *Nature*, 463(7281):657–661.

Dordek, Y., Soudry, D., Meir, R., and Derdikman, D. (2016). Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis. *Elife*, 5:e10094.

Dorrell, W., Latham, P. E., Behrens, T. E., and Whittington, J. C. (2022). Actionable neural representations: Grid cells from minimal constraints. *arXiv preprint arXiv:2209.15563*.

Doya, K., Ishii, S., Pouget, A., and Rao, R. P. (2007). *Bayesian brain: Probabilistic approaches to neural coding*. MIT press.

Duong, L. R., Lipshutz, D., Heeger, D. J., Chklovskii, D. B., and Simoncelli, E. P. (2023). Statistical whitening of neural populations with gain-modulating interneurons. *arXiv preprint arXiv:2301.11955*.

Eichenbaum, H. (2013). Memory on time. *Trends in cognitive sciences*, 17(2):81–88.

Eichenbaum, H. (2014). Time cells in the hippocampus: a new dimension for mapping memories. *Nature Reviews Neuroscience*, 15(11):732–744.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., and Rasmussen, D. (2012). A large-scale model of the functioning brain. *science*, 338(6111):1202–1205.

Feynman, R. (1972). *Statistical Mechanics; A Set of Lectures*. Addison-Wesley, Massachusetts.

Friedrich, J., Golkar, S., Farashahi, S., Genkin, A., Sengupta, A., and Chklovskii, D. (2021). Neural optimal feedback control with local learning rules. *Advances in Neural Information Processing Systems*, 34:16358–16370.

Friston, K. (2003). Learning and inference in the brain. *Neural Networks*, 16(9):1325–1352.

Friston, K. (2005). A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836.

Friston, K. (2008). Hierarchical models in the brain. *PLoS computational biology*, 4(11).

Friston, K. and Ao, P. (2012). Free energy, value, and attractors. *Computational and mathematical methods in medicine*, 2012.

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

Friston, K., Stephan, K., Li, B., and Daunizeau, J. (2010). Generalised filtering. *Mathematical Problems in Engineering*, 2010.

Friston, K., Trujillo-Barreto, N., and Daunizeau, J. (2008a). Dem: a variational treatment of dynamic systems. *Neuroimage*, 41(3):849–885.

Friston, K., Trujillo-Barreto, N., and Daunizeau, J. (2008b). Dem: a variational treatment of dynamic systems. *Neuroimage*, 41(3):849–885.

Friston, K. J., Harrison, L., and Penny, W. (2003). Dynamic causal modelling. *Neuroimage*, 19(4):1273–1302.

Fuhs, M. C. and Touretzky, D. S. (2006). A spin glass model of path integration in rat medial entorhinal cortex. *Journal of Neuroscience*, 26(16):4266–4276.

Fyhn, M., Hafting, T., Witter, M. P., Moser, E. I., and Moser, M.-B. (2008). Grid cells in mice. *Hippocampus*, 18(12):1230–1238.

Garrido, M. I., Kilner, J. M., Stephan, K. E., and Friston, K. J. (2009). The mismatch negativity: a review of underlying mechanisms. *Clinical neurophysiology*, 120(3):453–463.

George, T. M., Stachenfeld, K. L., Barry, C., Clopath, C., and Fukai, T. (2024). A generative model of the hippocampal formation trained with theta driven local learning rules. *Advances in Neural Information Processing Systems*, 36.

Ghahramani, Z., Beal, M. J., et al. (2000). *Graphical models and variational methods*. Advanced mean field methods-theory and practice. MIT Press.

Giocomo, L. M., Moser, M.-B., and Moser, E. I. (2011). Computational models of grid cells. *Neuron*, 71(4):589–603.

Gluck, M. A. and Myers, C. E. (1993). Hippocampal mediation of stimulus representation: A computational theory. *Hippocampus*, 3(4):491–516.

Golkar, S., Tesileanu, T., Bahroun, Y., Sengupta, A., and Chklovskii, D. (2022). Constrained predictive coding as a biologically plausible model of the cortical hierarchy. *Advances in Neural Information Processing Systems*, 35:14155–14169.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Gray, J. A. and McNaughton, N. (2003). *The Neuropsychology of Anxiety: An Enquiry Into the Function of the Septo-hippocampal System*. Oxford University Press.

Greedy, W., Zhu, H. W., Pemberton, J., Mellor, J., and Ponte Costa, R. (2022). Single-phase deep learning in cortico-cortical networks. *Advances in neural information processing systems*, 35:24213–24225.

Guerguiev, J., Lillicrap, T. P., and Richards, B. A. (2017). Towards deep learning with segregated dendrites. *Elife*, 6:e22901.

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., and Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806.

Hasselmo, M. E., Giocomo, L. M., and Zilli, E. A. (2007). Grid cell firing may arise from interference of theta frequency membrane potential oscillations in single neurons. *Hippocampus*, 17(12):1252–1271.

Hawkins, J., George, D., and Niemasik, J. (2009). Sequence memory for prediction, inference and behaviour. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1203–1209.

Haynsworth, E. V. (1968). On the schur complement. Technical report, BASEL UNIV (SWITZERLAND) MATHEMATICS INST.

Hebb, D. (1949). *The Organization of Behavior.* Wiley, New York.

Helmholtz, H. v. (1866). Concerning the perceptions in general. *Treatise on physiological optics,*.

Henson, R. N. (1998). Short-term memory for serial order: The start-end model. *Cognitive psychology*, 36(2):73–137.

Hertz, J., Krogh, A., and Palmer, R. G. (2018). *Introduction to the theory of neural computation*. CRC Press.

Hillis, J. M., Watt, S. J., Landy, M. S., and Banks, M. S. (2004). Slant from texture and disparity cues: Optimal cue combination. *Journal of vision*, 4(12):1–1.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500.

Hohwy, J., Roepstorff, A., and Friston, K. (2008). Predictive coding explains binocular rivalry: An epistemological review. *Cognition*, 108(3):687–701.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Howard, M. W. and Kahana, M. J. (2002). A distributed representation of temporal context. *Journal of mathematical psychology*, 46(3):269–299.

Howard-Jones, P. A., Bogacz, R., Yoo, J. H., Leonards, U., and Demetriou, S. (2010). The neural mechanisms of learning from competitors. *Neuroimage*, 53(2):790–799.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106.

Huxter, J. R., Senior, T. J., Allen, K., and Csicsvari, J. (2008). Theta phase–specific codes for two-dimensional position, trajectory and heading in the hippocampus. *Nature neuroscience*, 11(5):587–594.

Hyvärinen, A., Hurri, J., and Väyrynen, J. (2003). Bubbles: a unifying framework for low-level statistical properties of natural image sequences. *JOSA A*, 20(7):1237–1252.

Iatropoulos, G., Brea, J., and Gerstner, W. (2022). Kernel memory networks: A unifying framework for memory modeling. *arXiv preprint arXiv:2208.09416*.

Jacobs, R. A. (1999). Optimal integration of texture and motion cues to depth. *Vision research*, 39(21):3621–3629.

Jazwinski, A. H. (2007). *Stochastic processes and filtering theory*. Courier Corporation.

Jensen, O., Idiart, M., and Lisman, J. E. (1996). Physiologically realistic formation of autoassociative memory in networks with theta/gamma oscillations: role of fast nmda channels. *Learning & Memory*, 3(2-3):243–256.

Jiang, L. P. and Rao, R. P. (2022). Dynamic predictive coding: A new model of hierarchical sequence learning and prediction in the cortex. *bioRxiv*.

Jones, R. and Bühl, E. (1993). Basket-like interneurones in layer ii of the entorhinal cortex exhibit a powerful nmda-mediated synaptic excitation. *Neuroscience letters*, 149(1):35–39.

Jordan, M. I. (1990). Attractor dynamics and parallelism in a connectionist sequential machine. In *Artificial neural networks: concept learning*, pages 112–127.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45.

Kanerva, P. (1988). *Sparse distributed memory*. MIT press.

Katz, B. and Miledi, R. (1965). The measurement of synaptic delay, and the time course of acetylcholine release at the neuromuscular junction. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 161(985):483–495.

Keeler, J. D. (1988). Comparison between kanerva's sdm and hopfield-type neural networks. *Cognitive Science*, 12(3):299–329.

Keller, G. B. and Mrsic-Flogel, T. D. (2018). Predictive processing: a canonical cortical computation. *Neuron*, 100(2):424–435.

Kermani Nejad, K., Anastasiades, P., Hertäg, L., and Costa, R. P. (2024). Self-supervised predictive learning accounts for cortical layer-specificity. *bioRxiv*.

Kessy, A., Lewin, A., and Strimmer, K. (2018). Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314.

Keysers, C., Xiao, D.-K., Földiák, P., and Perrett, D. I. (2001). The speed of sight. *Journal of cognitive neuroscience*, 13(1):90–101.

Kilner, J. M., Friston, K. J., and Frith, C. D. (2007). Predictive coding: an account of the mirror neuron system. *Cognitive processing*, 8(3):159–166.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Knill, D. C. and Pouget, A. (2004). The bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, 27(12):712–719.

Knill, D. C. and Saunders, J. A. (2003). Do humans optimally integrate stereo and texture information for judgments of surface slant? *Vision research*, 43(24):2539–2558.

Kohonen, T. (1972). Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359.

Koutnik, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A clockwork rnn. In *International Conference on Machine Learning*, pages 1863–1871. PMLR.

Krizhevsky, A., Nair, V., and Hinton, G. (2014). The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5).

Kropff, E. and Treves, A. (2008). The emergence of grid cells: Intelligent design or just adaptation? *Hippocampus*, 18(12):1256–1269.

Krotov, D. and Hopfield, J. J. (2016). Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29.

Krupic, J., Bauza, M., Burton, S., Barry, C., and O'Keefe, J. (2015). Grid cell symmetry is shaped by environmental geometry. *Nature*, 518(7538):232–235.

Krupic, J., Burgess, N., and O'Keefe, J. (2012). Neural representations of location composed of spatially periodic bands. *Science*, 337(6096):853–857.

Ku, S.-p., Hargreaves, E. L., Wirth, S., and Suzuki, W. A. (2021). The contributions of entorhinal cortex and hippocampus to error driven learning. *Communications biology*, 4(1):1–12.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Kutschireiter, A., Surace, S. C., Sprekeler, H., and Pfister, J.-P. (2017). Nonlinear bayesian filtering and learning: a neuronal dynamics for perception. *Scientific reports*, 7(1):1–13.

Langston, R. F., Ainge, J. A., Couey, J. J., Canto, C. B., Bjerknes, T. L., Witter, M. P., Moser, E. I., and Moser, M.-B. (2010). Development of the spatial representation system in the rat. *Science*, 328(5985):1576–1580.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

LeCun, Y., Cortes, C., and Burges, C. J. (2010). Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, 7(23):6.

Lee, T. S. and Mumford, D. (2003). Hierarchical bayesian inference in the visual cortex. *JOSA a*, 20(7):1434–1448.

Levenstein, D., Efremov, A., Eyono, R. H., Peyrache, A., and Richards, B. (2024). Sequential predictive learning is a unifying theory for hippocampal representation and replay. *bioRxiv*, pages 2024–04.

Levy, W. B. (1989). A computational approach to hippocampal function. In *Psychology of learning and motivation*, volume 23, pages 243–305. Elsevier.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):1–10.

Lillicrap, T. P. and Santoro, A. (2019). Backpropagation through time and the brain. *Current opinion in neurobiology*, 55:82–89.

Lisman, J. and Redish, A. D. (2009). Prediction, sequences and the hippocampus. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1193–1201.

Lisman, J. E. (1999). Relating hippocampal circuitry to function: recall of memory sequences by reciprocal dentate–ca3 interactions. *Neuron*, 22(2):233–242.

Livingstone, M. S. and Hubel, D. H. (1987). Psychophysical evidence for separate channels for the perception of form, color, movement, and depth. *Journal of Neuroscience*, 7(11):3416–3468.

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

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., and Moser, M.-B. (2006). Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience*, 7(8):663–678.

Mehta, M. R., Quirk, M. C., and Wilson, M. A. (2000). Experience-dependent asymmetric shape of hippocampal receptive fields. *Neuron*, 25(3):707–715.

Mikulasch, F. A., Rudelt, L., Wibral, M., and Priesemann, V. (2022). Dendritic predictive coding: A theory of cortical computation with spiking neurons. *arXiv preprint arXiv:2205.05303*.

Mikulasch, F. A., Rudelt, L., Wibral, M., and Priesemann, V. (2023). Where is the error? hierarchical predictive coding through dendritic error computation. *Trends in Neurosciences*, 46(1):45–59.

Millidge, B., Salvatori, T., Song, Y., Bogacz, R., and Lukasiewicz, T. (2022a). Predictive coding: Towards a future of deep learning beyond backpropagation? *arXiv preprint arXiv:2202.09467*.

Millidge, B., Salvatori, T., Song, Y., Lukasiewicz, T., and Bogacz, R. (2022b). Universal hopfield networks: A general framework for single-shot associative memory models. In *International Conference on Machine Learning*, pages 15561–15583. PMLR.

Millidge, B. and Shillcock, R. (2019). Fixational eye movements: Data augmentation for the brain? *PsyArXiv*.

Millidge, B., Tang, M., Osanlouy, M., Harper, N. S., and Bogacz, R. (2024). Predictive coding networks for temporal prediction. *PLOS Computational Biology*, 20(4):e1011183.

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

Millidge, B., Tschantz, A., Seth, A., and Buckley, C. (2021). Neural kalman filtering. *arXiv preprint arXiv:2102.10021*.

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

Moeller, M., Manohar, S., and Bogacz, R. (2022). Uncertainty–guided learning with scaled prediction errors in the basal ganglia. *PLoS computational biology*, 18(5):e1009816.

Mumford, D. (1992). On the computational architecture of the neocortex. *Biological cybernetics*, 66(3):241–251.

Murray, S. O., Kersten, D., Olshausen, B. A., Schrater, P., and Woods, D. L. (2002). Shape perception reduces activity in human primary visual cortex. *Proceedings of the National Academy of Sciences*, 99(23):15164–15169.

Navajas, J., Hindocha, C., Foda, H., Keramati, M., Latham, P. E., and Bahrami, B. (2017). The idiosyncratic nature of confidence. *Nature human behaviour*, 1(11):810–818.

Navratilova, Z., Godfrey, K. B., and McNaughton, B. L. (2016). Grids from bands, or bands from grids? an examination of the effects of single unit contamination on grid cell firing fields. *Journal of neurophysiology*, 115(2):992–1002.

Neal, R. M. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Nowak, L. G., Azouz, R., Sanchez-Vives, M. V., Gray, C. M., and McCormick, D. A. (2003). Electrophysiological classes of cat primary visual cortical neurons in vivo as revealed by quantitative analyses. *Journal of neurophysiology*, 89(3):1541–1566.

Ocko, S. A., Hardcastle, K., Giocomo, L. M., and Ganguli, S. (2018). Emergent elasticity in the neural code for space. *Proceedings of the National Academy of Sciences*, 115(50):E11798–E11806.

Ogawa, S., Lee, T.-M., Kay, A. R., and Tank, D. W. (1990). Brain magnetic resonance imaging with contrast dependent on blood oxygenation. *proceedings of the National Academy of Sciences*, 87(24):9868–9872.

Ohzawa, I., DeAngelis, G. C., and Freeman, R. D. (1996). Encoding of binocular disparity by simple cells in the cat's visual cortex. *Journal of Neurophysiology*, 75(5):1779–1805.

O'Keefe, J. (1976). Place units in the hippocampus of the freely moving rat. *Experimental neurology*, 51(1):78–109.

O'Keefe, J. and Burgess, N. (2005). Dual phase and rate coding in hippocampal place cells: theoretical significance and relationship to entorhinal grid cells. *Hippocampus*, 15(7):853–866.

O'Keefe, J. and Recce, M. L. (1993). Phase relationship between hippocampal place units and the eeg theta rhythm. *Hippocampus*, 3(3):317–330.

Ólafsdóttir, H. F., Bush, D., and Barry, C. (2018). The role of hippocampal replay in memory and planning. *Current Biology*, 28(1):R37–R50.

Oliviers, G., Bogacz, R., and Meulemans, A. (2024). Learning probability distributions of sensory inputs with monte carlo predictive coding. *PLOS Computational Biology*, 20(10):e1012532.

Olshausen, B. A. (2002). Sparse coding of time-varying natural images. *Journal of Vision*, 2(7):130–130.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609.

Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325.

O'Reilly, R. C. and McClelland, J. L. (1994). Hippocampal conjunctive encoding, storage, and recall: Avoiding a trade-off. *Hippocampus*, 4(6):661–682.

Ororbia, A. and Kifer, D. (2022). The neural coding framework for learning generative models. *Nature communications*, 13(1):1–14.

Ororbia, A., Mali, A., Giles, C. L., and Kifer, D. (2020). Continual learning of recurrent neural networks by locally aligning distributed representations. *IEEE Transactions on Neural Networks and Learning Systems*.

Ouchi, A. and Fujisawa, S. (2024). Predictive grid coding in the medial entorhinal cortex. *Science*, 385(6710):776–784.

Pachitariu, M. and Sahani, M. (2012). Learning visual motion in recurrent neural networks. *Advances in Neural Information Processing Systems*, 25.

Pachitariu, M. and Sahani, M. (2017). Visual motion computation in recurrent neural networks. *bioRxiv*, page 099101.

Palm, R. B. (2012). Prediction as a candidate for learning deep hierarchical models of data. *Technical University of Denmark*, 5.

Parr, T. and Friston, K. J. (2018). The discrete and continuous brain: from decisions to movement—and back again. *Neural computation*, 30(9):2319–2347.

Pavlov, I. P. (1906). The scientific investigation of the psychical faculties or processes in the higher animals. *Science*, 24(620):613–619.

Pehlevan, C. and Chklovskii, D. B. (2019). Neuroscience-inspired online unsupervised learning algorithms: Artificial neural networks. *IEEE Signal Processing Magazine*, 36(6):88–96.

Pettersen, M., Schøyen, V. S., Østby, M. D., Malthe-Sørenssen, A., and Lepperød, M. E. (2024). Self-supervised grid cells without path integration. *bioRxiv*, pages 2024–05.

Pinchetti, L., Qi, C., Lokshyn, O., Olivers, G., Emde, C., Tang, M., M'Charrak, A., Frieder, S., Menzat, B., Bogacz, R., Lukasiewicz, T., and Salvatori, T. (2025). Benchmarking predictive coding networks – made simple.

Pouget, A., Beck, J. M., Ma, W. J., and Latham, P. E. (2013). Probabilistic brains: knowns and unknowns. *Nature neuroscience*, 16(9):1170.

Rainer, G., Asaad, W. F., and Miller, E. K. (1998). Selective representation of relevant information by neurons in the primate prefrontal cortex. *Nature*, 393(6685):577–579.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., et al. (2020). Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*.

Rao, R. (1997). Correlates of attention in a model of dynamic visual recognition. *Advances in neural information processing systems*, 10.

Rao, R. P. (1999). An optimal estimation approach to visual perception and learning. *Vision research*, 39(11):1963–1989.

Rao, R. P. and Ballard, D. H. (1997). Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural computation*, 9(4):721–763.

Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87.

Ribeiro, M. I. (2004). Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, 43(46):3736–3741.

Richards, B. A. and Lillicrap, T. P. (2019). Dendritic solutions to the credit assignment problem. *Current opinion in neurobiology*, 54:28–36.

Rolls, E. T. (2010). A computational theory of episodic memory formation in the hippocampus. *Behavioural brain research*, 215(2):180–196.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345.

Ruck, D. W., Rogers, S. K., Kabrisky, M., Maybeck, P. S., and Oxley, M. E. (1992). Comparative analysis of backpropagation and the extended kalman filter for training multilayer perceptrons. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):686–691.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sacramento, J. a., Ponte Costa, R., Bengio, Y., and Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31, pages 8721–8732. Curran Associates, Inc.

Salvatori, T., Pinchetti, L., Millidge, B., Song, Y., Bogacz, R., and Lukasiewicz, T. (2022a). Learning on arbitrary graph topologies via predictive coding. *arXiv preprint arXiv:2201.13180*.

Salvatori, T., Song, Y., Hong, Y., Sha, L., Frieder, S., Xu, Z., Bogacz, R., and Lukasiewicz, T. (2021). Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, 34.

Salvatori, T., Song, Y., Millidge, B., Xu, Z., Sha, L., Emde, C., Bogacz, R., and Lukasiewicz, T. (2022b). Incremental predictive coding: A parallel and fully automatic learning algorithm. *arXiv preprint arXiv:2212.00720*.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473.

Sargolini, F., Fyhn, M., Hafting, T., McNaughton, B. L., Witter, M. P., Moser, M.-B., and Moser, E. I. (2006). Conjunctive representation of position, direction, and velocity in entorhinal cortex. *Science*, 312(5774):758–762.

Schaeffer, R., Khona, M., and Fiete, I. (2022). No free lunch from deep learning in neuroscience: A case study through models of the entorhinal-hippocampal circuit. *Advances in neural information processing systems*, 35:16052–16067.

Schaeffer, R., Khona, M., Ma, T., Eyzaguirre, C., Koyejo, S., and Fiete, I. (2024). Self-supervised learning of representations for space generates multi-modular grid cells. *Advances in Neural Information Processing Systems*, 36.

Schultz, W. (1998). Predictive reward signal of dopamine neurons. *Journal of neurophysiology*, 80(1):1–27.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599.

Scoville, W. B. and Milner, B. (1957). Loss of recent memory after bilateral hippocampal lesions. *Journal of neurology, neurosurgery, and psychiatry*, 20(1):11.

Sherrington, C. S. (2023). The integrative action of the nervous system. In *Scientific and Medical Knowledge Production, 1796-1918*, pages 217–253. Routledge.

Siddiqui, S. A., Krueger, D., LeCun, Y., and Deny, S. (2023). Blockwise self-supervised learning at scale. *arXiv preprint arXiv:2302.01647*.

Singer, Y., Taylor, L., Willmore, B. D., King, A. J., and Harper, N. S. (2023). Hierarchical temporal prediction captures motion processing along the visual pathway. *Elife*, 12:e52599.

Singer, Y., Teramoto, Y., Willmore, B. D., Schnupp, J. W., King, A. J., and Harper, N. S. (2018). Sensory cortex is optimized for prediction of future input. *elife*, 7:e31557.

Snow, M. A. and Orchard, J. (2022). Biological softmax: Demonstrated in modern hopfield networks. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 44.

Sompolinsky, H. and Kanter, I. (1986). Temporal association in asymmetric neural networks. *Physical review letters*, 57(22):2861.

Song, Y., Lukasiewicz, T., Xu, Z., and Bogacz, R. (2020). Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33:22566–22579.

Song, Y., Millidge, B., Salvatori, T., Lukasiewicz, T., Xu, Z., and Bogacz, R. (2024). Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nature Neuroscience*, 27(2):348–358.

Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Sorscher, B., Mel, G. C., Ocko, S. A., Giocomo, L. M., and Ganguli, S. (2023). A unified theory for the computational and mechanistic origins of grid cells. *Neuron*, 111(1):121–137.

Spratling, M. W. (2017). A review of predictive coding algorithms. *Brain and cognition*, 112:92–97.

Squire, L. R. (1992). Memory and the hippocampus: a synthesis from findings with rats, monkeys, and humans. *Psychological review*, 99(2):195.

Squire, L. R. and Zola-Morgan, S. (1991). The medial temporal lobe memory system. *Science*, 253(5026):1380–1386.

Srinivasan, M. V., Laughlin, S. B., and Dubs, A. (1982). 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.

Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR.

Stachenfeld, K. L., Botvinick, M. M., and Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643–1653.

Stengel, R. F. (1986). *Stochastic optimal control: theory and application*. John Wiley & Sons, Inc.

Stensola, H., Stensola, T., Solstad, T., Frøland, K., Moser, M.-B., and Moser, E. I. (2012). The entorhinal grid map is discretized. *Nature*, 492(7427):72–78.

Tallec, C. and Ollivier, Y. (2018). Can recurrent neural networks warp time? *arXiv preprint arXiv:1804.11188*.

Tang, M., Barron, H., and Bogacz, R. (2024). Sequential memory with temporal predictive coding. *Advances in Neural Information Processing Systems*, 36.

Tang, M., Salvatori, T., Millidge, B., Song, Y., Lukasiewicz, T., and Bogacz, R. (2023). Recurrent predictive coding models for associative memory employing covariance learning. *PLOS Computational Biology*, 19(4):e1010719.

Tang, M., Yang, Y., and Amit, Y. (2022). Biologically plausible training mechanisms for self-supervised learning in deep networks. *Frontiers in computational neuroscience*, 16:789253.

Thorpe, S., Fize, D., and Marlot, C. (1996). Speed of processing in the human visual system. *nature*, 381(6582):520–522.

Thunell, E. and Thorpe, S. J. (2019). Memory for repeated images in rapid-serial-visual-presentation streams of thousands of images. *Psychological science*, 30(7):989–1000.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review*, 55(4):189.

Treves, A. and Rolls, E. T. (1994). Computational analysis of the role of the hippocampus in memory. *Hippocampus*, 4(3):374–391.

Tscshantz, A., Millidge, B., Seth, A. K., and Buckley, C. L. (2023). Hybrid predictive coding: Inferring, fast and slow. *PLoS computational biology*, 19(8):e1011280.

Tulving, E. (1972). Episodic and semantic memory.

Urbanczik, R. and Senn, W. (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528.

van Hateren, J. H. and Ruderman, D. L. (1998). Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1412):2315–2320.

Van Ravenzwaaij, D., Cassey, P., and Brown, S. D. (2018). A simple introduction to markov chain monte–carlo sampling. *Psychonomic bulletin & review*, 25(1):143–154.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vogels, T. P., Sprekeler, H., Zenke, F., Clopath, C., and Gerstner, W. (2011). Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science*, 334(6062):1569–1573.

Wallenstein, G. V., Hasselmo, M. E., and Eichenbaum, H. (1998). The hippocampus as an associator of discontiguous events. *Trends in neurosciences*, 21(8):317–323.

Walsh, K. S., McGovern, D. P., Clark, A., and O'Connell, R. G. (2020). Evaluating the neurophysiological evidence for predictive processing as a model of perception. *Annals of the New York Academy of Sciences*, 1464(1):242.

Watanabe, E., Kitaoka, A., Sakamoto, K., Yasugi, M., and Tanaka, K. (2018). Illusory motion reproduced by deep neural networks trained for prediction. *Frontiers in psychology*, 9:345.

Weber, S. N. and Sprekeler, H. (2018). Learning place cells, grid cells and invariances with excitatory and inhibitory plasticity. *Elife*, 7:e34560.

Weilnhammer, V., Stuke, H., Hesselmann, G., Sterzer, P., and Schmack, K. (2017). A predictive coding account of bistable perception-a model-based fmri study. *PLoS computational biology*, 13(5):e1005536.

Weiss, Y., Simoncelli, E. P., and Adelson, E. H. (2002). Motion illusions as optimal percepts. *Nature neuroscience*, 5(6):598–604.

Welch, G., Bishop, G., et al. (1995). An introduction to the kalman filter.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Whitehead, S. D. and Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83.

Whittington, J. C. and Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262.

Whittington, J. C. and Bogacz, R. (2019). Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250.

Whittington, J. C., Muller, T. H., Mark, S., Chen, G., Barry, C., Burgess, N., and Behrens, T. E. (2020). The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263.

Widloski, J. and Fiete, I. R. (2014). A model of grid cell development through spatial exploration and spike time-dependent plasticity. *Neuron*, 83(2):481–495.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Williams, R. J. and Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501.

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

Wills, T. J., Cacucci, F., Burgess, N., and O'Keefe, J. (2010). Development of the hippocampal cognitive map in preweanling rats. *science*, 328(5985):1573–1576.

Wilson, R. and Finkel, L. (2009). A neural implementation of the kalman filter. In *Advances in neural information processing systems*, pages 2062–2070.

Winter, S. S., Mehlman, M. L., Clark, B. J., and Taube, J. S. (2015). Passive transport disrupts grid signals in the parahippocampal cortex. *Current Biology*, 25(19):2493–2502.

Wirth, S., Avsar, E., Chiu, C. C., Sharma, V., Smith, A. C., Brown, E., and Suzuki, W. A. (2009). Trial outcome and associative learning signals in the monkey hippocampus. *Neuron*, 61(6):930–940.

Witter, M. P. and Moser, E. I. (2006). Spatial representation and the architecture of the entorhinal cortex. *Trends in neurosciences*, 29(12):671–678.

Wolpert, D. M., Doya, K., and Kawato, M. (2003). A unifying computational framework for motor control and social interaction. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):593–602.

Xu, D., Gao, R., Zhang, W.-H., Wei, X.-X., and Wu, Y. N. (2022). Conformal isometry of lie group representation in recurrent network of grid cells. *arXiv preprint arXiv:2210.02684*.

Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T., and Wang, X.-J. (2019). Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306.

Yartsev, M. M., Witter, M. P., and Ulanovsky, N. (2011). Grid cells without theta oscillations in the entorhinal cortex of bats. *Nature*, 479(7371):103–107.

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

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR.

Zilli, E. A. and Hasselmo, M. E. (2010). Coupled noisy spiking neurons as velocity-controlled oscillators in a model of grid cell spatial firing. *Journal of Neuroscience*, 30(41):13850–13860.