Information representation and processing in neuronal networks: from biological to artificial systems and from first to second-order statistics

Sofía A. Lawrie

TESI DOCTORAL UPF / year 2022

THESIS SUPERVISOR
Dr. Matthieu Gilson and Prof. Rubén Moreno-Bote
Department of Information and Communication Technologies

upf. Universitat Pompeu Fabra *Barcelona*

“(...) me dispongo a contar algunos acontecimientos, entremezclados, difusos, que han sido parte de tensiones profundas y contradictorias, de una vida llena de equivocaciones, desprolija, caótica, en una desesperada búsqueda de la verdad.”

Ernesto Sábato, *Antes del fin*

Agradecimientos

Si bien cuatro o cinco años son un plazo de tiempo relativamente corto en la vida de una persona, cuando se realiza un doctorado el tiempo parece por momentos contraerse o dilatarse de una manera extraña y absurda (o será que así es la vida en general, no lo sé). Así que a muchísima gente cabría agradecerle en este espacio, ya sea por su incondicional y sostenido apoyo a lo largo del desarrollo de esta tesis, o por su fugaz pero memorable compañía. A más de uno seguramente dejaré en el camino.

En primer lugar, mi agradecimiento completo es a Rubén y Matthieu. A Rubén por darme la oportunidad de explorar y depositar un grano de confianza en mí desde el principio. A Matthieu por su apoyo y orientación, en especial en la segunda mitad de este trayecto. Sin su guía gran parte de los resultados que aquí se presentan no hubieran sido posibles.

All CBC people also deserve a shout out, from my lovely office mates (Marc Lluís, Alice and Ege) to administrative staff and colleagues from my own and other labs. It has been fun sharing lunches, coffee breaks, seminars, book clubs and the occasional beer after work, sadly interrupted by the pandemic. Special mention goes to Kat, my big metal sister and partner in crime, and Gabriela, my walking companion, both for their emotional support.

Thanks are also dedicated to Megan Roussey, Rogelio Luna and Julio Martínez-Trujillo, at Western University, for data sharing and collaboration. Also to the organizers and classmates of the three wonderful summer schools I had the pleasure of attending during these four years.

En plano más personal, eternas gracias a todos aquellos que han logrado hacer de Barcelona mi casa: la gente que sí del Ático 1, de la Kebab y de mis tres hogares percusivos a lo largo de este tiempo, en especial Ketubara. Profundamente gracias a Nati, Jujuy, Paz, las Flores, Mechi, Cami, Francesca, Caro, Arnau, Nico, Leo, Lean y la Cris.

Por último, porque a menudo dejo lo más importante para el final, gracias a mi familia: mi madre Sandra y mi hermano Juan. Mi madre merece la mención mas grande de todas, por su ejemplo, su visión (no siempre igual a la mía), su esfuerzo constante y, sobretodo, su cariño.

Abstract

Neuronal networks are today hypothesized to the basis for the computing capabilities of biological nervous systems. In the same manner, artificial neuronal systems are intensively exploited for a diversity of industrial and scientific applications. However, how information is represented and processed by these networks remains under debate, meaning that it is not clear which sets of neuronal activity features are useful for computation. In this thesis, I present a set of results that link the first-order statistics of neuronal activity with behavior, in the general context of encoding/decoding to analyse experimental data collected while non human primates performed a working memory task. Subsequently, I go beyond the first-order and show that the second-order statistics of neuronal activity in reservoir computing, a recurrent artificial network model, make up a robust candidate for information representation and transmission for the classification of multivariate inputs.

Keywords: computational neuroscience, machine learning, neuronal representation, statistical features, covariance coding, reservoir computing, supervised learning, classification, multivariate time series, bio-inspired computing, covariance perceptron, working memory

Resumen

Las redes neuronales se presentan hoy, hipotéticamente, como las responsables de las capacidades computacionales de los sistemas nerviosos biológicos. De la misma manera, los sistemas neuronales artificiales son intensamente explotados en una diversidad de aplicaciones industriales y científicas. No obstante, cómo la información es representada y procesada por estas redes está aún sujeto a debate. Es decir, no está claro qué propiedades de la actividad neuronal son útiles para llevar a cabo computaciones. En esta tesis, presento un conjunto de resultados que relaciona el primer orden estadístico de la actividad neuronal con comportamiento, en el contexto general de codificación/decodificación, para analizar datos recolectados mientras primates no humanos realizaban una tarea de memoria de trabajo. Subsecuentemente, voy más allá del primer orden y muestro que las estadísticas de segundo orden en computación de reservorios, un modelo de red neuronal artificial y recurrente, constituyen un candidato robusto para la representación y transmisión de información con el fin de clasificar señales multidimensionales.

Palabras clave: neurociencia computacional, aprendizaje automático, representación neuronal, propiedades estadísticas, codificación por covarianzas, computación de reservorios, aprendizaje supervisado, clasificación, series temporales multidimensionales, computación bio-inspirada, perceptrón de covarianza, memoria de trabajo

Preface

Understanding how biological neuronal systems process information is a key goal in computational neuroscience, given their outstanding performance for diverse input types, their flexibility and their impressive generalization properties. Likewise, building machines that display these properties is a major goal in machine learning (or artificial intelligence in general). This thesis constitutes an effort in both these directions, independently, and is structured as follows.

Chapter 1 provides as introduction, where concepts such as neuronal representations, encoding/decoding, neuronal networks (perceptron and reservoir computing), learning schemes and the problem of time series classification are presented.

Chapter 2 is concerned with the study of biological neuronal representations. Particularly, we aim to elucidate how (mnemonic) stimuli are represented in the neuronal activity of simultaneously recorded populations of non human primates performing a working memory task within a naturalistic virtual reality environment. This dataset was collected and kindly shared by Julio Martínez-Trujillo's lab at the University of Western Ontario (Canada). The introductory section of this chapter contains a review of relevant working memory literature and further states the specific question we address with the proposed experimental set-up: whether neuronal populations in lateral prefrontal cortex sustain mnemonic stimulus representations when dynamic visual activity is underway.

Chapter 3 and Chapter 4 investigate within-trial variability, a ubiquitous trait of biological neuronal systems, as a basis for information representations to perform time series classification of real and synthetic datasets with reservoir computing systems. While the use of covariances as features for classification has been previously studied in other settings, we here do it for the first time for reservoir computing networks. We examine whether these recurrent models of cortex can represent or encode input properties useful for classification in their correlated activity, aiming towards the general question of unifying information representation and learning in biologically inspired systems under a consistent framework. We employ numerical simulations and analytical approaches to expose relationships between input and reservoir statistics and their link with classification performance. In Chapter 3, we follow a conventional machine learning approach that maps static features to class probability vectors with a multinomial logistic regression classifier. We show that the use of covariances boosts performance, reaching high accuracy levels with low reservoir resources (i.e. number of neurons). Later, in Chapter 4, we focus on a biologically more realistic setting that maps time series to time series, by building on previous work on the covariance perceptron. Using a reservoir to filter the input signals, we find that covariance-based classification accuracy stays high (matching or outperfoming a classical mean-based perceptron), while the training cost for the covariance perceptron readout is reduced, since we avoid recurrent connections at the output layer. Our work takes a step further towards theoretically investigating second-order information processing in neuronal networks.

Chapter 5 provides a general discussion of the results presented in previous chapters, points to the limitations and corresponding improvements of the results presented here to underlie future perspectives for neuronal information processing.

Contents

List of Figures

List of Abbreviations

ANN	Artificial neuronal network
ESN	Echo state network
GLM	Generalized linear model
LP	Linear perceptron
LPFC	Lateral prefrontal cortex
LSM	Liquid state machine
MLR	Multinomial logistic regression
ODR	Oculomotor delayed response
P	Perceptual (task)
RNN	Recurrent neuronal network
RC	Reservoir computing
SEM	Standard error of the mean
WM	Working memory

Chapter 1

INTRODUCTION

Biological neuronal networks are assumed to process information in terms of sequences of *spikes*, which are events resulting from the excitation of a neuron beyond a threshold. From an information processing point of view, in the context of perception leading to action, neuronal ensembles extract informational features or properties embedded in input sequences that are relevant for stimulus identification. Downstream, networks emit output spike sequences with embedded information to produce a behavioral response (top Fig 1.1). Associative relationships with varying degree of predictive power between stimulus/action and neuronal activity features fall under the umbrella of *neuronal representations* or *neuronal code*, a topic that has been intensively explored in computational neuroscience in the past decades [1, 2, 3, 4, 5, 6].

Framed from a machine learning perspective, the classification of input features and their transformation are an important aspect of neuronal coding. When features from the input signal are mapped by a neuronal network to features in the output signal in a class dependent fashion (bottom Fig 1.1), decision rules applied to the output features can lead to an input-dependent response. Two important questions that remain, both in biological and in artificial networks, are:

1. How do neurons represent stimulus information?

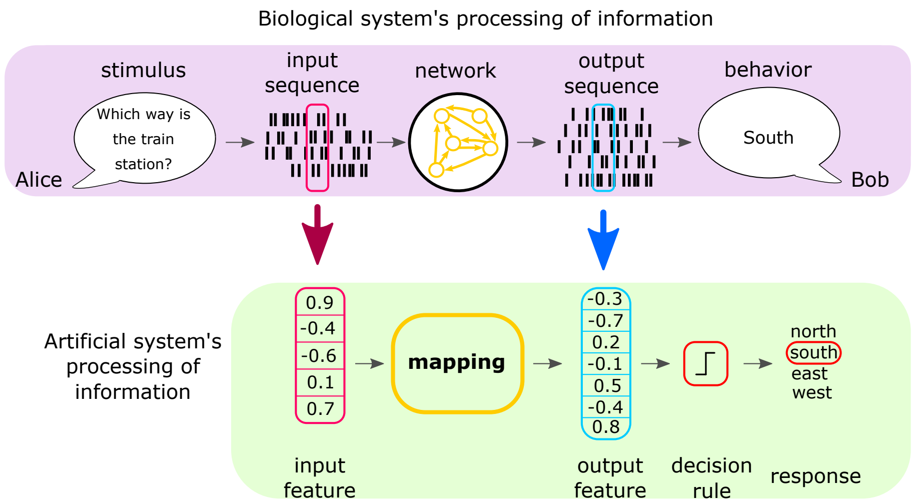


Figure 1.1: Information processing in biological and artificial systems. Top: a stimulus enters the network as an input time series and, after some processing, an output time series is emitted to produce a behavioral response. In biological systems, these time series are spike sequences. Bottom: information processing in artificial systems, from a machine learning point of view. Relevant features of the input time series, computed over a temporal window (colored rectangles in top figure), are mapped by some model or algorithm to output features upon which a decision rule is applied. In this thesis, we study mappings implemented via biological and artificial networks. Adapted from [7].

2. How can networks of neurons efficiently process these representations?

These are the main issues addressed in this thesis, with a strong focus on the temporal evolution of neuronal activity.

This introductory chapter first offers a background on the neuroscientific study of biological neuronal representations, introducing the encoding/decoding framework with an emphasis in populations (Section 1.1) and the role of correlated activity (Section 1.2). Afterwards, the focus is switched to bio-inspired networks (Section 1.3). We discuss learning schemes (supervised and unsupervised), input/output features mappings and network architectures. We then introduce in detail the perceptron (Section 1.4) and reservoir computing (Section 1.5), two network models that are the focus of Chapters 3 and 4. Chapter 2 presents the results of a study on biological neuronal representations in non human primates. Chapter 3 and Chapter 4 constitute the bulk of this work and are dedicated to the exploration of bio-inspired representations in artificial neuronal networks. Chapter 5 provides a general discussion of the results obtained during the development of this thesis, as well as future perspectives. Along this first chapter, we point to the reader how specific sections or concepts are related to subsequent chapters.

1.1 The encoding/decoding framework

From a signal processing (i.e. information processing) point of view, perception-induced behavior can be viewed as an instantiation of an input-process-output model. To fix ideas, let us consider that Alice is asking a question to Bob, as in Fig 1.1. The stimulus s (Alice's voice) is a physical phenomenon, a mechanical wave, that propagates through the air as pressure variations. When reaching Bob's ear, these vibrations produce movement in hair cells located in the cochlea, which is later transduced into electrical activity in a frequency-dependent manner and passed to downstream areas. The full process, then, elicits a cascade of spiking activity n in neuronal populations within auditory brain areas. Thus, from Bob's brain point of view, the electrical activity generated at the sensory organs by Alice's voice has to be processed to identify the words spoken and generate an answer. This constitutes a high-level cognitive example of perception leading to action. Indeed, speech processing is a complex task that can be hierarchically broken down into several sub-processes: speech recognition (Bob recognizes the phonemes that make up Alice's speech as words or sentences with specific associated meaning), information retrieval (Bob recalls where the train station is, a knowledge he had previously collected) and speech production (Bob finally answers the question). In this thesis, we work with simplified instances of the first two sub-processes: word recognition (Chapter 3 and Chapter 4) and information 'retrieval' (short-term information maintenance in Chapter 2).

Traditionally, the study of neuronal representations (particularly in behaving animals) involves experimental set-ups where the stimulus s is of lower complexity than in our previous example (e.g. a single frequency tone instead of the complex sequence of phonemes that make up speech). Researchers can carefully control and manipulate s, while simultaneously recording neuronal activity n. Mathematical models that aim to predict n given s are known as *encoding* models. Those that do the converse and predict stimulus (or response) s from observed neuronal activity n are referred to as *decoding* models [8]. In this context, decoding can be interpreted as inverse encoding [6].

1.1.1 Neuronal variability and probabilistic descriptions

While some experiments performed in vitro or in anaesthetized animals with carefully controlled non-constant stimulation have shown that neuronal spiking activity can be reproducible (even at the 1 milisecond time scale) across trials [9, 10, 11], this finding is not habitual in awake and behaving animals (but see [12] or [13]). In fact, spiking activity is typically variable: the number of spikes and the emission times do not replicate across trials [14, 15, 16, 17, 18, 19]. This occurs for stimuli across several domains, as well as self-initiated behaviors, such as motor commands [20]. The origins of variability are to present day a matter of discussion, although it has been attributed to several different factors. These include the internal state of the animal (arousal, attention), unobserved variables (brain activity in unrecorded areas or fluctuations in cell membrane potential or other metabolic variables), or even wrongly imposed external time reference frames, to name a few [20, 21].

Thus, to overcome variability, encoding and decoding models are better described in probabilistic terms [2, 3, 4, 22]. From the encoding perspective, we characterize the conditional probability of stimulus s in evoking neuronal response n, $P(n|s)$ [8]. Thus, the observer's point of view is taken. From the decoding perspective, we instead infer the probability of s being present in the environment given that we observe the neuronal response n, $P(s|n)$ [8]. Then, 'the brain's' point of view is taken [1, 6]. These two perspectives are related by Bayes' rule:

$$P(n|s)P(s) = P(s|n)P(n); \tag{1.1}$$

where $P(s)$ and $P(n)$ are the marginal probabilities of stimuli and responses respectively, and each side of the equation equals the joint probability distribution $P(n, s)$ [3, 22]. In practice, the modelling of these probabilities can be carried out with different tools, such as statistical Bayesian inference [8] or other machine learning methods.

A comprehensive theory for information processing in the brain thus requires the specification of spike-based representations: what patterns or statistical features in the spiking sequence n of neuronal ensembles are informative for perception and behavior [2, 3, 4, 7].

1.2 Neuronal activity features: averages and correlations

A historically de facto approach in neurophysiology data analysis is to characterize spiking activity by averaging or collapsing spike counts. For each neuron, the averaging procedure can be conducted in different ways. Particularly, we distinguish between averaging only within trials and averaging within and across trials (from here on termed simply across trials averaging) [2, 3, 23]. Both these procedures require that a time resolution d be specified for the within trial averaging (Fig 1.2).

When the averaging is conducted across trials, spiking activity is described by peri/post stimulus time histograms (PSTHs) and tuning curves (top Fig 1.2). PSTHs show the evolution of the mean firing rate within a stereotypical trial and allow to detect increases or decreases in activity that correlate with trial events, such as stimulus onset or offset. Thus, PSTHs display the (averaged across trials) time-resolved trajectory of neuronal activity. Tuning curves display how stimulus-conditioned mean firing rate varies across the whole stimulus set, within a certain trial epoch (as specified by the resolution d) [24, 25]. A classical example is the tuning curve for orientation selectivity in visual cortex: firing rate modulations produced by the orientation of a light bar with respect to a neuron's receptive field [26]. In neuroscience jargon, a neuron is said to encode or be tuned to a certain stimulus when its activity is significantly modulated by stimulus variations. These constitute the earliest descriptions of features (firing rate) of neuronal activity for information representation.

Under the view that the average across trials is the relevant information resulting from the stimulus encoding, any trial-to-trial deviation from the mean response (also termed trial-to-trial variability) is considered to be noise [27]. Nonetheless, neuronal variability has been suggested to have a different role other than corrupting noise. For example, some theories propose it as the fundamental support for computations such as Bayesian inference and fast sampling of probabilities represented in spike trains [28, 29].

A particular form of variability we are interested in is that of the correlated type [17, 30]. Two forms of activity correlations that have been widely studied, especially at the pairwise level, are signal and noise correlations [31, 32, 33]. Signal correlations measure the degree to which neurons fire similarly to any stimulus s in a given set across trials, i.e. correlations due to tuning curve congruences. Noise correlations measure the similarity in the trial-to-trial deviations from the mean stimulus-conditioned responses.

The effect of trial-to-trial correlated activity for information processing is a long debated issue in neuroscience [19]. Several, often early, studies saw correlated activity as detrimental for behavioral performance and robust stimulus decoding [34, 35, 36, 37, 38]. For instance, influential research by Britten and colleagues [34] linking single neuron spike counts to psychophysical performance in a random dot motion detection task numerically showed that correlated activity within neuronal pools degrades stimulus decoding and that there is an upper bound on the information that can be obtained by increasing the pool size. This idea was later expanded in study where experimentally measured low pairwise correlations coefficients (~ 0.12) were shown to significantly diminish the signal-to-noise ratio within simulated neuronal populations [36]. However, a growing body of work has suggested that correlated variability might not always be a hindrance for decoding, with positive, negative or negligible effects typically depending on the noise structure [30, 31, 39, 40, 41, 42, 43, 44, 45].

1.2.1 Generalized linear models for mean encoding/decoding

A common strategy to study the impact of correlated trial-to-trial variability on neuronal representations supported by population activity is to conduct the averaging within trials only. In this context, generalized linear models (GLMs) are useful statistical tools to analyse neuronal data [8, 46, 47] (bottom Fig 1.2). These models allow to probabilistically model neuronal activity on a trial-by-trial basis. From a decoding point of view, GLMs are much exploited in population studies, where the joint single trial activity of neuronal ensembles is used to predict the stimulus that elicited it. For example, for a random dot motion detection task where the stimulus direction can be leftwards or rightwards (binary classification task), a GLM allows to model the conditional probability that the stimulus was rightwards given that we recorded the multivariate (N-dimensional) neuronal activity pattern n on a given trial, $P(s = \text{right}|n)$, as:

$$P(s = \text{right}|n, \omega) = \mathcal{P}(f^{-1}(\sum_{i=1}^{N} \omega_i n_i)), \tag{1.2}$$

where ω is a vector of regressor weights or coefficients, $\mathcal{P}$ is a probability distribution from the exponential family and f is called a link function [46, 47]. Therefore, trial-to-trial correlations within the components of the activity pattern n are taken into account and prediction accuracy is used as a proxy for information content [22].

From an encoding perspective, GLMs allow to model the responses of single neurons and populations considering multiple explanatory factors at the same time, such as high-dimensional stimuli, previous history or interactions between neurons [8]. Therefore, if we encapsulate all possible variables into vector x of dimension M, then we can model the activity of a single observed neuron i as:

$$P(n_i|x,\alpha) = \mathcal{P}(f^{-1}(\sum_{j=1}^{M} \omega_j x_j)). \tag{1.3}$$

In this encoding context, these models thus broaden the concept of tuning curve by refining the quantitative description of the neuronal responses. We remark that, typically, how $\mathcal{P}$ and f are chosen depends on the properties of the dependent variable to model, such as whether it is continuous (as a firing rate) or discrete (as spike-counts).

In Chapter 2, we use GLMs to characterize neuronal activity in lateral prefrontal cortex of macaque monkeys performing a working memory task within a naturalistic virtual reality environment, both from encoding and decoding perspectives.

1.2.2 Within-trial correlations in biological networks

While GLMs for single trial firing rates or collapsed spike-counts are relatively easy to implement, they exploit static snapshots of neuronal activity during the temporal window d: coordinated fluctuations around the within-trial temporal average are ignored (i.e. single trial second-order statistics). However, the idea of exploiting reproducible patterns of synchronized spiking activity within group of cells as a means for information representation and learning goes back to the work of Donald Hebb [48]: coincident firing in connected units, generally in a way such that one takes part in triggering the firing of the others, drives them to become more connected, leading to the formation of neuronal assemblies. Indeed, spike synchronization has been observed even at the fine temporal resolution of 5 ms and in the absence of firing rate modulations in motor cortex neuronal ensembles [49]. Furthermore, such patterns of within-trial correlated activity have been linked to behavioral variables, such as stimulus expectation [49] and perceptual accuracy [50, 51], and to be shaped by learning [52].

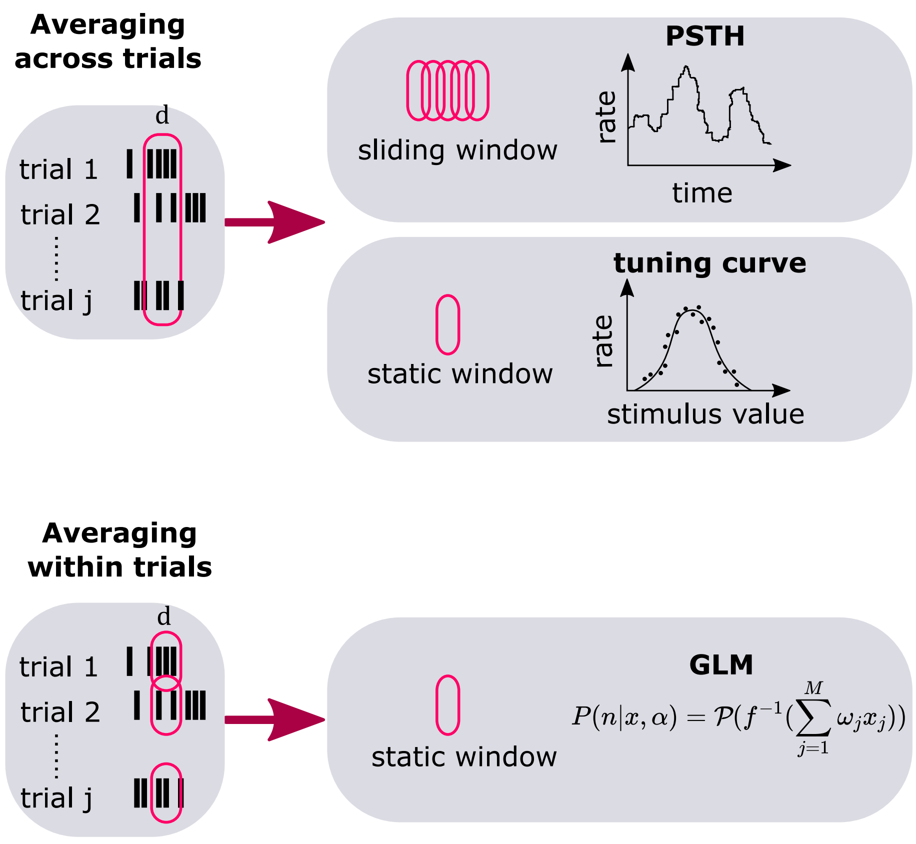


Figure 1.2: Neuronal features obtained by averaging. Top: spiking activity is averaged within trials with a time resolution d and across j trials. Descriptive statistics are given by PSTHs and tuning curves. Bottom: spiking activity is averaged within trials at a time resolution d. In the limit $d \rightarrow 1$ milisecond, activity is described at the spiking level. Analysis is carried out by means of GLMs that probabilistically model the neuronal response features n with a probability distribution $\mathcal{P}$ from the exponential family that depends on a linear combination of relevant experimental variables x_i weighted by ω_i via the link function f^{-1} [8, 47].

Even though further experimental studies are needed to elucidate the possible functions of within-trial correlated variability in biological systems, in Chapter 3 and Chapter 4, we employ within-trial correlations as features for information processing in artificial neural networks, expanding on recent previous work by Gilson and colleagues [53].

1.2.3 Other neuronal coding mechanisms

Thus far, we have only discussed features of the neuronal code based on the spike-count structure of first (rate) or second-order (variability). However, for completeness, we will also mention that other coding mechanisms have been explored in the literature. As a matter of fact, in our previous formulation based on averaging within the temporal window d, in the limit of d approaching 1 milisecond, the neuronal features are reduced to the full sequence of spike-timings, therefore constituting a "time code" [4, 23, 54]. Other possible features for coding include latency and phase-of-firing.

Latency codes have been suggested as faster information transmission channels than pure rate codes. Different ways to define latency have been proposed, but typically they rely on the relative timing between spike events within a neuronal population [55, 56], where spikes that occur earlier after stimulus presentation are more informative than those emitted at later times, as in rank-order coding [57, 58] or time-to-first spike mechanisms [59]. Other definitions of latency based on rate as features have also been proposed [60].

One of the main criticisms against latency as a possible neuronal feature is that it sometimes depends on correctly establishing the time of stimulus onset [59]. Phase-of-firing codes, on the other hand, propose that information is represented in the spike emission times relative to the phase of global network oscillatory behavior, which provides an internal time reference [61, 62], as measured by local field potentials.

Altogether, the variety of proposed codes suggests that population spiking activity patterns should be studied across different spatial and temporal scales to understand how different mechanisms interact with each other and the specific functions they perform [54, 63, 64, 65].

1.3 Bio-inspired network models

For each neuronal representation or coding mechanism, a dedicated processing is necessary, corresponding to specific dynamical properties in a neuronal network. Conversely, the same network may perform distinct operations depending on the information representation applied in inputs. A popular tool both to explore (as in computational neuroscience) and exploit (as in machine learning) neuronal representations and brain-like information processing are artificial neuronal networks (ANNs) [66, 67, 68, 69, 70].

ANNs come in many flavours, from shallow to deep, feedforward to fully recurrent and spiking to analogous. The only defining property is that their constituent elements, much like the brain, be neurons (also called nodes or units). These neurons can be chosen with different levels of abstraction, depending on their purpose. Machine learning approaches usually employ analogue units, which represent firing rates in biological terms. These are analytically more tractable and generally constitute models that are easier to train. Approaches aiming for stronger biological realism typically model the spiking event, as in integrate-and-fire neurons [71]. However, the strong nonlinearity inherent to spiking models makes the learning procedure harder, although recent alternatives that use surrogate gradients have shown improved results when training feedforward and recurrent spiking networks [72, 73, 74, 75].

Regardless of the neuron model used, ANNs function by mapping input features into output features by adjusting the connections (i.e. learning weights) between units in a goal-driven manner. Among the different tasks that can be implemented via networks, we can mention classification, pattern recall, pattern prediction, clustering or dimensionality reduction. The learning procedure aims to account for several molecular mechanisms that govern synaptic plasticity changes in real networks. Overall, for a given task, how the mapping is developed in these bio-inspired networks depends on learning scheme, input/output features and network architecture. We will expand on these points in the following subsections.

1.3.1 Learning: supervised and unsupervised schemes

Two common approaches to learning in bio-inspired networks are supervised and unsupervised learning [3, 47], although intermediate schemes exist (e.g. reinforcement learning, which has been argued to be relevant to model biological phenomena like reward-gated learning [76]).

Supervised learning relies on an explicit group of input features and associated target output features, which constitute a training set [77]. These associations act as a collection of constraints that limit the possible values that the network weights can take and effectively shape the mapping, although an optimal solution might not always exist or be found. Learning is achieved by minimizing an error or *cost function* that measures how different the network output is from the desired target output across the whole training set. Models are typically tested on a holdout portion of the data not used for training (i.e. a test set), in order to assess their capacity to generalize to new or unseen samples. Supervised learning has proven successful at solving tasks that are deemed as cognitively demanding, such as the recognition of real images or sounds. We study this type of learning in Chapter 3 and Chapter 4.

Limitations of supervised learning arise when labelled training data is scarcely available. In such a case, the small training set may lead to overfitting and poor generalization: a well-performing model in the training set that performs badly in the test set.

On the other hand, unsupervised learning techniques do not depend on precise previously established relationships between inputs and outputs. Instead, given some stream of input data, the network must self-organize using activity dependent learning rules established by the designer [78]. Unsupervised schemes have been shown to reproduce basic features of neuronal representations, like the tuning curves for orientation selectivity in visual cortex mentioned in Section 1.2 [79, 80], and are well suited for tasks such as blind source separation, clustering and dimensionality reduction [81]. Hebbian learning (previously mentioned in Section 1.2) constitutes a classical example of unsupervised learning in biological systems: synapses are strengthened or weakened when neurons are repeatedly active or silent simultaneously, giving rise to neuronal assemblies [48]. It was shown theoretically that this type of learning is sensitive to correlations between pre and post-synaptic spiking activity [82, 83, 84] aiming to reproduce experimental findings [85, 86].

In some cases, unsupervised learning can be used as a preprocessing step in a pipeline, to represent inputs in a more convenient space [87, 88, 89, 90], akin to feature detectors in visual brain area V1.

1.3.2 Output features for supervised multivariate time series classification

To expand on how input/output features can be exploited for information representation, transmission and transformation across a network, we will focus on the specific problem of multivariate time series classification: given a stream of multidimensional input data, our goal is to predict to which one of several classes it belongs to. As previously mentioned, encoding/decoding in biological systems can be formalized as a particular example of this general classification set-up. Furthermore, categorization is considered a large underlying component of cognitive behavior and a key ability for the survival of living organisms [91, 92].

First, let us formalize the problem of time series classification, relevant for our studies in Chapter 3 and Chapter 4. Let us consider that a dynamic multivariate input $u(t) \in \mathbb{R}^M$ is fed to an arbitrary network and, after it is processed, an output $y(t) \in \mathbb{R}^K$ is emitted (Fig 1.3). We aim to classify the input trajectories $u(t)$ into 1-out-of-K classes by applying a decision rule on some statistical feature or property of the observed output signal $y(t)$ [7, 53]. This feature is computed over an observation period of length d. Note that while the network operates by mapping input sequences to output sequences, as in biological systems, at the functional level we understand the transformation to occur between features (i.e. statistical properties of the input signal are transformed into statistical properties of the output signal).

A common output feature for time series classification is the mean activity vector (i.e. the temporal average) (top pathway in Fig 1.3). A decision rule can be implemented by choosing the node with highest mean activity, thus embedding information in the first-order statistics of the output time series. In essence, this type of decoding is similar to traditional decoding methods used in neuroscience, where mean firing rates are used as informative features and within-trial temporal fluctuations around the mean are detrimental noise, as in classical tuning curves (see Section 1.2).

However, drawing inspiration from biology where structured temporal fluctuations convey information and correlated activity sustains Hebbian learning (see Section 1.2), it has been shown that other higher-order statistical moments of the output signal are also suitable to set the decision rule, such as covariances (bottom pathway in Fig 1.3) [53]. Pairwise covariances measure the degree to which two neurons fluctuate around their mean activity in a correlated manner, within a single input presentation (i.e. within a single trial, not to be confused with noise correlations that measure correlation on stimulus conditioned averages *across trials*, see Section 1.2). Then, the output node with highest (co)variance can effectively determine input class [7, 53]. This means that the output signal is instead shaped at the second-order. The logic is that if structured correlations support learning, then networks should be able to transmit them for downstream processing and exploit them for information representation.

The first and second-order structure of the network outputs can be imposed without explicitly setting a time-resolved teacher signal during supervised training, but directly constraining output statistics instead [53].

We next describe in further detail the input/output mappings achievable by a minimal network architecture (perceptron) in a supervised learning context, when using means or covariances as output features for classification.

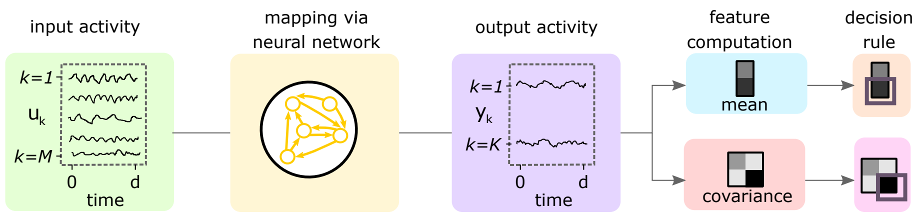


Figure 1.3: Multivariate time series classification based on temporal statistics. An input time series $u(t)$ is fed to a neural network and an output time series $y(t)$ is obtained. To predict the class, the readout must extract information from $y(t)$ dynamics. Here we focus on two alternative possibilities: the information relevant for classification is embedded either in the mean activity of $y(t)$ (top pathway), or in the covariances (bottom pathway). In both cases, a decision rule based on these statistics can predict input class.

1.4 The perceptron

The perceptron is one of the earliest neural network models used to perform a cognitive-like task, classification. It originally consists of a single layer of M input nodes linearly connected to a single layer of K output nodes [47, 93] (Fig 1.4). It was designed to act as a linear classifier of static input patterns [93]. The perceptron network can be naively extended to operate on time series as inputs and outputs. Formally, at each point in time, network output $y(t) \in \mathbb{R}^K$ is a linear combination of current network input $u(t) \in \mathbb{R}^M$:

$$y(t) = \Omega^{\text{out}} u(t), \tag{1.4}$$

where the weights $\Omega^{\text{out}} \in \mathbb{R}^{K \times M}$ are not time dependent. Whenever we refer to a perceptron in this thesis, we refer to the network dynamics given by Eq. 1.4[1]. Additionally, a nonlinearity can be included at the output neurons, which would result in a nonlinear perceptron. However, we stick to the linear version to fix ideas.

[1]Note that we do not refer with 'perceptron' to the specific learning algorithm developed by Rosenblatt [93].

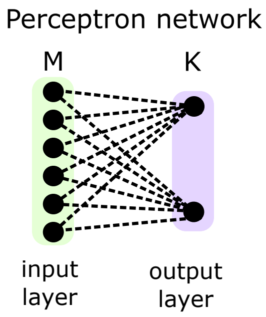


Figure 1.4: Perceptron architecture. M inputs are linearly connected to K outputs by plastic synapses.

1.4.1 Classical perceptron

A perceptron with output signal shaped at the first-order is a classical or mean-based perceptron. The linear nature of the network allows to write cost functions directly in terms of the output statistics to constrain, instead of the time-resolved output trajectories [53]. Thus, if we aim to constrain output mean activity $m_k = \frac{1}{d}\sum_{t=1}^{d} y_k(t)$ for each node k, then the cost function C to optimize (see Section 1.3.1) is:

$$C = \frac{1}{2}\sum_{k=1}^{K}(m_k - \bar{y}_k)^2, \tag{1.5}$$

where $\bar{y}$ is the vector of target mean activity patterns (defined by a 1-of-K coding scheme).

Analytically, the classical perceptron has been shown to have a capacity of $2M$ patterns, where capacity is defined as the maximum number of random input patterns that can be separated in a binary classification problem [94].

1.4.2 Covariance perceptron

When the output signal of a perceptron is shaped at the second-order, the linear network is a covariance perceptron [7, 53]. The mapping is in this case established to match network output covariances Y to target output covariances $\bar{Y}$ in time series by minimizing the cost function:

$$C = \frac{1}{2} \sum_{i_1,i_2=1}^{K} (Y_{i_1,i_2} - \bar{Y}_{i_1,i_2})^2, \tag{1.6}$$

where the difference is between matrix elements (i_1, i_2). Thus, the resulting output time series are structured at the second-order in a class-dependent fashion. An external observer can perform classification by applying the same winner-take-all decision (i.e. an *argmax* operator) as in the classical perceptron, but on output covariance matrices instead of output vectors of mean activity (compare Fig 1.3 pathways). In this thesis, we focus on variance-based decoding: the node with highest variance predicts input class. However, the rule can be based on cross-covariances as well [53].

The covariance approach has been formally shown to provide richer representational spaces (in terms of the number of patterns that can be linearly separated in a binary classification task) when compared to the classical perceptron [7]. Indeed, while a network with K output units is described by a mean activity vector of dimension K, its covariance matrix has dimension $\frac{K(K+1)}{2}$, which results in a larger space to code information.

While we do not distinguish between classical and covariance linear perceptrons in terms of architecture or dynamics, the first introduction of the covariance perceptron included recurrent connections at the output layer. While these connections offer greater versatility for the type of input structures that can be categorized, the computational learning cost to train them is bigger than for the feedforward connections, as they require that a Lyapunov equation be solved at each iteration of the training procedure. Approximations to avoid solving these equations have been suggested, but numerical instabilities remain in practice [53].

Applied to empirical datasets, the covariance perceptron has been proven successful at classifying real time series consisting of images of moving digits [53].

1.4.3 Strengths and limitations of the perceptron

The linear perceptron, both in its classical and covariance versions, has an important downside: due to the linear nature of the network, these models can only sustain consistent same-order feature mappings [53]. In other words, a covariance perceptron can classify input time series only when there is information embedded in its covariance matrix and a mean perceptron can classify input time series only when there is information embedded in its mean vector. Thus, each input/output mapping is in essence constrained by the input structure and information transmission across statistical orders from inputs to outputs is precluded.

The classical perceptron can be nonlinear and capture to some extent the input covariance structure. Nonetheless, this approach yields poor performance when compared to the linear covariance perceptron [53]. Thus, to satisfactorily capture both first and second-order structure in inputs, the previous scheme needs to be extended.

More evolved architectures than a perceptron can be obtained by stacking several nonlinear perceptron-like networks on top of each other: multilayer perceptrons (MLPs) or deep feedforward networks [66]. Although deep networks have been proposed as models for the visual system [95], they are not well suited to deal with sequential inputs. In fact, the most pertinent architecture for sequential inputs is that of recurrent neural networks (RNNs) [67]. These models present lateral or recurrent connections within neurons in the same layer. Therefore, the inputs are not processed in a purely feedforward manner, as in MLPs, and reverberating activity provides an intrinsic medium for memory storage. Nonetheless, supervised RNN training usually relies on unstable algorithms due to the vanishing gradient problem, such as in back-propagation through time [96].

1.5 Reservoir computing

Reservoir computing (RC) makes up a good compromise configuration between purely feedforward and fully recurrent models [97, 98, 99, 100]. In its most simple form, a RC network has an input layer randomly connected to a recurrent layer. This recurrent layer, called reservoir, feeds to an output layer, and the peculiarity is that only weights between these two layers are trained to minimize a cost function (Fig 1.5). This trait is the main advantage of RC networks when compared to fully recurrent models, as training is reduced to that of a single layer model for the read-out and does not require that errors be back-propagated into the recurrent layer and through time. Furthermore, the recurrent reservoir endows the system with a memory of past inputs well suited for time series processing in real time, which feedforward models lack.

Large reservoirs can act as a rich high-dimensional expansion of the inputs, in which linear separability is easier to achieve (for time series classification) and input dynamics are easier to predict or reconstruct (for time series prediction or recall) [97, 99]. The dynamical and structural properties of the reservoir, which are user defined, determine the quality of input representations [99, 101, 102, 103]. Optionally, feedback connections can be included between output and reservoir [99, 104, 105].

The first primitive implementation of RC can be traced back to an early model of language sequence learning in cortical circuits where the network was trained through reinforcement learning [106, 107]. Later (and simultaneously), the idea saw its renaissance with liquid state machines (LSM) [98] and echo state networks (ESN) [97].

Within the machine learning community, RC initially succeeded due to its remarkably good performance/learning cost trade-off in chaotic systems prediction [104]. However, it has been tested in diverse applications, such as biomedical data analysis (brain-machine interfaces, heart rate, fMRI, EEG, EMG), speech recognition, sensor control, cryptography and prediction of stock markets and seismic activity, to name a few (see [100] for a more comprehensive list and further references).

ESNs, which are implemented with analogue units, are the most exploited RC variant in machine learning [99, 101, 108]. However, these rate-based approaches have also been studied in computational neuroscience [105, 109, 110, 111, 112], where RC has been proposed as a model of the cerebral cortex. Indeed, LSMs are RC networks of spiking neurons used to model computation in generic cortical microcircuits

[98, 113, 114]. Typical tasks solved by LSMs include sequence classification and input recall, which were solved in a supervised learning scheme. Biologically more realistic learning rules, such as reward-modulated Hebbian learning [110] have been proposed for rate-based networks.

Experimentally, evidence for reservoir computing in biological networks is lacking, in the sense that while it is able to reproduce neuronal properties observed in a variety of circuits [115, 116, 117], it has not been able to generate falsifiable hypothesis. Furthermore, given the strong computational abilities of a large random reservoir, any network displaying similar properties could potentially constitute a powerful RC network.

A main goal of research in RC is to understand how reservoir structural and dynamical properties interact with inputs to produce high-performing systems. For example, the spectral properties of the reservoir connectivity matrix have been shown to affect how RC dynamics behave in diverse tasks, with special emphasis placed on memory capacity and chaotic time series prediction [97, 102, 103, 108, 118, 119, 120, 121, 122, 123, 124, 125, 126]. In this thesis, we explore the relationship between statistical features in inputs and reservoir units in simple ESNs for multivariate time series classification, when reservoir covariances are used as features for information representation. The next section is dedicated to RC approaches for time series classification.

1.5.1 RC for multivariate time series classification

For multivariate time series classification, several RC models exploring different input representations within the reservoir have been developed. A main distinction can be drawn between approaches mapping input time series into output time series and those mapping input time series into static class probability vectors at the output layer, like in more conventional machine learning applications. In the later, the reservoir is used as a kernel that feeds to a classifier such as multinomial logistic regression, support vector machine or extreme kernel machine [127, 128, 129, 130].

A naive method to implement the classifier is to feed it with the concatenated full sequence of reservoir states (i.e. the time resolved reservoir

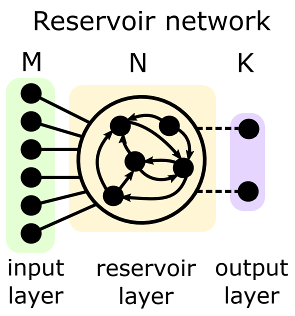


Figure 1.5: Reservoir computing architecture. M inputs are fed to a recurrent layer with N units (the reservoir), which connects to K outputs. Fixed random weights (solid lines) are used for input-to-reservoir and within reservoir connections. Trainable weights (dashed lines) only concern reservoir-to-output connections

trajectory) [127]. Note, however, that for a reservoir of size N, this would imply that the input for the classifier has dimension $d \times N$, instead of N, which would scale poorly (in number of trainable parameters) with input length d. Thus, this approach is better suited for short sequences or in regimes where there is enough data to avoid overfitting. Thus, other methods have been proposed, based on reservoir statistics or other properties. For example, common features for the readout classifier are mean reservoir states (i.e. temporal averages during input time series presentation, analogue to firing rates), or the final reservoir state after input presentation [108, 130, 131]. Other well performing approaches are based on model space representations. These train RC networks to perform input time series prediction and the resulting weights are fed to the classifier. A novel version of this method where the network instead predicts a low dimensional embedding of its own future state has recently been studied in [130].

In Chapter 3, we explore whether reservoir covariances constitute powerful representations when fed as features to a multinomial logistic regression classifier (akin to a nonlinear perceptron operating on static inputs, but with a cross-entropy cost function).

Approaches mapping time series into time series in RC, on the other hand, are interesting because of their higher biological relevance, since spike sequences are transformed into spike sequences in the brain. In this context, information relevant for classification is embedded at the output layer activity by choosing class-dependent target trajectories that arise as combinations of reservoir states at each point in time. Stereotypical trajectories silent all output nodes except the one that indicates the current input class, as in 1-out-of-K coding schemes for K-class classification problems. Thus, a prediction is made at each point in time and a final decision can be implemented by winner-take-all mechanisms at the end of input presentation [108]. For linear readouts and time-independent target trajectories, this is equivalent to mapping reservoir mean trajectories to output mean trajectories, thus embedding information in the first-order of the output time series, as in the classical mean perceptron (Section 1.4.1). In Chapter 4, we explore whether information can be embedded at the second-order in the output time series of RC systems, hence exploiting trial-to-trial variability, to solve multivariate time series classification tasks. We achieve this by implementing a covariance perceptron (see Section 1.4.2) as readout: a linear layer that maps time series to target time series with structured covariances in a supervised fashion [7, 53].

Chapter 2

CHARACTERIZING NEURONAL ACTIVITY IN LATERAL PREFRONTAL CORTEX

This work was presented in poster format at the Neuro-inspired Computation Course (University of Tokyo, Japan, March 2019) and at the Okinawa Computational Neuroscience Course (Okinawa Institute of Science and Technology, Japan, July 2019).

Abstract

Neuronal activity in lateral prefrontal cortex of non human primates is known to be tuned to visuospatial stimulus features, both when these are perceptually available or maintained in working memory. Here, we study perceptual and mnemonic spatial information encoding at the single neuron and population levels in simultaneously recorded units in lateral prefrontal cortex of macaque monkeys performing a delayed-response task. The experiment is developed in a novel virtual reality environment en-

dowed with naturalistic conditions, where ocular behavior is not externally imposed and visual exploration is free. We find that tuning for spatial stimulus coexists with tuning of gaze-behavioral variables in single neuron activity, and that the amount of spatial information, as measured by the accuracy of a linear classifier, remains stable across trial epochs in population codes.

2.1 Introduction

Working memory (WM) –the ability to store and maintain information about an absent stimulus for brief periods of time (usually spanning a few seconds)– is a fundamental higher cognitive function, since it allows to guide behavior by integrating information from recent percepts, not only those available at the time a decision is made [132]. When the information to recall is of spatial nature, a vast bibliography suggests that lateral prefrontal cortex (LPFC) is a core substrate for WM in non human primates [133, 134, 135, 136, 137]. In fact, neurophysiological markers of (spatial and non-spatial) memory maintenance in this area have been found both at the single neuron and at the population levels. These include stimulus selective persistent spiking activity (see [138] for a review) and bursts of oscillatory activity at the gamma ($\sim$ 50-120 Hz) and beta ($\sim$ 20-35 Hz) frequency bands of local field potentials [139, 140]. Furthermore, mnemonic population codes have been shown to have both stable and dynamic components: while some subspaces of the neuronal activity space are steady within an experimental trial, others change through time [141, 142, 143, 144, 145]. This temporal evolution has been linked with coding of additional task contingencies that may occur as the trial unfolds, such as the appearance of distractors [146]. In addition to WM, LPFC has also been linked with the encoding of perceptual gaze information [147, 148, 149, 150, 151] and voluntary (i.e top-down or goal-directed) allocation of attention [152, 153, 154, 155, 156, 157, 158]

Traditional experimental protocols in WM studies employ delayed-response or delayed-match-to-sample tasks where subjects' ocular behav-

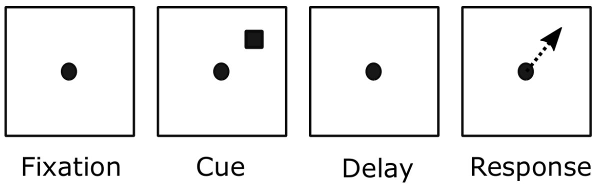


Figure 2.1: Traditional ODR task trial timeline. Subjects start each trial by fixating the central dot. During and after cue presentation (square), subjects maintain fixation and only break it to report their recalled cue location with a saccade at the response period, as indicated by the arrow [134].

ior is highly constrained throughout the trial. For example, in a basic oculomotor delayed response (ODR) task [134], subjects face a monitor screen and start each trial with a fixation period, followed by cue presentation and a delay, where fixation must be held, and subsequent responses are reported by a saccade (Fig. 2.1). These experimental designs prevent the subject from turning to ocular strategies to solve the task (i.e. fixating on cue location throughout the delay period) and therefore guarantee that observed neuronal activity during the delay period is exclusively due to mnemonic processes (or motor suppression). Nonetheless, hypothesis generated in such highly artificial and constrained conditions might not generalize well to more naturalistic settings, where behavior is typically dynamic and less restrained.

Task paradigms that involve WM and free-viewing components, such as cued search tasks [159, 160, 161], have been deployed in the literature, but the accompanying neuronal recordings were performed in brain areas other than LPFC. For instance, McGinty and colleagues studied how value and fixation location jointly influence neuronal activity in orbitofrontal cortex [162]. The question that remains, then, is how task-relevant spatial information is encoded along ongoing visual activity in LPFC activity. To address this, we characterized the information encoded in LPFC neurons of macaque monkeys during a free-viewing spatial WM task performed within a naturalistic virtual reality environment, to describe if and how representations of task relevant features (namely, cue location) can coexist with that of other behavioral variables in this area.

2.2 Results

2.2.1 Animals display target-biased gaze behavior

Two adult rhesus macaques (*Macaca mulatta*) were trained to perform a visuospatial task within a naturalistic virtual reality environment. The environment was presented on a computer monitor and the animals could navigate it using a two-axis joystick (Fig 2.2a). During all experiments, ocular behavior was free and recorded with an eye-tracking system. All eye movements were classified into saccades, fixations or smooth pursuits using a custom toolbox [164] (see Section 2.4.1 for further details).

Subjects begun each trial positioned within a rectangular box. A target stimulus (red volume) was randomly presented in 1 out of 9 possible locations within a circular arena (Fig 2.2b) for 3 seconds. After a 2 seconds delay period, subject had to use the joystick to navigate towards target location (see trial timeline for WM task, Fig 2.2c). A trial was considered correct if location was reached within 10 seconds and incorrect otherwise. Correct trials were rewarded and incorrect ones were not. A variant of this task with the stimulus presented during both the cue and delay epochs was performed by the subjects as well (perceptual (P) task, Fig 2.2c).

Subjects performed a total of 21 sessions of the WM task (13 mon-B) and 21 sessions of the P task (14 mon-B). In a subset of these sessions, neuronal recordings in LPFC area 8Ar were simultaneously performed using two chronically implanted 96-channel microelectrode arrays (5 WM sessions and 2 P sessions).

Both subjects correctly performed the two variants of the task above chance (0.11 for 1/9 locations) in all sessions (binomial test, $p < 10^{-5}$, Fig 2.3a, Section 2.4.2), with performance decreasing with initial distance to target and response time naturally increasing with distance to target (Fig 2.3b-c). On the WM task, subjects also displayed performance decreases when targets appeared on lateral locations with respect to the starting position (Fig 2.3d). This could indicate that either the animals found it hard to maintain lateral mnemonic target representations or, more likely, that redirecting navigation towards a nearly missed target was naturally more difficult when such a target was not visible (Fig 2.3e). However, due to experimental design, determining the source of error in incorrect trials was not straightforward.

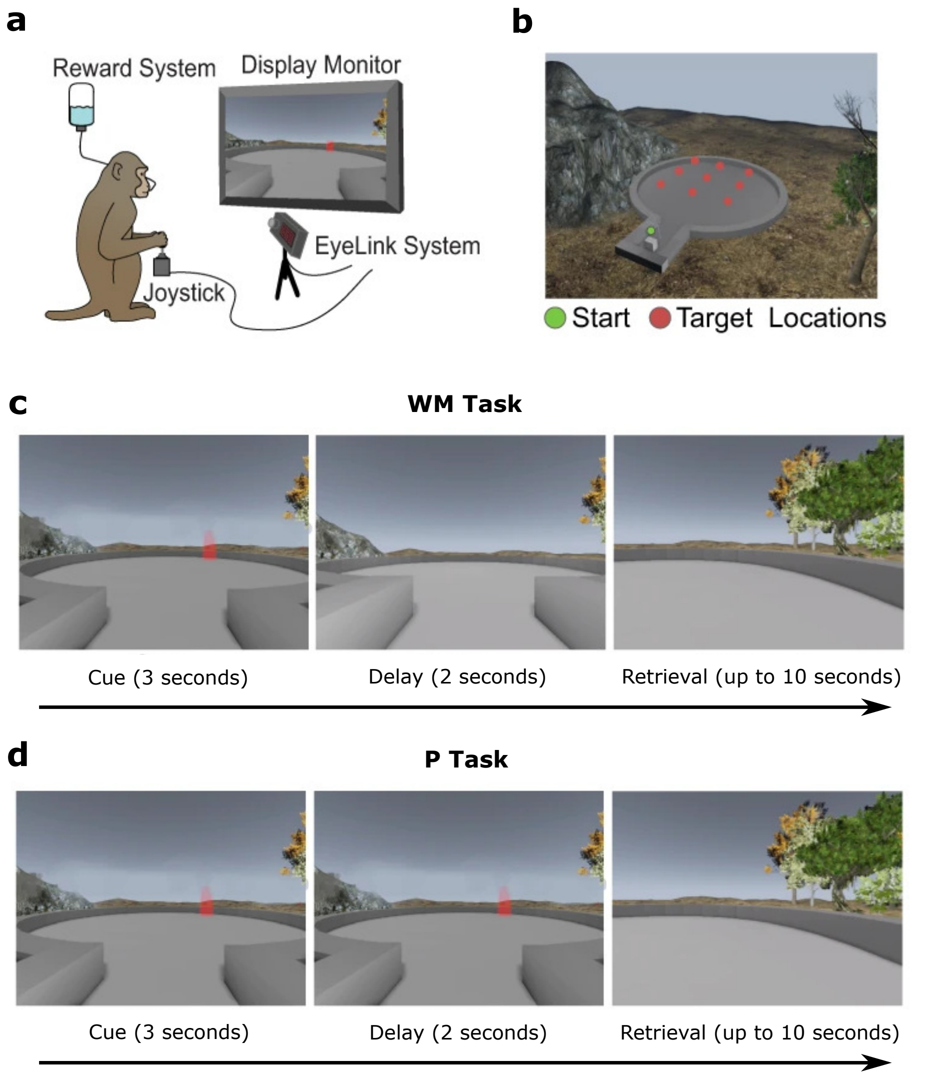


Figure 2.2: Task description. **a.** Experimental set-up. **b.** Monkey start location and target locations (red volumes) within virtual reality environment. **c.** Trial epoch timeline for WM task. **d.** Trial epoch timeline for P task. Figures adapted from [163].

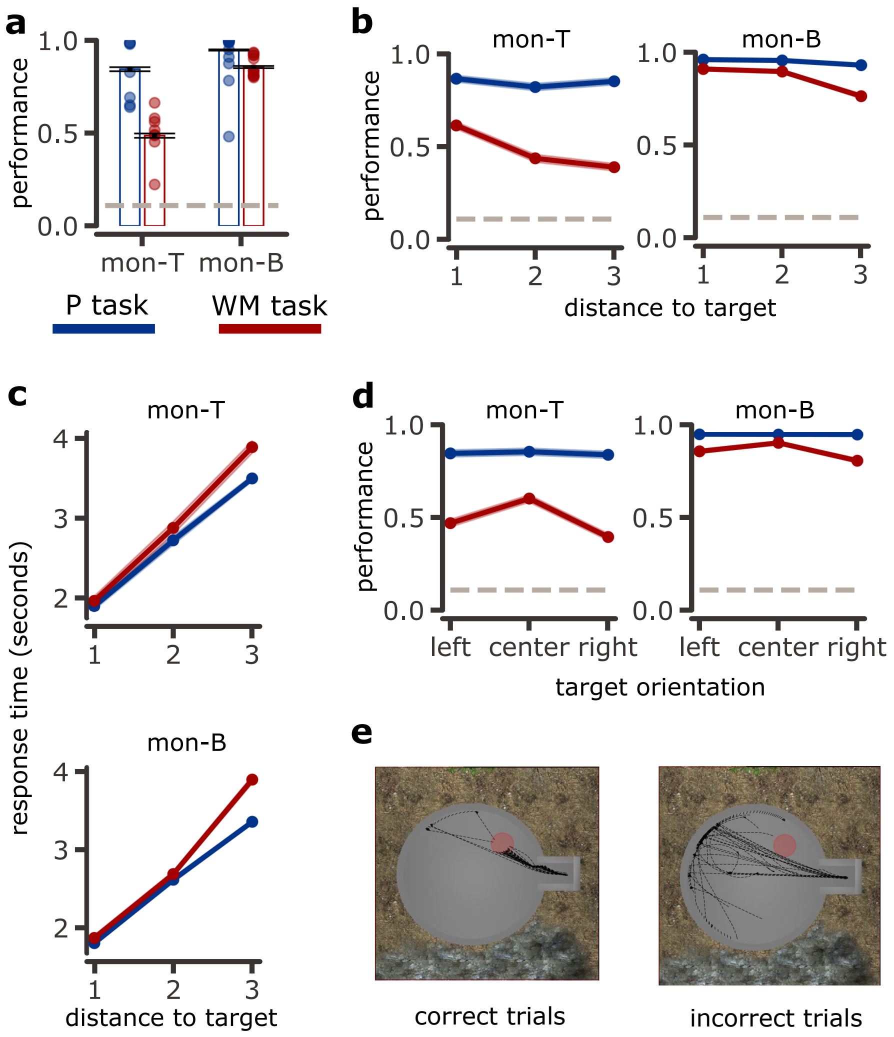


Figure 2.3: Performance for each task variant and subject. **a.** Average performance when pooling all trials across sessions together (bar height), and single session results (circles). Dashed gray lines indicate chance level. **b.** Performance as a function of distance to target (nominal units) from starting location within virtual reality environment. **c.** Response time vs distance to target. **d.** Performance vs target orientation relative to starting position. **e.** Retrieval trajectories for one target during an example WM task session from mon-T.

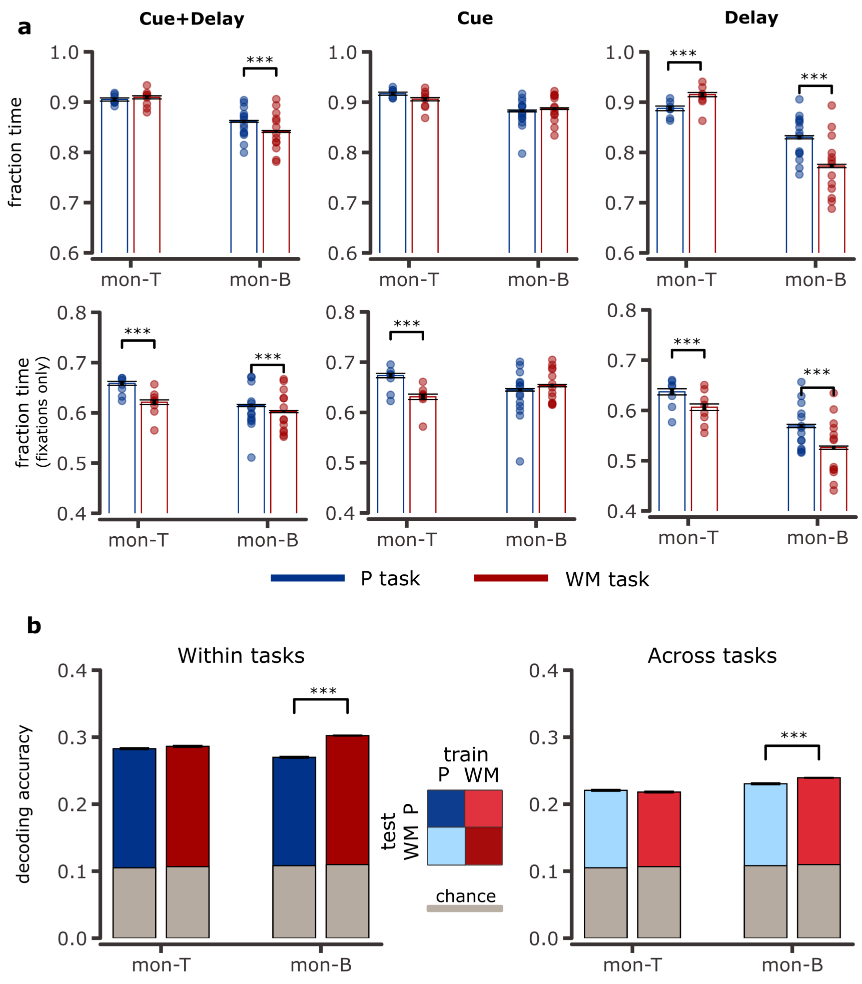


Figure 2.4: Gaze analysis. **a.** Fraction of trial time spent with gaze on screen for each task variant, trial epoch and monkey. Top row shows results when pooling all types of eye movements (fixations, saccades and smooth pursuits). Bottom row shows results for fixations only.

Figure 2.4 *(previous page)*: Bars indicate averages when pooling all correct trials across sessions together, while circles shows single session results. **b.** Decoding accuracy for target identity from gaze behavior on trial-by-trial basis. Results are distinguished when training and testing on data folds from same task variant (left panel) or from different task variants (right panel). Bars indicate averages across sub samples of data to match number of trials across conditions. All error bars are sem (standard error of the mean). $***p < 0.001$, $**p < 0.01$, $*p < 0.05$, two-tailed permutation test.

We explored whether there were naive gaze behavioral differences across subjects or tasks, as measured by the fraction of trial epoch time (cue, delay or both) with gaze within screen boundaries in correct trials (Fig 2.4a, Section 2.4.3). The only consistent observation that we found across subjects was that they fixated more on screen when performing the P task, particularly during the delay epoch (Fig 2.4a, bottom row). To relate within-trial fixation patterns to target identity, we divided the screen with a 4-by-3 grid and used the vector of time fraction fixated per bin to predict target (Fig 2.4b). We found that monkeys displayed target-biased gaze behavior in both tasks, as target identity could be decoded above chance from the time fraction vector. Furthermore, gaze behavior was not completely task specific, since trained decoders in one task variant also displayed above chance performance when tested on the other variant. As an interesting observation, for one subject (mon-B), gaze behavior was more informative of target identity when performing the WM task than the P task, which could indicate the subject was more overtly attentive during the first task.

All in all, we found target-biased gaze behavior during task performance, which needs to be accounted for when relating neuronal activity to task relevant variables.

2.2.2 Single-cell activity is modulated by target

Given the small number of sessions for which simultaneous behavioral and neuronal recordings were available, we pooled neurons across monkeys to reach a total of 587 single cells (437 recorded during WM sessions and 150 during P sessions) and did not discard incorrect trials. Prior to analysis, we discarded units with trial average firing rates below 1 Hz. Thus, we analyzed a total of 475 cells (344 during WM task, 131 during P task). Firstly, we looked for activity modulations in epoch-averaged firing rate with target identity within the three trial epochs (cue, delay, retrieval; Kruskal-Wallis test). We found a significant fraction of neurons tuned to target identity during all task epochs and task variants (Fig 2.5a, above $\sim$ 19 $\%$ for all conditions, one-tailed binomial test $p < 10^{-5}$). We note that there was a decrease in fraction of neurons tuned to target between epochs, with tuned neurons dropping from cue to delay, and rising again for the retrieval epoch. Such a tendency was, however, only significant for the P task ($p < 0.05$ in two-tailed permutation tests for fraction differences).

To preclude the possibility of target-activity modulations captured by our previous analysis arising as a by-product of target-biased gaze patterns, we conducted a second and more restrictive analysis that involved the spiking activity of each neuron during short periods of time where fixations on-screen lasting at least 100 ms took place. During these short time windows, gaze was naturally constrained. We used a Poisson Generalized Linear Model (GLM) to regress spike-counts during fixations to three different categorical variables simultaneously: task epoch (cue or delay), fixated screen region and target orientation (see details in Section 2.4.4 in Methods). To gain more power in our analysis, we pooled targets together by orientation relative to starting position (left, central, right), and only considered screen locations fixated at least 10 times within a session. We found a larger fraction of neurons encoding any of the task or behavioral variables during the WM task than the P task (Fig 2.5b). The source of this difference mostly arises from the larger fraction of neurons encoding task epoch during the WM task, which becomes an irrelevant variable during the P task (no difference in environment between cue and delay epochs). Importantly, when controlling for gaze and epoch, we found neurons tuned for target orientation, which we considered a proxy for target identity. Furthermore, 18 $\%$ of neurons were tuned for target in the WM task, while only 5.8 $\%$ displayed the same tuning during the P task ($p < 0.01$, permutation test for fraction differences). This significant difference is consistent with the results in Fig 2.5a during the delay period. Thus, our results hint that higher tuning for target in the WM task than the P task maybe related to mnemonic processes taking place in LPFC neurons even in this naturalistic setting with unconstrained ocular behavior.

2.2.3 Target information is stable across trial epochs in population codes

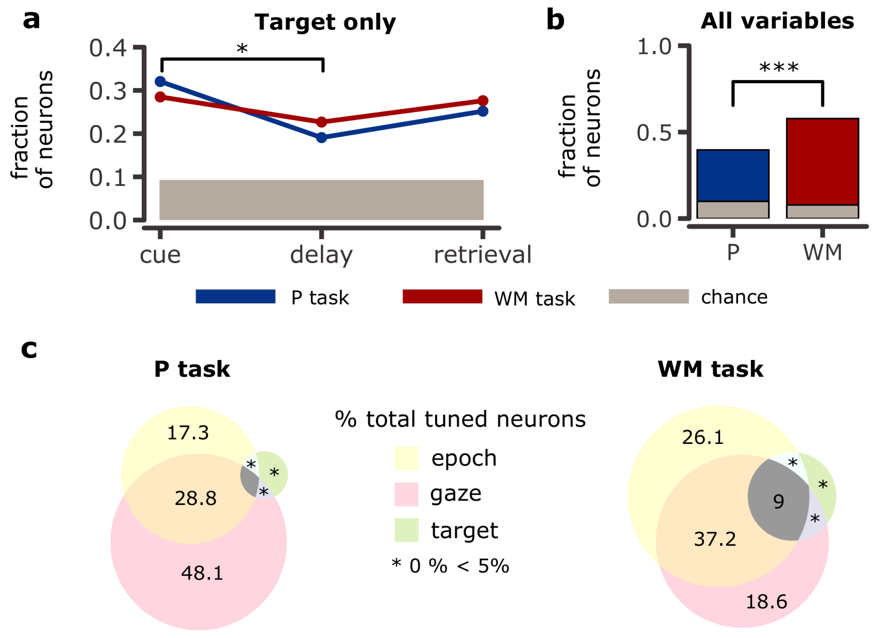


Figure 2.5: Single neuron tuning. **a.** Fraction of neurons tuned to target when averaging activity across whole task epochs for P task (blue) and WM task (red). Gray square shows fraction levels that could have been found by chance ($\alpha = 0.05$ binomial test). $* = p < 0.05$, permutation test for fraction difference between cue and delay in P task. **b.** Fraction of neurons tuned for task epoch, gaze location and/or target orientation when activity is measured during on-screen fixations. $*** = p < 0.001$, permutation test for fraction difference. Chance level in gray ($\alpha = 0.05$ binomial test). **c.** Distribution of percentage of tuned neurons reported in b) for encoded variables (epoch, gaze, target orientation), for the P task (left) and the WM task (right).

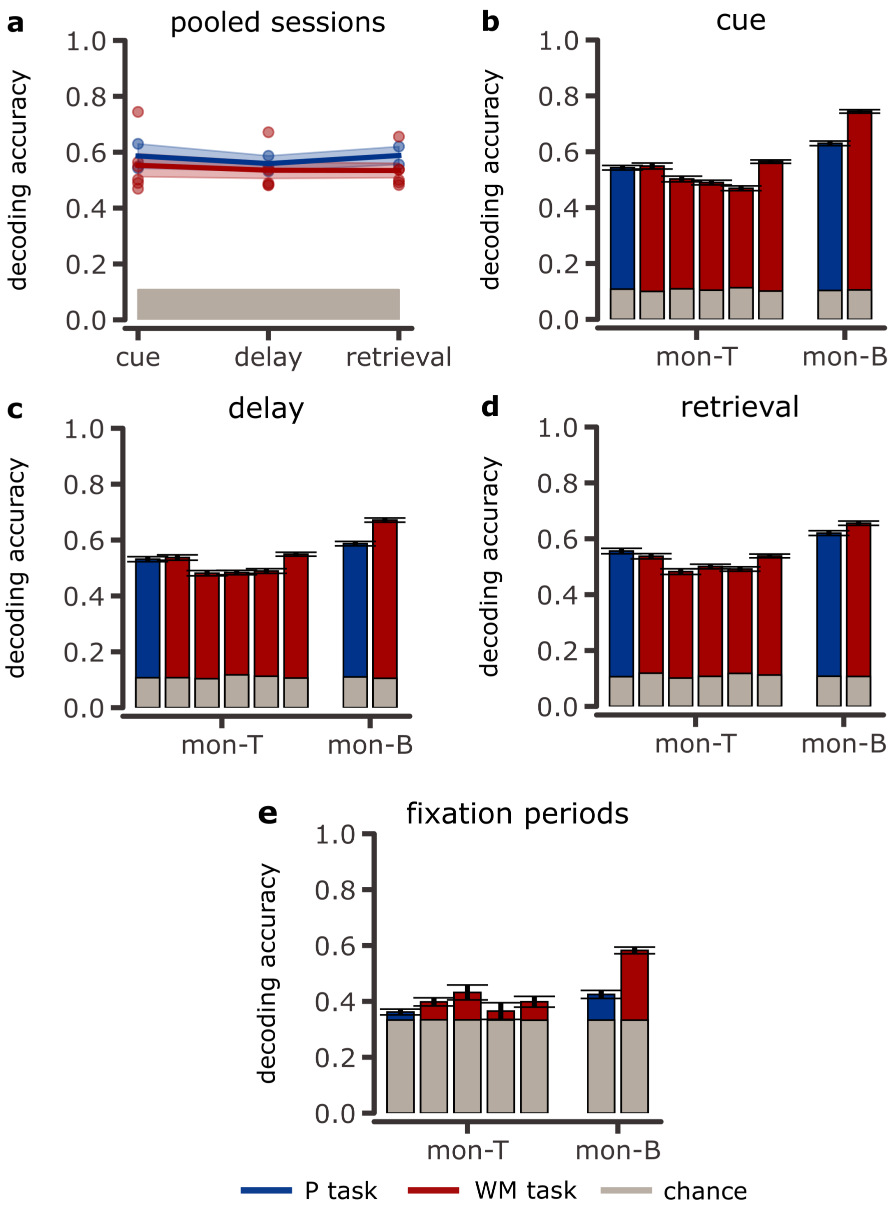


Figure 2.6: Decoding target identity from population activity.

Figure 2.6 *(previous page)*: **a.** Mean decoding accuracy across sessions (solid lines) for three trial epochs. Circles show individual session results and shaded areas are ± 1 sem across all sessions. Gray rectangle is decoding chance level accuracy. **b-d.** Mean decoding accuracy per session (one session per bar), distinguished per task (color) and subject. Error bars are ± 1 sem within session. One figure per trial epoch (**b**: cue, **c**: delay, **d**: retrieval). **e**: Mean decoding accuracy per session when firing rate is only measured during 100 ms of fixation periods. Error bars are ± 1 sem across random data subsamples per session. Note here that chance is at 0.33, since targets are pooled together by orientation. All results in this panel figure are significant with $p < 0.001$ (permutation test).

To study distributed target representations in LPFC, we performed a population level analysis. For each session, we trained logistic regression decoders to predict target identity using population firing-rate vectors as features (see Section 2.4.5 in Methods). Target identity could be decoded above chance in all trial epochs for both tasks, and accuracy was consistent across these two dimensions (epochs/tasks, Fig 2.6a-d).

To control for the effect of gaze behavior on neuronal activity, we conducted a second population analysis. We created datasets where the distribution of fixations was matched across target orientations. Thus, any above chance decoding of target information was guaranteed to arise from target-related activity. Furthermore, for the WM task, we only used fixations occurring during the delay epoch. For this analysis, we discarded one WM session from mon-T, as it failed to have enough number of fixations per screen bin and target orientation (see Methods). Decoding accuracy was significantly above chance in all sessions analyzed (Fig 2.6e), which confirms that neurons in LPFC encode spatial representations in naturalistic settings. We compared the mean accuracy of population codes during the WM task and the P task and do not report a significant difference (permutation test).

2.3 Discussion

Traditional experimental approaches to study the neuronal correlates of WM characterize by imposing heavy constrains both on the subject and the set of stimuli presented, which usually vary along a single dimension [165]. While these efforts have contributed with valuable knowledge about the mechanisms that sustain this cognitive function, it remains unknown whether hypothesis tested on such highly artificial settings generalize well to more naturalistic conditions. In such situations, allocation of attention is not forced and the brain must operate on larger sets of non-stationary sensory attributes, while at the same time performing the cognitive operations that guide behavior.

In this study, we have provided evidence that LPFC neurons encode information about the spatial location of a stimulus in a virtual reality environment, both when the stimulus is perceptually available and when it is not, thus sustaining both perceptual and mnemonic representations [151] in a freely-viewing setting. Target-related information was mostly encoded by neurons displaying mixed selectivity (i.e. tuned for more than one variable) [166], and the higher cognitive demands of the WM task translated into a higher fraction of neurons tuned towards behavioral variables (gaze location) and task features (target location and task epoch). Population codes were informative of target location across tasks, and this information remained stable across all task epochs. Furthermore, even in the regime where spike-counts where estimated in very small time-windows (100 ms) and with few ‘trials’ (fixations), population codes were still qualitatively informative. Our finding that higher tuning (in terms of percentage of neurons) is displayed for variables other than stimulus location, as are gaze position and trial epoch, are in line with previous research that proposes that LPFC participates in top-down control of attention, rather than in mnemonic processes [153, 156, 158, 167]. Indeed, visual attention, visual working memory and gaze control have been linked at behavioral and circuit levels [168]. A possible limitation of our study might be that we underestimated neurons tuned to gaze, since we measured neuronal activity while maintaining fixations, but it has been shown that the tuning is more pronounced for prospective gaze position [150, 168].

While the rich experimental design studied here provided, in principle, many possible alternatives to explore neuronal dynamics in LPFC in an innovative and more ecologically valid setting, the type of analysis that could in practice be performed were very much constrained. The small percentage of correct trials per target condition, especially mon-T in the WM task, degraded the quality of the data. While chance performance level was set at 11% for each target, this is possibly a lower bound. Indeed, the time needed to go from one target to an adjacent one was ~ 0.5 seconds. Thus, in the 10 seconds window that the retrieval epoch was elapsed for, a subject could visit more than one target location. Since only passing by the target location was enough for a trial to be correct, 'random' correct retrievals were likely. Furthermore, by looking at navigation trajectories, it was not straightforward to determine when a trial was random or the subject (again, particularly mon-T) was having trouble directing the joystick. Another possibility for analysis could have been to compare neuronal activity in correct/incorrect trials, to find markers of errors that could be related to mnemonic disruptions. However, this could only have been done for mon-T and not mon-B, whose performance was more consistent and thus lacked enough incorrect trials.

An expanded version of this dataset that studies the effect of ketamine on working memory was fully analyzed in [163]. One of the main results of that study is consistent with the one presented here: LPFC neurons encode spatial stimulus information in naturalistic settings. However, they do not report a significant fraction of neurons tuned towards gaze variables (saccade position), which is at odds with the results here reported.

While modern neuroscience advocates for more complex experimental designs leading to richer datasets that give rise to new hypothesis or clarify old controversies [169], especially in the working memory community [138, 140], researchers must keep in mind the curse of dimensionality: going for larger complexity will probably require larger amounts of trials to ensure that the experimental task space is properly sampled and neuronal activity can be accurately characterized.

2.4 Methods

2.4.1 Behavioral task and recordings

Two adult male rhesus macaques (*Macaca mulatta*, age: 10, 9; weight: 12, 10 kg) were used in this study. Animal care, training and surgical procedures pre-approved by the University of Western Ontario Animal Care Committee. Experimental design, surgeries, data collection and pre-processing were performed by Julio Martínez-Trujillo's lab at University of Western Ontario (Canada). A recent publication from their research group involving an expanded version of this dataset and a more exhaustive description of experimental procedures can be found at [163].

The task environment was developed using Unreal Engine 3 (UDK, May 2012 release; Epic Games). Target locations within the environment were placed in a 3×3 grid and spaced 290 unreal units apart. Movement speed was fixated during navigation, which translates in a $\sim$ 0.5 seconds time-interval between adjacent targets.

The task was displayed on an computer LCD monitor located 80 cm from the subjects' eyes (27" ASUS VG278H monitor, 1024 $\times$768 pixel resolution, 75 Hz refresh rate, 33.5 cm screen height and 45 cm screen width). Experimental rooms were isolated and had no illumination sources other than the monitor. Eye positions were recorded using a video-oculography system with 500 Hz sampling rate (EyeLink 1000, SR Research). Stimulus presentation, reward delivery and ocular and behavioral recordings were controlled by a custom computer program.

Two 10 $\times$ 10 microelectrode Utah arrays (96 channels, Blackrock Microsystems) were chronically implanted in each subject. They were located in the left LPFC (anterior to the arcuate sulces and on either side of the posterior end of the principal sulcus).

Neuronal activity was recorded using a Cerebus Neural Signal Processor (Blackrock Microsystems) via a Cereport adapter. The neural signal was digitized (16-bit) at a sample rate of 30 kHz. Spike waveforms were detected online by thresholding at 3.4 standard deviations of the signal. Resulting spikes were sorted semi-automatically with Plexon Offline

Sorter (Plexon Inc.) into single and multiunits. Only single-unit activity was analyzed in this study.

Behavioral data was collected across 21 sessions of the WM task (13 mon-B, 8 mon-T) and 21 sessions of the P task (14 mon-B, 7 mon-T). In a subset of these sessions, simultaneous neuronal data was collected: 5 sessions of the WM task (1 mon-B, 4 mon-T) and 2 sessions of the P task (1 mon-B, 1 mon-T). The total number of single cells recorded was 587 (150 P, 437 WM). Prior to any analysis, we discarded units with average firing rate below 1 Hz measured within cue and delay epochs, across all trials. Thus, the total number of cells analyzed was 475 (131 P, 344 WM). On average, 73 $\pm$6 single cells were simultaneously recorded in a session, with 270 $\pm$20 (average) number of trials per session.

2.4.2 Task performance analysis

Performance was defined as the percentage of correct trials (targets reached within the 10 seconds period), regardless of target location. Chance level was naively set at 1/9. Pooling results across sessions for each subject, we computed performance as a function of distance from the starting position to target. Thus, targets located within the same row of the 3×3 grid were pooled to the same distance, which was measured in nominal units from 1 to 3. Average response times, defined as the time elapsed between end of delay epoch and end of trial in correct trials, were also computed as a function of distance. Performance was finally computed when distinguishing target orientation relative to starting position (left, central, right).

2.4.3 Gaze analysis

Ocular recordings for the first two epochs of each task (cue and delay) were analyzed separately for each subject. For each correct trial, we computed the fraction of epoch time with gaze on screen, regardless of eye-movement type (smooth pursuits, saccades or fixations) or during fixations only. For each subject and epoch, we tested the null hypothesis that the mean fraction of time with gaze on screen (across trials) was equal across task variants for each epoch with a two-tailed permutation test. We used a permutation test instead of a t-test because the distributions were highly skewed towards the right in the $[0, 1]$ interval. We rejected the null hypothesis at $\alpha = 0.05$, and p-values resulting from the permutation test were Bonferroni corrected for multiple comparisons ($n = 6$). The number of permutations to compute the null hypothesis distribution was set to 10,000.

Target decoding from fixated time per screen location within trials

For each subject and task, all correct trials across sessions were pooled together. For each trial, we collected fixations that complied with the following three requisites: lasted for at least 5 ms, were located within the monitor screen boundaries and took place during the cue or delay epochs.

The screen was binned with a 4×3 grid and we computed the fraction of time spent fixating on each bin by collapsing the previously identified fixation lengths. The vectorized grid was later used as a feature vector to predict target identity on a trial-by-trial basis with a regularized (L2 penalization) multinomial logistic regression model (MLR). The MLR is a generalized linear model for classification, where the conditional probabilities of each class (target) C_k given regressor vector $\vec{x}$ (fraction time per screen region vector, in our case) are given by:

$$P(C_k|\vec{x}) = \frac{\exp(\vec{\omega}_k^\top \vec{x})}{\sum_{i=1}^K \exp(\vec{\omega}_k^\top \vec{x})}, \tag{2.1}$$

for $k = 1, ..., K$ in the K-class problem [47]. The cost function (log-probabilities) is in practice optimized subject to the constrain $\sum_{k=1}^K ||\vec{\omega}_k||^2 = 0$ to regularize the model (L2 regularization [47]).

To assure that differences observed in decoding performance were not due to dataset size imbalances across tasks or targets, for each subject we matched the number of trials across targets and tasks by identifying the minimum number of correct trials per target and randomly subsampling trials without replacement to be the same across all conditions. Thus, we used datasets of 53 trials per target for mon-T (477 total) and 267 trials per target for mon-B (2,403 total). Each of these sub-datasets was randomly split in a 80/20 train-test set with balanced classes. We fitted the multinomial logistic regression model using a ten-fold cross-validation procedure on the train set to find the best regularization parameter. Final model was refitted using the whole batch of training data. Decoding accuracies reported in Fig 2.4b (left) are the mean accuracy scores on the hold-out test set across 1,000 random subsampling iterations. Chance decoding levels were computed for each subsample by randomly shuffling the class labels on the training set and repeating the training procedure. Decoding accuracies reported in Fig 2.4b (right) are the mean accuracy scores on a random subsample of data taken from the complementary task for each trained classifier, with sample sizes matched to the test-set size of the original train-test split.

2.4.4 Generalized linear model for spike-counts

For the GLM analysis (Fig 2.5b-c), for each neuron we fitted the spike-count n during fixations occurring during cue or delay epochs according to:

$$n \sim Poisson\left(f^{-1}\left(\sum_{k=1}^{3}\vec{\omega}_k^{\top}\vec{x}_k + \omega_0\right)\right) \tag{2.2}$$

where the link function f^{-1} was the natural logarithm [47]. The argument of the link function is a sum over the product between each categorical one-hot encoded regressor $\vec{x}_k$ (trial epoch, target orientation, fixated screen region) and its corresponding (transposed) regression weight $\vec{\omega}_k$, plus a bias term ω_0. Since we used a consistent time window to compute spike-counts (100 ms), we did not fit the offset of the model, which is used to account for variations in counts due to different observation windows. For this analysis, we used a 4×3 grid for the screen, but only kept bins that had at least 10 fixations per session. Thus, we discarded 3 bins from one P session and 3 from one WM (both corresponding to mon-B).

To determine whether a neuron's spike-count was significantly modulated by at least one of the previous regressors (Fig 2.5b), we created the distribution of regressors under the null-hypothesis that they had no explanatory effect over the spike-counts by randomly shuffling the spike-count vector and fitting the GLM model for 10,000 iterations. This allowed us to maintain correlation structures within regressors [170]. For a neuron to be considered significantly tuned, it had to fulfill two conditions. First, the deviance of the true GLM model had to be in the top 5% of the distribution of deviances fitted under the null hypothesis. Second, at least one regressor had to be statistically significant. To asses this, we defined the probability that a particular regressor component was not modulating neuron's spike count by the fraction of samples that fell above or below the real regressor value for $\omega_{k,i} > 0$ or $\omega_{k,i} < 0$, respectively, for each regressor component i. Two-tailed p-values for each regressor component and neuron were twice that fraction, multiplied by the dimension of the regressor vector (Bonferroni correction). The reported fraction of neurons was the number of neurons that had the spike-counts significantly modulated by each regressor $\vec{\omega_k}$ over the total number of neurons used in the analysis, where a regressor $\vec{\omega_k}$ is significantly modulating the activity if at least one of its components is doing so ($\alpha = 0.05$). A binomial test was used to asses whether reported fractions were significant, with Bonferroni corrected p-values for $n = 3$ and $\alpha = 0.05$.

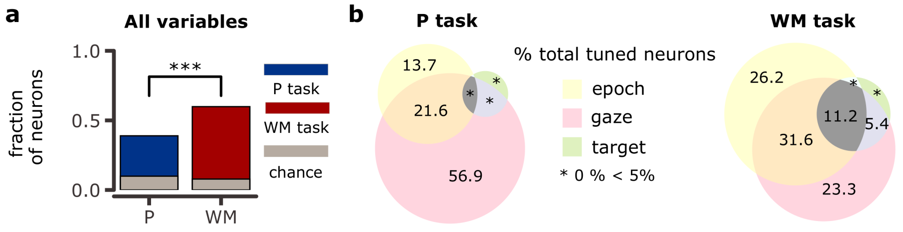


Figure 2.7: Single neuron tuning with linear GLM. **a.** Same as Fig 2.5b. **b.** Same as Fig 2.5c

.

These results were qualitatively reproduced (Fig 2.7) when using a linear model instead of a Poisson model to fit the spike counts:

$$n \sim Normal\left(f^{-1}\left(\sum_{k=1}^{3} \vec{\omega}_k^\top \vec{x}_k + \omega_0\right), \sigma^2\right) \tag{2.3}$$

where the link function f^{-1} was the identity, and the distribution was normal or gaussian instead of Poisson (Fig 2.7) [47].

2.4.5 Population level analysis of neuronal activity

For each epoch within a trial (3 seconds for cue, 2 seconds for delay, first second of retrieval), we computed a vector of neuron firing-rates (each entry corresponding to a neuron). This set of vector features was used to predict target identity (9 locations) using a multinomial logisitic regression model with L2 regularization (see Methods 2.4.3) . For each session, we identified the target with less available trials and we matched the number of trials across target locations by subsampling to avoid data-biased models. The reported test accuracies in Fig2.6b-d are the mean ($\pm$ sem) across 1,000 random subsamples for each session, split in a 80/20 train-test split. For each subsample, we estimated chance performance by randomly shuffling target locations across trials 1,000 times and fitting the MLR model. Results pooled across sessions and subjects are reported in Fig 2.6a.

To control for the effect of gaze behavior in the decoding of target from neuronal activity, we went through each trial and computed the mean firing rate of each neuron during screen fixations. We only used fixations during the delay epoch in the WM task, and both from cue and delay in the P task. Thus, we considered that each fixation was in this case a 'trial', and the goal was to predict target information from the vector of firing rates during 100 ms of fixations. Since we aimed for fixations to be matched (in number) across targets and screen locations, we pooled targets together by orientation (left, central, right). Any screen location with less than 10 fixations per target orientation was removed from analysis, which led to one session from mon-T (WM task) not being analyzed due to lack of data. As in our first analysis, fixation random subsampling was performed 1,000 times and we used a 80/20 train-test split. Since datasets were very small, we tuned the regularization parameter through 4-fold cross-validation in the training set. Final model was fitted using the whole training data. Mean ($\pm$ sem) test accuracies per session are reported in Fig 2.6e. Chance decoding accuracies were computed for each data subsample by randomly shuffling trial orientations in the training split and fitting the model 1,000 times.

2.4.6 Code availability

All analyses were performed in Python, using custom code based on standard libraries. Code is available upon request.

Chapter 3

COVARIANCE FEATURES IMPROVE LOW-RESOURCE RESERVOIR COMPUTING PERFORMANCE IN MULTIVARIATE TIME SERIES CLASSIFICATION

This work was presented at the 5th International Conference on Computational Vision and Bio Inspired Computing (November 25th, 2021). Conference Proceedings to be published on Springer - Advances in Intelligent Systems and Computing Series. Series Ed.: Kacprzyk, Janusz ISSN: 2194-5357.

Abstract

Biological systems exhibit tremendous performance and flexibility in learning for a broad diversity of inputs, which are in general time series. Inspired by their biological counterpart, artificial neural networks used in machine learning for classification aim to extract activity patterns within input signals to transform them into stereotypical output patterns that represent categories. For the vast majority, they rely on fixed target values in output to represent probabilities or implement winner-take-all decisions, which corresponds in the case of time series to first-order statistics. In other words, the basis for such classification of time series is the transformation of input high-order statistics into output first-order statistics. However, the transformation of input statistics to second-or-higher order statistics has not been much explored yet. Here we consider a computational scheme based on a reservoir that maps information engrained in input multivariate time series statistics to second-order statistics of its own activity, before being fed to a usual classifier (logistic regression). We compare this covariance decoding with the "classical" mean decoding applied to the reservoir for classification with both synthetic and real datasets of multivariate time series. We show that covariance decoding can extract a broader diversity of second-order statistics from the input signals, yielding higher performance with smaller resources (i.e. reservoir size). Our results pave the way for the characterization of elaborate input-output mappings between statistical orders to efficiently represent and process input signals with complex spatio-temporal structures.

3.1 Introduction

Multivariate time series classification traverses diverse scientific areas and industrial applications, such as medical diagnosis, ambient assisted living and finance. Reservoir computing systems have become usual tools for the classification of time series, in addition to their most known application to time series forecasting. A reservoir computing model consists of an untrained recurrent layer that processes the inputs by projecting them into a higher-dimensional space, to be then fed to a trainable readout layer (Fig 3.1a) [97, 98, 99]. For time series forecasting, the most common approach is to map the current input activity to a target output activity (future input states) at the readout layer [97, 99]. This type of input-output mapping has also been employed for time series classification by using units in the readout layer that each represents a class and choosing stereotypical target output activities [99, 108]. For example, when the target configuration at each time step corresponds to all outputs being silent except the one indicating the desired class, a winner-take-all decision applied to the output layer over the sequence presentation (thereby pooling its activity over time) can perform the classification. Equivalently, one can consider the mean activity of the reservoir over the input stimulation taken as a state that is associated to the (static) class label. Thus, the reservoir is used as a temporal kernel and its time averaged (mean) state constitutes an informational feature relevant to classify the input time series (Fig 3.1b).

Here we go beyond the use of first-order statistics to define the reservoir activity states and consider the covariances (i.e. second-order statistical moments) of the reservoir activity as features for classification. Although it has previously been shown that (co)variances convey relevant information for multivariate time series classification [53, 171, 172], this has not been fully formalized nor exploited in a reservoir computing framework. Our approach combines the high-dimensional nonlinear expansion operated on inputs by the reservoir with a decoding of its spatial (zero-lag) covariances (Fig 3.1b). We benchmark this classification scheme with both synthetic and real datasets. Firstly, we use multivariate time series generated with controlled zero-lag or one-lag covariances that are the relevant information for the classification in two groups. We show that reservoir computing is beneficial to covariance decoding and that it allows for capturing not only the spatial structure of input time series, but also their temporal structure (which the classical mean-based approach completely misses). Importantly, we assess the influence of the reservoir parameters (Fig 3.1c) on the input-output mapping for the first and second-order statistical moments. We then demonstrate that our method is efficient for practical applications with two real datasets in a task of spoken digits recognition. In particular, it outperforms mean decoding, even with much smaller reservoirs.

After introducing the reservoir model and readout training (Section 3.2),

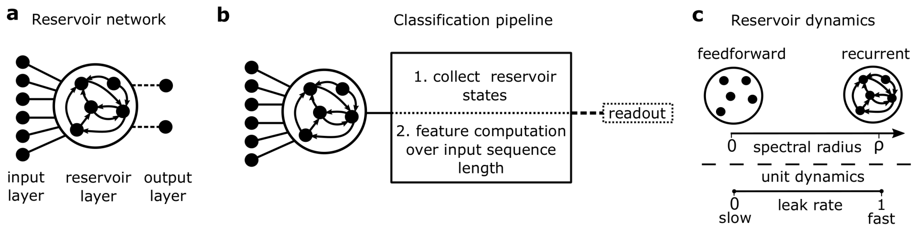


Figure 3.1: Reservoir computing model and classification pipeline. **a.** Reservoir computing network model where only the dashed connections to the readouts are trained. **b.** Classification pipeline where an input sequence is filtered (or processed) by the reservoir, whose activity is measured via statistical features (means or covariances) computed from the reservoir time-resolved states and then fed to a decoder. Only the dashed connections from reservoir features to decoder are trained to predict the labels for the classes. **c.** Reservoir parameters governing its dynamical regime: spectral radius (global parameter) and leak rate (local parameter).

we present the synthetic and real datasets to benchmark the mean and covariance decoding of the reservoir activity. Section 3.3 highlights the advantages of our covariance-based scheme, which is further discussed in Section 3.4.

3.2 Methods

3.2.1 Reservoir implementation

We implement the reservoir computing system with an echo state network, where the reservoir is composed of N leaky integrator neurons [99, 108]. When driven at time t by a multivariate time-dependent input $u^{\mathrm{t}} \in \mathbb{R}^{\mathrm{M}}$, the multivariate activity state $x^{\mathrm{t}}_{\mathrm{res}} \in \mathbb{R}^{\mathrm{N}}$ of the N neurons (or units) in the reservoir changes according to:

$$\bar{x}_{\mathrm{res}}^{\mathrm{t}} = \mathcal{F}(\Omega_{\mathrm{in}} u^{\mathrm{t}} + \Omega_{\mathrm{res}} x_{\mathrm{res}}^{\mathrm{t}-1}), \tag{3.1}$$

$$x_{\mathrm{res}}^{\mathrm{t}} = (1-\alpha) x_{\mathrm{res}}^{\mathrm{t}-1} + \alpha \bar{x}_{\mathrm{res}}^{\mathrm{t}}, \tag{3.2}$$

where the function $\mathcal{F} = \tanh$ is the hyperbolic tangent function applied to each vector component and α is a leak rate. The matrices $\Omega_{\mathrm{in}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ and $\Omega_{\mathrm{res}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ determine the input and recurrent connectivities, respectively. In practice, both connectivity matrices have elements randomly drawn from the uniform distribution over the interval $[-0.5, 0.5)$.

To characterize the reservoir, we consider two important parameters. First, we calculate its spectral radius $\rho\left(\Omega_{\mathrm{res}}\right)$, which is the largest absolute eigenvalue of the recurrent connectivity matrix Ω_{res}. This parameter has been shown to influence the reservoir performance for memory-based tasks commonly studied in the literature [97, 173, 174]. Note that a spectral radius equal to 0 corresponds to $\Omega_{\mathrm{res}} = 0$, meaning that the reservoir is in fact a feedforward layer (left-hand side in Fig 3.1c). To adjust the spectral radius, we uniformly rescale the weights in Ω_{res}, exploring values ranging from 0 to 1.8. Second, we vary the leak rate α in Eq 3.2 that controls how each neuron's dynamics is affected by its immediate past activity state [99, 108]. Smaller α values produce slower dynamics by smoothing the unit activity, thereby integrating the inputs over time. In contrast, no such integration is performed when $\alpha = 1$, corresponding to fully leaky neurons.

As illustrated in Fig 3.1c, we explore the joint influence of the two aforementioned parameters, which govern the reservoir dynamical regime, on the classification performance using numerical simulation. As an example from a previous study [99], if $\alpha = 1$, a spectral radius close to 1 is required for tasks that need long memory depth. In contrast, a feedforward reservoir with $\alpha = 1$ simply transforms the inputs by a pointwise nonlinearity determined by $\mathcal{F}$ on the inputs randomly mixed by Ω_{in}, which does not implement any memory via temporal integration.

Note that all reservoirs in our simulations start with a zero activity state when transforming a short-sequence mode, so initial transient states are present and used in the classification task.

3.2.2 Features and logistic regression as readout

We use logistic regression as a readout or decoder to map features, which are computed from the reservoir activity, to the class labels [47]. We distinguish between mean and covariance decoding, according to which feature is fed to the classifier (Fig 3.1b).

For the mean decoding, the output $y_{\mathrm{out}} \in \mathbb{R}^{\mathrm{K}}$ that predicts the class labels corresponds to

$$\begin{aligned} y_{\mathrm{out}} &= \phi(\Omega_{\mathrm{out}} V), & (3.3) \\ V_i &= \langle x_i^{\mathrm{t}} \rangle =: \frac{1}{d} \sum_{1 \le t \le d} x_i^{\mathrm{t}}. & (3.4) \end{aligned}$$

The vector $V \in \mathbb{R}^{\mathrm{N}}$ is the mean of the reservoir activity over a period of duration d, which corresponds to the input stimulation duration. $\Omega_{\mathrm{out}} \in \mathbb{R}^{\mathrm{K}\times\mathrm{N}}$ is the matrix of weights that are trained and ϕ denotes the softmax function (multivariate according to the number of classes).

For the covariance decoding, we simply replace the mean vector V by the covariance Q:

$$\begin{aligned} y_{\mathrm{out}} &= \phi(\Omega_{\mathrm{out}} Q), & (3.5) \\ Q_{ij} &= \langle \tilde{x}_i^{\mathrm{t}} \tilde{x}_j^{\mathrm{t}} \rangle. & (3.6) \end{aligned}$$

Here the zero-mean time series $\tilde{x}^{\mathrm{t}}$ to compute the covariances comes from the centering of x^{t}. Then, the covariances are vectorized, keeping only independent entries (upper triangle including the diagonal made of variances), meaning that $\Omega_{\mathrm{out}} \in \mathbb{R}^{\mathrm{K}\times\mathrm{N}(\mathrm{N}+1)/2}$.

The training of the readout is performed using the standard algorithms available in the Python library ‘scikit-learn’ [175]. We minimize a L2-regularized cross-entropy loss function, with regularization parameter adjusted using a grid search in a 5-fold cross-validation scheme performed on the training set. We use standard scaling of each feature as a preprocessing step in all implementations, and the logistic regression involves a unit bias with trained weight in all implementations.

3.2.3 Synthetic datasets

For a binary classification task ($K = 2$), we consider two groups of multivariate time series, each being characterized by specific second-order statistics. We constrain either the spatial structure (zero-lag covariances) or the temporal structure (one-lag covariances) of these time series. The point here is to compare how the mean and covariance readouts can capture the relevant information for classification from the reservoir activity. Note that, for a linear reservoir, mean decoding cannot extract second-order statistics, implying that the nonlinearity is necessary for mean decoding to work here. A more thorough comparison would involve a mean structure encoded in the inputs to compare mean and covariance decoding, but this is left for future work.

In practice, we use dynamic systems to generate input time series for $M = 10$ inputs in discrete time $1 \leq t \leq d$, with duration $d = 20$. Denoting the input activity by $u^{\mathrm{t}} \in \mathbb{R}^{\mathrm{M}}$, we consider their covariances without and with lag to quantify their spatio-temporal structure at the second statistical order:

$$\begin{aligned} P^0_{kl} &= \left\langle u^{\mathrm{t}}_k u^{\mathrm{t}}_l \right\rangle_d - \left\langle u^{\mathrm{t}}_k \right\rangle_d \left\langle u^{\mathrm{t}}_l \right\rangle_d, & (3.7) \\ P^1_{kl} &= \left\langle u^{\mathrm{t}}_k u^{\mathrm{t+1}}_l \right\rangle_d - \left\langle u^{\mathrm{t}}_k \right\rangle_d \left\langle u^{\mathrm{t+1}}_l \right\rangle_d, & (3.8) \end{aligned}$$

where $\langle \cdots \rangle_d$ is the temporal averaging over d. The spatial structure corresponds to time series with distinct patterns for P^0 as a “defining statistic”, whereas the time series for the temporal structure correspond to distinct patterns for P^1 (but the same P^0). In both cases, we randomly draw a number of such reference patterns and assign them to one of two classes in a balanced manner. Then we simulate for each pattern several realizations of the time series, which include further empirical noise. Thus, there are two sources of variability in our sample time series: one endowed with their spatio-temporal structure, relevant for classification, and another that relates to observational noise and must be discarded for the classification.

We rely on a 70/30 train/test validation splitting to assess the classification performance, reporting test accuracy. In addition, we use the logistic classifier directly on the empirical statistics P^0 and P^1 for each sample to evaluate the linear separability of the two classes in the feature space (mean or covariances) via the classification performance. We work with datasets whose linear separability is below 100% in order to observe how the performance improves or degrades across the decoding strategies and reservoir configurations.

Spatial structure

The following dynamics are used to produce time series with specific covariances P^0:

$$u^{\mathrm{t}} = W z^{\mathrm{t}}, \tag{3.9}$$

where $W \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$ is a random sparse matrix (with density equal to 0.1) and its non-zero entries are drawn from a normal distribution. The input noise $z^{\mathrm{t}} \in \mathbb{R}^{\mathrm{M}}$ is drawn from a normal distribution, yielding white noise. The resulting time series has zero mean and the zero-lag covariance matrix for this time series is $P^0 = WW^T$, while $P^1 = 0$ [53]. For each dataset with controlled spatial structure, we use 60 different reference W matrices divided in 2 equal groups. For each reference matrix, we produce 500 noisy time series samples, obtained by simulating the dynamical process, then split them with 70/30 ratio for the train/test sets.

Temporal structure

The following dynamics are used to produce time series with specific covariances P^1:

$$u^{\mathrm{t}} = W u^{\mathrm{t}-1} + z^{\mathrm{t}}. \tag{3.10}$$

To assure both that the zero-lag covariances P^0 are the same for reference patterns P^1, we set the matrix $W = \exp(\beta \mathcal{I}_{\mathrm{M}\times\mathrm{M}} + J)$, where parameter $\beta < 0$, $\mathcal{I}_{\mathrm{M}\times\mathrm{M}} \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$ is the identity matrix and $J \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$ is an antisymmetric matrix [53]. This ensures that $P^1 = WP^0$, where P^0 is a diagonal matrix equal to $\exp(2\beta)\mathcal{I}_{\mathrm{M}\times\mathrm{M}}$. Each dataset with temporal structure has 6 such reference W matrices, evenly assigned to one of two classes. To generate each W, we set $\beta = -0.5$ and create the J matrices by drawing the modulus of the entries located above the diagonal from the uniform distribution over $[0.5, 1)$. The entries' signs are chosen in a random fashion, and the density is set to 0.3 to obtain a sparse matrix. For each W, we again simulate 500 noisy time series samples and keep the 70/30 ratio for the train/test sets.

3.2.4 Spoken digits datasets

We test our covariance decoding approach using two real datasets for the recognition of spoken digits: the spoken Arabic digits [176, 177, 178] and the S-MNIST, a dataset derived from the Google speech commands [179, 180]. Both datasets have $K = 10$ classes (one for each digit from 0 to 9) and their multivariate time series consist of 13 Mel Frequency Cepstral Coefficients (MFCC) [181], which are widely employed features for preprocessing of audio signals [182].

The spoken Arabic digits datasets has 8,800 samples, corresponding to 88 utterances of 10 digits spoken by 10 different native Arabic speakers, split in a training/test set with a 75/25 ratio [177, 178]. Sample length varies between 5-92 elements (median 40) in a digit dependent manner. To have a consistent length ($d = 50$) across samples, we shorten or enlarge them through zero-padding in order to make code implementations more straightforward.

The S-MNIST consists of 70,000 (60,000/10,000 train/test split) multivariate time series of digits 0-9, recorded in real-world environments [180, 183], with consistent sequence length across samples ($d = 39$).

3.3 Results

We first present the results for the synthetic dataset, to explore the reservoir configurations that allow for efficient mean or covariance decoding of controlled input signals. Then we compare mean and covariance decoding for the real datasets of spoken digits.

3.3.1 Covariance based readouts capture a larger diversity of input structures

Using the binary classification of synthetic inputs determined by specific structures embedded in their second-order statistics (Section 3.2.3), we explore how the classification accuracy varies with the reservoir parameterization (spectral radius and leak rate), as illustrated in Fig 3.2.

For the spatial structure (top rows, corresponding to Section 3.2.3), the best performance is obtained for a reservoir that is a fully leaky feed-forward layer, namely $\rho(\Omega_{\text{res}}) = 0$ and $\alpha = 1$, with a decaying trend when the spectral radius increases. This holds for all leak rates considered and both decoding types are only weakly sensitive to the leak parameter: a slower integration mechanism (smaller α) marginally enhances the performance. Importantly, we find that the reservoir can boost the covariance decoding (but not the mean decoding) above the benchmark performance of a classifier directly operating on input statistics (light gray lines).

For the temporal structure (bottom rows, corresponding to Section 3.2.3), the performance for the covariance decoding is much better than that obtained with the classical mean decoding, the latter being only slightly better than chance. Moreover, it goes beyond the benchmark of the logistic classifier directly applied to one-lag covariances (dashed light gray line). In other words, the zero-lag covariances of the reservoir are more easily separable than the lagged covariances of the inputs. To achieve this, the reservoir requires a distinct configuration for the spectral radius and leak rate. To begin with, for all leak rates tested, both readouts reach their top accuracy for non-null radii. Moreover, the leak rate has a strong influence here, where a fully leaky reservoir with $\alpha = 1$ performs best. This hints that reservoir global dynamics are vital when the goal is to map the input temporal covariances into reservoir spatial statistics of first or second-order. In this case, retaining information about past input states is crucial and can be best achieved by having recurrent connections within the reservoir. These optimal reservoir configurations with $\rho(\Omega_{\text{res}}) \approx 1$ have been shown to maximize memory capacity for inputs of Gaussian nature [101, 174]. In those studies, the reservoir transforms current in-

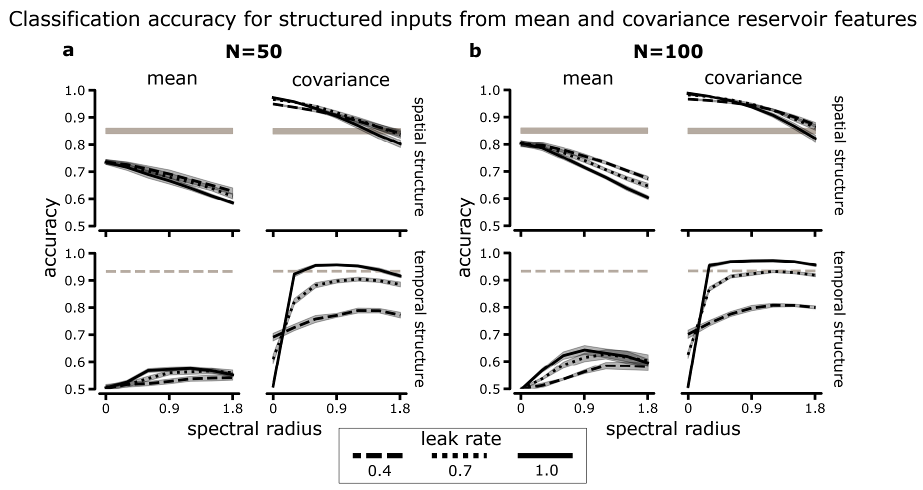


Figure 3.2: Classification accuracy of synthetic datasets. **a.** Accuracy in binary classification of structured multivariate time series that differ by their spatial structure (top row) and temporal structure (bottom row). We compare two types of features fed to the decoder: the mean reservoir activity (left column) and its spatial covariances (right column). We show test accuracy versus spectral radius (x-axis) and for different leak rates (indicated by the various line styles), for reservoirs of size $N = 50$. The results for each reservoir model (leak and radius) are averaged across 5 different random seeds for reservoir fixed connections and synthetic data generation, where shaded areas are ± 1 standard error of the mean. Horizontal lines indicate the benchmark performance of the two decoders when no reservoir is used: logistic regression on input spatial covariances (solid gray) and logistic regression on input one-lag covariances (dashed gray). **b.** Same as Fig 3.2a, but for reservoirs of size $N = 100$.

put activation into specific time-dependent signals (i.e. lagged inputs). Here it does the converse, by transforming “time-dependent” (lagged) information into zero-lagged information. Furthermore, accuracy profiles increase with leak rate for the two decoding configurations. Nonetheless, in feedforward reservoirs ($\rho(\Omega_{\text{res}}) = 0$), the leaky mechanism is essential for covariance decoding, since it implements a temporal integration of the inputs.

As a summary for the synthetic datasets, statistics embedded in time series can be efficiently processed by reservoirs with distinct configurations depending on the nature of the input spatio-temporal structure. A good compromise for both configurations studied here lies at reservoirs without integration (i.e. fully leaky with $\alpha = 1$) and medium-sized spectral radius ($\rho(\Omega_{\text{res}}) \approx 0.9$). The spatial covariances of the reservoir convey relevant information that can be used for classification, but the mean activity of the reservoir is much less informative.

3.3.2 Covariance-based decoding is efficient for spoken digits recognition

Now, we test whether covariance decoding coupled to reservoir computing systems works with real datasets for the recognition of spoken digits. Reservoir computing approaches exhibit good performance for speech recognition, as the task has a sequential nature [119, 125, 128, 129, 184, 185, 186, 187, 188, 189]. Indeed, for the spoken Arabic digits dataset [176, 177, 178], reservoir computing constitutes the state-of-the-art model [129]. With an echo state network of size bounded by $N = 1000$ coupled to a classical mean-based readout, the test performance is 99.9923%, and it has been reported to reach 99.9945% when switching the representation from the natural reservoir state space to a predictive model state space one [129]. With a logistic regression classifier trained on input means, we obtain for this dataset a 69.67% test accuracy, while using as features spatial and temporal covariances, we obtain test performances of 91.68% and 91.13% respectively. This indicates that both statistics contain potentially rich information to classify the time series and are thus suitable to test a

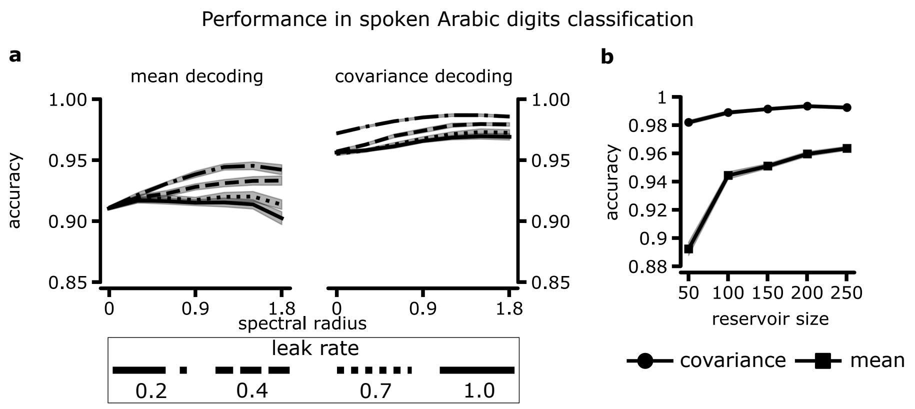


Figure 3.3: Classification accuracy for spoken digits time series. **a.** Accuracy when using mean (left panel) and covariance (right panel) features for decoding, versus spectral radius (x-axis) and for different leak rates (indicated by the various line styles), for reservoirs of size $N = 100$. **b.** Classification accuracy versus reservoir size (for $\alpha = 0.2$ and $\rho(\Omega_{\text{res}}) = 1.2$) for mean (squares) and covariance (circles) decoding. Results for both figures are averaged across 10 different random seeds for reservoir fixed connections, shaded areas are ± 1 standard error of the mean.

covariance decoding scheme.

As with synthetic data, we examine the influence of the spectral radius and the leak rate. We find that both decoders reach top performance for small leak rate ($\alpha = 0.2$) and large spectral radius ($\rho(\Omega_{\text{res}}) > 1$). This is consistent with our results for the temporal structure rather than those for the spatial structure (Fig. 3.2), suggesting that the temporal structure of the real data is relevant to classify the digits. For this task (and unlike the synthetic data), small leak rates are optimal, which could be explained by the slow rate at which speech features generally vary as compared to the sampling frequency [119]. Importantly, covariance decoding outperforms mean decoding across all the explored model parameters, with an average increase of $4.9 \pm 0.1\%$. This gap in accuracy between decoders can be reduced by increasing the reservoir size (Fig 3.3b). In fact, covariance decoding remains quite stable, reaching $99.23 \pm 0.04\%$ accuracy for $N = 250$, while mean decoding results in $96.33 \pm 0.04\%$. We remark that with as little as $N = 50$, our covariance framework already reaches $98.2 \pm 0.2\%$ accuracy, which is way above that obtained with mean decoding, and even comparable to the performance given by much sophisticated models, such as the bidirectional long-short-term-memory (LSTM) network that yields 98.77% accuracy [190].

The classification task for S-MNIST [183] turns out to be more challenging than the spoken Arabic digits, as samples for a single digit display more variability and noise due to recording conditions [179]. To our knowledge, the best reported accuracy in the literature is $75.9 \pm 0.2\%$, obtained with a self-organizing map [180]. A logistic regression classifier using input means as features (i.e. without a reservoir) reaches a rather poor performance of 32.92%, while one trained on spatial covariances reaches 80.58%. For this dataset, we also tried a naive method that consists in feeding the whole input sequence, without feature computation, to the logistic classifier, which yielded 48.56% accuracy. This confirms that the method to compute features (means or covariances) dramatically affects the classification accuracy of the downstream pipeline: here covariances of those spoken signals convey more accurate information that is specific to the spoken digits than their means.

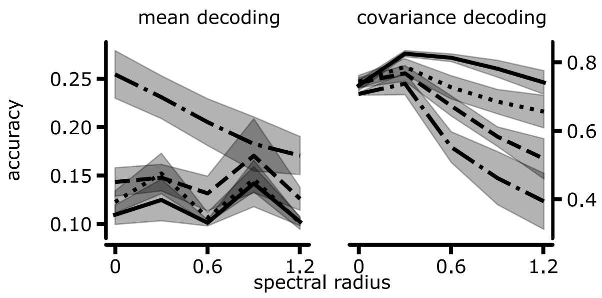


Figure 3.4: Classification accuracy for S-MNIST time series. Accuracy when using mean (left panel) and covariance (right panel) features for decoding versus spectral radius (x-axis), for different leak rates (indicated by the various line styles), for reservoirs of size $N = 100$. Note that the y-axes have different scales for the subfigures. Results are averaged across 5 different random seeds for reservoir fixed connections, shaded areas are ± 1 standard error of the mean.

Now using a reservoir with $N = 100$, this further filtering of the time series does not improve performance for mean decoding (Fig 3.4), and only marginally for covariance decoding with $\approx 2\%$ increase. Nevertheless, the interesting outcome of our study here is the strong effect of the reservoir configuration on the classification performance (as earlier with the other datasets), corresponding to $\rho(\Omega_{\text{res}}) = 0.3$ and $\alpha = 1$ for the optimal configuration. Once again, covariance decoding performs much better than mean decoding and displays a more robust behavior across the dynamical parameters space.

All in all, our findings show that covariance decoders can be successfully applied to small-size reservoirs to classify real time series.

3.4 Discussion

In this paper we have shown that covariance patterns can be powerful features for the classification of real time series and that reservoir computing can boost the performance even with limited number of resources, as measured by the reservoir size here. Our study systematically compares a decoding method that leverages the second-order statistical moments of time series to the classical approach based on first-order statistics. Although this has been tried in a few previous studies [53, 171, 172], it had not been explored when coupled with reservoir computing. Reservoir computing usually relies on first-order statistics that depend on target being e.g. mean activity or fixed profiles of activity for which variability is undesired noise. Instead, we exploit the structured variability of the reservoir activity in our novel decoding scheme. Together, our results pave way to the use of higher-order statistics as features to classify time series in bio-inspired networks dedicated to machine learning.

The reservoir operates on the covariances of the input time series by projecting them into a higher-dimensional space where linear separability is easier to achieve. This transformation by the reservoir is similar to that for usual mean patterns [97, 101, 108, 125], but supposedly with an order of magnitude larger: $O(N^2)$ for spatial covariance patterns instead of $O(N)$ for mean patterns, with N being the reservoir size. This partly explains why covariance decoding outperforms mean decoding even with much smaller reservoirs. This advantage in terms of dimensionality of the feature space (fed to the readout decoder) is in line with previous theoretical results that showed that a single-layer network can separate more covariance patterns than mean patterns [7]. The performance enhancement for low resources, in terms of reservoir size, posits covariance decoding as an interesting option to further explore in neuromorphic systems [191, 192]. For large reservoir size, the dimensionality of the corresponding activity covariance scales up quadratically and other regularization than the L2 considered here may give better accuracy, like L1 to impose stronger sparsity on the contributing features to the readout classifier.

Moreover, we have demonstrated that covariance decoding can capture a broader range of input structures than mean decoding after the operation of the reservoir, using synthetic datasets where time series have controlled second-order statistics, either with their spatial or temporal covariances. In this set-up, decoding temporal structures works best for reservoir with recurrent connectivity (see Fig 3.2 bottom rows, $0.3 \leq$

$\rho(\Omega_{\text{res}}) \leq 1.8$ with $\alpha = 1$), whereas decoding spatial structures works best for feed-forward reservoir ($\rho(\Omega_{\text{res}}) \simeq 0$), without much constraint on α. A good compromise is thus $\rho(\Omega_{\text{res}}) \simeq 0.3$ with $\alpha = 1$ (i.e. fully leaky). For the real dataset of spoken Arabic digits, the top reservoir configuration has spectral radius within the range of good values found for the synthetic dataset with temporal structure, while the leak rate is smaller (see Fig 3.3, $\rho(\Omega_{\text{res}}) \geq 0.9$ with $\alpha = 0.2$). This smaller leak is in agreement with previous work using reservoirs with mean decoding, which suggested that small leak rates are suitable to the inherent time scale of the speech sequences whose structure changes at a slow pace in relation to the sampling frequency [108, 121]. Nevertheless, this signifies that the reservoir extracts a complex spatio-temporal structure from these input signals. For S-MNIST, the performance increase when adding the reservoir is comparatively much smaller than for the Arabic digits data. This is due to the inherent richness of MFCC as features for speech recognition, coupled with the overall bigger size of the S-MNIST dataset [191]. Therefore, our method remains to be benchmarked in multivariate times series data of different nature and further compared to other approaches, such as dynamic time warping and ROCKET [193].

Reservoir computing performance in time series forecasting has been shown to be also sensitive to the detailed configurations of the input-to-reservoir and within-reservoir connectivity matrices, as they affect the quality of the reservoir representations of the inputs [122, 124, 126, 194]. This direction remains to be explored in our covariance set-up for classification, as well as possible extensions to deep or layered reservoir architectures [195, 196]. Recent work has shown that plasticity mechanisms within the reservoir improve performance by decreasing its pairwise correlations [125, 197], which could imply that tailoring the reservoir for mean-based representations degrades higher-order ones. This naturally raises the question of which mechanisms could shape the reservoir for covariance representations, and whether both statistical orders combined can have a synergistic effect on performance.

A comparison with deep learning is also left for future work. In the same manner that high-order statistics can be extracted from images by convolutional layers in deep networks, the same scheme has been applied across time for time series to automatically extract high-order statistical features [198, 199]. The comparison of extracted patterns by reservoir computing and deep networks on time series would be interesting, to assess whether they fundamentally perform similar or distinct operations for the same classification task [200]. In addition, extension of deep networks (possibly with recurrent connectivity) could be developed to process structures embedded in distinct statistical orders of the time series in a cross-talk manner, rather than fanning in first-order statistics in outputs.

Chapter 4

COVARIANCE BASED INFORMATION PROCESSING IN RESERVOIR COMPUTING SYSTEMS

This chapter expands on the concepts of Chapter 3 and exploits variability in a biologically more plausible setting where time series are mapped to time series with embedded information in their structure. Part of the methodology and synthetic datasets are, thus, the same.

The work here presented is currently in preparation for submission. Preprint details: Lawrie, S., Moreno-Bote, R. and Gilson, M. (2021). Covariance-based information processing in reservoir computing systems. Bioarxiv, doi: https://doi.org/10.1101/2021.04.30.441789

Abstract

In biological neuronal networks, information representation and processing are achieved through plasticity learning rules that have been empirically characterized as sensitive to second and higher-order statistics in spike trains. However, most models in both computational neuroscience and machine learning aim to convert diverse statistical properties in inputs into first-order statistics in outputs, like in modern deep learning networks. In the context of classification, such schemes have merit for inputs like static images, but they are not well suited to capture the temporal structure in time series. In contrast, the recently developed covariance perceptron uses second-order statistics by mapping input covariances to output covariances in a consistent fashion. Here, we explore the applicability of covariance-based perceptron readouts in reservoir computing networks to classify synthetic multivariate time series structured at different statistical orders (first and second). We show that the second-order framework outperforms or matches the classical mean paradigm in terms of accuracy. Our results highlight a nontrivial relationship between input and reservoir properties in generating the output reservoir activity, which suggests an important role for recurrent connectivity in transforming information representations in biologically inspired architectures. Finally, we solve a speech recognition task for the classification of spoken digits to further demonstrate the potential of covariance-based decoding for real data.

4.1 Introduction

The variability of spiking activity is a hallmark of biological neuronal networks. It has been observed both in vivo and in vitro, in resting and active states and across a wide variety of species, brain areas and time scales [17, 20, 201]. The role that trial-by-trial variability plays in information processing in neuronal networks is currently under debate, after early considerations that saw it as detrimental to (de)coding and learning [18, 202], but were later proven to not always be the case [42, 43, 203]. In particular, it has recently been shown that structured variability, corresponding to reproducible correlation patterns, can be a substrate for robust information processing [53]. That study showed that patterns determined by second-order statistics can be learned and transformed by a simple linear analogue network, thereby implementing a ‘covariance perceptron’.

More generally, covariance coding offers a middle ground alternative between the much-debated rate coding and temporal coding theories [12, 23, 204]. In the new view, neither the mean nor the full probability distribution of neuronal activity are used for information transmission, but rather the pairwise correlation between neurons, as quantified by second-order statistics. This view is backed up by recent experimental findings in neurophysiological data, where spike patterns —corresponding to second-or-higher statistical orders— are informative about stimulus or behavior [5, 50]. Considering statistics up to the second order to define informational patterns, we examine how neurons can classify multivariate time series, which has been also used in a variety of applications [171, 172].

In this context, we specifically focus on the processing of patterns determined by structured variability by a reservoir computing system. It usually consists of an untrained network of neurons used to filter incoming signals before feeding a readout layer [97, 98, 99, 100], which draws on the computational power arising from the interplay between recurrent connectivity and neuronal nonlinearities. In particular, nonlinearities can map inputs into spaces where linear separability is easier to achieve, as used in many network architectures like multilayer or deep networks [66, 205]. In addition, reservoirs of larger size than the inputs can map them to a higher-dimensional space, which can be beneficial to separate different input patterns. A third point is that, by only training connections from the reservoir to the readout to perform the classification, the weight optimization procedure is simpler (fewer resources to tune) and often more stable as compared to the training of recurrent connections. Reservoir networks thus appear as an interesting candidate to process statistical structures of time series in a classification task, which has been traditionally exploited by reading out the mean activity of the reservoir [97, 99, 108, 119], although other more complex methods have been tried, such as the model space representation [206].

This study explores the combination of reservoir computing systems with mean/covariance decoding to classify multivariate time series (Fig. 4.1a, top pathway), using both synthetic and real data. Specifically, we examine the cross-talk between the first and second statistical orders of input times series in the reservoir activity, including both spatial and temporal structure for the second order (i.e. zero-lag and lagged covariances). To do so, we explore through exhaustive numerical simulations and analytically derived insights the influence of the reservoir parameters to elucidate their interplay in shaping the reservoir processing. We implement our reservoirs by means of echo state networks, whose parameters of interest are the spectral radius (quantifying the overall strength of the recurrent connectivity) and the leak rate. These parameters have been shown to influence strongly reservoir properties such as memory capacity for Gaussian inputs [101, 174], or the processing of input time series that involve low frequency signals [121], so we aim to test whether similar tendencies are observed for the processing of second-order statistics.

In addition, we compare two types of decoders: a single layer of neurons that are fed from the reservoir inspired by biology (linear perceptron, LP); and the multinomial logistic regression (MLR) coming from machine learning [47] (Fig 4.2b). The main difference between these two decoders is that the LP processes its inputs in real time, producing an output time series that has class-dependent structure at the first (mean-LP) or second orders (cov-LP). The MLR, on the other hand, is fed by precomputed input statistics of either first (mean-MLR) or second-order (cov-MLR) and outputs a single class-probabilities vector.

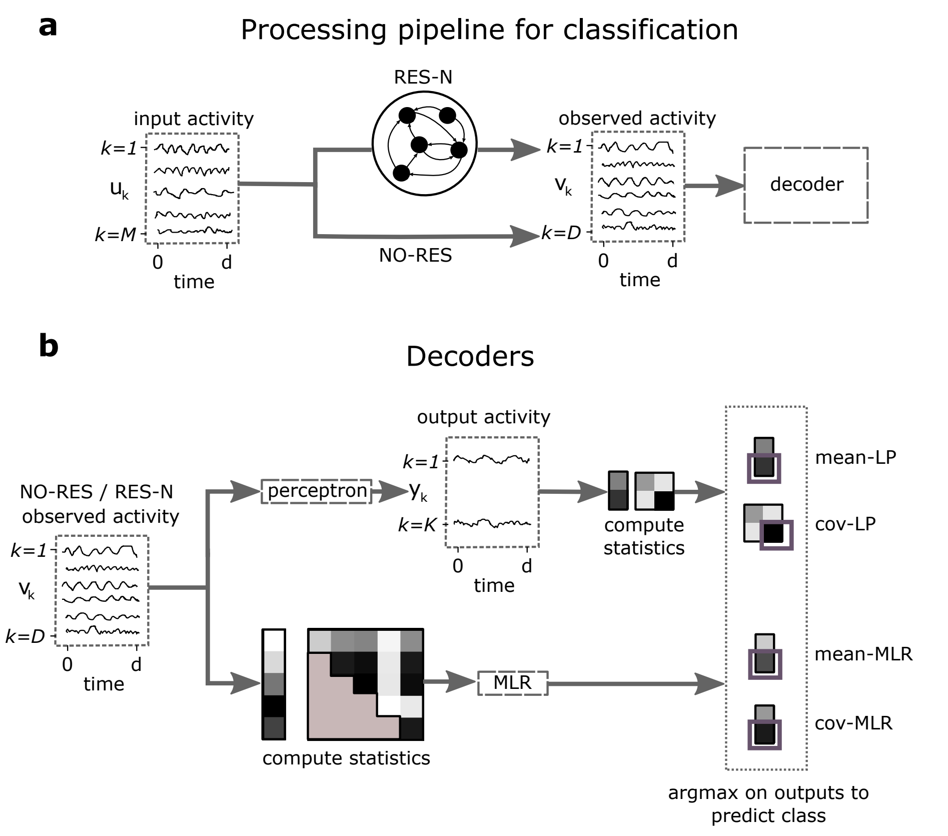


Figure 4.1: Processing pipeline for multivariate time series classification. **a:** An M-dimensional input time series observed for d time steps is either filtered by a reservoir layer of N neurons (‘RES-N’, top pathway) or directly fed to a linear decoder (‘NO-RES’, bottom pathway) to perform a classification task. The bottom pathway was studied in [53], and we implement it here for comparison purposes. **b:** Our main focus of study is the linear perceptron (LP) decoder (top pathway), but we also implement a multinomial logistic regression (MLR) decoder (bottom pathway) whose accuracy we use as a reference.

Figure 4.1 *(previous page)*: There is an important operational difference between the LP and the MLR, given by the order in which the statistics are computed for classification. The MLR pathway first computes the statistics of the observed activity, and outputs a class probability vector. This implies the time series are transformed to static features. The perceptron, on the other hand, maps the observed activity time series to output time series, transforming time series to time series. The output time series thus convey information in its statistics, which are afterwards evaluated to make the decision.

The manuscript is structured as follows. Section 4.2 describes the pipeline that we use for classification of multivariate time series, namely the reservoir implementation and the decoders (Fig. 4.1), followed by the datasets used to test its performance. Results are then presented in Section 4.3, starting with synthetic time series to uncover general principles and followed by real data to further verify our proposed classification scheme. They are then discussed in Section 4.4 to contextualize them with respect to their biological and machine-learning implications.

4.2 Methods

This section first presents the reservoir implementation used in our simulations. Among the variety of reservoir implementations that have been proposed [100], we rely on echo state networks that employ analogue sigmoid neurons [97, 99, 108, 119], since they provide a formalism that is compatible with the covariance perceptron (used as a decoder). We also provide an analysis of the reservoir first and second-order statistics using a weakly nonlinear approximation, inspired by previous work on reservoir dynamics and memory capacity [103, 125].

Then, we explain the training of the decoder, which is done by performing a gradient descent on its weights (from the reservoir units) to minimize the mean-squared error between output activity and target activity as a cost function. The target activity is defined such that the classification can be performed by comparing the values (means or variances) of the readout outputs in a winner-take-all fashion. Importantly, the gradient descent depends on the metric applied to the reservoir activity and the type of target activity, which differ across the decoders as illustrated in Fig. 4.1b.

Last, we detail the generation of the input time series that are used to test the classification pipeline. The classification task consists of separating the time series into K classes, with $K = 2$ for the synthetic datasets and $K = 10$ for the real dataset. Each of these classes consist of samples that differ by their statistical features, which are transformed by the reservoir and must be captured by the decoder.

We remark that in all the equations to be presented throughout this work, we use lower case greek letters for real parameters, lower case latin letters for vectors and upper case letters (greek or latin) for matrices, except when noted (e.g M and N are natural numbers).

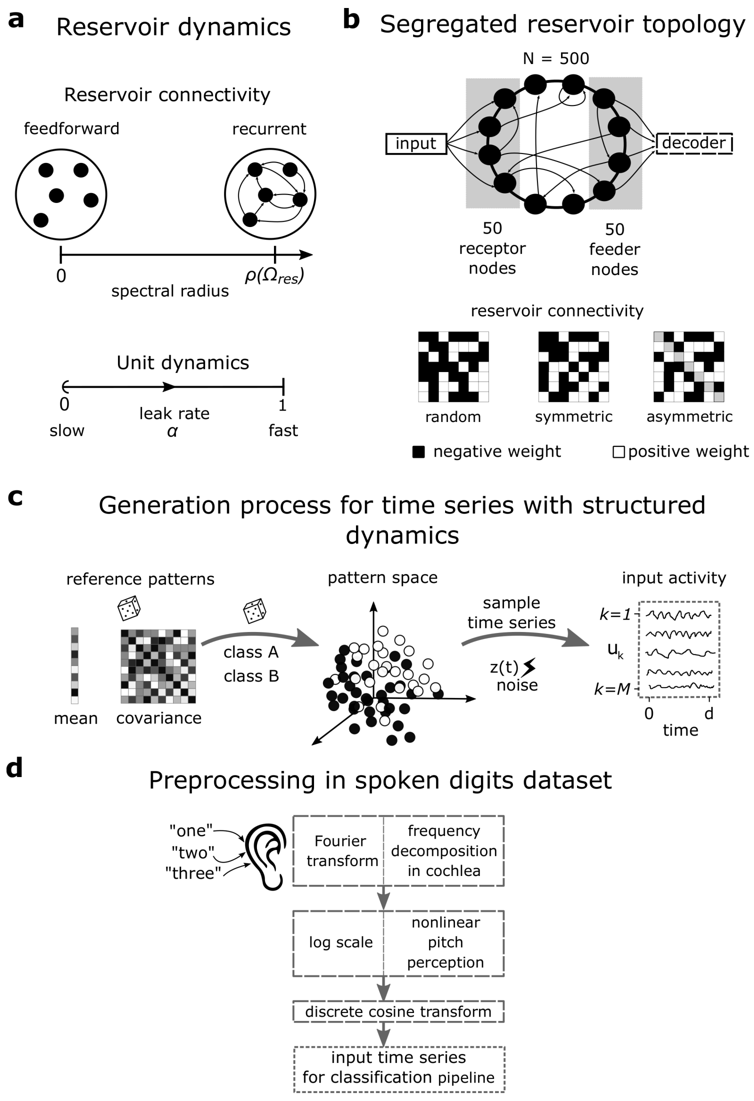


Figure 4.2: Description of reservoir properties and input datasets. **a:** The reservoir dynamics are characterized by the spectral radius $\rho(\Omega^{\text{res}})$, which is the largest absolute eigenvalue of the reservoir's connectivity matrix. By convention, a reservoir with $\rho(\Omega^{\text{res}}) = 0$ means $\Omega^{\text{res}} = 0$ and corresponds to a feedforward layer without recurrent connectivity. We also modulate the reservoir processing via the local units' dynamics by adding a leak parameter α.

Figure 4.2 *(previous page)*: Small leak values indicate an integration of the input over time yielding slower update dynamics, while $\alpha = 1$ indicates that the units do not integrate past input information. **b:** We explore an additional reservoir structure where nodes connected to inputs (receptors) are segregated from nodes connected to outputs (feeders) [207]. We choose the reservoir connectivity matrix to be fully random, symmetric or asymmetric (see Section 4.2.1). **c:** For synthetic inputs, we first sample reference patterns from given probability distributions. Afterwards, we randomly split them in two balanced classes. To generate a sample time series for a given pattern, we add noise at each time point through specific dynamics (see Eqs. 4.10-4.13). **d:** The real data consist of input time series that correspond to spoken digits. To approximate the way the human cochlea processes sound when entering the ear, the speech signals are framed and windowed. For each time bin, a frequency spectrum is computed through a Fourier transform, simulating the frequency-tuning of nerve cells in the cochlea, and a logarithmic scale (mel scale) is used to represent the power coefficients, simulating the nonlinear perception of pitch in humans. These coefficients are decorrelated by means of a discrete cosine transform, keeping 13 amplitude coefficients per time bin. This multivariate time series is afterwards passed through the processing pipeline in B for prediction of the spoken digit. Note that the dataset is only available as preprocessed MFCC coefficients [176, 178].

4.2.1 Reservoir implementation

The reservoir used in the classification pipeline ('RES-N' in Fig. 4.1a, top pathway) is an echo state network with N leaky integrator neurons (or units), similar to previous studies [99, 108]. The update equations for the activity state at time t of the N leaky neurons inside the reservoir, denoted by $x(t) \in \mathbb{R}^{\mathrm{N}}$ when fed by a multivariate input $u(t) \in \mathbb{R}^{\mathrm{M}}$, are

$$\begin{aligned} \bar{x}(t) &= \mathcal{F}\left(\Omega^{\mathrm{in}}u(t) + \Omega^{\mathrm{res}}x(t-1)\right), && (4.1) \\ x(t) &= (1-\alpha)\,x(t-1) + \alpha\bar{x}(t), && (4.2) \end{aligned}$$

where the function $\mathcal{F} = \tanh$ has a sigmoidal profile, $\Omega^{\text{in}} \in \mathbb{R}^{\text{N}\times\text{M}}$ is the connection matrix from the input time series to the reservoir units and $\Omega^{\text{res}} \in \mathbb{R}^{\text{N}\times\text{N}}$ is the weight matrix of recurrent connections within the reservoir. All connection weights are randomly sampled from $[-0.5, 0.5)$, and the resulting matrices are dense. The parameter α is a leak rate, $\alpha \in (0, 1]$, which governs how each reservoir unit integrates its own dynamical state over time. When α tends to zero, the neuron's dynamics becomes slower and more dependent on previous history than on the current input state [99, 108]. When $\alpha = 1$, the activity of the reservoir (Eq. 4.2) is $x(t) = \bar{x}(t)$, thus, no integration is performed and each unit's activity state only depends on the instantaneous inputs and activities of other neurons.

Using numerical simulation, we explore the different dynamical regimes of the reservoir, by varying a local parameter (leak rate applied homogeneously to all units) or a global parameter (spectral radius), see Fig. 4.2a. The spectral radius $\rho\left(\Omega^{\text{res}}\right)$ is the largest absolute eigenvalue of the reservoir's weight matrix Ω^{res}. It affects the reservoir performance in different benchmark tasks typically reported in the literature, such as memory [97, 173, 174]. A general heuristic when $\alpha = 1$ is that $\rho\left(\Omega^{\text{res}}\right)$ should approximate 1 (from below) for tasks that require long memory and be smaller for tasks where a too long memory might be detrimental [99]. Since we are agnostic to the effects of this parameter for multivariate time series classification (especially those with structured dynamics as we have detailed in the previous section), we vary it spanning a range that goes from 0 to 1.8. In the absence of inputs, a spectral radius larger than 1 implies that a linear reservoir (i.e. when $\mathcal{F}$ is the identity operator in Eq. 4.1) is unstable, in the sense that its trajectory will deviate away from the zero fixed-point when started from a non-zero state [121]. However, in practice, the sigmoid function bounds the growth of the trajectory and effectively produces a reservoir that is dynamically less excitable. Note that, by convention, a null spectral radius implies a zero connection matrix ($\Omega^{\text{res}} = 0$), corresponding to a feedforward layer (left-hand side in Fig. 4.2a). Thus, $\alpha = 1$ and $\Omega^{\text{res}} = 0$ implies a nonlinear and memoryless transformation of the inputs randomly mixed by Ω^{in}. When $\alpha < 1$, an effective spectral radius can be calculated for the reservoir, corresponding to the linearization of its dynamics: $\alpha\Omega^{\text{res}} + (1-\alpha)\mathcal{I}_{\text{N}\times\text{N}}$, instead of Ω^{res} [108]; here $\mathcal{I}_{\text{N}\times\text{N}}$ is the identity matrix.

Other possible parameters for exploration include the input scaling and the choice for the sigmoid function $\mathcal{F}$, which we leave for future work.

All the results we present in this study are averaged across 10 different reservoir configurations, where a configuration is given by specific connection matrices Ω^{in} and Ω^{res}, and we always start each reservoir from a zero state. Importantly, our work focuses on transient states, since we are interested in learning and representations in short time scales.

Reservoirs propagate diverse input statistics

Under slightly different conditions than the ones we have stated previously (see Supplementary Material 4.5.1 for full details), it can be proven that the first-order statistics of a neuron x_i inside a reservoir with $\alpha = 1$ and spectral radius ρ when fed by a multivariate input time series $u(t) \in \mathbb{R}^M$, with $u_1(t)$ a bias unit, is given by:

$$\begin{aligned}\langle x_i(t)\rangle \approx &\sum_{m=1}^{M} \frac{\varepsilon}{1-\rho^2}\langle u_m(t)\rangle \\ &- \sum_{m,r=2}^{M} \frac{\varepsilon^3}{1-\rho^2}\frac{1}{1-\rho^4}\langle u_m(t)u_r(t)\rangle \\ &- 2\sum_{m,r=2}^{M} \frac{\varepsilon^3}{1-\rho^2}\frac{\rho^2}{1-\rho^4}\langle u_m(t)u_r(t+1)\rangle,\end{aligned} \tag{4.3}$$

where the angular brackets denote temporal average, ε denotes the strength of connections from input to reservoir and we have assumed the neuron is in a weakly nonlinear regime (Supplementary Material 4.5.1).

Likewise, the second-order zero-lag statistics of neurons x_i and x_j in the reservoir are given by:

$$\begin{aligned}\langle x_i(t)x_j(t)\rangle \approx & 2\sum_{m=1}^{M}\frac{\varepsilon^2}{(1-\rho^2)^2}\langle u_m(t)\rangle \\ & + \sum_{m,r=2}^{M}\frac{\varepsilon^2}{1-\rho^4}\langle u_m(t)u_r(t)\rangle \\ & + 2\sum_{m,r=2}^{M}\frac{\varepsilon^2\rho^2}{1-\rho^4}\langle u_m(t)u_r(t+1)\rangle,\end{aligned} \tag{4.4}$$

where in the above derivation it is enough to assume that the neuron is behaving in a linear regime (Supplementary Material 4.5.1).

Thus, the reservoir mixes input statistics and these are reflected in reservoir statistics of first and second-order, with a dependence on input-to-reservoir and within-reservoir connections (i.e. ε and ρ). Full details of this derivation can be found in Supplementary Material 4.5.1. Below we provide some insights for a feedforward reservoir and for the case of temporally correlated inputs.

Feedforward reservoir In the limit $\rho \to 0$, the reservoir becomes essentially a feedforward layer. In this case, reservoir spatial statistics of first and second order do not reflect information embedded in second-order temporal covariances (see Eq. 4.3 and Eq. 4.4 when setting $\rho = 0$). Thus, a purely feedforward reservoir is not useful to process this type of structure. Nonetheless, the nonlinear behavior of the sigmoid function introduces a cross-talk between first and zero lag second-order statistics, where we see that the reservoir mean activity and covariances are influenced in both cases by both input means and covariances. Furthermore, in terms of the strength of the input-to-reservoir connections ε, first order input statistics have $\mathcal{O}(\varepsilon)$ contribution to reservoir mean and $\mathcal{O}(\varepsilon^2)$ contribution to reservoir covariances, while input covariances have $\mathcal{O}(\varepsilon^2)$ contribution to reservoir mean and $\mathcal{O}(\varepsilon^2)$. This hints that maintaining consistency betweeen statistical orders for 'encoding/decoding' is better than mixed schemes when inputs have first or second-order zero lag structure. Thus, when information about class is embedded in the input mean activity, this is more strongly reflected in the reservoir mean activity but can in principle also be recovered from the (zero-lag) second-order statistics (to a lesser extent). However, when the information is embedded in the zero-lag covariances, the above derivation suggests that second-order statistics of the reservoir provide better representations than first-order ones. This constitutes a strong departure from purely linear networks, where covariance-based information representations are only possible if indeed such a representation is present in the inputs [53].

Temporally correlated inputs This type of input structure requires $\rho \neq 0$. This can be seen via the coefficients of the form $\frac{1}{1-\rho^2}$ and $\frac{\rho^2}{1-\rho^4}$ that increase with ρ. Note that they diverge as $\rho \to 1$, but the present analysis based on linearization does not capture well the reservoir regime then. Therefore, the optimal value for the spectral radius is expected to be non-zero and below 1, as it maximizes the influence of input one lag correlation in reservoir mean and spatial covariance statistics. In our calculations, from the coefficients in the Taylor series, we observe that reservoir means are more influenced by input lag statistics than reservoir covariances. However, this only holds if the Taylor series approximation remains valid when ρ increases and we thus verify that a mean-based representation be better than a covariance based one using numerical simulation.

Analysis of reservoir dynamical regime

Since the above analysis relies on the assumption that the reservoir behaves in a weakly nonlinear regime (see Supplementary Material 4.5.1), we need to define in a quantitative manner the different dynamical regimes, in order to investigate its relationship with the efficiency of the mapping on input statistics performed by the reservoir. In practice, we define for each neuron three possible dynamical regimes according to its activation state z, where the activation state is the argument inside the nonlinearity

$\mathcal{F} = \tanh$ at any given time point (Eq. 4.1). If $|z| \leq 0.3$, the regime is considered linear. If $0.3 < |z| \leq |0.6|$, it is considered to be weakly nonlinear. Any other case is likely to involve strong nonlinearities. We set these bounds for the regimes by upper-bounding by 0.01 the error in the truncated Taylor series approximation to the hyperbolic tangent of first and third order. For each of the datasets we work with, we numerically compute the probability of finding a neuron in each dynamical regime as function of spectral radius and leak rate.

Reservoir topology

In most of our study we use a reservoir implementation as detailed above, where each neuron is fed by all the input nodes (i.e. Ω^{in} is a full matrix) and likewise feeds the decoder (corresponding to a full Ω^{out} matrix).

However, it is known that the anatomical connectivity in the brain is not full, but sparse and constrained. Getting inspiration from one of the most known pathways of cortex from sensory areas to motor areas, we also consider a reservoir topology that mimics this structure by simply separating input receptor nodes from output feeder ones, as studied in previous work (Fig. 4.2b) [207]. To study how activity propagates across this specialized reservoir, we place the neurons along a ring and choose the input receptor neurons to be opposite the output feeder ones. This implies constrains in Ω^{in} and Ω^{out}. For all our simulations using the segregated reservoir, we use a total of $N = 500$ neurons, where two disjoint sets of size 50 form the receptor and feeder groups.

Within the reservoir, we try three different naive and sparse (0.1 density) connectivity patterns for Ω^{res}: random, symmetric and asymmetric. The motivation behind exploring these different patterns comes from the design of the synthetic inputs in Section 4.2.3, where a mixing matrix given by an asymmetric matrix J guarantees, via the exponential function, that the inputs are temporally but not spatially correlated. To generate these topologies, all non-null reservoir connectivity matrix elements are uniformly sampled from $[-0.5, 0.5)$. For the symmetric (asymmetric) matrices, we only sample elements Ω^{res}_{ij} for the upper triangular part, while the lower triangular elements are assigned to fulfil the symmetry (asymmetry) condition $\Omega^{\text{res}}_{ji} = \Omega^{\text{res}}_{ij}$ ($\Omega^{\text{res}}_{ji} = -\Omega^{\text{res}}_{ij}$).

4.2.2 Decoders

We have four different learning and decoding schemes, as outlined in Fig. 4.1b: mean-LP, cov-LP, mean-MLR and cov-MLR, with names following the convention 'feature-classifier', where the feature corresponds to the statistical order used by the classifier to predict the class. We denote by $v(t) \in \mathbb{R}^{\text{D}}$ the observed activity in Fig. 4.1b, which either comes directly from the input time series (i.e. $v(t) = u(t)$ with $D = M$) or is filtered by the reservoir ($v(t) = x(t)$ with $D = N$). The prefix 'mean' indicates that the classifier relies on the mean vector of the observed activity, namely $\mathcal{S}(v(t)) = \frac{1}{d}\sum_{t=1}^{d} v(t)$. The prefix 'cov' indicates that the decoder relies on the matrix of zero-lag (or one-lag) covariances of the observed activity, $\mathcal{S}(v(t)) = \frac{1}{d-1}\sum_{t=1}^{d-1}\left(v(t) - \frac{1}{d}\sum_{t'=1}^{d} v(t')\right)\left(v(t) - \frac{1}{d}\sum_{t'=1}^{d} v(t')\right)^{\top}$, with the superscript $\top$ indicating the matrix transpose. In each case, we consider two options for the classifier: a linear perceptron [47, 93] (LP in Fig. 4.1b) or a (multinomial) logistic regression classifier [47] (MLR in Fig. 4.1b). Our main focus is the performance of the LP and we implement the MLR decoder only as a reference.

We stress that there is an operational difference between the two classifiers. The LP is biologically inspired in the sense that it generates, for a K-class classification problem, an output time series $y(t) \in \mathbb{R}^{\text{K}}$ at each time step, and the statistical moments for classification are computed for this vector when the observation period is over (top pathway in Fig. 4.1b). Thus, the output activity at time t is given by:

$$y(t) = \Omega^{\text{out}} v(t), \tag{4.5}$$

where $\Omega^{\text{out}} \in \mathbb{R}^{\text{K}\times\text{D}}$ is the matrix of classifier parameters. We remark that Ω^{out} has a fixed dimension independently of the statistic used for classification. In the mean-based instance, the predicted class is given by the output node with highest mean activity during the observation period.

In the covariance-based instance, the predicted class is given by the output node with highest variance during the observation period. Note that when using the covariance perceptron learning rule, we always implement a mapping between spatial covariances, since we do not consider the case of recurrent connections in the readout layer, as is needed to map temporal covariances to spatial covariances [53].

On the other hand, the MLR is a conventional machine-learning approach which first computes the desired time series statistics and afterwards uses it as a vector entry for the classifier (bottom pathway in Fig. 4.1b). To do so, the MLR produces a single output vector $y \in \mathbb{R}^{\mathrm{K}}$ from the statistics $\mathcal{S}(v(t))$ of the observed time series, which is not time dependent, but instead represents the probabilities of the input feature vector to belong to each class:

$$y = \exp\left(\Omega^{\mathrm{out}} \mathcal{S}\left(v(t)\right)\right). \tag{4.6}$$

In Eq. 4.6, $\mathcal{S}\left(v(t)\right)$ is D-dimensional for the mean vector and $D(D+1)/2$-dimensional for the vectorized covariance matrix (taking symmetries into account), so the output matrix (Ω^{out}) dimension depends on the statistic used as feature. In practice, the output is L1-normalized so that all the elements in y sum to 1. The class is then given by the index of the maximum element in y. In other words, the order to the transformation by Ω^{out} and the calculation of the statistics is swapped between the two types of classifiers (Fig.4.1b). Therefore, the MLR for covariances is not equivalent to the linear perceptron with an extra nonlinearity (logistic function). We further emphasize that, once both decoders are trained, while MLR always computes the statistics of the observed activity for classification, the perceptron instead embeds this information directly in the output time series, which are then further processed to compute the statistics and predict the class (Fig. 4.1b).

In addition, all our models make use of a bias unit at the input-to-reservoir and reservoir-to-output levels, which can be straightforwardly included in all our previous equations. This unit consists of a time series with constant (unit) activity as additional entry. Thus, it has a mean equal to 1, null variance and null cross-covariance with other input nodes. Another possibility would be to choose the bias unit that feeds the decoder unit as a signal with zero-mean activity and variance of 1. Intuitively, this would correspond to adjusting the offset in covariance space. However, to keep consistency at all layers, we keep the bias as a unit constant, given its importance for input representations (see Section 4.2.1 and Supplementary Material 4.5.1).

Learning procedures

Once that the pipeline is set (with or without reservoir, statistical order of feature and decoder), the final step is training the decoder by tuning the matrix weights Ω^{out}, the only plastic aspect of the network. In all cases, we rely on a gradient descent that aims to minimize a cost C.

For the mean-LP, learning is achieved by minimizing a regularized mean squared cost function between the output mean activity $m = \langle y(t) \rangle_d$ and a target output mean activity $\bar{y} \in \mathbb{R}^{\mathrm{K}}$ during the observation period d:

$$C = \frac{1}{2} \sum_{k=1}^{K} \left(m_k - \bar{y}_k\right)^2 + \frac{\lambda}{2} \sum_{k=1}^{K} \sum_{i=1}^{D} \left(\Omega_{ki}^{\mathrm{out}}\right)^2 . \tag{4.7}$$

Given the linear nature of the readout, when $\lambda = 0$, this is equivalent to matching an output time-dependent trajectory to a constant target output trajectory [53], which corresponds to the common winner-take-all readouts typically used in reservoir computing applications for classification [108, 119, 185, 186]. We use the scikit-learn library [175] to minimize this cost function. Importantly, we do not highly tune the regularization parameter λ, but set it to 0.02 for all models.

For the cov-LP decoder, learning consists in minimizing a squared error cost function from output spatial (zero-lag) covariances $Y^0 \in \mathbb{R}^{\mathrm{K}\times\mathrm{K}}$ to target spatial covariances $\bar{Y}^0 \in \mathbb{R}^{\mathrm{K}\times\mathrm{K}}$ [53]:

$$C = \frac{1}{2} \sum_{k=1}^{K} \left(Y_{kk}^0 - \bar{Y}_{kk}^0\right)^2 . \tag{4.8}$$

Note that since only the diagonal elements of these matrices are useful for classification, we only constrain them during training, leaving the cross-covariances to vary freely. This is achieved through a gradient-descent learning rule derived for linear dynamics [53], where the weight updates $\Delta\Omega^{\mathrm{out}}_{ik}$ for the connection between element k in the output and element i in the observed activity are given by:

$$\Delta\Omega^{\mathrm{out}}_{ik} = \eta \left(\bar{Y}^0 - Y^0\right) \odot \left(G^{ik} V^0 \Omega^{\mathrm{out}\top} + \Omega^{\mathrm{out}} V^0 G^{ik\top}\right), \qquad (4.9)$$

where η is the learning rate, $V^0 \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$ is the spatial covariance matrix of the observed activity and $G^{ik} \in \mathbb{R}^{\mathrm{K}\times\mathrm{D}}$ has 0s everywhere except in element (i,k) that is equal to 1. Symbol $\odot$ denotes the element-wise (Hadamard) product followed by summation of resulting elements. The learning rate is set for all models to 0.01 and 100 optimization steps are performed.

For the MLR decoders, learning is done through stochastic gradient descent to optimize an L2-regularized cross entropy cost function [47]. For this, we use the scikit-learn library [175].

Subsampling procedure for observed activity

Generally, statistical models with larger number of free parameters will yield better performing models than those with lower resources, provided they do not overfit the data [47]. Since the cov-MLR decoder differs in the number of parameters to learn when compared to the other three schemes, to fairly compare them we subsample the dimensions of the observed activity vector $v^(t)$ so that its vectorized covariance matrix has dimension close to D. In the reservoir pipeline, D represents the size N. Thus, if $N = 25$, we train the cov-MLR decoders with neuron subsamples of size $S = 6$ and $S = 7$. The resulting decoders have, on average, 24.5 free parameters, thus approximately matching the complexity of cov-LP, mean-LP and mean-MLR when trained on the full size reservoir. Each reservoir initialization is randomly subsampled 100 times for each S, so reported performance is not heavily dependent on a given subsampling (as it is averaged across 10 reservoir initializations, with a total of $2,000$ subsampling iterations). For $N = 50$ we use $S = 9, 10$, and for $N = 100$, $S = 13, 14$. For the pipeline without reservoir, we subsample the number of inputs following the same approach. Therefore, for the real dataset with 13 input features (Section 4.2.4) we use $S = 4$ and $S = 5$.

4.2.3 Synthetic datasets

This section introduces the synthetic datasets of multivariate time series whose 'information' relevant for classification is embedded in one of their statistics up to second order. We also consider a mixed scenario where the information is embedded both in means and zero-lag covariances, thus either of these statistics can be used for classification. Note that we use the term 'information' in a colloquial manner in this study, without a specific reference to information theory.

We consider a multivariate time series given by $u(t) \in \mathbb{R}^{\mathrm{M}}$ that represents the activity of $M = 10$ input nodes observed at discrete times $1 \leq t \leq d$ with $d = 20$. We rely on dynamical systems to enforce a specific spatio-temporal structure that constrains its statistics up to the second order, namely, the empirical mean activity and their zero-lag and one-lag covariance matrices, defined as follows:

- vector of mean activity $p \in \mathbb{R}^{\mathrm{M}}$, with elements $p_k = \langle u_k(t) \rangle_d = \frac{1}{d} \sum_{t=1}^{d} u_k(t)$;

- zero-lag covariance matrix $P^0 \in \mathbb{R}^{\mathrm{M \times M}}$, with elements

$$\begin{aligned} P^0_{kl} &= cov\left(u_k(t), u_l(t)\right)_d \\ &= \langle u_k(t) u_l(t) \rangle_d - \langle u_k(t) \rangle_d \langle u_l(t) \rangle_d \\ &= \frac{1}{d-1} \sum_{t=1}^{d-1} \left[u_k(t) - \frac{1}{d} \sum_{t'=1}^{d} u_k(t') \right] \left[u_l(t) - \frac{1}{d} \sum_{t'=1}^{d} u_l(t') \right]; \end{aligned}$$

- one-lag covariance matrix $P^1 \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$, with elements

$$\begin{aligned}
P^1_{kl} &= cov(u_k(t),\ u_l(t+1)_d \\
&= \langle u_k(t)u_l(t+1)\rangle_d - \langle u_k(t)\rangle_d \langle u_l(t+1)\rangle_d \\
&= \frac{1}{d-2}\sum_{t=2}^{d-1}\left[u_k(t) - \frac{1}{d-1}\sum_{t'=2}^{d} u_k(t')\right] \\
&\quad\times\left[u_l(t+1) - \frac{1}{d-1}\sum_{t'=1}^{d-1} u_l(t'+1)\right].
\end{aligned}$$

Here our goal is to classify the synthetic time series into $K = 2$ classes. We thus generate two groups of such time series according to one of the above structures, where the defining statistics —p, P^0 and P^1— correspond to distinct patterns that are randomly assigned to one of two classes (Fig. 4.2c). We first draw a number of such "reference" patterns by sampling given probability distributions, then we generate for each pattern several sample time series for our classification task (each sample involving further stochastic randomness). In each category, there are then two sources of "noise" or variability: the different patterns belonging to a same class, and the empirical noise due to the individual probabilistic realization of each sample. The rationale behind our choice is to account for empirical noise that is typically observed in real time series, such as speech sounds, where distinct phonemes have a similar spatio-temporal structure, which is altered at each pronunciation.

We use cross-validation to assess the classification performance, relying on a 70/30 train/test split that is maintained in all synthetic datasets. For each dataset, we evaluate how separable the two classes are in the relevant feature space by applying a multinomial logistic regression decoder (details in Section 4.2.2) directly on the input sample time series statistics(see bottom pathways in Fig. 4.1). The performance of this benchmark decoding is affected by the number of patterns to classify per class (intuitively, densely populated feature spaces are more likely to involve overlapping classes) and by the properties of the probability distribution the patterns are drawn from. Note that we focus on synthetic datasets with a benchmark classification accuracy below 100%, so we can detect performance improvements and decreases across different decoding schemes and models.

In the following subsections we describe the generative dynamics for each statistical structure.

Mean or first-order structure

We use the following generative process for the time series:

$$u(t) = p + z(t), \tag{4.10}$$

where $z(t) \in \mathbb{R}^{\mathrm{M}}$ is a normally distributed random variable, with zero-mean and identity covariance matrix. To characterize the mean activity of $u(t)$, we use a pattern vector $p \in \mathbb{R}^{\mathrm{M}}$, with non-null elements sampled from a zero-mean and unit variance normal distribution. This vector is created sparse, with a 0.1 density. This means that 90% of the input nodes will have zero-mean activity.

To generate a dataset of this type, we first draw 20 patterns of such p vectors and randomly split them in two classes (10 in each), as shown in Fig. 4.2c. Once defined the patterns for the two classes, we generate noisy samples by using Eq. 4.10 with 500 repetitions for each pattern. From each set of 500 samples, we use 350 samples for the training set and the remaining 150 for the test set.

Spatial covariance or second-order zero-lag structure

We use the following generative process for the time series:

$$u(t) = Wz(t), \tag{4.11}$$

where $z(t) \in \mathbb{R}^{\mathrm{M}}$ is, as before, a standard normal random variable and $W \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$ is a random sparse matrix with 0.1 density and non-zero elements sampled from a standard normal distribution. The resulting zero-lag covariance matrix is given by $P^0 = WW^T$ [53]. Note that the generated time series has zero-mean activity over time up to the empirical noise, as well as zero temporal correlations ($P^1 = 0$). Thus, the only discriminative information for the binary classification is in P^0 (or equivalently, W), which is the defining statistic.

To generate a dataset of this type, we sample 60 W matrices and randomly split them in two classes before simulating the dynamical processes. As before, we then generate 500 noisy samples with a 70/30 ratio for the train and test.

Temporal covariance or second-order one-lag structure

We use the following generative process for the time series:

$$u(t) = Wu(t-1) + z(t), \tag{4.12}$$

where we choose the mixing matrix $W = \exp(\beta \mathcal{I}_{\mathrm{M}\times\mathrm{M}} + J)$, with parameter $\beta < 0$, $\mathcal{I}_{\mathrm{M}\times\mathrm{M}} \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$ is the identity matrix and $J \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$ is an antisymmetric matrix. This guarantees that the time series will not differ neither in their mean activity vectors p (which are null) nor in their spatial correlation structure P^0 (which only depends on β), but only in their one-lag covariances $P^1 = WP^0$ [53].

To generate a dataset of this type, we sample 6 W matrices and randomly split them in two classes before simulating the dynamical processes to generate the noisy samples in the same manner as before. Without loss of generality, we set $\beta = -0.5$ and create the matrices J by sampling unsigned upper diagonal elements from the uniform distribution over $[0.5, 1)$. The elements' signs are randomly assigned, and the resulting J matrices have by construction a 0.3 density.

Mixed spatial inputs with first and second-order zero-lag structure

To create time series that differ in mean and spatial covariance structure, we use a superposition of the signals given in Eqs. 4.10 and 4.11:

$$u(t) = p + Wz(t). \tag{4.13}$$

To generate a dataset of this type, we randomly sample 20 W matrices and 20 p vectors. Each p is randomly paired to a W matrix and the tuple is randomly assigned to one of two classes. This is done with the purpose of having mean patterns and covariance patterns that are evenly separable. Afterwards, the dynamical processes are simulated. Note that p and W are generated with the same density and normal distribution for their non-zero elements, as in Sections 4.2.3 and 4.2.3 respectively.

4.2.4 Spoken Arabic digits dataset

To test our covariance-based decoding applied to reservoir computing in a real application, we work with the spoken Arabic digits dataset [176, 177, 178]. The motivation is to use time series with spatio-temporal structures. The dataset contains 8800 multivariate time series (10 digits x 10 repetitions x 88 speakers) recorded from native Arabic speakers (44 females, 44 males, ages 18-40 years old) with the purpose of classification, split in a training (75%) and test set (25%). Here the classification is not binary, but there are $K = 10$ classes (one per digit).

The time series are represented by 13 Mel Frequency Cepstral Coefficients (MFCC) [181], which constitute a widely used feature for tasks such as speech recognition [182]. They mimic the transformation of the audio signal by the inner ear and are a model of how sound stimuli are "perceived" by the early neuronal auditory system. As represented in Fig. 4.2d, when a mechanical sound wave reaches the ear, it produces vibrations that propagate throughout the cochlea, with high frequencies entraining the early part of the cochlea and low frequencies the end part. Hair cells in the cochlea translate these vibrations into electrical activity in a frequency dependent manner (depending on their position), so the sound is spectrally decomposed. This is performed by means of the Fourier transform in the MFCC computation. Afterwards, the spectrum is represented using a logarithmic scale (mel scale), emulating the nonlinear perception of pitch. Finally, a discrete cosine transformation is applied with the purpose of decorrelating the resulting coefficients.

Since the MFCC sequences vary in length (5-92 elements, median 40), as is natural in speech, we opt to shrink or expand them (through zero-padding) to have the same size and a consistent observation window

($d = 50$). This is strictly not necessary since all our classification methods can operate on varying sequence lengths, but is only motivated to make the implementations more straightforward and does not make the classification problem easier. Another possibility to avoid losing the information about sequence length, which strongly relates to digit identity, is to add it as a normalized constant input to the reservoir [108], but we choose to not follow this approach.

4.3 Results

The purpose of our study is the comparison between mean-based and (co)variance-based linear perceptron readouts applied to a reservoir of sigmoid neurons for the classification of multivariate time series. To do so, we firstly consider synthetic time series with controlled structures, characterized by either their means, spatial covariances (zero-lag) or temporal covariances (one-lag) (see Section 4.2.3).

To determine the usefulness of the reservoir in the pipeline for classification, we also compute baseline classification accuracies (i.e. without reservoir, 'NO-RES' pathway in Fig. 4.2a) for each perceptron readout type. Furthermore, we use additional multinomial logistic regression decoders (mean-MLR and cov-MLR) to quantify in terms of their accuracy how 'class-informative' the noisy statistics of the time series that reach the perceptron readouts are. We remark that the purpose of the MLR decoders is merely to provide a reference, and that these decoders are operationally different from the perceptron. The later receives time series as input and likewise produces time series as output, with class 'information' embedded in its statistics. MLR instead receives pre-computed statistics as input features and directly outputs a single static class-probability vector (Fig.4.1b and see Section 4.2.2 for further details).

We systematically vary reservoir parameters to allow for the identification of which properties of the reservoir (size, spectral radius, leak rate) are important to extract the relevant information for classification, in line with previous work that used mean-based readouts with similar reservoirs [97, 174, 208, 209]. In addition, we explore the influence of the reservoir connectivity by comparing fully connected random reservoirs and segregated reservoirs where input-receptor neurons (receptors) and output-sending ones (feeders) are separated by at least one neighbor (Fig.4.2b and see Section 4.2.1) for the decoding of inputs with different structures.

Last, we apply our analysis under the same considerations of the first part of our study to real data for speech recognition (see Section 4.2.4), which is a practical problem where reservoir computing has been efficiently applied [108, 119, 128, 184, 185, 186, 187, 188, 189].

4.3.1 Reservoirs enhance covariance perceptron performance and efficiently represent second-order statistics from input time series

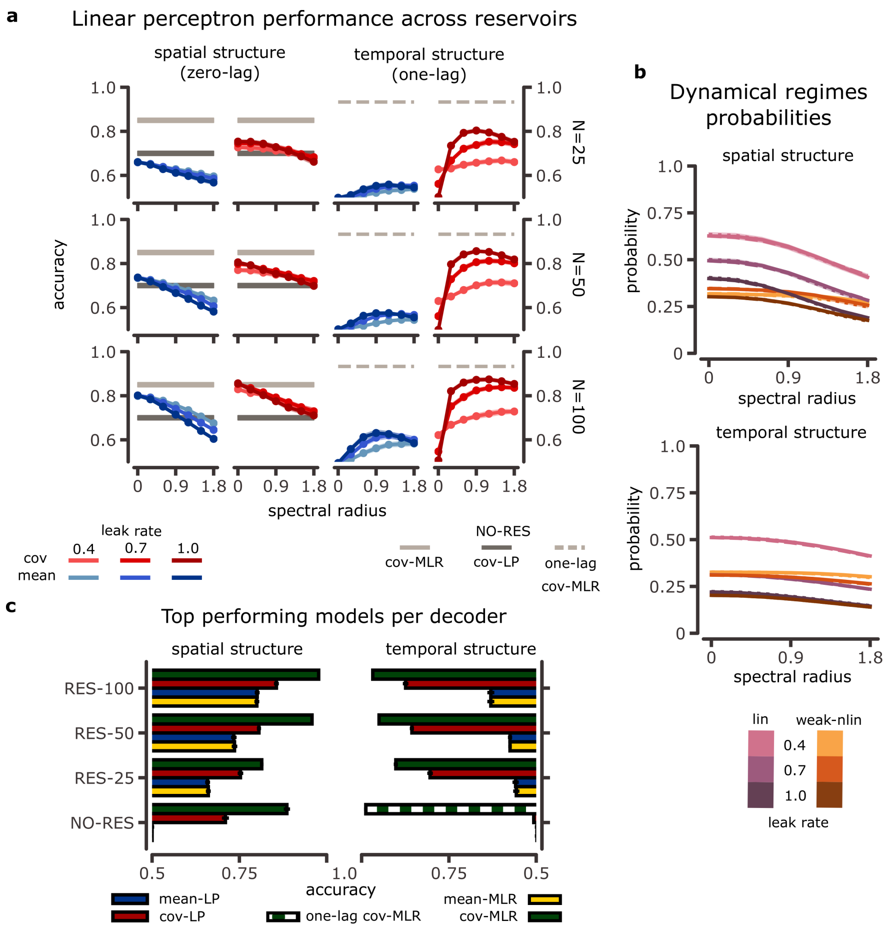


Figure 4.3: Decoding performance for spatial and temporal structure and reservoir dynamics. **a:** Reservoir classification performance for a spatial structure (left panel) and a temporal structure (right panel) embedded in the input time series when the decoder is a linear perceptron. Accuracy is shown as a function of spectral radius (on the x-axis) and for different reservoir sizes (see N) and leak rates (indicated by the various contrasts). We compare mean-based decoder (mean-LP in blue) and covariance-based decoder (cov-LP in red). The light-gray lines on each subplot indicate the performance of a MLR classifier directly applied to the statistics (see bottom pathways in Fig. 4.1) that characterizes the

Figure 4.3 *(previous page)*: information embedded in the input time series (zero-lag covariances for the left panel and one-lag covariances for the right panel). The dark gray line shows the performance of a covariance perceptron directly applied to the input time series (top pathways in Fig. 4.1). Shaded areas represent ± 1 standard error of the mean (sem) across 10 different simulations of time series and reservoir configurations. Each class has 30 different covariance patterns in the left panel, and 6 in the right panel. **b:** Probability of finding a reservoir neuron in the linear dynamics regime (see Section 4.2.1) (pink) or in the weakly nonlinear regime (orange) versus spectral radius, for different reservoir sizes (overlapping dotted, dashed and solid lines for $N = 25$, $N = 50$ and $N = 100$ respectively) and leak rates (shades) and input structures (spatial, top; temporal, bottom). Results are averaged across 10 different data simulations and reservoir configurations. Shaded areas are ± 1 sem. **c:** Accuracy for the best models for each decoder type: blue for mean-LP, red for cov-LP, yellow for mean-MLR and green for cov-MLR. Error bars represent ± 1 sem across 10 different simulations of time series and reservoir configurations. Numbers in model names indicate the size of the reservoir (RES-N, $N = 25$, 50, 100 as above, see the bottom pathway in Fig. 4.2a), while NO-RES indicates that the input time series is directly fed to the decoder (the bottom pathway in Fig. 4.1a). In the right panel, we also include the performance of MLR on one-lag covariances (one-lag cov-MLR) for the NO-RES case, since the other features are not informative in this case.

The linear perceptron readout (mean-LP and cov-LP in Fig. 4.1b) is trained to perform a binary ($K = 2$) classification task of synthetic input time series that differ by their second-order statistics. We create these time series in such a way that the statistical features that determine to which of the two classes they belong to is embedded either in their zero-lag covariances (spatial structure, Section 4.2.3 in Methods) or in their one-lag covariances (temporal structure, Section 4.2.3 in Methods).

For the spatial structure where the categories differ by the zero-lag covariance patterns, we first train mean-LP and cov-LP readouts without the reservoir in the pre-processing stage. As expected, mean-LP operating directly on the input time series does not produce above chance classification accuracy, since the inputs have, by construction, zero-mean activity (up to some observation noise). On the other hand, cov-LP achieves an accuracy of $71 \pm 2\%$. While this accuracy is much better than chance, the decoder is not able to fully extract the second-order information embedded in the input time series covariances, as quantified by the cov-MLR performance ($88.6 \pm 0.9\%$).

We add the reservoir to the pipeline and vary its parameters (namely size, leak rate and spectral radius) to assess how they influence the classification performance (Fig. 4.3a, left panel). For the classical mean-based perceptron readout (in blue), we find similar results to previous work on reservoir computing applied to others tasks (e.g. types of inputs): performance monotonically increases with the number of units forming the reservoir. This is due to the increased dimensionality of the representation of the inputs in the reservoir activity, which makes it easier to find a separating hyperplane for the two categories of inputs. The new approach with the cov-LP readout (in red) displays the same trend. Furthermore, the reservoir can boost the decoding performance beyond that of a cov-LP directly applied to the inputs (dark gray line) with as little as $N = 25$ neurons (half the amount needed by mean-based readouts) and it even reaches the performance of the MLR directly applied on input covariances (light gray line) for $N = 100$. Note, however, that the number of trained weights per class is then equal to $N = 100$ for the cov-LP, whereas it is equal to $10 \times 11/2 = 55$ for the (cov)MLR-NO-RES (see Section 4.2.2). For both readout types (mean/cov), we find that the performance decays with spectral radius, the best being achieved when the reservoir is a feedforward layer where $\rho(\Omega^{\text{res}}) = 0$. This points to interesting relationships between reservoir dynamics and representations. Indeed, it can be shown that when the neurons inside the reservoir behave linearly, then mean-based reservoir representations are not able to classify spatially structured inputs at the second-order, while covariance-based ones can (see Section 4.2.1). However, when neurons are driven in a weakly nonlinear regime, mean-based representations become possible. On the other hand, when a reservoir is excited at a strongly nonlinear regime (i.e. saturating the nonlinearity), it will provide representations (at both orders) that are degraded when compared to the weakly nonlinear case.

In fact, a neuron saturating the nonlinearity can only behave in three possible ways: continuously saturating the nonlinearity at the top limit, at the bottom one, or alternating from one to the other. Neurons constantly saturating the same limit will have average state equal to 1 or -1, while flipping neurons will have mean states that depend on the switching probability, with higher probabilities more likely leading to zero-mean states that degrade the input representation. This degradation, nonetheless, is expected to be more prominent in the mean than the covariance space, since saturated neurons can display coordinated switching behavior.

Succinctly, when the goal is to process input spatial covariances, randomly mixing inputs (via Ω^{in}) and applying a point-wise nonlinearity is enough to achieve this, while keeping a memory of past states through reservoir dynamics appears detrimental. Indeed, strong recurrent connections within the reservoir drive it away from the linear/weakly nonlinear regime (Fig. 4.3b, top) and the corresponding accuracy drops as representations lose quality by becoming sparser. In line with this, performance degrades more slowly for the cov-LP than the mean-LP, although the effect becomes similar as reservoir size increases.

We also observe that cov-LP performance is largely insensitive to changes in the leak rate, as shown by the overlapping red lines in Fig.4.3a (left panel). On the other side, mean-LP performance degrades when increasing the leak rate for reservoirs whose spectral radius is close to or larger than 1, and this effect increases with reservoir size. Thus, the leaky integration mechanism becomes useful when using mean-based readouts (smaller α values yield better performance). Indeed, leaky integration drives the reservoir away from the nonlinear regime, therefore improving representations (see profiles in 4.3b with varying leak rate).

Then, we consider the temporal structure signal (Fig. 4.3a, right panel, see Section 4.2.3 in Methods for details) for which the input time series from the two classes only differ by their one-lag covariances, their spatial covariances being identical. For these type of input structure, mean-LP and cov-LP alone (i.e. without reservoir) cannot capture the relevant statistics for classification, so they produce chance level accuracy. The reservoir thus becomes fundamental for this task. As with the spatial structure, we find that performance increases with reservoir size for both readout types. However, the performance for the covariance-based decoders is well above that of mean-based ones and approximates that of the MLR directly applied to the one-lag covariances of the inputs ($98.8 \pm 0.5\%$, dashed light gray line). Thus, the conversion from temporal second-order patterns in the inputs to spatial second-order patterns in the reservoir is more efficient than to spatial first-order patterns. The temporal structure yields different accuracy dependencies on the spectral radius and leak rate, as compared to the spatial structure. First, we note that mean-based readouts perform very poorly for all tested configurations and that both decoders achieve their best performance for non-zero spectral radii. This indicates that the reservoir recurrent dynamics are essential to transform the input lag covariances into output spatial statistics of the reservoir activity, of either first or second order. The optimal reservoir configurations have recurrent connectivity with $\rho(\Omega^{\text{res}}) \approx 1$, which have also been shown to maximize memory capacity for Gaussian inputs [101, 174]. In those studies, the reservoir transforms dynamic signals in a way that allows to retrieve past information for a given range of delays. Instead, we here do the converse and transform the temporal structure (lag covariances) into a spatial structure (zero-lag covariances). For a reservoir in a linear or weakly nonlinear regime, indeed it can be shown that optimal representations at both orders are obtained when $\rho(\Omega^{\text{res}}) \to 1$ from below (see Section4.2.1 in Methods). Nonetheless, as these temporally structured inputs produce reservoir nonlinear dynamics that are stronger than the ones induced by spatially structured inputs (Fig. 4.3b, bottom), representation degradations seem much more pronounced at the first-order than the second-one, explaining the larger gap in accuracy between mean and covariance based decoders in Fig. 4.3a.

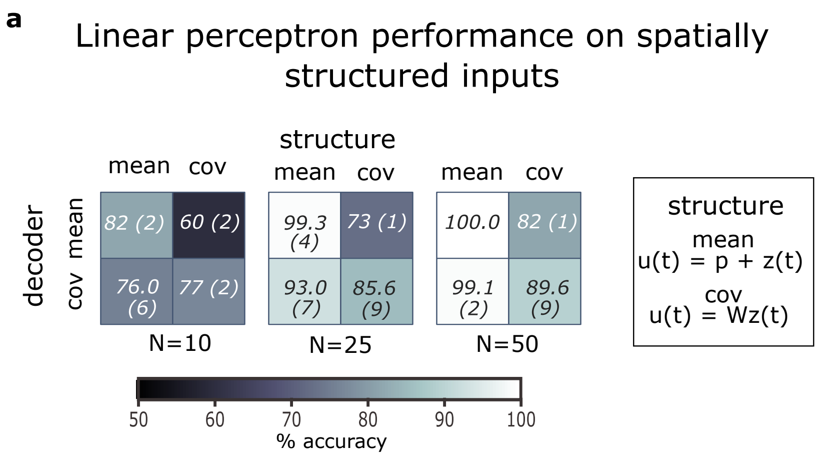
Second, accuracy increases with leak rate for both readout types. Nonetheless, for covariance readouts in a feedforward reservoir with $\rho(\Omega^{\mathrm{res}}) = 0$, leaky integration is key, as it allows neurons to keep a memory of past inputs in their current state.

We further compare these LP decoders to the MLR decoders, in both mean-based and covariance-based versions as illustrated in Fig. 4.3c. As before, we focus on the case where a reservoir is involved in the classification pipeline (RES-N, corresponding to the bottom pathway in Fig. 4.2a), as well as the decoders directly applied to the inputs (NO-RES, see the top pathway in Fig. 4.2a). The best decoder performance for each case in Fig. 4.3a across the considered reservoir parameters is displayed in Fig. 4.3c. It can be seen that covariance-based decoders outperform mean-based ones when the information to extract is in the second-order statistics of the inputs across all models tested, and that the best decoder is the cov-MLR, which is also the readout with highest complexity (number of parameters).

Ultimately, temporal and spatial input structures are efficiently processed by reservoirs, but distinct characteristics give the best performance (especially the radius). A good compromise for both types of inputs and covariance-based readouts is a reservoir without leaky integration and medium-sized spectral radius, namely $\alpha = 1$ and $\rho(\Omega^{\mathrm{res}}) \simeq 1$.

4.3.2 Reservoirs with covariance-based readouts can also extract first-order statistics

Thus far, we have shown that a consistent processing scheme between inputs and outputs via the reservoir for covariance-based information processing is better than a mixed scheme that maps input covariances to means in the reservoir activity, which are then used by the readouts for decoding. Now, we explore for comparison the case where the inputs have embedded information in their mean activity. When class information is embedded in spatial statistics, we have shown analytically and numerically that a pointwise nonlinearity is key to produce reservoir representations shaped at first and second order, while the recurrent dynamics are less important. Thus, we restrict this investigation to a feedforward reser-



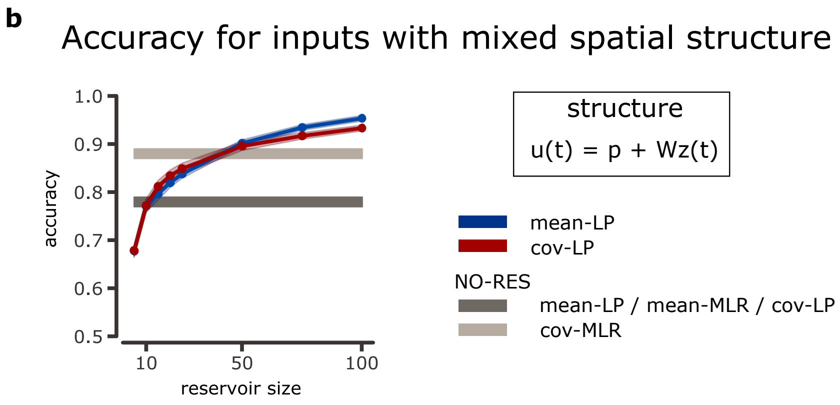


Figure 4.4: Decoding mixed information for the mean structure and spatial structure using feedforward reservoirs. **a:** Accuracy table for feedforward reservoirs without leaky integration coupled to a linear perceptron when using mean or covariance readouts to classify time series characterized by its mean or its spatial covariance structure. We focus on small reservoirs ($N = 10, 25, 50$), since the mean structure task gives perfect performance for such reservoir size ($N \geq 50$). Each entry in the table is the mean accuracy (with sem in parentheses) across 10 different simulations. Datasets were designed to match the performances using the perceptron as readout: cov-LP $79(\pm 2)\%$, 15 patterns per class, mean-LP $81(\pm 2)\%$, 10 patterns per class.

Figure 4.4 *(previous page)*: **b:** Classification accuracy as a function of reservoir size for a feedforward reservoir without leaky integration coupled to a linear perceptron as a readout when the categories of input time series differ by both their means and their spatial covariances. As before in Fig. 4.3a, the performance for the mean-LP is represented in blue, that of the cov-LP in red. Shaded areas represent ± 1 sem across 10 different simulations and gray lines indicate the performance of decoders that are directly applied to the inputs (NO-RES). Note that the datasets are generated in such a way that the classification performance of a mean-LP and cov-LP without reservoir is matched, equal to $78 \pm 1\%$, with 10 patterns of each statistic to distinguish per class (see Section 4.2.3 in Methods).

voir and no leaky integration, namely $\rho(\Omega^{\mathrm{res}}) = 0$ and $\alpha = 1$, as they give the best performance in Fig. 4.3A. Note that the leak rate is less crucial than the absence of recurrent connectivity here.

We firstly compare the performance of mean-LP and cov-LP decoders when the input information is embedded in two different statistical orders: either means (input dynamics in Section 4.2.3) or spatial covariances (input dynamics in Section 4.2.3). We intentionally work with small reservoirs to avoid the case where the mean classification task becomes trivial, with perfect performance. To fairly compare the two structure schemes, we create the respective datasets such that the performance of a LP classifier acting on the distinctive input statistics (either mean or covariances) is matched up to 1% of performance. Our results in Fig. 4.4a indicate that, typically, the best strategy is to use for decoding the same order the input is structured at, for small reservoirs. However, when the reservoirs are big enough ($N \geq 50$), switching to mean-based decoding can be beneficial, as learning is computationally faster.

Then, we consider the situation where the input time series can be categorized by either of their first/second-order statistics, corresponding to the dynamics in Section 4.2.3. In this setting, knowing only one of these statistics is enough to perform the binary classification, and the question is whether a type of decoding can make use of both types of information in a synergistic manner. As before, the inputs are designed to match decoding performances when LP is applied directly to them. We find that both decoders perform equally well across various reservoir sizes (Fig. 4.4b), with a slight advantage for cov-LP over mean-LP for smaller sizes and conversely for larger sizes.

These results are consistent with the fact that in a feedforward reservoir, if input-to-reservoir weights are $\mathcal{O}(\varepsilon)$, with $|\varepsilon| < 1$, then the statistical moments of reservoir neurons will depend on input statistics with different leading orders in ε (see Methods 4.2.1). We have shown that reservoir mean activity has $\mathcal{O}(\varepsilon)$ dependence on input mean activity and $\mathcal{O}(\varepsilon^2)$ dependence on input spatial covariances. On the other hand, reservoir spatial covariances display the same $\mathcal{O}(\varepsilon^2)$ dependence on both input statistics. Thus, if information is on input means, it is more strongly reflected on reservoir means, and likewise for covariances. However, when information is embedded in both statistical orders, then fixing a representation for the reservoir settles the other one as 'noise'. Therefore, in a mixed scheme, mean-based representations have a better signal-to-noise ratio than covariance-based ones.

Together, these results show that the application of covariance-based decoding, when combined with a reservoir, goes beyond that of its same order encoding: information about multiple input statistical orders can be effectively mapped to output second-order moments.

4.3.3 Covariance-based information is more efficiently propagated in short time scales

As a last point with the synthetic data, we consider a reservoir with separated input receptor and output feeder units (Section 4.2.1) to study the role of the reservoir topology in the transmission of different statistical orders (Fig. 4.2b). In this architecture motivated by biology with segregated functions, signals have to propagate from end-to-end of the reservoir to reach the decoder in a short-time window $d = 20$. We compare segregated reservoirs with different sparse connectivities (Fig.4.2b), as well as the architecture studied until now with full connectivity from inputs and to

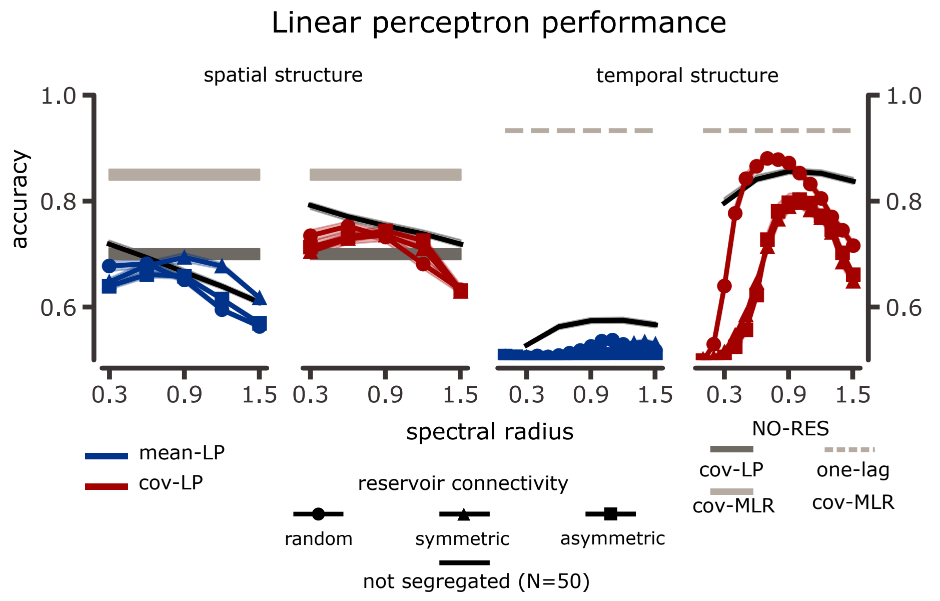


Figure 4.5: Segregated reservoir (Fig 4.2b) classification performance of spatial structure (left panel) and temporal structure (right panel) embedded in the input time series when the decoder is a linear perceptron. Accuracy is shown as a function of spectral radius and for different reservoir topologies (marker styles). Color coding is similar to Fig. 4.3a. The black lines show the performance of a non-segregated reservoir of 50 neurons without leaky integration ($\alpha = 1$, same as in Fig. 4.3a, middle row) for comparison.

outputs. Importantly, we design these configurations such that they share the same number of decoder resources, as given by the number of connections from reservoir units to output units (Fig. 4.3a, middle row, $N = 50$). The leak rate is always set to $\alpha = 1$. Since the density of within reservoir connections does not affect performance in the full connectivity configuration (it is only important that it is non-null, see Supplementary Material Fig. 4.8), we set the reservoirs to have 0.1 density.

For the task of decoding input spatial covariances, we observe that the performance is improved for mean-decoding (Fig. 4.5a, left panel, blue) with large spectral radii when the segregated reservoir has symmetric structure. On the other hand, covariance-based decoding is only reaching the not segregated reservoir performance for medium radii ($\rho(\Omega^{\text{res}}) = 0.9$), where the segregation of receptor and feeders impedes that this task be achieved by a feedforward reservoir.

When the information to decode is in the temporal covariances (Fig. 4.5a, right panel), mean-LP performance is degraded. Covariance decoding, on the other hand, is better preserved and even reaches a better performance than a fully connected random topology for $\rho(\Omega^{\text{res}}) \approx 0.7$. For symmetric and asymmetric connectivities, accuracy is significantly degraded across all spectral radii when compared to the not segregated reservoir, and we also find that the relationship between optimal performance for these topologies and spectral radius changes. This suggests a complex interplay between the input structure and the reservoir topology, which would be interesting to explore in a more analytical manner in future work.

On average, information about input covariances is better recovered by covariance decoders even when the patterns must propagate along a network to reach the output units in a short time window.

4.3.4 Covariance-based schemes work in classification of spoken digits

Finally, we explore the applicability of covariance-based readouts in reservoir computing for the classification of spoken Arabic digits [176, 177,

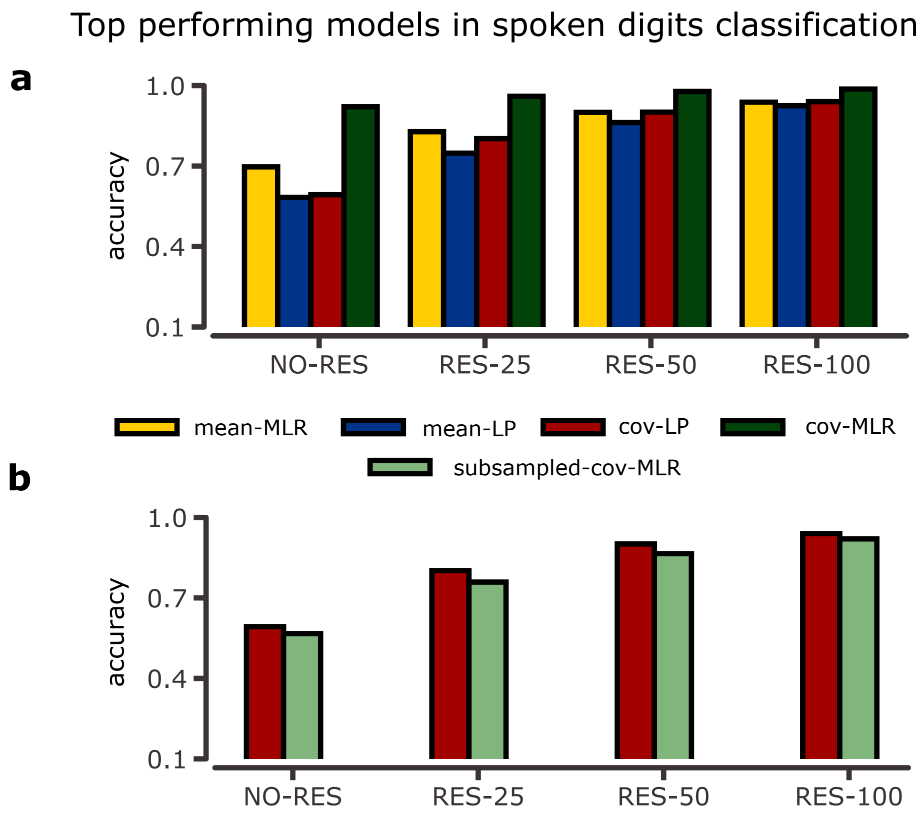


Figure 4.6: Decoding performance in spoken Arabic digits classification. **a:** Accuracy for the best performing models of each size, distinguished by decoder type. Colors and model label conventions are the same as in Fig. 4.3b. Results are averaged across 10 different reservoir instantiations. Error bars displaying ± 1 sem are also included. **b:** Accuracy for the best performing models with varying reservoir size, when the decoder is covariance-based (cov-LP and cov-MLR) and the number of parameters of each decoder are approximately matched. Error bars displaying ± 1 sem are also included.

178]. The dataset consist of digits $0-9$, represented by 13 input nodes obtained after some preprocessing mimicking the cochlea as illustrated in Fig. 4.2d (see Section 4.2.4 in Methods for further details). Our goal here is not so much to improve the best performance obtained so far on this dataset. Rather we aim to evaluate our covariance-based decoding in a real case study and speech recognition is a common area of application for reservoir computing, given the sequential nature of the task [119, 125, 128, 184, 185, 186, 187, 188, 189]. Indeed, the classical mean-based decoding reaches a competitive performance of 99.9923% for a reservoir of size bounded by $N = 1,000$ [129], and it can be further increased to 99.9945% when using a predictive model space representation instead of the natural reservoir space [129].

As before, we use the two pipelines for classification, with and without reservoir (same as Fig. 4.1a) and we vary reservoir properties (size, spectral radius and leak rate) across a grid. For brevity, we only report the test accuracy of the best performing models for each reservoir size and decoder in Fig. 4.6a, which are obtained for $\alpha = 0.2$ and $\rho(\Omega^{\text{res}}) = 1.2$ (see Supplementary Material Fig. 4.9 for all results). This is in line with decoding temporal structure (Fig. 4.3a), rather than spatial structure, suggesting that the temporal structure of the real data is best captured by the reservoir to perform efficient classification. First, we note that when we do not use a reservoir, the best decoders are the nonlinear ones (mean-MLR and cov-MLR), and that both statistical orders contain relevant information to classify the time series. However, the higher accuracy obtained with the cov-MLR indicates that the covariance patterns are potentially easier to extract than the mean patterns. Indeed, the performance of cov-MLR with $N = 100$ is $98.7 \pm 0.1\%$, better than using a mean-based decoder with an echo state network of $N = 900$ neurons (96.91%[127]), but obtained with 9 times fewer neurons within the reservoir. Furthermore, the performance is at the same level of that obtained with a much more complex network model, as is the long short-term memory network in [190] (98.77%).

In addition, the linear decoders (mean-LP and cov-LP) perform well above chance level, which corresponds here to 10% because the dataset is balanced across the 10 possible digits. The use of a reservoir in the pipeline is beneficial to all decoders, and the resulting performance increases with reservoir size. When $N = 100$, the performances of the mean-MLR, mean-LP and cov-LP are closely matched, but they are still below that of the cov-MLR. Our insight is that this gap could be further reduced by increasing the reservoir size.

To further compare the cov-LP and cov-MLR, we subsample the number of neurons in the reservoir, so that its vectorized covariance matrix has dimensionality close to N, instead of $N(N+1)/2$ (see Section 4.2.2 for details). We find that both models exhibit similar performance under this constrain (see Fig. 4.6b), which confirms that the cov-LP decoder extracts information about covariances in an optimal manner given its limited number of resources.

All in all, covariance-based decoders can be successfully applied within reservoir computing frameworks to classify spoken digits. The covariance perceptron (cov-LP) applied to reservoir computing offers a good compromise between good performance and limited resources as in a biological context.

4.4 Discussion

In this study, we have explored how a neuronal reservoir can be efficiently paired with covariance-based readouts for the classification of time series. Our goal was to investigate the potential of this new type of decoding as compared to the classical mean-based decoding that has been used with reservoir computing until now, with a two-fold motivation. First, we aimed to characterize how reservoir dynamics can process the statistical structure embedded in the input time series, up to the second order including spatial and temporal structures. Second, we wanted to compare a biologically inspired configuration —a perceptron network mapping time series to time series— with a machine learning configuration —multinomial logistic regression mapping static features to class probabilities. Our results demonstrate the efficiency of covariance-based readouts applied to reservoirs, even in the biological configuration that involves limited resources as implemented by a (linear) covariance perceptron.

We have shown using synthetic data that covariance decoding allows for capturing a broad diversity of input structures after their transformation by the reservoir. To do so, a compromise configuration for our echo state networks, corresponding to $\alpha = 1$ and $\rho(\Omega^{\mathrm{res}}) = 1$, is robust to variations in second-order input structure. These findings are confirmed with the classification of spoken digits from a real dataset, for which we find better performance for covariance decoding: the performance of cov-MLR with $N = 100$ ($98.7 \pm 0.1\%$) is better than the mean decoding with $N = 900$ (96.91%) [128, 210]. Moreover, the cov-LP maximally extracts information about covariances given its restricted resources. For the real digits dataset, the best radius is similar to the compromise configuration for synthetic data, but the best leak rate is smaller. This is in line with previous work in reservoir computing studies with mean-based readouts, which suggest that small leak rates better suit the intrinsic time scale of the input time series, as speech spectral features vary slowly when compared to the sampling frequency (i.e. the spacing between the windows used to compute MFCC) [121]. In such a case, the usefulness of the reservoir is to transform signals at those slow timescales into zero-lag correlations of the reservoir activity.

In more detail, the collective dynamics of the reservoir, as governed by the interplay between the spectral radius and the leak rate, have a crucial effect on the input-output mapping in terms of statistics. For large radii, small leak rates produce slower dynamics that tend to drive the reservoir towards the linear or weakly non linear regimes, which in turn enhances mean decoding of spatial structure. When inputs are endowed with spatial structure, covariance decoding is less dependent on leak rate, which suggests that reservoir pairwise correlation patterns mostly depend on global dynamic features. On the other hand, mean decoding of temporal structure performs very poorly and is negatively affected by slow unit and global dynamics, as information from the past must rapidly produce spatial patterns distinguishable by the mean decoder. For covariance decoding when the information source are the one-lag covariances, the global memory mechanism given by the spectral radius and the individual one given by the leak rate operate best when acting alone, as we find that the presence of slow unit dynamics decreases performance when the reservoir is scaled to operate close to the unstable regime ($\rho(\Omega^{\text{res}}) \approx 1$). The optimal reservoir in this case is one without leaky integration and $\rho(\Omega^{\text{res}}) \approx 1$, which is balanced between the linear, weakly nonlinear and nonlinear regimes. Thus, driving the system closer to the linear regime by decreasing the leak rate (and effectively reducing the strength of the recurrent connections for a given spectral radius) degrades performance. While most analytical studies in the reservoir computing literature focus on the linear approximation [97, 101, 108, 125], our numerical results suggest that other input-induced dynamical regimes should be further examined theoretically [102, 103].

Early studies in echo state networks failed to report performance improvements in time series prediction when using small-world or scale-free topologies [211, 212]. However, it was later shown that connectivity plays a role when the reservoir network displays cortex-like topological properties [122, 207], a finding that had been observed in reservoirs of spiking neurons earlier [213]. In those studies as replicated in ours, nodes receiving inputs differ from those feeding the readouts, thereby mimicking the separation of sensory areas from motor areas. This scheme is most efficient to transform input temporal covariances into output spatial covariances when the reservoir connectivity is fully random; note that in this case the optimal spectral radius is smaller than for the non-segregated reservoirs ($\rho(\Omega^{\text{res}}) = 0.7$ instead of 1). This could be due to the overall bigger reservoir size that provides a faster mixing of the inputs ($N = 500$). Surprisingly, symmetric connectivity enhances mean decoding of spatial structure. Overall, there is a nontrivial interaction between reservoir topology and input structure which should be further investigated. We note that future work could explore more elaborate topological properties, like small-world, scale-free, clustered [90] or even real connectome patterns [214, 215], beyond the simple traits studied here. Furthermore, we constrained our study to random input projections, which take part in shaping the input representations that arise in the reservoir (as hinted in Section 4.2.1). Along this line, it has been shown that unsupervised plasticity at the input-to-reservoir layer improves performance in pattern recognition tasks with mean-based decoding [90], so it is natural to question whether this effect is also observed, or even further enhanced, for covariance-based readouts.

Last, we stress that the use of the reservoir here offers several advantages with respect to the linear network studied in [53]. First, it avoids the training of recurrent connections in the readout layer to classify temporal covariance structures, by retaining past information in its own activity. Thus, learning is computationally cheaper, as there is no need to numerically solve Lyapunov equations, which is the case in the recurrent covariance perceptron [53]. Second, the cross-talk among statistical orders induced by the reservoir allows the covariance perceptron to capture a broader variety of input statistics, in particular when the input information is embedded in the first statistical order. Meanwhile, the neuronal system remains biologically plausible and takes advantages of the interplay between nonlinearities and recurrent connectivity [98, 116]. Although we have not examined the influence of the nonlinearity used in the reservoir, we expect a variety of biologically inspired functions to lead to efficient computations, provided they keep the recurrent dynamics under control for medium radii (i.e. bounded activity). Our work may also bring a novel perspective in training recurrent networks with feedback in the line of the FORCE algorithm, where the focus is on generating patterns (time series) that consist of trajectories [105, 216, 217]. A possible extension is to generate time series with desired covariance-based patterns instead.

Another extension in the direction of biological realism is to transpose the scheme to spiking neurons [98]. Following from our results when our analogue reservoir is close to a linear regime, we expect our covariance-based decoding framework to give interesting perspectives in terms of operations on covariances for spiking networks in similar linear regimes. However, the specific nonlinearities involved in spiking neurons should have significant effects on the input-output mapping and require a thorough study. Such covariance-based learning for spiking neurons would be an intermediate between learning spike rate patterns and precise spike trains, as with the 'tempotron' [218] or ReSuMe [219].

An important limitation of our study has been on the size of the reservoirs we were able to implement, as the perceptron learning procedure for covariances becomes numerically unstable as the number of parameters to tune increases. Current work on overcoming this and studying the scaling behavior of covariance-based reservoirs is focused on using techniques such as gradient-clipping [96]. Nonetheless, this approach makes the learning procedure slow as the number of optimization steps needed to find a good solution increases. Another issue to also address in the future is how to efficiently regularize covariance-based decoders, since going for larger reservoirs might lead to overfitting. These considerations underlie a cost-benefit trade-off between covariance and mean-based representations. While covariances offer higher-dimensional representational spaces than means for the same number of resources, learning is still computationally cheaper and more stable for mean-based representations [7, 53].

4.5 Supplementary Material

4.5.1 Reservoirs propagate diverse input statistics

Let $u(t) \in \mathbb{R}^M$ be a multivariate time series fed to an echo state network, with $u_1(t)$ a bias unit. The update equations for an arbitrary neuron $x_i(t)$ inside a reservoir with N units are given by:

$$\begin{aligned} \bar{x}_i(t) &= \mathcal{F}\left(\sum_{m=1}^{M} \Omega^{\text{in}}_{im} u_m(t) + \sum_{l=1}^{N} \frac{\rho}{\gamma} \Omega^{\text{res}}_{il} x_l(t-1)\right), && (4.14) \\ x_i(t) &= (1-\alpha)\, x_i(t-1) + \alpha \bar{x}_i(t), && (4.15) \end{aligned}$$

where the function $\mathcal{F} = \tanh$ has a sigmoidal profile, $\Omega^{\text{in}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ is the connection matrix from the input time series to the reservoir units and $\Omega^{\text{res}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ is the weight matrix of recurrent connections within the reservoir. Factor $\frac{\rho}{\gamma}$ allows to control the spectral radius of the effective matrix of recurrent connections $\Omega^{\text{eff}} := \frac{\rho}{\gamma}\Omega^{\text{res}}$, where γ is the spectral radius of Ω^{res}.

In a fully leaky reservoir ($\alpha = 1$) and assuming a left-infinite sequence of inputs is presented to the reservoir, then the state at time t of a neuron can be written as

$$x_i(t) = \tanh\left(\sum_{m=1}^{M}\sum_{k=0}^{\infty}\left(\left(\frac{\rho}{\gamma}\Omega^{\text{res}}\right)^k \Omega^{\text{in}}\right)_{im} u_m(t-k)\right) \tag{4.16}$$

If $\alpha = 1$ and neurons are in a weakly nonlinear activation state, we can approximate Eq. 4.16 by a Taylor series truncated at the third order:

$$\begin{aligned} x_i(t) \approx & \sum_{m=1}^{M}\sum_{k=0}^{\infty}\left(\left(\frac{\rho}{\gamma}\Omega^{\text{res}}\right)^k \Omega^{\text{in}}\right)_{im} u_m(t-k) \\ & - \frac{1}{3}\sum_{m,r,s=1}^{M}\sum_{k,l,n=0}^{\infty}\left(\left(\frac{\rho}{\gamma}\Omega^{\text{res}}\right)^k \Omega^{\text{in}}\right)_{im}\left(\left(\frac{\rho}{\gamma}\Omega^{\text{res}}\right)^l \Omega^{\text{in}}\right)_{ir} \\ & \times \left(\left(\frac{\rho}{\gamma}\Omega^{\text{res}}\right)^n \Omega^{\text{in}}\right)_{is} u_m(t-k)u_r(t-l)u_s(t-n), \end{aligned} \tag{4.17}$$

where we have collapsed the sums $\sum_{m,r,s=1}^{M} = \sum_{m=1}^{M}\sum_{r=1}^{M}\sum_{s=1}^{M}$ and $\sum_{k,l,n=0}^{\infty} = \sum_{k=0}^{\infty}\sum_{l=0}^{\infty}\sum_{n=0}^{\infty}$ for ease of notation and have used that $\tanh(z) \approx z - \frac{z^3}{3}$ when $z \approx 0$.

In the following, we will assume:

- Ω^{in} elements are independently sampled from the uniform distribution over $[-\varepsilon/M, \varepsilon/M]$
- Ω^{res} elements are independent and identically distributed random variables with zero mean and variance $\sigma = \frac{1}{N}$, and thus in the limit $N \to \infty$, $\gamma \to 1$ [220].

We want to observe how reservoir first and second-order statistics are influenced by input statistics. Therefore, note that:

- In magnitude, elements in $\frac{\rho}{\gamma}\Omega^{\text{res}}$ follow $\left(\frac{\rho}{\gamma}\Omega^{\text{res}}\right)_{pq} \sim \frac{\rho^2}{\gamma^2}\sigma \implies \left(\frac{\rho}{\gamma}\Omega^{\text{res}}\right)^k_{pq} \sim N^{k-1}\frac{\rho^{2k}}{\gamma^{2k}}\sigma^k = \frac{N^{k-1}}{N^k}\rho^{2k} = \frac{\rho^{2k}}{N}$
- Elements in Ω^{in} follow $\Omega^{\text{in}}_{jr} \sim \frac{\varepsilon}{M}$
- $\implies \left(\left(\frac{\rho}{\gamma}\Omega^{\text{res}}\right)^k \Omega^{\text{in}}\right)_{im} \sim \varepsilon\rho^{2k}$

Thus, we get that:

$$\begin{aligned} x_i(t) &\approx \sum_{m=1}^{M}\sum_{k=0}^{\infty} \varepsilon\rho^{2k} u_m(t-k) \\ &\quad - \frac{1}{3}\sum_{m,r,s=1}^{M}\sum_{k,l,n=0}^{\infty} \varepsilon^3\rho^{2(k+l+n)} u_m(t-k)u_r(t-l)u_s(t-n), \end{aligned} \tag{4.18}$$

Thus, if we compute the mean $\langle x_i(t)\rangle$, we obtain:

$$\begin{aligned} \langle x_i\rangle &= \lim_{T\to\infty}\frac{1}{T}\int_t^{t+T} x_i(t')dt' \\ &\approx \sum_{m=1}^{M}\sum_{k=0}^{\infty}\varepsilon\rho^{2k}\lim_{T\to\infty}\frac{1}{T}\int_t^{t+T} u_m(t'-k)dt' \\ &\quad - \frac{1}{3}\sum_{m,r,s=1}^{M}\sum_{k,l,n=0}^{\infty}\varepsilon^3\rho^{2(k+l+n)} \\ &\quad \times \lim_{T\to\infty}\frac{1}{T}\int_t^{t+T} u_m(t'-k)u_r(t'-l)u_s(t'-n)dt' \end{aligned} \tag{4.19}$$

$$\begin{aligned} \implies \langle x_i(t)\rangle &\approx \sum_{m=1}^{M}\sum_{k=0}^{\infty}\varepsilon\rho^{2k}\langle u_m(t-k)\rangle \\ &\quad - \frac{1}{3}\sum_{m,r,s=1}^{M}\sum_{k,l,n=0}^{\infty}\varepsilon^3\rho^{2(k+l+n)} \\ &\quad \times \langle u_m(t-k)u_r(t-l)u_s(t-n)\rangle, \end{aligned} \tag{4.20}$$

If we assume that the input time series are stationary, then their statistical moments do not change when time shifted, i.e. $\langle u_m(t-k)\rangle = \langle u_m(t)\rangle$ and $\langle u_m(t-k)u_r(t-k)u_s(t-k)\rangle = \langle u_m(t)u_r(t)u_s(t)\rangle$ (they are time invariant). Thus, we can compute the infinite sums (which are convergent geometrical series if $\rho < 1$), and get the dependence of $\langle x_i(t)\rangle$ in terms of the first order and second (zero and one lag) covariances of the inputs:

$$\begin{aligned}\langle x_i(t)\rangle \approx &\sum_{m=1}^{M} \frac{\varepsilon}{1-\rho^2}\langle u_m(t)\rangle \\ &- \sum_{m,r=2}^{M} \frac{\varepsilon^3}{1-\rho^2}\frac{1}{1-\rho^4}\langle u_m(t)u_r(t)\rangle \\ &- 2\sum_{m,r=2}^{M} \frac{\varepsilon^3}{1-\rho^2}\frac{\rho^2}{1-\rho^4}\langle u_m(t)u_r(t+1)\rangle,\end{aligned} \tag{4.21}$$

Following the same procedure, we can compute $\langle x_i(t)x_j(t)\rangle$ using the linear approximation to the state of each neuron. Thus, we get:

$$\langle x_i(t)x_j(t)\rangle \approx \sum_{m,r=1}^{M}\sum_{k,l=0}^{\infty} \varepsilon^2\rho^{2(k+l)}\langle u_m(t-k)u_r(t-l)\rangle \tag{4.22}$$

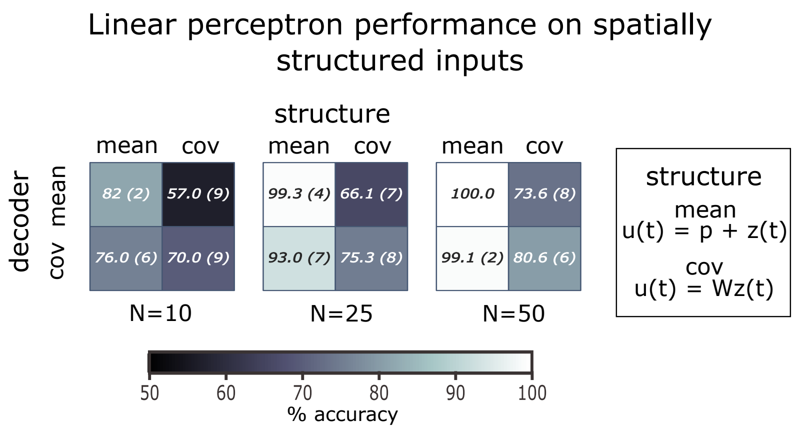
And in terms of first and second order moments (zero and one-lag), we find:

$$\begin{aligned}\langle x_i(t)x_j(t)\rangle \approx &2\sum_{m=1}^{M} \frac{\varepsilon^2}{(1-\rho^2)^2}\langle u_m(t)\rangle \\ &+ \sum_{m,r=2}^{M} \frac{\varepsilon^2}{1-\rho^4}\langle u_m(t)u_r(t)\rangle \\ &+ 2\sum_{m,r=2}^{M} \frac{\varepsilon^2\rho^2}{1-\rho^4}\langle u_m(t)u_r(t+1)\rangle.\end{aligned} \tag{4.23}$$

Thus, we observe that ρ plays a key role in how input statistics are reflected in reservoir statistics.

4.5.2 Section 4.3.2 results when using MLR decoders as benchmarks

a



b

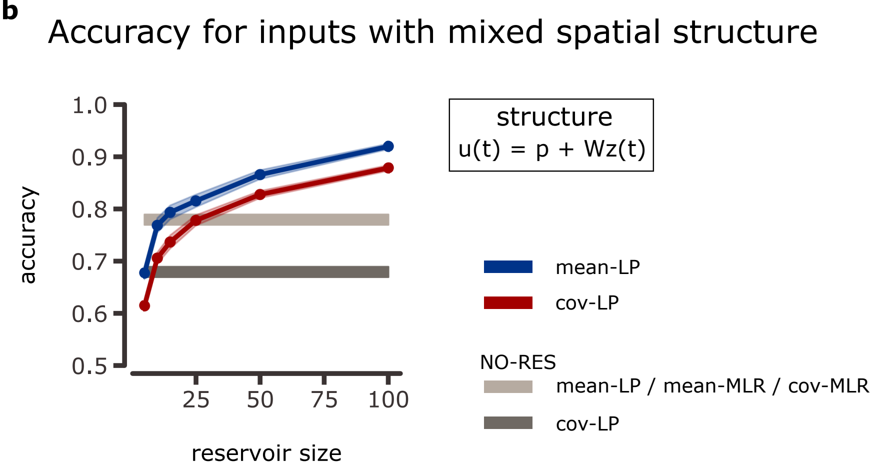


Figure 4.7: Decoding mixed information for the mean structure and spatial structure using feedforward reservoirs. **a:** Accuracy table for feedforward reservoirs without leaky integration coupled to a linear perceptron when using mean or covariance readouts to classify time series characterized by its mean or its spatial covariance structure. Same conventions as Fig 4.4. Datasets were designed to match the performances using the MLR as readout: cov-MLR $85(\pm 1)\%$, 30 patterns per class, mean-MLR $83(\pm 2)\%$, 10 patterns per class.

Figure 4.7 *(previous page)*: **b:** Classification accuracy as a function of reservoir size for a feedforward reservoir without leaky integration coupled to a linear perceptron as a readout when the categories of input time series differ by both their means and their spatial covariances. Same conventions as in Fig 4.4. Note that the datasets are generated in such a way that the classification performance of a mean-MLR and cov-MLR without reservoir is matched, equal to $78 \pm 1\%$, with 10 patterns of mean vectors and 20 patterns of covariances to distinguish per class.

In Section 4.3.2, we compare the performance of cov-LP and mean-LP decoding when the statistical information is embedded in the first or second-order moments of the inputs (Fig 4.4a) or in both (Fig 4.4b). To be able to compare performance across different datasets, we chose the number of patterns to distinguish per class such that mean-LP and cov-LP decoders acting directly on the inputs obtained matching performances. Using instead MLR decoders to match datasets, we can produce the same figure. Note that in this case, for Fig 4.7a, we have 30 covariance patterns per class (as in Fig 4.3a, left pannel) instead of 20. For Fig 4.7b we have 20 covariance patterns per class, instead of the 10 in Fig 4.4b. There is now a difference between the number of patterns to distinguish in each space, given mainly because cov-MLR decoders have a higher model complexity. Thus, to generate the dataset each mean pattern is randomly paired to 2 covariance patterns, and then the set is randomly assigned to a class. We observe, nonetheless, that the results of Fig 4.4 are mostly preserved. The difference relies in that in the mixed scenario, cov-LP decoders now lag behind mean-LP across all sizes tested. We venture that this difference is due to two sources. One is that when using mean decoders, each noisy pattern is observed twice when compared with each covariance pattern when using covariance decoders. The other is that resources are low for small reservoirs when the input covariance space needs to be covered. This could be overcomed with increasing reservoir size. We observe such decrease in the performance gap between decoders up to 25 neurons. Afterwards, the decreasing stops. This might be due to numerical instabilities when trying to optimize covariance decoders for large number of weights. Overall, Fig 4.7 suggests that the first-order information overrides the second-order one.

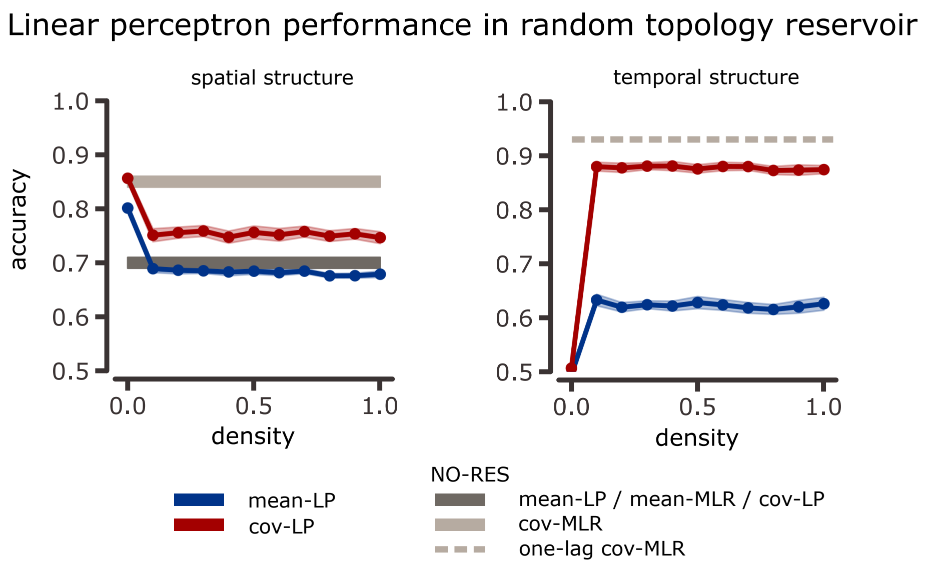


Figure 4.8: Linear perceptron classification performance in synthetic time series with second-order structure. Similar color conventions as in Fig 4.3a. Reservoir parameters are $N = 100$, $\alpha = 1$ and $\rho(\Omega^{\text{res}}) = 1.2$.

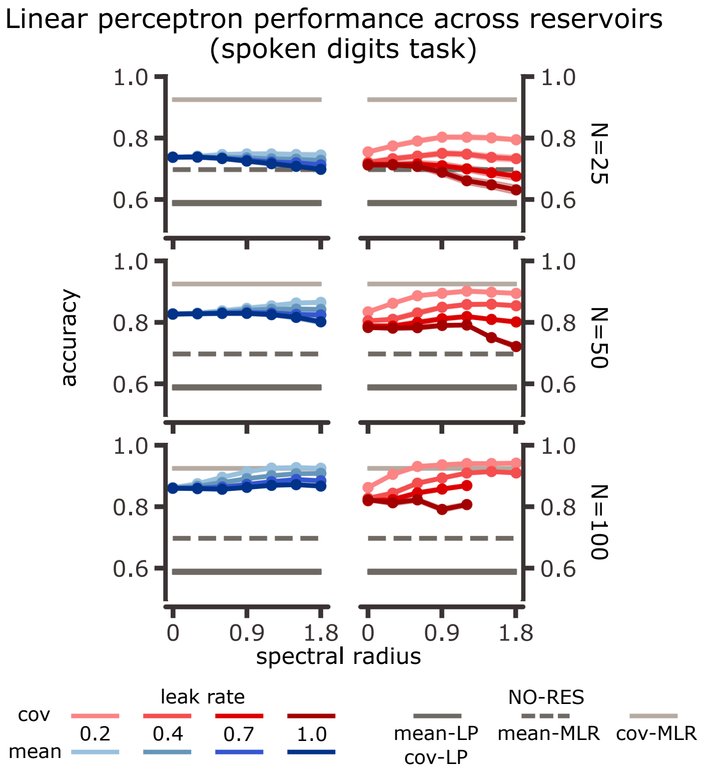


Figure 4.9: Linear perceptron classification performance for spoken digits across reservoir configurations. Similar color conventions as in Fig 4.5a. Missing points for $N = 100$, large leaks and spectral radii are due to numerical instabilities during covariance learning.

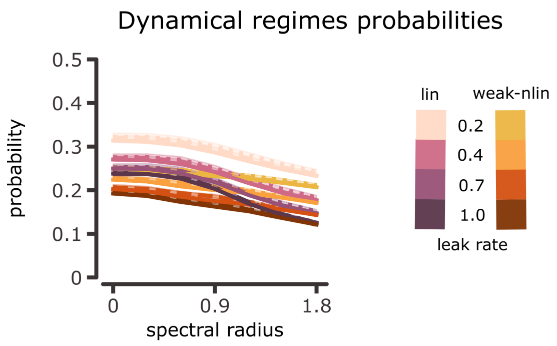


Figure 4.10: Dynamical regime probabilities for the spoken digits task. Probability of finding a reservoir neuron in the linear or weakly nonlinear activation regimes versus spectral radius, for different reservoir sizes (overlapping dotted, dashed and solid lines for $N = 25$, $N = 50$ and $N = 100$ respectively) and leak rates (color shades).

Chapter 5

GENERAL DISCUSSION

5.1 Summary of contributions

Understanding how perception and behavior come to be has long been a subject of study in humankind's quest for knowledge. Thus, this has led to a diversity of fields approaching the problem from different perspectives. Here, we have embraced the materialist view that cognitive functions are sustained by computations, where the term 'computation' is to be understood as a mapping between inputs and outputs. Particularly, we have shown interest in mappings that emerge from the interactions among inputs and the internal states of neuronal networks, either of biological or artificial nature, to determine the kind or class of inputs. This links directly to the neuronal coding problem, which is concerned with how properties of the outer world (as in external stimuli) and the inner world (as in memories, emotions or imagination) are reflected or represented in the activity of neuronal populations. A fundamental question, thus, is what mappings can be sustained. To address this issue, we combined several tools: real neuronal data analyses in collaboration with an experimental team (Chapter 2), intensive simulations for synthetic and real data (Chapters 3 and 4) and analytic work (Chapter 4). We recall our results here.

We first used a classical methodology based on the first-order statistics of biological neurons recorded in the lateral prefrontal cortex (LPFC) of behaving macaque monkeys (Chapter 2). In particular, the subjects were trained to perform two conditions of a free-viewing delayed-response task. On one task variant, the stimulus was always available (perceptual variant), while on the other it was only presented during a cue period (mnemonic variant). To begin with, we showed that the spike-count of neurons in LPFC is modulated by mnemonic and perceptual stimulus spatial location, by current gaze location and by the trial timeline epoch. We also found a large fractions of neurons displaying mixed selectivity [166]. As a last point, we showed that the information content of population codes for stimulus spatial location remained stable within trials, as indicated by the performance of a linear decoder. Altogether, these results provide evidence for mappings between the first-order statistics of neurons in LPFC and behavioral and task-relevant variables. The diversity of encoded variables is consistent with previous findings that suggest that LPFC is involved in the control of visual attention in a top-down manner [153, 156, 158, 167, 221]: signals from upper levels in the somatosensory pathway that bias the selection of attended features in earlier areas.

Afterwards, we changed our focus from biological to artificial systems, to study the transformation of representations achievable by recurrent neural networks in the framework of reservoir computing. The question, in short, was whether second-order statistics in randomly connected neuronal populations could constitute powerful representations, thus going beyond the first-order. We used the neural reservoir as a tool to re-encode information from input statistics in a cross-talk manner. Therefore, information about input averages could be recovered from reservoir covariances and vice versa. In Chapter 3, we followed a machine learning approach where we determined the usefulness of the second-order statistics of the reservoir as features fed to a multinomial logistic regression classifier to perform input classification. We stress the good performance with limited resources, as given by the size of the reservoir.

Once proven that reservoir covariances constitute powerful features to sustain mappings, we worked towards stronger biological plausibility in Chapter 4. We developed reservoir computing systems with an output signal shaped at the second-order in an input dependent fashion. We achieved this by implementing a covariance perceptron [7, 53] as readout. In this case, we were able to obtain good performance, again with limited resources, as given now by the number of learnable connections from reservoir to output layer.

While our results in Chapter 3 and Chapter 4 were developed in the context of time series classification, it has recently been shown that pairwise interactions provide robust features for the forecasting of chaotic systems [222]. This suggests that the second-order framework might be even more powerful than shown here and able to sustain other type of computations.

In the following sections, we address some specific points relevant to the strengths and limitations of our work. Section 5.2 is related to aspects developed in Chapter 2, while the remaining sections are mostly oriented to Chapter 3 and Chapter 4.

5.2 Naturalistic experiments and data analysis

The main novelty in the study in Chapter 2 was the experimental design, fully developed by our collaborators (Martínez-Trujillo's lab at Western University). In fact, it involved (to the best of our knowledge) the first recordings from LPFC of non human primates while they performed a spatial working memory task in free-viewing conditions, with a response report that required navigation within a virtual reality environment [163]. While experiments in virtual environments are nowadays quite common in the monkey hippocampus literature [223, 224] and free-viewing experiments have also been reported in the literature for orbitofrontal cortex in studies regarding the encoding of value [162], none had been performed before to probe LPFC activity during working memory. However, we were not able to fully exploit the innovative task design. This was mostly due to the inconsistent behavior across subjects, the low performance displayed by one of them and the impossibility to conduct additional experiments with other subjects.

One limitation of our study was on how we exploited the free-viewing setting. Eye movements are a relatively easy behavior to quantify, since equipment for noninvasive recordings is accessible and automated classification software tools are readily available [164]. Thus, gaze is an obvious candidate to study in naturalistic conditions. Although movement implies changes through time, we turned to analyses where we artificially recovered "static" conditions: neuronal activity was correlated to fixation location. We also made the assumption that each fixation was independent of all the others, even those taking place within the same experimental trial. This probably might have not been the case, since the frequency of fixated locations was predictive of target within a trial. Thus, it would have been interesting to perform analyses using single-trial continuous-time paradigms that more naturally align with the dynamics both in behavior (eye movements) and neuronal activity.

Adapting our generalized linear model scheme to a mix of continuous behavior (gaze) and static stimuli (target location) is, nonetheless, not trivial. For example, simply using a small sliding window technique might not yield a good enough signal-to-noise ratio or adapt well to the time scales of different gaze patterns. Applying tools from control theory could be another possibility. The Kalman filter, for instance, has previously been employed in the study of smooth pursuits and their relationship to continuous stimulus motion [225, 226]. Still, implementing Kalman filter models in a setting where the stimulus is not smoothly varied is not straightforward [227]. Altogether, there is an evident need to develop novel data analysis techniques to deal with naturalistic or highly-complex behaviors and continuous neural signals [169, 227, 228, 229, 230].

An alternative worth further exploring on Chapter 2 was population code dynamics. It has been shown that a mix of both dynamic and stable codes support working memory [142, 143, 145], and that LPFC sustains flexible mnemonic representations in the presence of distractors [146, 231]. Our task did not involve distractors, but we hypothesize that the free-viewing condition drives population codes towards a more unstable regime. Thus, it would have been interesting to compare eye-restrained versus unrestrained task settings, for example through cross-temporal decoding [141, 231]. The lack of restrained versions of the experiment prohibited this.

As a last point, applying the covariance decoding framework developed in Chapter 3 (without a reservoir) to the data from Chapter 2 would have been not only a nice way to link the two chapters, but could also have provided new valuable insights on the biological encoding of information from the second-order perspective. Indeed, we have mentioned that the experimental evidence for the functional role of trial-to-trial covariances is lacking [50, 51, 52]. Nevertheless, the low amount of trials (approximately 30 per condition and neuron in the best of cases) would have required some subsampling procedure to pre-select neurons and avoid overfitting. In this sense, an approach using principal component analysis on mean and covariance space of neuronal activity before feeding to a classifier constitutes a more viable alternative and is left for future work.

5.3 The functional role of covariances

One of the oldest questions in neuroscience is how neuronal spiking patterns reflect characteristics of the world. Representational accounts aiming to link specific features of brain activity to stimulus properties or behavioral responses have proposed a variety of codes. These mainly differ in the temporal scales used to describe the spike sequences, with rate-based and full-sequence descriptions laying at the extremes of a continuum [64]. Building on previous work [7, 53], we have shown that in a minimalist model of brain cortex (reservoir computing), a middle-ground coding mechanism based on structured covariances constitutes a powerful representation, as given by the decoding accuracy of a logistic classifier across a diverse set of classification tasks (Chapter 3). Furthermore, we have shown that covariance patterns can propagate across networks, as given by our covariance perceptron study in Chapter 4. In both of these chapters, the novel use of the reservoir was key to induce a cross-talk between statistical orders in the input layer signal and the output layer one, while also avoiding the training of recurrent connections at the output layer.

Covariance coding proposes that information is encoded and transmitted in the within-trial activity co-fluctuations of neuronal units. This can occur both in the presence or absence of mean-rate modulations. Thus, while coding mechanisms are usually discussed in opposing terms (e.g. rate versus time), we propose covariance coding as a plausible alternative that can coexist with other schemes, such as rated-based representations. This is consistent with some theories that support that the brain might concurrently employ several information coding mechanisms [63, 64, 232, 233]. Therefore, information can be integrated across a wider range of spatiotemporal scales than would be achievable by a single neuronal mechanism. It has been shown, for example, that slow rhythmic activity (local field potential oscillations in the frequency range below 30 hertz) does not constitute noise, but can instead complement the information (theoretic) content of rate codes [63].

Interestingly, the fact that in our work different coding schemes at the output share a common neuronal substrate opens up the way to consider the multiplexing of information [234, 235, 236]. Indeed, we can independently shape readouts at the first or second-order through the same reservoir. It remains to be tested whether simultaneous signals of different nature can be efficiently routed through different reservoir/readout statistical orders.

A further important aspect to consider when discussing coding schemes is information transmission [64]. One common argument against pure rate-based coding is that information transmission should be faster and able to rely on few emitted spikes. For instance, the processing of visual information for image recognition is believed to last less than 50 miliseconds, that is, for information to travel from the lower level sensory areas (retina) to the higher level ones (cortex) [55]. In our work, we observed that information represented in input one-lag covariances is better transformed by a recurrent network into zero-lag covariance patterns than into mean activity patterns. Importantly, this occurred when the stimulation period was short and no wash-out was allowed to remove transient effects. Thus, covariance representations appear as a suitable candidate for fast information transmission. Additional work is needed to determine whether lagged covariances can be efficiently represented by first-order statistics given enough time to mix.

In a Bayesian-brain context, neuronal networks are thought to represent stimulus as probability distributions [28, 237]. In this view, different statistical orders of neuronal activity can be linked to different parameters in the perceptual probability distribution over stimuli. As an example, it has been suggested that averaged activity encodes the mean of the posterior distribution, while (co)variances represent the spread or uncertainty around said mean [238, 239]. Thus, statistical moments of neuronal activity serve distinct purposes and can vary independently of each other. In our reservoir scheme, reservoir statistics depend on a linear sum of diverse input statistics, due to the nonlinearity. Therefore, the "independence" between input-driven orders is lost. Furthermore, as we developed it, larger variance on a single output node sustained more reliable stimulus identification, which is at odds with Bayesian coding theories. However, by switching the decision rule to choose the output node with lowest variance (consistently setting target output covariances during learning), our signal processing approach can be driven closer to the Bayesian inference one. In our case, lower variability would signal stronger evidence or confidence towards a certain stimulus class.

5.4 The reservoir approach

Two key concepts in this thesis are covariance decoding, which we previously discussed, and reservoir computing. We have here merged them, to the best of our knowledge, for the first time. Reservoirs can be used to shape the activity of output neurons beyond the first order. We remark the importance of nonlinear reservoir behavior to achieve this. Nonlinearities mix how input statistical orders contribute to the shaping of reservoir statistics, and thus potentially allow to develop mappings from first, second or higher-order statistics to any desired output statistic. In this thesis, we particularly examined the cases involving the first two orders, so extensions to third and higher-order moments are pending. Extensions of Chapter 3 in this direction are straightforward. On the other hand, for Chapter 4, it would be necessary to derive a new learning rule, for example, to map third-order moments in reservoir to third-order moments in outputs and so on for other orders.

Reservoir computing power relies on high-dimensional heterogeneous randomness. The more varied the dynamics inside the reservoir, the bigger the chances of finding subspaces where data is linearly separable or can be reconstructed. The way we exploited heterogeneous dynamics was through random connectivities among neurons and from inputs to reservoir, although it has also been suggested that variations in single neuron time constants can also provide richer spaces [115]. This also remained an unvisited path in our work.

Small improvements (mostly due to the larger computing capacity available nowadays) for training fully recurrent models have resulted in their rise of popularity for modelling neural systems [68]. A common approach is to train the network to perform a specific task or general purpose computation to later reverse engineer the model, looking for emergent dynamics and structures that can be contrasted with data from real circuits. This is not feasible in a reservoir computing framework, since the reservoir is pre-designed and fixed. Hence, no emergent properties can arise during training. However, it is possible to construct reservoirs with imposed biological constraints [70], for example in the connectivity structure [215], to examine how specific bio-realistic characteristics influence dynamics and performance when compared to the fully random case. This is another line of research to further extend the work of Chapters 3 and 4.

In an interesting article about the evolutionary aspects of reservoir computing [240], it was suggested that it might have evolved to provide circuits in high-order brain areas that perform a variety of complex tasks with large generalization properties. Therefore, reservoir computing as a means for brain computation appears as an appealing hypothesis, but not a universal one: circuits that solve specific dedicated computations might be better explained by models that are highly tuned for a single task. Thus, a question that remains, regarding our work, is whether a specific topography that optimally propagates covariances might emerge when a recurrent network is fully trained.

5.4.1 Scalability and robustness to noise

Biological neuronal systems are typically formed by large number of neurons [241]. However, the reservoir systems we modelled in Chapter 3 and Chapter 4 were of small size (smaller than 300 units), although still comparable to the neural system of *C. elegans*. Furthermore, from a machine learning perspective, it is always important to characterize the ability of a model to improve performance as computational resources increase. Hence, scalability studies for our reservoir computing covariance decoding approach should be performed. This requires the identification of tools to efficiently regularize the training procedure when implementing the logistic classifier, as wells as the development of methods to numerically stabilize the learning procedure for the covariance perceptron.

We remark, nonetheless, that the fact that the covariance approach works best for small resources is a solid good point. In fact, the deployment of large scale models is known to be energy consuming, so more economic machine learning approaches that leave a softer environmental footprint should be devised in the near future [192, 242].

On another matter, all our simulations in Chapter 3 and Chapter 4 were performed in the same conditions. Our reservoir always started from a zero-state and the only noise in the system was inherently present in the input time series, due to stochastic variability in each sample presentation. This raises two questions.

First, since we exploited transient dynamics for classification, it is important to know how these states vary with changing initial conditions. Biological systems are known to have dynamically structured ongoing activity (e.g. resting states in fMRI [243, 244, 245] or up/down states in membrane potential [246, 247]), so reproducing the exact initial state across trials is not possible. Therefore, our scheme should be robust to changing initial conditions to show stronger biological plausibility.

Second, we have mentioned that performance in living organisms can stay high while displaying variability due to a large variety of possible sources: internal states, unobserved variables, unidentified inputs from other areas, etc [21]. However, we have only examined how the reservoir behaves when the noise comes exclusively from inputs. Specially, our results on the S-MNIST data in Chapter 3 seem to suggest that the reservoir is very sensitive to noise. Indeed, those speech signals were recorded in ambient conditions, thus displaying larger variability, which significantly diminished the classifiers performance. Hence, exploring artificial settings with controlled noise levels would help better identify the origin of performance drops and how they relate to internal dynamics. Possible sources of noise that can be studied are noise directly injected into the reservoir activity, or indirectly through noise synapses, randomly "killing" neurons on a trial-by-trial basis and switching off synapses.

5.4.2 Spiking reservoirs

Our choice to use analogue networks was mainly driven by the desire to keep our models compatible with the covariance perceptron, the novel readout we implemented in Chapter 4. A natural extension of our work is, thus, switching to more biologically plausible spike-based models. Nonetheless, a few non trivial points should be addressed in doing so.

Spiking neuron models are excited by incoming input spikes. However, real world data often come in analogue values, as the spectral coefficients of speech we worked with. Thus, encoding methods to transform input real values into discrete spike sequences are needed. Many studies use rate-based encoding mechanisms, where continuous signals are transformed into spikes by means of Poisson processes [98, 248]. Hence, the rate of the process is given by the intensity of the signal. However, this neglects any information contained in other (temporal) aspects, and therefore might hinder or opaque the true computing power of a spiking network. How to efficiently represent real world data with spikes is a currently active field of research within the neuromorphic computing community [249, 250, 251]. Importantly, early implementations of liquid state machines, the spiking version of echo state networks, were usually tested on synthetic datasets. Efforts performing speech recognition tasks with real data also remarked on the importance of the encoding method used to transform the analogue values into spike sequences. Yet, these mostly exploited rate-based mechanisms, such as Bens Spiker Algorithm (BSA) [184, 188, 189, 252].

Additionally, defining a cost function in supervised learning for spiking models is not straightforward and it is not uncommon to find that the error surface has discontinuities that make learning unstable when using error backpropagation [248, 253]. When the objective is to map input patterns into specific output patterns (i.e. spikes emitted at predefined times), cost functions used in the literature include mean-squared-error in firing times (SpikeProp [254]), mean-squared-error in membrane potential of output neurons (Tempotron [218]) and the Victor-Purpura distance [255]. However, these algorithms become unstable as the number of neurons to train at the output layer increases. Therefore, they have mostly been studied to shape the activity of a single neuron. In our case, the gradient descent algorithm for covariance learning should be adapted, for example for implementation via surrogate gradients [73, 75]. In this setting, it might be useful to first test whether a variance approach for classification can be implemented via a single output node if a proper threshold value is chosen, so that training cost is reduced for the binary classification task.

5.5 Endnote

A main personal take-home message of the work here presented is that information representation, learning and transmission are intrinsically entangled and should not be studied in isolation, since constraints in one of these aspects could imply further constrains in another in turn. In this sense, there is strong potential in the interaction between computational neuroscience and machine learning (or artificial intelligence in general). Exploring representation and learning mechanisms exploited by the nervous system can lead to efficient representation and learning in artificial ones, while the converse can provide novel hypothesis and experimental ideas to look for correlates of neuronal representations and the processes by which these originate.

In terms of computation, any given process might be multiply realized. Thus, finding a solution does not mean that such solution is universal or unique. Indeed, the space of solutions might be infinite, include optimal and sub-optimal ones, and any given system might settle for one or the other depending on current and previous input and internal history, available resources or future state/input estimates. Therefore, computation (especially that of the natural type) appears as a system-subjective context-dependent interactive process. However, as a physicist by training, I personally hold within the hope that more general underlying principles be discovered by the scientific community in the years to come.

Bibliography

[1] William Bialek, Fred Rieke, Rob R. De Ruyter van Steveninck, and David Warland. Reading a Neural Code. *Science*, 252, 1991.

[2] Fred Rieke, David Warland, Rob de Ruyter van Steveninck, and William Bialek. *Spikes: Exploring the Neural Code*. The MIT Press, Cambridge, Massachusetts, 1999.

[3] Laurence F. Abbott and Peter Dayan. *Theoretical neuroscience: Computational and mathematical modeling of neural systems.* Massachusetts Institute of Technology Press, Cambridge, Massachusetts, 2001.

[4] Garrett B. Stanley. Reading and writing the neural code. *Nature Neuroscience*, 16(3):259–263, 2013.

[5] Stefano Panzeri, Christopher D. Harvey, Eugenio Piasini, Peter E. Latham, and Tommaso Fellin. Cracking the Neural Code for Sensory Perception by Combining Statistics, Intervention, and Behavior. *Neuron*, 93(3):491–507, feb 2017.

[6] Nikolaus Kriegeskorte and Jorn Diedrichsen. Peeling the Onion of Brain Representations. *Annual Review of Neuroscience*, 42:407–432, 2019.

[7] David Dahmen, Matthieu Gilson, and Moritz Helias. Capacity of the covariance perceptron. *Journal of Physics A: Mathematical and Theoretical*, 53(35):354002, sep 2020.

[8] Liam Paninski, Jonathan Pillow, and Jeremy Lewi. Statistical models for neural encoding, decoding, and optimal stimulus design, 2007.

[9] Z. Mainen and T. Sejnowski. Reliability of spike timing in neocortical neurons. *Science*, 268(5216):1503–1506, jun 1995.

[10] L. Nowak, Maria V. Sanchez-Vives, and D. A. McCormick. Influence of low and high frequency inputs on spike timing in visual cortical neurons. *Cerebral Cortex*, 7(6):487–501, sep 1997.

[11] Daniel S. Reich, Jonathan D. Victor, Bruce W. Knight, Tsuyoshi Ozaki, and Ehud Kaplan. Response Variability and Timing Precision of Neuronal Spike Trains In Vivo. *Journal of Neurophysiology*, 77(5):2836–2841, may 1997.

[12] Wyeth Bair and Christof Koch. Temporal Precision of Spike Trains in Extrastriate Cortex of the Behaving Macaque Monkey. *Neural Computation*, 8(6):1185–1202, aug 1996.

[13] Giedrius T. Buračas, Anthony M. Zador, Michael R. DeWeese, and Thomas D. Albright. Efficient discrimination of temporal patterns by motion-sensitive neurons in primate visual cortex. *Neuron*, 20(5):959–969, 1998.

[14] D.J. Tolhurst, J.A. Movshon, and A.F. Dean. The statistical reliability of signals in single neurons in cat and monkey visual cortex. *Vision Research*, 23(8):775–785, jan 1983.

[15] WR Softky and C Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs. *The Journal of Neuroscience*, 13(1):334–350, jan 1993.

[16] Moshe Gur, Alexander Beylin, and D. Max Snodderly. Response Variability of Neurons in Primary Visual Cortex (V1) of Alert Monkeys. *The Journal of Neuroscience*, 17(8):2914–2920, apr 1997.

[17] Michael N. Shadlen and William T. Newsome. The Variable Discharge of Cortical Neurons: Implications for Connectivity, Computation, and Information Coding. *The Journal of Neuroscience*, 18(10):3870–3896, may 1998.

[18] Richard B. Stein, E. Roderick Gossen, and Kelvin E. Jones. Neuronal variability: Noise or part of the signal? *Nature Reviews Neuroscience*, 6(5):389–397, 2005.

[19] A. Aldo Faisal, Luc P.J. Selen, and Daniel M. Wolpert. Noise in the nervous system. *Nature Reviews Neuroscience*, 9(4):292–303, 2008.

[20] Alfonso Renart and Christian K. Machens. Variability in neural activity and behavior. *Current Opinion in Neurobiology*, 25:211–220, apr 2014.

[21] Timothée Masquelier. Neural variability, or lack thereof. *Frontiers in Computational Neuroscience*, 7, 2013.

[22] Alexandre Pouget, Peter Dayan, and Richard Zemel. Information processing with population codes. *Nature Reviews Neuroscience*, 1(November):125–132, 2000.

[23] Romain Brette. Philosophy of the Spike: Rate-Based vs. Spike-Based Theories of the Brain. *Frontiers in Systems Neuroscience*, 9(November):1–14, nov 2015.

[24] A. J. Parker and W. T. Newsome. SENSE AND THE SINGLE NEURON: Probing the Physiology of Perception. *Annual Review of Neuroscience*, 21(1):227–277, mar 1998.

[25] Daniel A. Butts and Mark S. Goldman. Tuning curves, neuronal variability, and sensory coding. *PLoS Biology*, 4(4):639–646, 2006.

[26] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591, oct 1959.

[27] Michael N. Shadlen and William T. Newsome. Noise, neural codes and cortical organization. *Current Opinion in Neurobiology*, 4(4):569–579, aug 1994.

[28] Wei Ji Ma, Jeffrey M. Beck, Peter E. Latham, and Alexandre Pouget. Bayesian inference with probabilistic population codes. *Nature Neuroscience*, 9(11):1432–1438, 2006.

[29] GergÅ Orbán, Pietro Berkes, József Fiser, and Máté Lengyel. Neural Variability and Sampling-Based Probabilistic Representations in the Visual Cortex. *Neuron*, 92(2):530–543, 2016.

[30] Bruno B. Averbeck, Peter E. Latham, and Alexandre Pouget. Neural correlations, population coding and computation. *Nature Reviews Neuroscience*, 7(5):358–366, 2006.

[31] Bruno B. Averbeck and Daeyeol Lee. Effects of noise correlations on information encoding and decoding. *Journal of Neurophysiology*, 95(6):3633–3644, 2006.

[32] Marlene R. Cohen and Adam Kohn. Measuring and interpreting neuronal correlations. *Nature Neuroscience*, 14(7):811–819, 2011.

[33] Adam Kohn, Ruben Coen-Cagli, Ingmar Kanitscheider, and Alexandre Pouget. Correlations and Neuronal Population Information. *Annual Review of Neuroscience*, 39:237–256, 2016.

[34] KH Britten, MN Shadlen, WT Newsome, and JA Movshon. The analysis of visual motion: a comparison of neuronal and psychophysical performance. *The Journal of Neuroscience*, 12(12):4745–4765, dec 1992.

[35] T. J. Gawne and B. J. Richmond. How independent are the messages carried by adjacent inferior temporal cortical neurons. *Journal of Neuroscience*, 13(7):2758–2771, 1993.

[36] Ehud Zohary, Michael N. Shadlen, and William T. Newsome. Correlated neuronal discharge rate and its implications for psychophysical performance. *Nature*, 370(6485):140–143, 1994.

[37] Leslie C. Osborne, William Bialek, and Stephen G. Lisberger. Time Course of Information about Motion Direction in Visual Area MT of Macaque Monkeys. *Journal of Neuroscience*, 24(13):3210–3222, 2004.

[38] Marlene R. Cohen and John H.R. Maunsell. Attention improves performance primarily by reducing interneuronal correlations. *Nature Neuroscience*, 12(12):1594–1600, 2009.

[39] L. F. Abbott and Peter Dayan. The effect of correlated variability on the accuracy of a population code. *Neural Computation*, 11(1):91–101, 1999.

[40] Stefano Panzeri, Simon R Schultz, Alessandro Treves, and Edmund T Rolls. Correlations and the encoding of information in the nervous system. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 266(1423):1001–1012, may 1999.

[41] Ranulfo Romo, Adrián Hernández, Antonio Zainos, and Emilio Salinas. Correlated neuronal discharges that increase coding efficiency during perceptual discrimination. *Neuron*, 38(4):649–657, 2003.

[42] Matthieu Gilson, Timothée Masquelier, and Etienne Hugues. STDP Allows Fast Rate-Modulated Coding with Poisson-Like Spike Trains. *PLoS Computational Biology*, 7(10):e1002231, oct 2011.

[43] Martin Boerlin, Christian K. Machens, and Sophie Denève. Predictive Coding of Dynamical Variables in Balanced Spiking Networks. *PLoS Computational Biology*, 9(11):e1003258, nov 2013.

[44] Rubén Moreno-Bote, Jeffrey Beck, Ingmar Kanitscheider, Xaq Pitkow, Peter Latham, and Alexandre Pouget. Information-limiting correlations. *Nature Neuroscience*, 17(10):1410–1417, 2014.

[45] Felix Franke, Michele Fiscella, Maksim Sevelev, Botond Roska, Andreas Hierlemann, and Rava Azeredo da Silveira. Structures of Neural Correlation and How They Favor Coding. *Neuron*, 89(2):409–422, 2016.

[46] J. A. Nelder and R W M Wedderburn. Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370, 1972.

[47] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006.

[48] Donald O Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, 1949.

[49] A. Riehle, Sonja Grün, Markus Diesmann, and Ad Aertsen. Spike Synchronization and Rate Modulation Differentially Involved in Motor Cortical Function. *Science*, 278(5345):1950–1953, dec 1997.

[50] Neda Shahidi, Ariana R. Andrei, Ming Hu, and Valentin Dragoi. High-order coordination of cortical spiking activity modulates perceptual accuracy. *Nature Neuroscience*, 22(7):1148–1158, 2019.

[51] Alireza Hashemi, Ashkan Golzar, Jackson E.T. Smith, and Erik P. Cook. The magnitude, but not the sign, of MT single-trial spike-time correlations predicts motion detection performance. *Journal of Neuroscience*, 38(18):4399–4417, 2018.

[52] Bjørg E. Kilavik, Sébastien Roux, Adrián Ponce-Alvarez, Joachim Confais, Sonja Grün, and Alexa Riehle. Long-term modifications in motor cortical dynamics induced by intensive practice. *Journal of Neuroscience*, 29(40):12653–12663, 2009.

[53] Matthieu Gilson, David Dahmen, Rubén Moreno-Bote, Andrea Insabato, and Moritz Helias. The covariance perceptron: A new paradigm for classification and processing of time series in recurrent neuronal networks. *PLOS Computational Biology*, 16(10):e1008127, oct 2020.

[54] A. L. Jacobs, G. Fridman, R. M. Douglas, N. M. Alam, Peter. E. Latham, G. T. Prusky, and S. Nirenberg. Ruling out and ruling in neural codes. *Proceedings of the National Academy of Sciences*, 106(14):5936–5941, 2009.

[55] Rufin Van Rullen and Simon J. Thorpe. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Computation*, 13(6):1255–1283, 2001.

[56] T. Gollisch and M. Meister. Rapid Neural Coding in the Retina with Relative Spike Latencies. *Science*, 319(5866):1108–1111, feb 2008.

[57] Simon Thorpe and Jacques Gautrais. Rank Order Coding. In *Computational Neuroscience*, pages 113–118. Springer US, Boston, MA, 1998.

[58] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14(6-7):715–725, jul 2001.

[59] Oren Shriki, Adam Kohn, and Maoz Shamir. Fast coding of orientation in primary visual cortex. *PLoS Computational Biology*, 8(6), 2012.

[60] Timothy J. Gawne, Troels W. Kjaer, and Barry J. Richmond. Latency: Another potential code for feature binding in striate cortex. *Journal of Neurophysiology*, 76(2):1356–1360, 1996.

[61] J. J. Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535):33–36, jul 1995.

[62] Timothée Masquelier, Etienne Hugues, Gustavo Deco, and Simon J. Thorpe. Oscillations, phase-of-firing coding, and spike timing-dependent plasticity: An efficient learning scheme. *Journal of Neuroscience*, 29(43):13484–13493, 2009.

[63] Christoph Kayser, Marcelo A. Montemurro, Nikos K. Logothetis, and Stefano Panzeri. Spike-Phase Coding Boosts and Stabilizes Information Carried by Spatial and Temporal Spike Patterns. *Neuron*, 61(4):597–608, 2009.

[64] Arvind Kumar, Stefan Rotter, and Ad Aertsen. Spiking activity propagation in neuronal networks: Reconciling different perspectives on neural coding. *Nature Reviews Neuroscience*, 11(9):615–627, 2010.

[65] Douglas McLelland and Rufin VanRullen. Theta-Gamma Coding Meets Communication-through-Coherence: Neuronal Oscillatory Multiplexing Theories Reconciled. *PLoS Computational Biology*, 12(10):4–10, 2016.

[66] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, may 2015.

[67] Jürgen Schmidhuber. Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[68] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6, oct 2017.

[69] Guangyu Robert Yang and Xiao Jing Wang. Artificial Neural Networks for Neuroscientists: A Primer. *Neuron*, 107(6):1048–1070, 2020.

[70] Friedemann Pulvermüller, Rosario Tomasello, Malte R. Henningsen-Schomers, and Thomas Wennekers. Biological constraints on neural network models of cognitive function. *Nature Reviews Neuroscience*, 22(8):488–502, 2021.

[71] A. N. Burkitt. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. *Biological Cybernetics*, 95(1):1–19, 2006.

[72] Friedemann Zenke and Tim P. Vogels. The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks. *Neural Computation*, 33(4):899–925, 2021.

[73] Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541, jun 2018.

[74] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):787–797, 2018.

[75] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[76] RÇzvan V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.

[77] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Overview of Supervised Learning. pages 9–41. 2009.

[78] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised Learning. pages 485–585. 2009.

[79] EL Bienenstock, LN Cooper, and PW Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience*, 2(1):32–48, jan 1982.

[80] Arnaud Delorme, Laurent Perrinet, and Simon J. Thorpe. Networks of integrate-and-fire neurons using Rank Order Coding B: Spike timing dependent plasticity and emergence of orientation selectivity. *Neurocomputing*, 38-40:539–545, jun 2001.

[81] Geoffrey Hinton and Terrence J. Sejnowski. *Unsupervised Learning: Foundations of Neural Computation*. The MIT Press, 1999.

[82] Adam Kepecs, Mark C.W. Van Rossum, Sen Song, and Jesper Tegner. Spike-timing-dependent plasticity: Common themes and divergent vistas. *Biological Cybernetics*, 87(5-6):446–458, 2002.

[83] K. D. Miller, L. F. Abbott, and S. Song. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3:919–926, 2000.

[84] Daniel E. Feldman. The Spike-Timing Dependence of Plasticity. *Neuron*, 75(4):556–571, 2012.

[85] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs. *Science*, 275(5297):213–215, jan 1997.

[86] Huibert D. Mansvelder, Matthijs B. Verhoog, and Natalia A. Goriounova. Synaptic plasticity in human cortical circuits: cellular mechanisms of learning and memory in the human brain? *Current Opinion in Neurobiology*, 54:186–193, 2019.

[87] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 9:201–208, 2010.

[88] Yingyezhe Jin and Peng Li. AP-STDP: A novel self-organizing mechanism for efficient reservoir computing. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1158–1165. IEEE, jul 2016.

[89] Dmitry Krotov and John J. Hopfield. Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences of the United States of America*, 116(16):7723–7731, 2019.

[90] Philipp Weidel, Renato Duarte, and Abigail Morrison. Unsupervised learning and clustered connectivity enhance reinforcement learning in spiking neural networks. pages 1–27, 2020.

[91] Stevan Harnad. To Cognize is to Categorize. In *Handbook of Categorization in Cognitive Science*, pages 21–54. Elsevier, 2017.

[92] J. David Smith, Alexandria C. Zakrzewski, Jennifer M. Johnson, Jeanette C. Valleau, and Barbara A. Church. Categorization: The view from animal cognition. *Behavioral Sciences*, 6(2):1–24, 2016.

[93] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[94] E Gardner. The space of interactions in neural network models. *Journal of Physics A: Mathematical and General*, 21(1):257–270, jan 1988.

[95] Daniel L.K. Yamins and James J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19(3):356–365, mar 2016.

[96] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, number 2, pages III–1310–III–1318, Atlanta, GA, USA, 2013. JMLR.org.

[97] Herbert Jaeger. The ”echo state” approach to analysing and training recurrent neural networks. Technical report, dec 2001.

[98] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation*, 14(11):2531–2560, nov 2002.

[99] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[100] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review, jul 2019.

[101] Herbert Jaeger. *Short term memory in echo state networks*. Sankt Augustin: GMD Forschungszentrum Informationstechnik, 2001, 60 pp. GMD Report, 152, 2002.

[102] G. Manjunath and H. Jaeger. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural Computation*, 25(3):671–696, 2013.

[103] Pietro Verzelli, Cesare Alippi, Lorenzo Livi, and Peter Tino. Input representation in recurrent neural networks dynamics. *arXiv*, 2020.

[104] Herbert Jaeger and Harald Haas. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304(5667):78–80, 2004.

[105] David Sussillo and L.F. Abbott. Generating Coherent Patterns of Activity from Chaotic Neural Networks. *Neuron*, 63(4):544–557, aug 2009.

[106] P F Dominey. Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological cybernetics*, 73(3):265–74, aug 1995.

[107] Peter F. Dominey. Recurrent temporal networks and language acquisition: from corticostriatal neurophysiology to reservoir computing. *Frontiers in Psychology*, 4(August):1–14, 2013.

[108] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20(3):335–352, apr 2007.

[109] Razvan Pascanu and Herbert Jaeger. A neurodynamical model for working memory. *Neural Networks*, 24(2):199–207, mar 2011.

[110] Gregor M. Hoerzer, Robert Legenstein, and Wolfgang Maass. Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning. *Cerebral Cortex*, 24(3):677–690, mar 2014.

[111] Christian Rössert, Paul Dean, and John Porrill. At the Edge of Chaos: How Cerebellar Granular Layer Network Dynamics Can Provide the Basis for Temporal Filters. *PLoS Computational Biology*, 11(10):1–28, 2015.

[112] Zhewei Zhang, Zhenbo Cheng, Zhongqiao Lin, Chechang Nie, and Tianming Yang. A neural network model for the orbitofrontal cortex and task space acquisition during reinforcement learning. *PLoS Computational Biology*, 14(1):1–24, 2018.

[113] S. Haeusler and W. Maass. A Statistical Analysis of Information-Processing Properties of Lamina-Specific Cortical Microcircuit Models. *Cerebral Cortex*, 17(1):149–162, feb 2006.

[114] Dean V. Buonomano and Wolfgang Maass. State-dependent computations: Spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, 10(2):113–125, 2009.

[115] Alberto Bernacchia, Hyojung Seo, Daeyeol Lee, and Xiao Jing Wang. A reservoir of time constants for memory traces in cortical neurons. *Nature Neuroscience*, 14(3):366–372, 2011.

[116] Pierre Enel, Emmanuel Procyk, René Quilodran, and Peter Ford Dominey. Reservoir Computing Properties of Neural Dynamics in Prefrontal Cortex. *PLoS Computational Biology*, 12(6):e1004967, jun 2016.

[117] Nicolas Cazin, Martin Llofriu Alonso, Pablo Scleidorovich Chiodi, Tatiana Pelc, Bruce Harland, Alfredo Weitzenfeld, Jean Marc Fellous, and Peter Ford Dominey. *Reservoir computing model of prefrontal cortex creates novel combinations of previous navigation sequences from hippocampal place-cell replay with spatial reward propagation*, volume 15. 2019.

[118] Olivia L. White, Daniel D. Lee, and Haim Sompolinsky. Short-term memory in orthogonal neural networks. *Physical Review Letters*, 92(14):1–4, 2004.

[119] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, apr 2007.

[120] Surya Ganguli, Dongsung Huh, and Haim Sompolinsky. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 105(48):18970–18975, 2008.

[121] David Verstraeten and Benjamin Schrauwen. On the quantification of dynamics in reservoir computing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intel-*

ligence and Lecture Notes in Bioinformatics), 5768 LNCS(PART 1):985–994, 2009.

[122] Qingsong Song and Zuren Feng. Effects of connectivity structure of complex echo state network on its prediction performance for nonlinear time series. *Neurocomputing*, 73(10-12):2177–2185, 2010.

[123] Lars Büsing, Benjamin Schrauwen, and Robert Legenstein. Connectivity, Dynamics, and Memory in Reservoir Computing with Binary and Analog Neurons. *Neural Computation*, 22(5):1272–1311, may 2010.

[124] Tobias Strauss, Welf Wustlich, and Roger Labahn. Design Strategies for Weight Matrices of Echo State Networks. *Neural Computation*, 24(12):3246–3276, dec 2012.

[125] Pau Vilimelis Aceituno, Gang Yan, and Yang Yu Liu. Tailoring Echo State Networks for Optimal Learning. *iScience*, 23(9):101440, 2020.

[126] Claudio Gallicchio. Sparsity in Reservoir Computing Neural Networks. In *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–7. IEEE, aug 2020.

[127] Abdulrahman Alalshekmubarak and Leslie S. Smith. A novel approach combining recurrent neural network and support vector machines for time series classification. *2013 9th International Conference on Innovations in Information Technology, IIT 2013*, pages 42–47, 2013.

[128] Abdulrahman Alalshekmubarak and Leslie S. Smith. On Improving the Classification Capability of Reservoir Computing for Arabic Speech Recognition. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lec-*

ture Notes in Bioinformatics), volume 8681 LNCS, pages 225–232. 2014.

[129] Witali Aswolinskiy, René Felix Reinhart, and Jochen Steil. Time Series Classification in Reservoir- and Model-Space. *Neural Processing Letters*, 48(2):789–809, 2018.

[130] Filippo Maria Bianchi, Simone Scardapane, Sigurd Lokse, and Robert Jenssen. Reservoir Computing Approaches for Representation and Classification of Multivariate Time Series. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):2169–2179, 2021.

[131] Manasij Venkatesh, Joseph Jaja, and Luiz Pessoa. Brain dynamics and temporal trajectories during task and naturalistic processing. *NeuroImage*, 186(1):410–423, feb 2019.

[132] A Baddeley. Working memory. *Science*, 255(5044):556–559, jan 1992.

[133] J. M. Fuster and G. E. Alexander. Neuron Activity Related to Short-Term Memory. *Science*, 173(3997):652–654, aug 1971.

[134] S. Funahashi, C. J. Bruce, and P. S. Goldman-Rakic. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *Journal of Neurophysiology*, 61(2):331–349, feb 1989.

[135] P S Goldman-Rakic. Cellular and circuits basis of working memory in prefrontal cortex of nonhuman primates. *The prefrontal cortex: its structure, function and pathology*, 85:325–336, 1990.

[136] P. S. Goldman-Rakic. Cellular basis of working memory, mar 1995.

[137] Christos Constantinidis, Matthew N. Franowicz, and Patricia S. Goldman-Rakic. The sensory nature of mnemonic representation in the primate prefrontal cortex. *Nature Neuroscience*, 4(3):311–316, mar 2001.

[138] Christos Constantinidis, Shintaro Funahashi, Daeyeol Lee, John D. Murray, Xue Lian Qi, Min Wang, and Amy F.T. Arnsten. Persistent spiking activity underlies working memory. *Journal of Neuroscience*, 38(32):7020–7028, 2018.

[139] Mikael Lundqvist, Jonas Rose, Pawel Herman, ScottÂ L L. Brincat, TimothyÂ J J. Buschman, and EarlÂ K K. Miller. Gamma and Beta Bursts Underlie Working Memory. *Neuron*, 90(1):152–164, 2016.

[140] Mikael Lundqvist, Pawel Herman, and Earl K Miller. Working memory: Delay activity, yes! persistent activity? maybe not. *Journal of Neuroscience*, 38(32):7013–7019, aug 2018.

[141] Mark G. Stokes, Makoto Kusunoki, Natasha Sigala, Hamed Nili, David Gaffan, and John Duncan. Dynamic coding for cognitive control in prefrontal cortex. *Neuron*, 78(2):364–375, 2013.

[142] John D. Murray, Alberto Bernacchia, Nicholas A. Roy, Christos Constantinidis, Ranulfo Romo, and Xiao Jing Wang. Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proceedings of the National Academy of Sciences of the United States of America*, 114(2):394–399, 2017.

[143] Eelke Spaak, Kei Watanabe, Shintaro Funahashi, and Mark G. Stokes. Stable and dynamic coding for working memory in primate prefrontal cortex. *Journal of Neuroscience*, 37(27):6503–6516, 2017.

[144] Sean E. Cavanagh, John P. Towers, Joni D. Wallis, Laurence T. Hunt, and Steven W. Kennerley. Reconciling persistent and dynamic hypotheses of working memory coding in prefrontal cortex. *Nature Communications*, 9(1):3498, dec 2018.

[145] Pierre Enel, Joni D. Wallis, and Erin L. Rich. Stable and dynamic representations of value in the prefrontal cortex. *eLife*, 9:1–23, 2020.

[146] Aishwarya Parthasarathy, Roger Herikstad, Jit Hon Bong, Felipe Salvador Medina, Camilo Libedinsky, and Shih Cheng Yen. Mixed selectivity morphs population codes in prefrontal cortex. *Nature Neuroscience*, 20(12):1770–1779, 2017.

[147] Kazuyoshi Takeda and Shintaro Funahashi. Prefrontal task-related activity representing visual cue location or saccade direction in spatial working memory tasks. *Journal of Neurophysiology*, 87(1):567–588, 2002.

[148] Cory R. Hussar and Tatiana Pasternak. Flexibility of Sensory Representations in Prefrontal Cortex Depends on Cell Type. *Neuron*, 64(5):730–743, 2009.

[149] Chadwick B. Boulay, Florian Pieper, Matthew Leavitt, Julio Martinez-Trujillo, and Adam J. Sachs. Single-trial decoding of intended eye movement goals from lateral prefrontal cortex neural ensembles. *Journal of Neurophysiology*, 115(1):486–499, 2016.

[150] Kelly R Bullock, Florian Pieper, Adam J Sachs, and Julio C Martinez-Trujillo. Visual and presaccadic activity in area 8Ar of the macaque monkey lateral prefrontal cortex. *Journal of Neurophysiology*, 118(1):15–28, 2017.

[151] Diego Mendoza-Halliday and Julio C. Martinez-Trujillo. Neuronal population coding of perceived and memorized visual features in the lateral prefrontal cortex. *Nature Communications*, 8:15471, jun 2017.

[152] G. Rainer, W. F. Asaad, and E. K. Miller. Selective representation of relevant information by neurons in the primate prefrontal cortex. *Nature*, 393(6685):577–579, 1998.

[153] Mikhail A. Lebedev, Adam Messinger, Jerald D. Kralik, and Steven P. Wise. Representation of attended versus remembered locations in prefrontal cortex. *PLoS Biology*, 2(11), 2004.

[154] Therese Lennert and Julio C. Martinez-Trujillo. Prefrontal neurons of opposite spatial preference display distinct target selection dynamics. *Journal of Neuroscience*, 33(22):9520–9529, 2013.

[155] Sébastien Tremblay, Florian Pieper, Adam Sachs, and Julio Martinez-Trujillo. Attentional Filtering of Visual Information by Neuronal Ensembles in the Primate Lateral Prefrontal Cortex. *Neuron*, 85(1):202–215, 2015.

[156] T. Pasternak, L. L. Lui, and P. M. Spinelli. Unilateral Prefrontal Lesions Impair Memory-Guided Comparisons of Contralateral Visual Motion. *Journal of Neuroscience*, 35(18):7095–7105, 2015.

[157] Theda Backen, Stefan Treue, and Julio C. Martinez-Trujillo. Encoding of spatial attention by primate prefrontal cortex neuronal ensembles. *eNeuro*, 5(1):1–19, 2018.

[158] Sofia Paneri and Georgia G. Gregoriou. Top-down control of visual attention by the prefrontal cortex. Functional specialization and long-range interactions. *Frontiers in Neuroscience*, 11(SEP):1–16, 2017.

[159] N. P. Bichot, A. F. Rossi, and R Desimone. Parallel and Serial Neural Mechanisms for Visual Search in Macaque Area V4. *Science*, 308(5721):529–534, apr 2005.

[160] Huihui Zhou and Robert Desimone. Feature-Based Attention in the Frontal Eye Field and Area V4 during Visual Search. *Neuron*, 70(6):1205–1217, 2011.

[161] Narcisse Bichot, Rui Xu, Azriel Ghadooshahy, Michael Williams, and Robert Desimone. The role of prefrontal cortex in the control

of feature attention in area V4. *Nature Communications*, 10(1):1–12, 2019.

[162] Vincent B. McGinty, Antonio Rangel, and William T. Newsome. Orbitofrontal Cortex Value Signals Depend on Fixation Location during Free Viewing. *Neuron*, 90(6):1299–1311, 2016.

[163] Megan Roussy, Rogelio Luna, Lyndon Duong, Benjamin Corrigan, Roberto Gulli, Ramon Nogueira, Ruben Moreno-Bote, Adam Sachs, Lena Palaniyappan, and Julio Martinez-Trujillo. Naturalistic coding of working memory in primate prefrontal cortex. *Molecular Psychiatry*, 2020.

[164] Benjamin W. Corrigan, Roberto A. Gulli, Guillaume Doucet, and Julio C. Martinez-Trujillo. Characterizing Eye Movement Behaviors and Kinematics of Non-Human Primates During Virtual Navigation Tasks. *Journal of Vision*, 17(2017):Forthcoming, oct 2018.

[165] Shintaro Funahashi. Functions of delay-period activity in the prefrontal cortex and mnemonic scotomas revisited. *Frontiers in Systems Neuroscience*, 9(FEB):1–14, 2015.

[166] Mattia Rigotti, Omri Barak, Melissa R. Warden, Xiao Jing Wang, Nathaniel D. Daw, Earl K. Miller, and Stefano Fusi. The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497(7451):585–590, 2013.

[167] Antonio H. Lara and Jonathan D. Wallis. The role of prefrontal cortex in working memory: A mini review. *Frontiers in Systems Neuroscience*, 9(DEC):1–7, 2015.

[168] Donatas Jonikaitis and Tirin Moore. The interdependence of attention, working memory and gaze control: behavior and neural circuitry. *Current Opinion in Psychology*, 29:126–134, 2019.

[169] Sandeep Robert Datta, David J. Anderson, Kristin Branson, Pietro Perona, and Andrew Leifer. Computational Neuroethology: A Call to Action. *Neuron*, 104(1):11–24, 2019.

[170] Ramon Nogueira, Juan M. Abolafia, Jan Drugowitsch, Emili Balaguer-Ballester, Maria V. Sanchez-Vives, and Rubén Moreno-Bote. Lateral orbitofrontal cortex anticipates choices and integrates prior with current information. *Nature Communications*, 8:14823, mar 2017.

[171] Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Classification of covariance matrices using a Riemannian-based kernel for BCI applications. *Neurocomputing*, 112:172–178, jul 2013.

[172] Md Sahidullah and Tomi Kinnunen. Local spectral variability features for speaker verification. *Digital Signal Processing*, 50:1–11, mar 2016.

[173] Michiel Hermans and Benjamin Schrauwen. Memory in reservoirs for high dimensional input. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, jul 2010.

[174] Igor Farkaš, Radomír Bosák, and Peter GergeÄ¾. Computational analysis of memory capacity in echo state networks. *Neural Networks*, 83:109–120, nov 2016.

[175] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[176] N. Hammami and M. Sellam. Tree distribution classifier for automatic spoken Arabic digit recognition. In *2009 International Conference for Internet Technology and Secured Transactions, (ICITST)*, pages 1–4. IEEE, nov 2009.

[177] Nacereddine Hammami and Mouldi Bedda. Improved tree model for arabic speech recognition. In *2010 3rd International Conference on Computer Science and Information Technology*, pages 521–526. IEEE, jul 2010.

[178] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2019.

[179] Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv*, 2018.

[180] Lyes Khacef, Laurent Rodriguez, and Benoît Miramond. Brain-inspired self-organization with cellular neuromorphic computing for multimodal unsupervised learning. *Electronics (Switzerland)*, 9(10):1–32, 2020.

[181] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, aug 1980.

[182] Mohammed Usman. On the Performance Degradation of Speaker Recognition System due to Variation in Speech Characteristics Caused by Physiological Changes. *International Journal of Computing and Digital Systemss*, 6(3):119–127, may 2017.

[183] Lyes Khacef, Laurent Rodriguez, and Benoit Miramond. Written and spoken digits database for multimodal learning, 2019.

[184] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout. Isolated word recognition with the Liquid State Machine: a case study. *Information Processing Letters*, 95(6):521–528, sep 2005.

[185] D. Verstraeten, B. Schrauwen, and D. Stroobandt. Reservoir-based techniques for speech recognition. In *The 2006 IEEE International*

Joint Conference on Neural Network Proceedings, pages 1050–1053. IEEE, 2006.

[186] Mark D. Skowronski and John G. Harris. Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, 20(3):414–423, apr 2007.

[187] Fabian Triefenbach, Azarakhsh Jalalvand, Benjamin Schrauwen, and Jean-pierre Martens. Phoneme Recognition with Large Hierarchical Reservoirs. In *Advances in Neural Information Processing Systems 23*, pages 2307—-2315. Curran Associates, Inc., 2010.

[188] Yong Zhang, Peng Li, Yingyezhe Jin, and Yoonsuck Choe. A Digital Liquid State Machine With Biologically Inspired Learning and Its Application to Speech Recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 26(11):2635–2649, nov 2015.

[189] Yingyezhe Jin and Peng Li. Performance and robustness of bio-inspired digital liquid state machines: A case study of speech recognition. *Neurocomputing*, 226:145–160, feb 2017.

[190] Naima Zerari, Samir Abdelhamid, Hassen Bouzgou, and Christian Raymond. Bidirectional deep architecture for Arabic speech recognition. *Open Computer Science*, 9(1):92–102, apr 2019.

[191] Flavio Abreu Araujo, Mathieu Riou, Jacob Torrejon, Sumito Tsunegi, Damien Querlioz, Kay Yakushiji, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Mark D. Stiles, and Julie Grollier. Role of non-linear data processing on speech recognition task in the framework of reservoir computing. *Scientific Reports*, 10(1):1–11, 2020.

[192] James B. Aimone. A Roadmap for Reaching the Potential of Brain Derived Computing. *Advanced Intelligent Systems*, 3(1), jan 2021.

[193] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. *The great multivariate time series*

classification bake off: a review and experimental evaluation of recent algorithmic advances, volume 35. Springer US, 2021.

[194] Diana Roca, Liang Zhao, Alex Choquenaira, Daniela Milón, and Roselli Romero. Echo State Network Performance Analysis Using Non-random Topologies. pages 133–146. 2021.

[195] Matthias Freiberger, Peter Bienstman, and Joni Dambre. A training algorithm for networks of high-variability reservoirs. *Scientific Reports*, 10(1):1–11, 2020.

[196] Claudio Gallicchio and Alessio Micheli. Reservoir Topology in Deep Echo State Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11731 LNCS:62–75, 2019.

[197] Guillermo B. Morales, Claudio R. Mirasso, and Miguel C. Soriano. Unveiling the role of plasticity rules in reservoir computing. 2021.

[198] Wei Chen and Ke Shi. Multi-scale Attention Convolutional Neural Network for time series classification. *Neural Networks*, 136:126–140, apr 2021.

[199] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), 2021.

[200] Cory Stephenson, Jenelle Feather, Suchismita Padhy, Oguz Elibol, Hanlin Tang, Josh McDermott, and Sue Yeon Chung. Untangling in invariant speech recognition. *arXiv*, (NeurIPS), 2020.

[201] Ramon Nogueira, Sofía Lawrie, and Rubén Moreno-Bote. Neuronal Variability as a Proxy for Network State. *Trends in Neurosciences*, 41(4):170–173, apr 2018.

[202] Lubomir Kostal, Petr Lansky, and Jean Pierre Rospars. Neuronal coding and spiking randomness. *European Journal of Neuroscience*, 26(10):2693–2701, 2007.

[203] Rubén Moreno-Bote. Poisson-Like Spiking in Circuits with Probabilistic Synapses. *PLoS Computational Biology*, 10(7):e1003522, jul 2014.

[204] Michael N. Shadlen and J.Anthony Movshon. Synchrony Unbound. *Neuron*, 24(1):67–77, sep 1999.

[205] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[206] Witali Aswolinskiy, René Felix Reinhart, and Jochen Steil. Time Series Classification in Reservoir- and Model-Space: A Comparison. pages 197–208. 2016.

[207] Yuji Kawai, Jihoon Park, and Minoru Asada. A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks*, 112:15–23, 2019.

[208] Joschka Boedecker, Oliver Obst, Joseph T. Lizier, N. Michael Mayer, and Minoru Asada. Information processing in echo state networks at the edge of chaos. *Theory in Biosciences*, 131(3):205–213, sep 2012.

[209] Nils Schaetti, Michel Salomon, and Raphael Couturier. Echo State Networks-Based Reservoir Computing for MNIST Handwritten Digits Recognition. In *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pages 484–491. IEEE, aug 2016.

[210] Abdulrahman Alalshekmubarak. *Towards A Robust Arabic Speech Recognition System Based On Reservoir Computing*. PhD thesis, University of Stirling, 2014.

[211] Benjamin Liebald. *Exploration of effects of different network topologieson the esn signal crosscorrelation matrix spectrum.* Bachelor thesis, University of Bremen, 2004.

[212] Ali Ajdari Rad. Dynamical Networks (miniproject) Effect of Topology of the Reservoir on Performance of Echo State Networks, 2008.

[213] S. Haeusler and W. Maass. A Statistical Analysis of Information-Processing Properties of Lamina-Specific Cortical Microcircuit Models. *Cerebral Cortex*, 17(1):149–162, feb 2006.

[214] Laura E. Suárez, Ross D. Markello, Richard F. Betzel, and Bratislav Misic. Linking Structure and Function in Macroscale Brain Networks. *Trends in Cognitive Sciences*, 24(4):302–315, apr 2020.

[215] Fabrizio Damicelli, Claus C Hilgetag, and Alexandros Goulas. Brain Connectivity meets Reservoir Computing. *bioRxiv*, page 2021.01.22.427750, 2021.

[216] Thomas Miconi. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife*, 6, feb 2017.

[217] Christian Klos, Yaroslav Felipe Kalle Kossio, Sven Goedeke, Aditya Gilra, and Raoul-Martin Memmesheimer. Dynamical Learning of Dynamics. *Physical Review Letters*, 125(8):088103, aug 2020.

[218] Robert Gütig and Haim Sompolinsky. The tempotron: A neuron that learns spike timing-based decisions. *Nature Neuroscience*, 9(3):420–428, 2006.

[219] Filip Ponulak and Andrzej Kasiński. Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting. *Neural Computation*, 22(2):467–510, feb 2010.

[220] Jean Ginibre. Statistical Ensembles of Complex, Quaternion, and Real Matrices. *Journal of Mathematical Physics*, 6(3):440–449, mar 1965.

[221] Matthew F. Panichello and Timothy J. Buschman. Shared mechanisms underlie the control of working memory and attention. *Nature*, 592(7855):601–605, 2021.

[222] Daniel J. Gauthier, Erik Bollt, Aaron Griffith, and Wendson A.S. Barbosa. Next generation reservoir computing. *Nature Communications*, 12(1):1–8, 2021.

[223] Sylvia Wirth, Pierre Baraduc, Aurélie Planté, Serge Pinède, and Jean René Duhamel. Gaze-informed, task-situated representation of space in primate hippocampus during virtual navigation. *PLoS Biology*, 15(2):1–28, 2017.

[224] Roberto A. Gulli, Lyndon R. Duong, Benjamin W. Corrigan, Guillaume Doucet, Sylvain Williams, Stefano Fusi, and Julio C. Martinez-Trujillo. Context-dependent representations of objects and space in the primate hippocampus during virtual navigation. *Nature Neuroscience*, 23(1):103–112, 2020.

[225] Jean Jacques Orban de Xivry, Sébastien Coppe, Gunnar Blohm, and Philippe Lefèvre. Kalman filtering naturally accounts for visually guided and predictive smooth pursuit dynamics. *Journal of Neuroscience*, 33(44):17301–17313, 2013.

[226] Trishna Mukherjee, Bing Liu, Claudio Simoncini, and Leslie C. Osborne. Spatiotemporal filter for visual motion integration from pursuit eye movements in humans and monkeys. *Journal of Neuroscience*, 37(6):1394–1412, 2017.

[227] Alexander Huk, Kathryn Bonnen, and Biyu J. He. Beyond trial-based paradigms: Continuous behavior, ongoing neural activity, and natural stimuli. *Journal of Neuroscience*, 38(35):7551–7558, 2018.

[228] David J. Anderson and Pietro Perona. Toward a science of computational ethology. *Neuron*, 84(1):18–31, 2014.

[229] Alex Gomez-Marin, Joseph J. Paton, Adam R. Kampff, Rui M. Costa, and Zachary F. Mainen. Big behavioral data: Psychology, ethology and the foundations of neuroscience. *Nature Neuroscience*, 17(11):1455–1462, 2014.

[230] Mackenzie Weygandt Mathis and Alexander Mathis. Deep learning tools for the measurement of animal behavior in neuroscience. *Current Opinion in Neurobiology*, 60:1–11, feb 2020.

[231] Aishwarya Parthasarathy, Cheng Tang, Roger Herikstad, Loong Fah Cheong, Shih Cheng Yen, and Camilo Libedinsky. Time-invariant working memory representations in the presence of code-morphing in the lateral prefrontal cortex. *Nature Communications*, 10(1):1–11, 2019.

[232] Pascal Fries. Rhythms for Cognition: Communication through Coherence. *Neuron*, 88(1):220–235, 2015.

[233] Benjamin Voloh, Mariann Oemisch, and Thilo Womelsdorf. Phase of firing coding of learning variables across the fronto-striatal network during feature-based learning. *Nature Communications*, 11(1):1–16, 2020.

[234] Stefano Panzeri, Nicolas Brunel, Nikos K. Logothetis, and Christoph Kayser. Sensory neural codes using multiplexed temporal scales. *Trends in Neurosciences*, 33(3):111–120, 2010.

[235] Stéphanie Ratté, Sungho Hong, Erik DeSchutter, and Steven A. Prescott. Impact of neuronal properties on network coding: Roles of spike initiation dynamics and robust synchrony transfer. *Neuron*, 78(5):758–772, 2013.

[236] Milad Lankarany, Dhekra Al-Basha, Stéphanie Ratté, and Steven A. Prescott. Differentially synchronized spiking enables

multiplexed neural coding. *Proceedings of the National Academy of Sciences of the United States of America*, 116(20):10097–10102, 2019.

[237] József Fiser, Pietro Berkes, GergÅ Orbán, and Máté Lengyel. Statistically optimal perception and learning: from behavior to neural representations. *Trends in Cognitive Sciences*, 14(3):119–130, mar 2010.

[238] GergÅ Orbán, Pietro Berkes, József Fiser, and Máté Lengyel. Neural Variability and Sampling-Based Probabilistic Representations in the Visual Cortex. *Neuron*, 92(2):530–543, 2016.

[239] Dylan Festa, Amir Aschner, Aida Davila, Adam Kohn, and Ruben Coen-Cagli. Neuronal variability reflects probabilistic inference tuned to natural image statistics. *Nature Communications*, 12(1):1–11, 2021.

[240] Luís F. Seoane. Evolutionary aspects of reservoir computing. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 374(1774), 2019.

[241] Suzana Herculano-Houzel. The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. *Proceedings of the National Academy of Sciences of the United States of America*, 109(SUPPL.1):10661–10668, 2012.

[242] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The Computational Limits of Deep Learning. 2020.

[243] Bharat Biswal, F. Zerrin Yetkin, Victor M. Haughton, and James S. Hyde. Functional connectivity in the motor cortex of resting human brain using echo-planar mri. *Magnetic Resonance in Medicine*, 34(4):537–541, oct 1995.

[244] J. L. Vincent, G. H. Patel, M. D. Fox, A. Z. Snyder, J. T. Baker, D. C. Van Essen, J. M. Zempel, L. H. Snyder, M. Corbetta, and M. E. Raichle. Intrinsic functional architecture in the anaesthetized monkey brain. *Nature*, 447(7140):83–86, 2007.

[245] A. Mitra, A. Z. Snyder, C. D. Hacker, and M. E. Raichle. Lag structure in resting-state fMRI. *Journal of Neurophysiology*, 111(11):2374–2391, 2014.

[246] Maria V. Sanchez-Vives and David A. McCormick. Cellular and network mechanisms of rhytmic recurrent activity in neocortex. *Nature Neuroscience*, 3(10):1027–1034, 2000.

[247] Rosa Cossart, Dmitriy Aronov, and Rafael Yuste. Attractor dynamics of network UP states in the neocortex. *Nature*, 423(6937):283–288, 2003.

[248] André Grüning and Sander M. Bohte. Spiking neural networks: Principles and challenges. *22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2014 - Proceedings*, (April):1–10, 2014.

[249] Michael Pfeiffer and Thomas Pfeil. Deep Learning With Spiking Neurons: Opportunities and Challenges. *Frontiers in Neuroscience*, 12(October), 2018.

[250] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[251] Wenzhe Guo, Mohammed E. Fouda, Ahmed M. Eltawil, and Khaled Nabil Salama. Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems. *Frontiers in Neuroscience*, 15(March):1–21, 2021.

[252] Benjamin Schrauwen and Jan Van Campenhout. BSA, a Fast and Accurate Spike Train Encoding Scheme. *Proceedings of the International Joint Conference on Neural Networks*, 4:2825–2830, 2003.

[253] Aboozar Taherkhani, Ammar Belatreche, Yuhua Li, Georgina Cosma, Liam P. Maguire, and T. M. McGinnity. A review of learning in biologically plausible spiking neural networks. *Neural Networks*, 122:253–272, 2020.

[254] Sander M. Bohte, Joost N. Kok, and Han La Poutré. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

[255] RÇzvan V. Florian. The chronotron: A neuron that learns to fire temporally precise spike patterns. *PLoS ONE*, 7(8), 2012.