Covariance-based information processing in reservoir computing systems

Sofía Lawrie[1,*], Rubén Moreno-Bote[1,2] and Matthieu Gilson[3,1,*]

[1] *Center for Brain and Cognition, and Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain*
[2] *Serra Hunter Fellow Programme, Universitat Pompeu Fabra, Barcelona, Spain*
[3] *Institute of Neuroscience and Medicine (INM-6) and Institute for Advanced Simulation (IAS-6) and JARA Institute Brain Structure-Function Relationships (INM-10), Jülich Research Centre, Jülich, Germany*
[*] *Corresponding authors: sofia.lawrie@upf.edu, m.gilson@fz-juelich.de*

April 30, 2021

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 tools. 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. We expose nontrivial relationships between input, reservoir and output dynamics, which suggest an important role for recurrent connectivity in transforming information representations in biologically inspired architectures. Finally, we solve a real automatic speech recognition task for the classification of spoken digits to further demonstrate the potential of covariance-based decoding.

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 (Shadlen and Newsome, 1998; Renart and Machens, 2014; Nogueira et al., 2018). The role that trial-by-trial variability plays in information processing in neuronal networks is currently under scrutiny, after early considerations that saw it as detrimental to robust (de)coding and learning (Stein et al., 2005; Kostal et al., 2007), but were later proven to be not always the case (Gilson et al., 2011; Boerlin et al., 2013; Moreno-Bote, 2014). In particular, it has recently been shown

that structured variability, corresponding to reproducible correlation patterns, can be a substrate for robust information processing (Gilson et al., 2020). 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'.

Covariance coding offers a middle ground alternative between the much-debated rate coding and temporal coding theories (Bair and Koch, 1996; Shadlen and Movshon, 1999; Brette, 2015). In the new view, neither the mean nor the full probability distribution of neuronal activity is used for information transmission, but rather the pairwise correlation between neurons, as quantified by second-order statistics. Nonetheless, the question about which type of patterns in neuronal activity –understood here as which statistical order– are informative for stimuli and behavior has just started to be experimentally investigated (Panzeri et al., 2017; Shahidi et al., 2019). Altogether, in the covariance-based paradigm, second-order statistics become a resource for computation, enabling neurons to perform tasks like the classification of multivariate time series.

Although linear systems like the covariance perceptron are able to achieve this (Gilson et al., 2020), it has been shown that the computational power of neuronal networks can be dramatically increased by nonlinearities, for example in deep networks (Hornik, 1991; LeCun et al., 2015). The reason behind this comes from the fact that nonlinearities allow to combine several statistical orders of the inputs. Among the computational systems that exploit this trait, reservoir computing has turned out to be a powerful solution for pattern classification: a randomly connected and untrained recurrent layer of nonlinear units (the reservoir) is used to filter an incoming signal before passing it to a readout layer (Jaeger, 2001; Maass et al., 2002; Lukoševičius and Jaeger, 2009; Tanaka et al., 2019). The high-dimensional space given by the reservoir provides rich dynamics where information about different statistical orders of the input mixes and can in turn collectively shape the dynamics of the outputs through linear connections. The specific values for these connections are found by minimizing a mean-squared error between output activity and target activity as cost function, thus producing in classification tasks an output signal that is shaped at the first-order (Jaeger, 2001; Jaeger et al., 2007; Verstraeten et al., 2007; Lukoševičius and Jaeger, 2009).

In this study we explore the effects of combining a high-dimensional nonlinear input expansion with a covariance-based information processing scheme in the classification performance of multivariate time series. With this aim, we work with reservoir computing systems where the output signal is shaped at the second-order, by minimizing a cost function between output covariances and target covariances. Even though a wide variety of reservoir implementations exist (Tanaka et al., 2019), our study focuses on echo state networks (Jaeger, 2001; Jaeger et al., 2007; Verstraeten et al., 2007; Lukoševičius and Jaeger, 2009), which employ analogue sigmoid neurons, since they provide a formalism that is close to the covariance perceptron.

We show how to exploit the cross-talk between the first and second statistical orders of input times series in the activity of echo state networks. In practice, we investigate how reservoir nonlinearities allow to mix spatial (zero-lag) information, and how recurrent connections are relevant to transform temporal (one-lag) information into spatial (zero-lag) information. Note that we use the term 'information' in a colloquial manner in this study, without a specific reference to information theory. We look for the best strategy for classification based on first or second order decoding when the input information is embedded either in the first or second order, or mixed in both. We show that second-order decoding is robust in a scheme where information must propagate

from input receptor to output feeder reservoir neurons that are segregated. Finally, we apply the covariance framework to classify time series of spoken Arabic digits. Our work introduces alternatives to the standard reservoir computing readouts and takes a step further towards theoretically investigating second-order information processing in more realistic neuronal networks.

The manuscript is structured as follows. Section 2 is dedicated to methods: we describe the datasets that we work with (synthetic and real), reservoir configurations and the pipeline that we use for classification of multivariate time series. Section 3 presents our results and Section 4 is dedicated to discussion.

2 Methods

2.1 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. In all the equations presented in 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.

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

- vector of mean activity $p \in \mathbb{R}^M$, with elements $p_k = \langle u_k^t \rangle_d = \frac{1}{d} \sum_{t=1}^{d} u_k^t$;
- zero-lag covariance matrix $P^0 \in \mathbb{R}^{M \times M}$, with elements $P^0_{kl} = cov(u_k^t,\ u_l^t)_d = \langle u_k^t u_l^t \rangle_d - \langle u_k^t \rangle_d \langle u_l^t \rangle_d = \frac{1}{d} \sum_{t=1}^{d} u_k^t u_l^t - \frac{1}{d^2} \sum_{t=1}^{d} u_k^t \sum_{t=1}^{d} u_l^t$;
- one-lag covariance matrix $P^1 \in \mathbb{R}^{M \times M}$, with elements $P^1_{kl} = cov(u_k^t,\ u_l^{t+1})_d = \langle u_k^t u_l^{t+1} \rangle_d - \langle u_k^t \rangle_d \langle u_l^{t+1} \rangle_d = \frac{1}{d} \sum_{t=1}^{d} u_k^t u_l^{t+1} - \frac{1}{d^2} \sum_{t=1}^{d} u_k^t \sum_{t=1}^{d} u_l^{t+1}$.

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. 1A). 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 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 decoder (details in Section 2.4) directly on the input sample time series (see top pathway in Fig. 1B). 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 improvements and decreases across different decoding schemes and models.

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

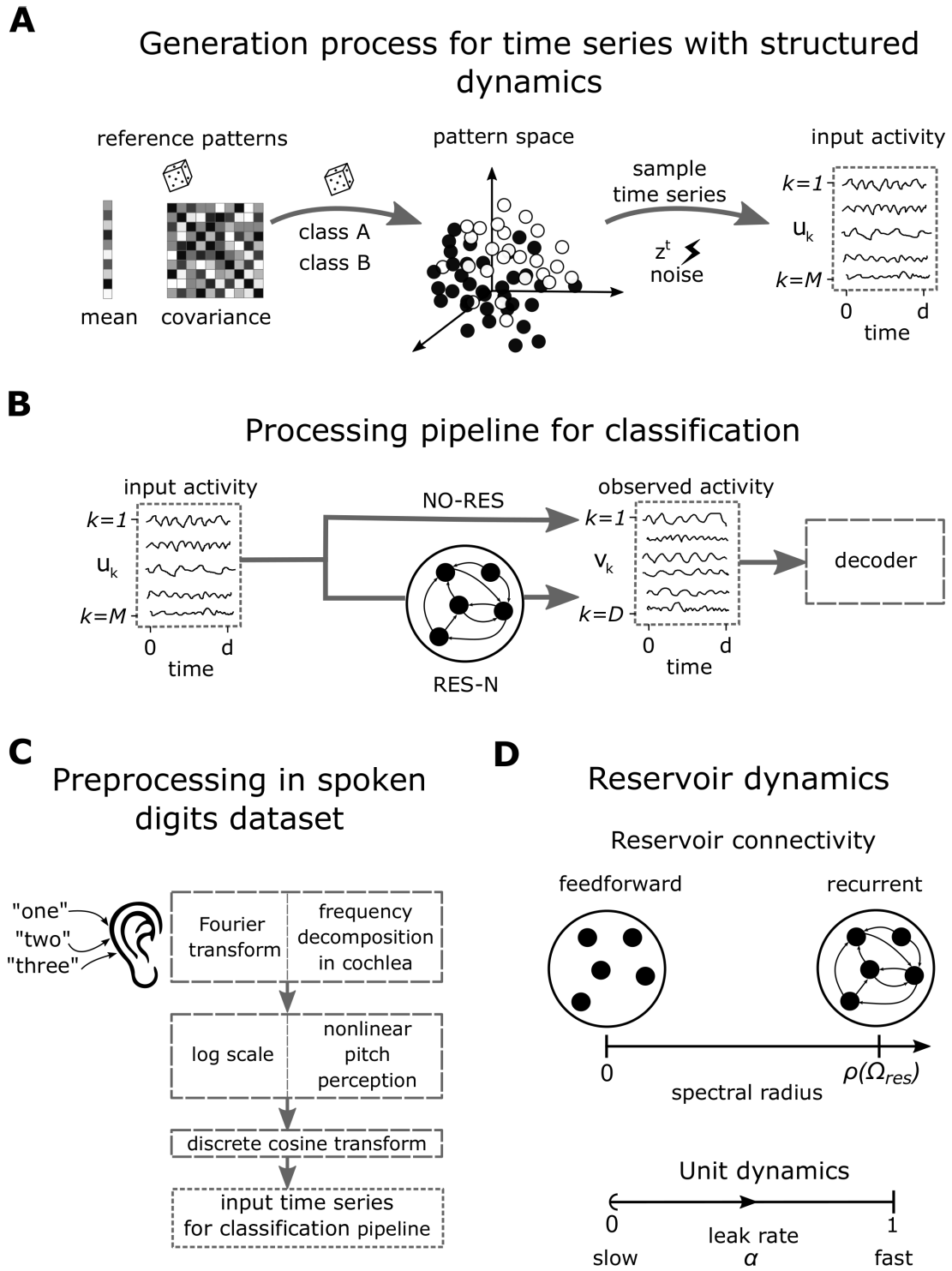

Figure 1: **Description of input datasets and reservoir processing. A:** For synthetic inputs, we first sample reference patterns from given probability distributions. Afterwards, we randomly split them in two balanced classes. To generate a sample time series for a given pattern, we add noise at each time point through specific dynamics (see Eqs. 1-4).

Figure 1 *(previous page)*: **B:** An M-dimensional input time series observed for d time steps is either filtered by a reservoir layer of N neurons ('RES-N' bottom pathway) or directly fed to a linear decoder ('NO-RES', top pathway) to perform a classification task. The top pathway was studied in (Gilson et al., 2020), and we implement it here for comparison purposes. **C:** 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 (Hammami and Sellam, 2009; Dua and Graff, 2019). **D:** The reservoir dynamics are characterized by the spectral radius $\rho(\Omega_{res})$, which is the largest absolute eigenvalue of the reservoir's connectivity matrix. By convention, a reservoir with $\rho(\Omega_{res}) = 0$ means $\Omega_{res} = 0$ and corresponds to a feed-forward layer without recurrent connectivity. We also modulate the reservoir processing via the local units' dynamics by adding a leak parameter α. 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.

2.1.1 Mean or first-order structure

We use the following generative process for the time series:

$$u^t = p + z^t, \tag{1}$$

where $z^t \in \mathbb{R}^M$ is a normally distributed random variable, with zero-mean and diagonal covariances equal to 1. To characterize the mean activity of u^t, we use a pattern vector $p \in \mathbb{R}^M$, 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. 1A. Once defined the patterns for the two classes, we generate noisy samples by using Eq. 1 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.

2.1.2 Spatial covariance or second-order zero-lag structure

We use the following generative process for the time series:

$$u^t = W z^t, \tag{2}$$

where $z^t \in \mathbb{R}^M$ is, as before, a standard normal random variable and $W \in \mathbb{R}^{M \times M}$ 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 $P^0 = WW^T$

(Gilson et al., 2020). 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.

2.1.3 Temporal covariance or second-order one-lag structure

We use the following generative process for the time series:

$$u^t = Wu^{t-1} + z^t, \tag{3}$$

where we choose the mixing matrix $W = \exp(\beta\mathcal{I}_{M\times M} + J)$, with parameter $\beta < 0$, $\mathcal{I}_{M\times M} \in \mathbb{R}^{M\times M}$ is the identity matrix and $J \in \mathbb{R}^{M\times M}$ 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 $P^1 = WP^0$ (Gilson et al., 2020).

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.

2.1.4 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. 1 and 2:

$$u^t = p + Wz^t. \tag{4}$$

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 2.1.1 and 2.1.2 respectively.

2.2 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 (Hammami and Sellam, 2009; Hammami and Bedda, 2010; Dua and Graff, 2019). 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) (Davis and Mermelstein, 1980), sampled at 11025 Hz. MFCC constitute a widely used feature for tasks such as speech recognition (Usman, 2017). 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. 1C, 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 (Jaeger et al., 2007), but we chose to not follow this approach.

2.3 Reservoir implementation

The reservoir used in the classification pipeline ('RES-N' in Fig. 1B, bottom pathway) is an echo state network with N leaky integrator neurons (or units), similar to previous studies (Jaeger et al., 2007; Lukoševičius and Jaeger, 2009). The update equations for the activity state at time t of the N leaky neurons inside the reservoir, denoted by $x^t_{res} \in \mathbb{R}^N$, are

$$\bar{x}^t_{res} = \mathcal{F}(\Omega_{in} u^t + \Omega_{res} x^{t-1}_{res}), \quad (5)$$

$$x^t_{res} = (1-\alpha) x^{t-1}_{res} + \alpha \bar{x}^t_{res}, \quad (6)$$

where the function $\mathcal{F} = \tanh$ has a sigmoidal profile, $\Omega_{in} \in \mathbb{R}^{N \times M}$ is the connection matrix from the input time series to the reservoir units and $\Omega_{res} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of recurrent connections within the reservoir. All connection weights are randomly sampled from $[-0.5, 0.5)$, and the resulting adjacency 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 (Jaeger et al., 2007; Lukoševičius and Jaeger, 2009). When $\alpha = 1$, the activity of the reservoir (Eq. 6) is $x^t_{res} = \bar{x}^t_{res}$, 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. 1D. The spectral radius $\rho(\Omega_{res})$ is the largest absolute eigenvalue of the reservoir's adjacency matrix Ω_{res}. It affects the reservoir performance in different benchmark tasks typically reported in the literature, such

as memory (Jaeger, 2001; Hermans and Schrauwen, 2010; Farkaš et al., 2016). A general heuristic when $\alpha = 1$ is that $\rho(\Omega_{res})$ should approximate 1 (from below) for tasks that require long memory and be smaller for tasks where a too long memory might be detrimental (Lukoševičius and Jaeger, 2009). 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, in the linear regime, the reservoir is chaotic, in the sense that its trajectory will deviate away from the zero fixed-point (Verstraeten and Schrauwen, 2009). However, in practice, the sigmoid function bounds the growth of the trajectory and effectively produces a reservoir that is dynamically less excitable. This dynamics could also be modulated by an input, since it drives the reservoir away from the linear regime. Note that, by convention, a null spectral radius implies a zero adjacency ($\Omega_{res} = 0$), corresponding to a feedforward layer (left-hand side in Fig. 1D). Thus, $\alpha = 1$ and $\Omega_{res} = 0$ 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: $\alpha\Omega_{res} + (1-\alpha)\mathcal{I}_{N\times N}$, instead of Ω_{res} (Jaeger et al., 2007); here $\mathcal{I}_{N\times N}$ 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.

2.3.1 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 (Kawai et al., 2019). 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 2.1.3, 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 adjacency matrix elements are uniformly sampled from $[-0.5, 0.5)$. For the symmetric (asymmetric) matrices, we only sample elements $\Omega_{res,ij}$ for the upper triangular part, while the lower triangular elements are assigned to fulfill the symmetry (asymmetry) condition $\Omega_{res,ji} = \Omega_{res,ij}$ ($\Omega_{res,ji} = -\Omega_{res,ij}$).

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.

2.3.2 Reservoirs propagate both mean and covariance spatial structures, which are mixed by nonlinearities

We here show that the statistics of the neurons inside the reservoir are affected by input statistics with different leading orders. This property only depends on the pointwise nonlinear behavior of the reservoir (i.e. the sigmoidal function), so we will consider that the network is feedforward ($\Omega_{res} = 0$) and no leaky integration is performed ($\alpha = 1$) (see Eqs. 5 and 6).

Let u^t be the multivariate input time series, including a constant unit bias term, that is fed to the reservoir by means of the matrix Ω_{in}. The activity at time t of neuron i inside the reservoir can then be approximated by a Taylor series expansion when driven close to the linear regime:

$$x^t_{res,i} = \tanh(\Omega_{in,ik} u^t_k) \approx \Omega_{in,ik} u^t_k + \frac{1}{3}\Omega_{in,ik}\Omega_{in,il}\Omega_{in,im} u^t_k u^t_l u^t_m, \tag{7}$$

where we use the Einstein notation convention where repeated indexes are summed over and we have used that $\tanh(z) \approx z + \frac{z^3}{3}$ when $z \approx 0$. We remark that $x^t_{res,i}$ would not depend on input pairwise interactions if it were not because of the inclusion of a bias term as one of the components of u^t. If all entries in $\Omega_{in} \approx \mathcal{O}(\epsilon)$, with $\epsilon < 1$ since they are sampled from $[-0.5, 0.5)$, then $x^t_{res,i}$ has a leading term dependence with input pairwise interaction of $\mathcal{O}(\epsilon^3)$ and a leading term of input dependence of $\mathcal{O}(\epsilon)$.

Following the same approach, we can see that the pairwise interaction between two neurons $x^t_{res,i}$ and $x^t_{res,j}$ inside the reservoir is given by:

$$\begin{aligned} x^t_{res,i} x^t_{res,j} \approx & (\Omega_{in,ik} u^t_k + \frac{1}{3}\Omega_{in,ik}\Omega_{in,il}\Omega_{in,im} u^t_k u^t_l u^t_m)(\Omega_{in,jn} u^t_n \\ & + \frac{1}{3}\Omega_{in,jn}\Omega_{in,js}\Omega_{in,jr} u^t_n u^t_s u^t_r). \end{aligned} \tag{8}$$

Expanding the above product, from the first term involving $\Omega_{in,ik}\Omega_{in,jn} u^t_k u^t_n$, we can see that the pairwise interactions inside the reservoir depend on input pairwise interaction with $\mathcal{O}(\epsilon^2)$. The same $\mathcal{O}(\epsilon^2)$ dependence holds for the dependence on input u^t, since we recover it when one of the input components is the bias term.

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 second-order statistics. However, when the information is embedded in the zero-lag covariances, the above derivation suggests that both the first and second-order statistics of the reservoir should be equally good at representing it. This constitutes a strong departure from simple linear networks, where covariance-based information representations are only possible if indeed such a representation is present in the inputs (Gilson et al., 2020).

2.3.3 Analysis of linearity of units' regime in the reservoir

The analytical insights previously developed (Section 2.3.2) are valid only in the linear dynamical regime. To assess the practical validity of this assumption, we numerically compute the probability of finding a neuron in the linear activation state, as a function of spectral radius and leak rate. The activation regime is considered linear if at time

t, the absolute value of the argument inside $\mathcal{F} = \tanh$ in Eq. 5 is less than or equal to 1, given that in this region the linear approximation to the hyperbolic tangent is acceptable.

2.4 Decoders

We have four different learning and decoding schemes, as outlined in Fig. 2A: mean-MLR, mean-LP, cov-MLR and cov-LP, 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 $v^t \in \mathbb{R}^D$ the observed activity in Fig. 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 $\mathcal{S}(v^t) = \frac{1}{d}\sum_{1\leq t\leq d} v^t$. The prefix 'cov' indicates that the decoder relies on the matrix of zero-lag (or one-lag) covariances of the observed activity, with matrix elements $\mathcal{S}(v^t) = \frac{1}{d}\sum_{1\leq t\leq d} v^t v^{t\top} - \frac{1}{d^2}\sum_{1\leq t\leq d} v^t \sum_{1\leq t\leq d} v^{t\top}$ with the superscript $\top$ indicating the matrix transpose. In each case, we consider two options for the classifier: a (multinomial) logistic regression classifier (Bishop, 2006) (MLR in Fig. 2A) or a linear perceptron (Rosenblatt, 1958; Bishop, 2006) (LP in Fig. 2A). Our main focus is the performance of the LP and we implement the MLR decoder for comparison.

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 $y^t_{out} \in \mathbb{R}^K$ at each time step, and the statistical moments for classification are computed for this vector when the observation period is over (bottom pathway in Fig. 2B). Thus, the output activity at time t is given by:

$$y^t_{out} = \Omega_{out} v^t, \tag{9}$$

where $\Omega_{out} \in \mathbb{R}^{K\times D}$ 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 (Gilson et al., 2020).

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 (top pathway in Fig. 2B). To do so, the MLR produces a single output vector $y_{out} \in \mathbb{R}^K$ 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:

$$y_{out} = \exp(\Omega_{out}\mathcal{S}(v^t)). \tag{10}$$

In Eq. 10, $\mathcal{S}(v^t)$ 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_{out} sum to 1. The class is then

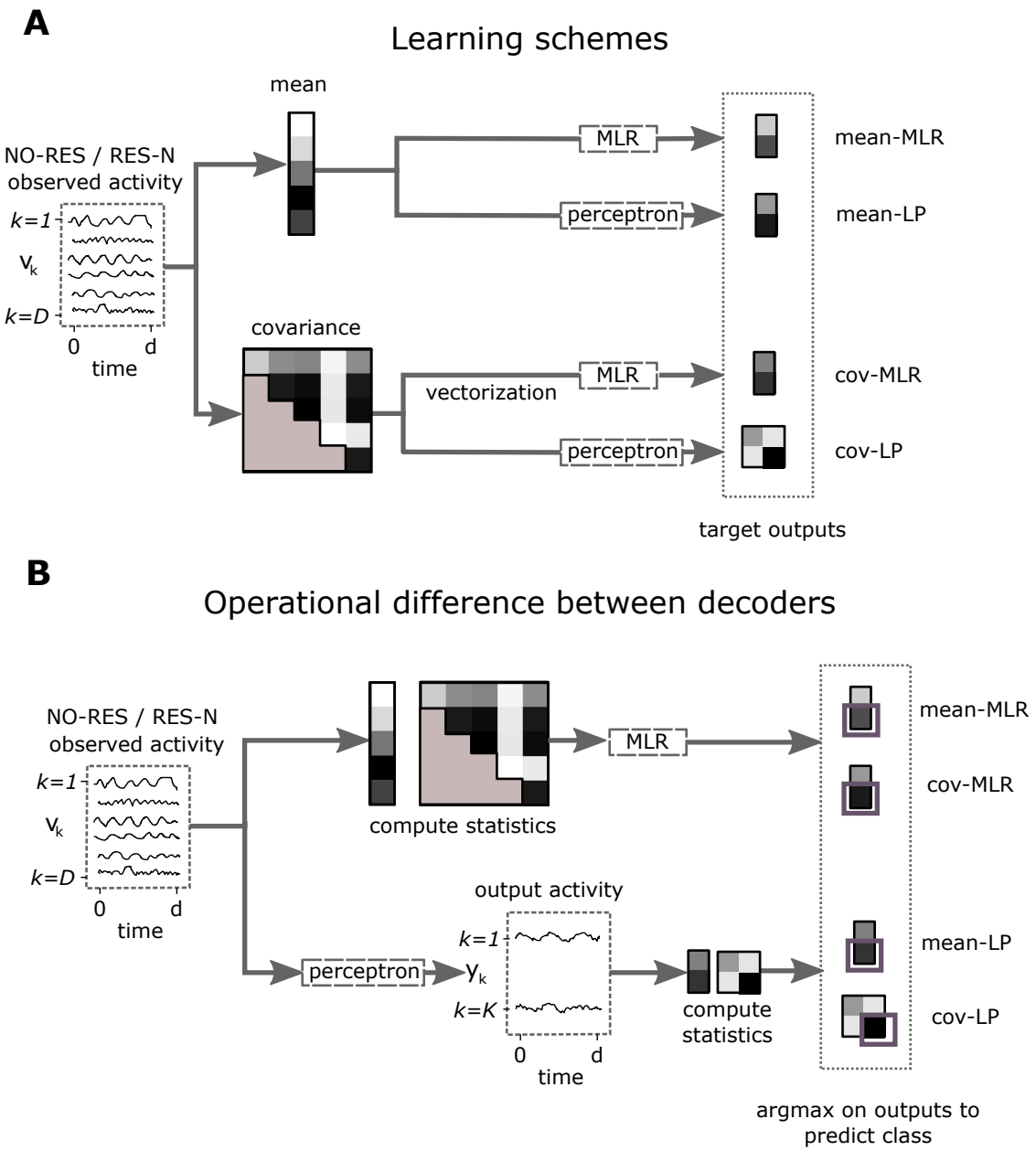

Figure 2: **Decoder properties. A:** The classification of the input time series is based on observed activity, either coming from the reservoir ('RES-N') or directly from the inputs ('NO-RES') in Fig. 1B. During learning, this activity has its mean or covariance statistic mapped to the mean or covariance statistic of the output layer of a linear perceptron or is instead directly classified using a multinomial logistic regression classifier that outputs class probability vectors. Thus, we have the possibility of implementing four different decoders defined by the statistic and the classifier. **B:** There is an important operational difference between the linear perceptron and the MLR, given by the order in which the statistics are computed for classification. MLR 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 are afterwards processed to compute a statistic for classification.

given by the index of the maximum element in y_{out}. 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. 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. 2B).

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 2.3.2).

2.4.1 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 $m = \langle y^t_{out} \rangle_d$ and a target output mean activity $\bar{y} \in \mathbb{R}^K$ during the observation period d:

$$C = \frac{1}{2} \sum_{k=1}^{K} (m_k - \bar{y}_k)^2 + \frac{\lambda}{2} \sum_{k=1}^{K} \sum_{i=1}^{D} \Omega^2_{out,ki}. \tag{11}$$

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 (Gilson et al., 2020), which corresponds to the common winner-take-all readouts typically used in reservoir computing applications for classification (Verstraeten et al., 2007, 2006; Skowronski and Harris, 2007; Jaeger et al., 2007). We use the scikit-learn library (Pedregosa et al., 2011) 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 $Y^0 \in \mathbb{R}^{K \times K}$ to target spatial covariances $\bar{Y}^0 \in \mathbb{R}^{K \times K}$ (Gilson et al., 2020):

$$C = \frac{1}{2} \sum_{k=1}^{K} (Y^0_{k,k} - \bar{Y}^0_{k,k})^2. \tag{12}$$

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 (Gilson

et al., 2020), where the weight updates $\Delta\Omega_{out,ik}$ for the connection between element k in the output and element i in the observed activity are given by:

$$\Delta\Omega_{out,ik} = \eta(\bar{Y}^0 - Y^0) \odot (G^{ik} V^0 \Omega_{out}^T + \Omega_{out} V^0 {G^{ik}}^T), \tag{13}$$

where η is the learning rate, $V^0 \in \mathbb{R}^{D\times D}$ is the spatial covariance matrix of the observed activity and $G^{ik} \in \mathbb{R}^{K\times D}$ has 0s everywhere except in element (i, k) that is equal to 1. Symbol $\odot$ denotes the 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(Bishop, 2006). For this, we use the scikit-learn library (Pedregosa et al., 2011).

2.4.2 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 (Bishop, 2006). 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 spoken digits with 13 MFCC coefficients as inputs, we use $S = 4$ and $S = 5$.

3 Results

The purpose of our study is the comparison between mean-based and (co)variance-based 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). This allows 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 (Jaeger, 2001; Farkaš et al., 2016; Boedecker et al., 2012; Schaetti et al., 2016). In addition, we explore the influence of the reservoir topology by comparing the standard random reservoir and a segregated reservoir (see Section 2.3) 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, which is a practical problem where reservoir computing has been efficiently applied (Verstraeten et al., 2005, 2006, 2007; Skowronski and Harris, 2007; Jaeger et al., 2007; Triefenbach et al., 2010; Alalshekmubarak and Smith, 2014; Zhang et al., 2015; Jin and Li, 2017).

3.1 Nonlinear reservoirs with covariance-based readouts can efficiently extract second-order statistics from input time series

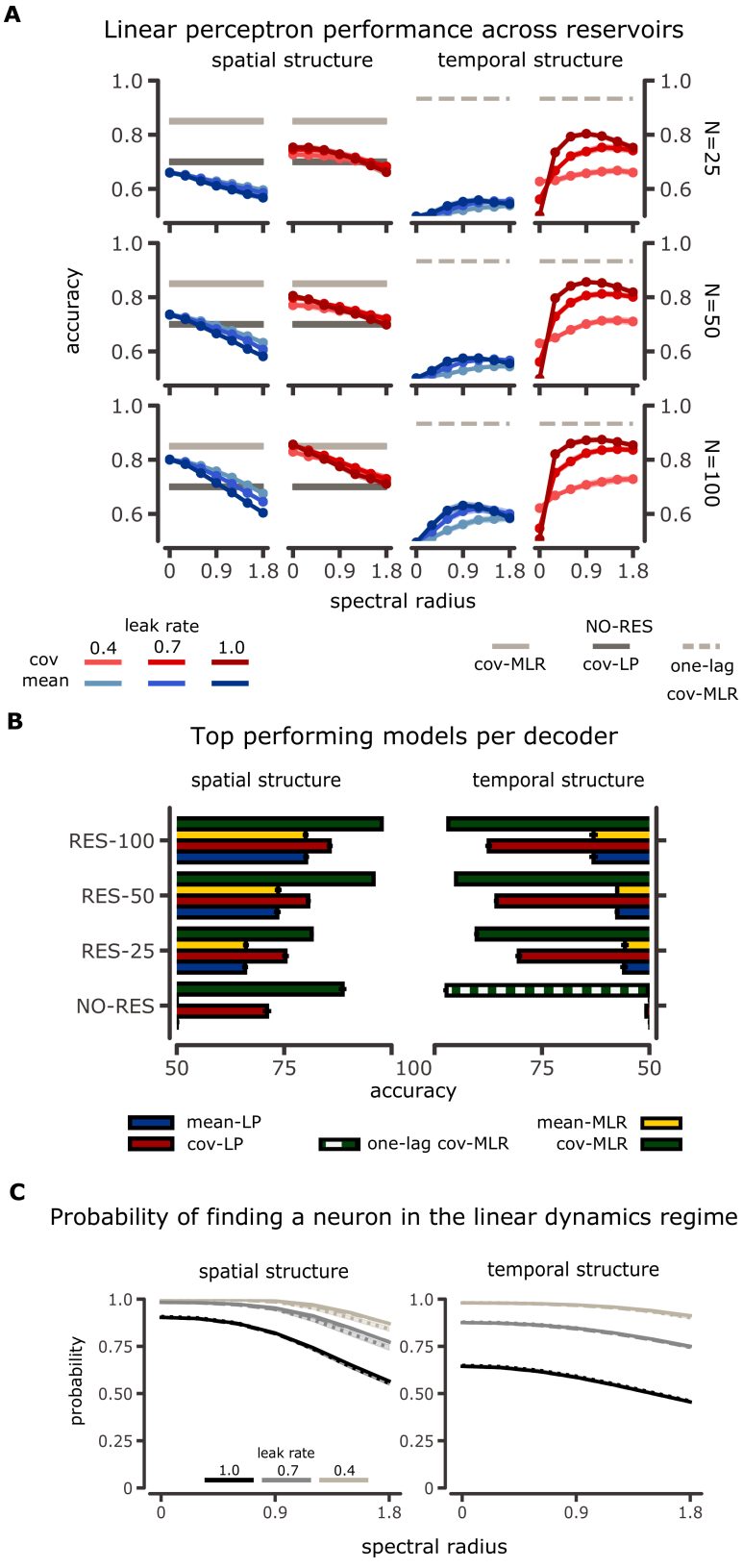
A
Linear perceptron performance across reservoirs
spatial structure
temporal structure
accuracy
N=25
N=50
N=100
spectral radius
leak rate
0.4
0.7
1.0
cov
mean
NO-RES
cov-MLR
cov-LP
one-lag cov-MLR
B
Top performing models per decoder
spatial structure
temporal structure
RES-100
RES-50
RES-25
NO-RES
accuracy
mean-LP
cov-LP
one-lag cov-MLR
mean-MLR
cov-MLR
C
Probability of finding a neuron in the linear dynamics regime
spatial structure
temporal structure
probability
leak rate
1.0
0.7
0.4
spectral radius

Figure 3

Figure 3 *(previous page)*: **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 top pathway in Fig. 1B) that characterizes the 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 pathway in Fig. 1B). 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:** 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. 1B), while NO-RES indicates that the input time series is directly fed to the decoder (see the top pathway in Fig. 1B). 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. **C:** Probability of finding a reservoir neuron in the linear activation regime 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 of gray). Results are averaged across 10 different data simulations and reservoir configurations. Shaded areas are ± 1 sem.

We focus on the usefulness of the reservoir in the pipeline for classification depicted by the bottom pathway of Fig. 1B. The perceptron readout (mean-LP and cov-LP in Fig. 2) is trained to perform a binary ($K = 2$) classification task of input time series that differ by their second-order statistics.

For the spatial structure where the categories differ by the zero-lag covariance patterns (Fig. 3A, left panel, see Section 2.1.2 in Methods for details), we vary the reservoir parameters to assess their influence on the classification performance, namely its size, leak rate and spectral radius. For the classical mean-based readouts (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 separate the two categories of inputs for the decoder. The new approach with the covariance-based readouts (in red) displays the same trend. Furthermore, the reservoir can boost the decoding performance beyond that of a covariance perceptron directly applied to the inputs (dark gray line) with as little as $N = 25$ neurons (half the amount needed by mean-based readouts). 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 MLR-

NO-RES (see Section 2.4). For both readout types, we find that the performance decays with the spectral radius, the best being achieved when the reservoir is a feedforward layer where $\rho(\Omega_{res}) = 0$. Thus, 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 (see Section 2.3.2), and keeping a memory of past states through reservoir dynamics appears detrimental. Note, nonetheless, that the 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.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).

Then, we consider the temporal structure (Fig. 3A, right panel, see Section 2.1.3 in Methods for details) for which the input time series from the two categories only differ by their one-lag covariances, their spatial covariances being identical. 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 (dashed light gray line). 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 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 $\rho(\Omega_{res}) \approx 1$, which have also been shown to maximize memory capacity for Gaussian inputs (Jaeger, 2002; Farkaš et al., 2016). 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). Second, accuracy increases with leak rate for both readout types. Nonetheless, for covariance readouts in a feedforward reservoir with $\rho(\Omega_{res}) = 0$, 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. 3B. 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. 1B), as well as the decoders directly applied to the inputs (NO-RES, see the top pathway in Fig. 1B). The best decoder performance for each case in Fig. 3A across the considered reservoir parameters is displayed in Fig. 3B. 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).

All in all, the previous results suggest a nontrivial relationship between input dynamics, reservoir parameters and decoder performance. To gain further insights, we numerically compute for each input structure type the probability of finding a neuron in the reservoir in the linear regime, as a function of spectral radius and leak rate (see

Section 2.3.3 for details).

For the spatial input structure (left panel, Fig. 3C), we find that the reservoir mostly operates in the linear regime, and that nonlinear behavior becomes more likely with increasing spectral radius. However, the nonlinear dynamics can be counterbalanced by small leak rates. Thus, when the information about inputs is embedded in the spatial covariances, the linear regime allows the reservoir to optimally represent the information across multiple statistical orders. This is very intuitive, since this task requires no memory, and is consistent with our Taylor series analysis in Section 2.3.2.

Next, we consider a temporal structure (right panel, Fig. 3C). Although we find the same probability trends as for the spatial structure (it decreases with spectral radius and leak rate), from the accuracy profiles in Fig. 3A (right panel), we note that the best dynamical regime to transform temporal to spatial covariances is tightly balanced between linearity and nonlinearity. Indeed, the only case where pure linear dynamics are preferable is when there is no recurrent connectivity ($\rho(\Omega_{res}) = 0$) and the neurons must rely on their individual memory, as given by the leaky mechanism.

Ultimately, temporal and spatial input structure 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 $\rho(\Omega_{res}) \simeq 1$.

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. From our Taylor series analysis of Section 2.3.2, we know that, in the linear regime, mean activity patterns in the inputs are transformed into patterns of both statistical orders in the reservoir, albeit to a lesser extent for the second order. Since the task of classifying input means is similar to the spatial task for covariances (i.e. no use of the lagged activity in the reservoir), we restrict this investigation to a feedforward reservoir and no leaky integration, namely $\rho(\Omega_{res}) = 0$ and $\alpha = 1$, as they give the best performance in Fig. 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 2.1.1) or spatial covariances (input dynamics in Section 2.1.2). 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. 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 ei-

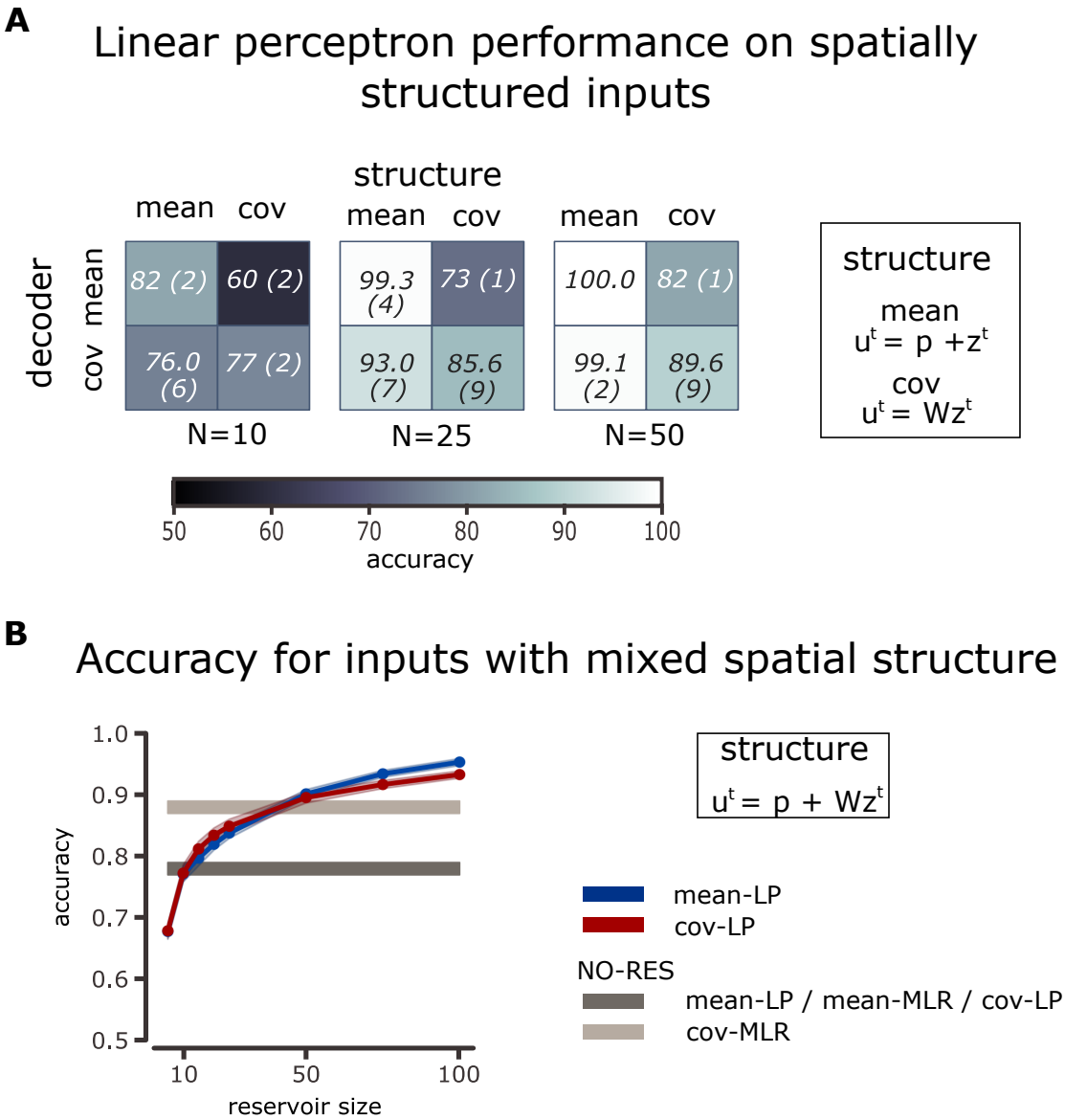

Figure 4: **Decoding mixed information for the mean structure and spatial structure using feedforward reservoirs. A:** Accuracy table for feedforward reservoirs without leaky integration coupled to a linear perceptron when using mean or covariance readouts to classify time series characterized by its 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 *(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. 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 2.1.4 in Methods).

ther of their first/second-order statistics, corresponding to the dynamics in Section 2.1.4. 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 the inputs. We find that both decoders perform equally well across various reservoir sizes (Fig. 4B), with a slight advantage for cov-LP over mean-LP for smaller sizes and conversely for larger sizes.

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.

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/output feeder units (Section 2.3.1) to study the role of the reservoir topology in the transmission of different statistical orders (Fig. 5A). 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, as well as the architecture studied until now with full connectivity from inputs and to 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. 3A, middle row, $N = 50$). The leak rate is always set to $\alpha = 1$. Since the density of connections does not affect performance in the full connectivity configuration (it is only important that it is non-null, see Supplementary Material Fig. 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. 5B, 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 ($\rho(\Omega_{res}) = 0.9$), 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. 5B, 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

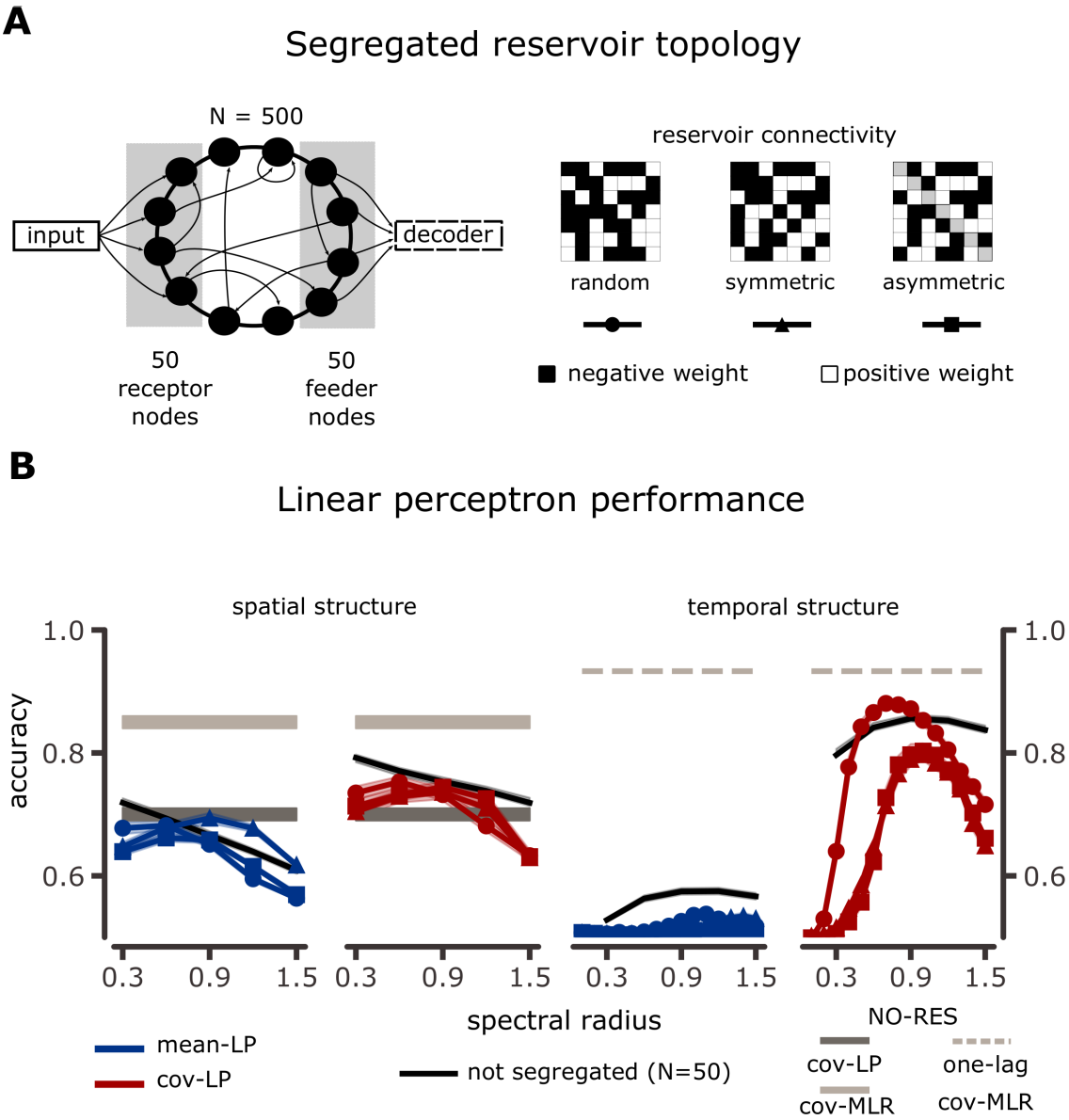

Figure 5: **Decoding performance for spatial and temporal structure in segregated reservoirs. A:** We explore an additional reservoir structure where nodes connected to inputs (receptors) are segregated from nodes connected to outputs (feeders) (Kawai et al., 2019). We choose the reservoir connectivity matrix to be fully random, symmetric or asymmetric (see Section 2.3.1). **B:** Segregated reservoir 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. 3A. The black lines show the performance of a non-segregated reservoir of 50 neurons without leaky integration ($\alpha = 1$, same as in Fig. 3A, middle row) for comparison.

for $\rho(\Omega_{res}) \approx 0.7$. 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.

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 (Hammami and Sellam, 2009; Hammami and Bedda, 2010; Dua and Graff, 2019). The dataset consist of digits $0-9$, represented by 13 input nodes obtained after some preprocessing mimicking the cochlea as illustrated in Fig. 1C (see Section 2.2 in Methods for further details). Our goal here is not so much to improve the best performance obtained so far on this dataset, which to the best of our knowledge is currently at $99.16 \pm 0.12\%$ using an echo state network of unreported size with an extreme kernel machine as readout (Alalshekmubarak and Smith, 2014). 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 (Verstraeten et al., 2005, 2006, 2007; Skowronski and Harris, 2007; Triefenbach et al., 2010; Alalshekmubarak and Smith, 2014; Zhang et al., 2015; Jin and Li, 2017; Aceituno et al., 2020). Indeed, the classical mean-based decoding reaches a competitive performance of 96.91% for a reservoir of size $N = 900$ (Alalshekmubarak and Smith, 2013; Alalshekmubarak, 2014).

As before, we use the two pipelines for classification, with and without reservoir (same as Fig. 1B) 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. 6A, which are obtained for $\alpha = 0.2$ and $\rho(\Omega_{res}) = 1.2$ (see Supplementary Material Fig. 9 for all results). This is in line with decoding temporal structure (Fig. 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 (Alalshekmubarak and Smith, 2013), 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 (Zerari et al., 2019) (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

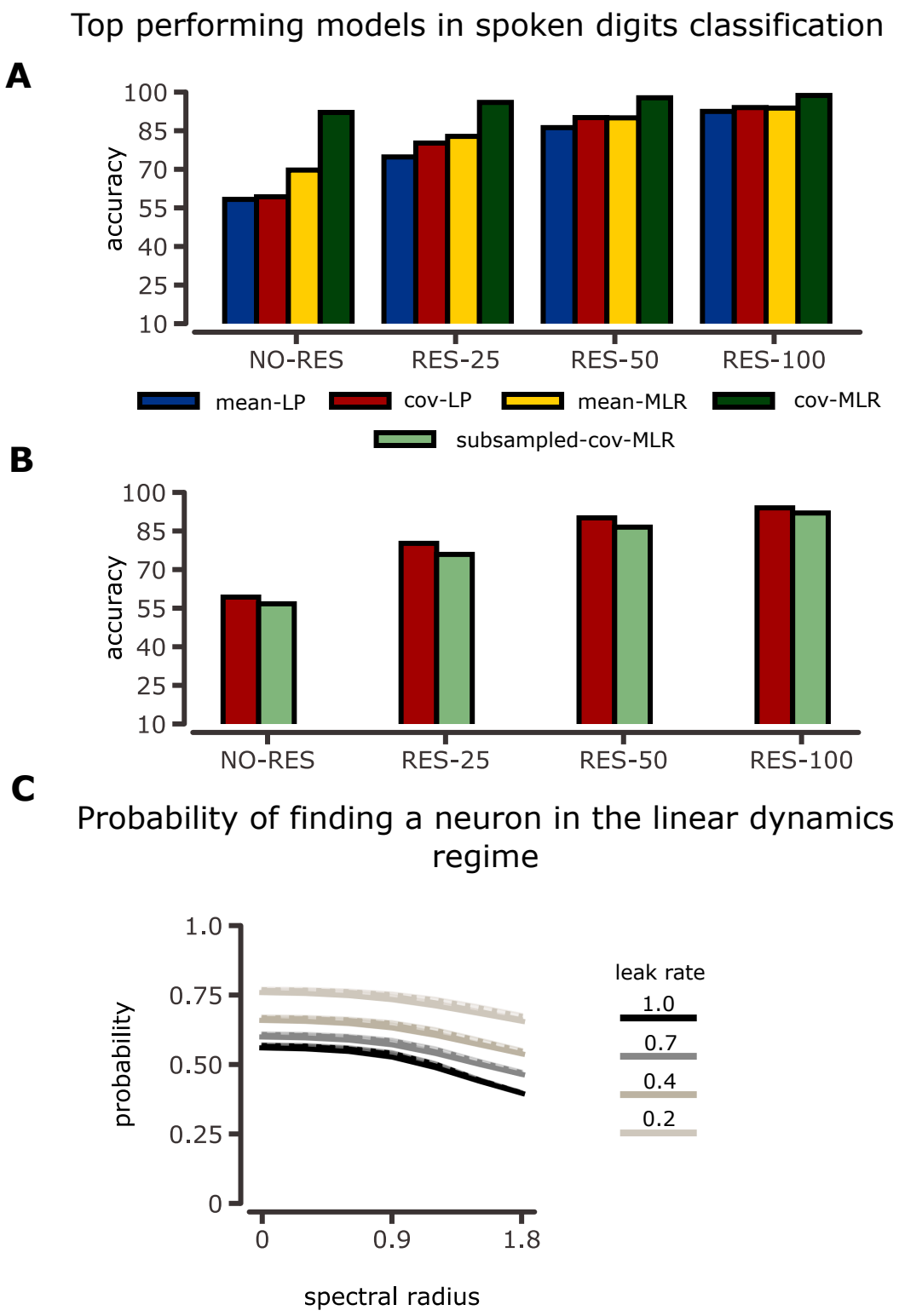

Figure 6: **Decoding performance in spoken Arabic digits classification. A:** Accuracy for the best performing models of each size, distinguished by decoder type. Colors and model label conventions are the same as in Fig. 3B. Results are averaged across 10 different reservoir instantiations. Error bars displaying ±1 sem are also included.

Figure 6 *(previous page)*: **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. **C:** Probability of finding a reservoir neuron in the linear activation regime 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 of gray).

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 2.4.2 for details). We find that both models exhibit similar performance under this constrain (see Fig. 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 Discussion

In this study, we have explored the application of covariance-based decoding to reservoir computing in classification tasks for time series. Our goal was to compare this new type of decoding to the classical mean-based one that has been used until now. The motivation is twofold: to compare a biologically inspired configuration (perceptron mapping time series to time series) and a machine learning configuration (multinomial logistic regression mapping static features to class probabilities) in the time series classification, and to characterize how reservoirs can process statistical orders of the input time series. Our results demonstrate the robustness of covariance-based readouts applied to a reservoir, even in the biological configuration that involves limited resources (perceptron).

We have shown with synthetic data that covariance decoding works well for a broad diversity of input structures and that a compromise network configuration with $\alpha = 1$ and $\rho(\Omega res) = 1$ 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%) (Alalshekmubarak, 2014; Alalshekmubarak and Smith, 2014). Moreover, the cov-LP maximally extracts information about covariances given its restricted resources. For the 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 represent better the intrinsic time scale of the input time series, as speech varies slowly when

compared to the sampling frequency (Jaeger et al., 2007; Verstraeten and Schrauwen, 2009).

The use of the reservoir offers two advantages with respect to the linear network studied in (Gilson et al., 2020). 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. Second, the cross-talk among statistical orders induced by its nonlinear behavior allows to represent information in reservoir and output covariances even when the input is only structured at the first-order. Although strictly we have not examined the influence of the choice for the nonlinearity in the reservoir, we made use of it following a biological motivation, as well as to keep the recurrent dynamics under control for large radii.

A main result lies in the characterization of the interplay between reservoir collective dynamics, as governed by the spectral radius and the leak rate, and the input-output mapping in terms of statistics. For large radii and small leak rates, the reservoir exhibits slow dynamics that tend to drive it towards the linear regime, which enhances mean decoding of 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 has to rapidly produce spatial patterns distinguishable by the perceptron decoder, a feat that maybe requires a longer memory span. 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 at the edge of chaos ($\rho(\Omega_{res}) \approx 1$). The optimal reservoir in this case is one without leaky integration and at the edge of chaos, which is balanced between the linear and nonlinear regimes. Thus, driving it 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 (Jaeger, 2001, 2002; Jaeger et al., 2007; Aceituno et al., 2020), our numerical results suggest that other input-induced dynamical regimes should be further examined theoretically (Manjunath and Jaeger, 2013), especially in the short time scale of transients studied here.

Early studies in echo state networks failed to report performance improvements in time series prediction when using small-world or scale-free topologies over random ones (Liebald, 2004; Rad, 2008). However, it was later shown that connectivity plays a role when the network displays cortex-like properties (Song and Feng, 2010; Kawai et al., 2019), a finding that had been observed in reservoirs of spiking neurons earlier (Haeusler and Maass, 2006). One of the best established properties of cortex is the separation of sensory areas from motor areas, which we mimicked in a part of our study. We did not use leaky mechanisms, as information transmission within the reservoir is more likely to be dependent on recurrent dynamics. This scheme is most efficient to transform input temporal covariances into output spatial covariances when the reservoir connectivity is random; note that in this case the optimal spectral radius is smaller than for the non-segregated reservoirs ($\rho(\Omega_{res}) = 0.7$ 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 we explored naive connectivities within the segregated reservoir and topologies displaying more biologically realistic traits, like small-world, scale-free, clustered (Weidel et al., 2020) or even real connectome patterns (Suárez et al., 2020; Damicelli et al., 2021), could be investigated. Furthermore, we constrained our study to random input projections, which take part in shaping the input representations that arise in the reservoir. 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 (Weidel et al., 2020), so it is natural to question whether this effect is also observed, or even further enhanced, for covariance-based readouts.

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 (Pascanu et al., 2013). 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 draw back the attention to the cost-benefit relationship betweeen covariance and mean-based representations (Gilson et al., 2020; Dahmen et al., 2020). While covariances offer higher-dimensional representational spaces than means at the same number of resources, learning is computationally cheaper and more stable for mean-based representations.

Even though working with echo state networks was a natural first step when building on previous work done in linear analogue networks to operate on covariances, we expect our covariance-based decoding framework to give interesting perspectives for more biologically realistic spiking network models.

Acknowledgements

S.L is supported by a FI fellowship from the Secretariat for Universities and Research of the Ministry of Business and Knowledge of the Government of Catalonia and the European Social Fund. R.M.-B is supported by the Howard Hughes Medical Institute (HHMI, ref 55008742), MINECO (Spain; BFU2017-85936-P) and ICREA Academia (2016). M.G acknowledges funding from the German Excellence Strategy of the Federal Government and the Länder (G:(DE-82)EXS-PF-JARA-SDS005) and the European Union's Horizon 2020 research and innovation programme under grant agreement No. 785907 (Human Brain Project SGA2).

Code availability

Python code samples to generate synthetic datasets and reservoirs, training methods, as well as some pretrained models can be found at `http://www.github.com/slawrie/covariance-reservoir`.

References

Aceituno, P. V., Yan, G., and Liu, Y. Y. (2020). Tailoring Echo State Networks for Optimal Learning. *iScience*, 23(9):101440.

Alalshekmubarak, A. (2014). *Towards A Robust Arabic Speech Recognition System Based On Reservoir Computing*. PhD thesis, University of Stirling.

Alalshekmubarak, A. and Smith, L. S. (2013). 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.

Alalshekmubarak, A. and Smith, L. S. (2014). 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 Lecture Notes in Bioinformatics)*, volume 8681 LNCS, pages 225–232.

Bair, W. and Koch, C. (1996). Temporal Precision of Spike Trains in Extrastriate Cortex of the Behaving Macaque Monkey. *Neural Computation*, 8(6):1185–1202.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*.

Boedecker, J., Obst, O., Lizier, J. T., Mayer, N. M., and Asada, M. (2012). Information processing in echo state networks at the edge of chaos. *Theory in Biosciences*, 131(3):205–213.

Boerlin, M., Machens, C. K., and Denève, S. (2013). Predictive Coding of Dynamical Variables in Balanced Spiking Networks. *PLoS Computational Biology*, 9(11):e1003258.

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

Dahmen, D., Gilson, M., and Helias, M. (2020). Capacity of the covariance perceptron. *Journal of Physics A: Mathematical and Theoretical*, 53(35):354002.

Damicelli, F., Hilgetag, C. C., and Goulas, A. (2021). Brain Connectivity meets Reservoir Computing. *bioRxiv*, page 2021.01.22.427750.

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

Dua, D. and Graff, C. (2019). {UCI} Machine Learning Repository.

Farkaš, I., Bosák, R., and Gergeľ, P. (2016). Computational analysis of memory capacity in echo state networks. *Neural Networks*, 83:109–120.

Gilson, M., Dahmen, D., Moreno-Bote, R., Insabato, A., and Helias, M. (2020). The covariance perceptron: A new paradigm for classification and processing of time series in recurrent neuronal networks. *PLOS Computational Biology*, 16(10):e1008127.

Gilson, M., Masquelier, T., and Hugues, E. (2011). STDP Allows Fast Rate-Modulated Coding with Poisson-Like Spike Trains. *PLoS Computational Biology*, 7(10):e1002231.

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

Hammami, N. and Bedda, M. (2010). Improved tree model for arabic speech recognition. In *2010 3rd International Conference on Computer Science and Information Technology*, pages 521–526. IEEE.

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

Hermans, M. and Schrauwen, B. (2010). Memory in reservoirs for high dimensional input. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.

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

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. Technical report.

Jaeger, H. (2002). *Short term memory in echo state networks*. Sankt Augustin: GMD Forschungszentrum Informationstechnik, 2001, 60 pp. GMD Report, 152.

Jaeger, H., Lukoševičius, M., Popovici, D., and Siewert, U. (2007). Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20(3):335–352.

Jin, Y. and Li, P. (2017). Performance and robustness of bio-inspired digital liquid state machines: A case study of speech recognition. *Neurocomputing*, 226:145–160.

Kawai, Y., Park, J., and Asada, M. (2019). A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks*, 112:15–23.

Kostal, L., Lansky, P., and Rospars, J. P. (2007). Neuronal coding and spiking randomness. *European Journal of Neuroscience*, 26(10):2693–2701.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Liebald, B. (2004). *Exploration of effects of different network topologieson the esn signal crosscorrelation matrix spectrum*. Bachelor thesis, University of Bremen.

Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.

Maass, W., Natschläger, T., and Markram, H. (2002). Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation*, 14(11):2531–2560.

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

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

Nogueira, R., Lawrie, S., and Moreno-Bote, R. (2018). Neuronal Variability as a Proxy for Network State. *Trends in Neurosciences*, 41(4):170–173.

Panzeri, S., Harvey, C. D., Piasini, E., Latham, P. E., and Fellin, T. (2017). Cracking the Neural Code for Sensory Perception by Combining Statistics, Intervention, and Behavior. *Neuron*, 93(3):491–507.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). 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. JMLR.org.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Rad, A. A. (2008). Dynamical Networks (miniproject) Effect of Topology of the Reservoir on Performance of Echo State Networks.

Renart, A. and Machens, C. K. (2014). Variability in neural activity and behavior. *Current Opinion in Neurobiology*, 25:211–220.

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

Schaetti, N., Salomon, M., and Couturier, R. (2016). 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.

Shadlen, M. N. and Movshon, J. (1999). Synchrony Unbound. *Neuron*, 24(1):67–77.

Shadlen, M. N. and Newsome, W. T. (1998). The Variable Discharge of Cortical Neurons: Implications for Connectivity, Computation, and Information Coding. *The Journal of Neuroscience*, 18(10):3870–3896.

Shahidi, N., Andrei, A. R., Hu, M., and Dragoi, V. (2019). High-order coordination of cortical spiking activity modulates perceptual accuracy. *Nature Neuroscience*, 22(7):1148–1158.

Skowronski, M. D. and Harris, J. G. (2007). Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, 20(3):414–423.

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

Stein, R. B., Gossen, E. R., and Jones, K. E. (2005). Neuronal variability: Noise or part of the signal? *Nature Reviews Neuroscience*, 6(5):389–397.

Suárez, L. E., Markello, R. D., Betzel, R. F., and Misic, B. (2020). Linking Structure and Function in Macroscale Brain Networks. *Trends in Cognitive Sciences*, 24(4):302–315.

Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A. (2019). Recent advances in physical reservoir computing: A review.

Triefenbach, F., Jalalvand, A., Schrauwen, B., and Martens, J.-p. (2010). Phoneme Recognition with Large Hierarchical Reservoirs. In *Advances in Neural Information Processing Systems 23*, pages 2307—-2315. Curran Associates, Inc.

Usman, M. (2017). 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.

Verstraeten, D. and Schrauwen, B. (2009). On the quantification of dynamics in reservoir computing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5768 LNCS(PART 1):985–994.

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

Verstraeten, D., Schrauwen, B., and Stroobandt, D. (2006). Reservoir-based techniques for speech recognition. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1050–1053. IEEE.

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

Weidel, P., Duarte, R., and Morrison, A. (2020). Unsupervised learning and clustered connectivity enhance reinforcement learning in spiking neural networks. pages 1–27.

Zerari, N., Abdelhamid, S., Bouzgou, H., and Raymond, C. (2019). Bidirectional deep architecture for Arabic speech recognition. *Open Computer Science*, 9(1):92–102.

Zhang, Y., Li, P., Jin, Y., and Choe, Y. (2015). 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.

A Supplementary Material

A.1 Section 3.2 results when using MLR decoders as benchmarks

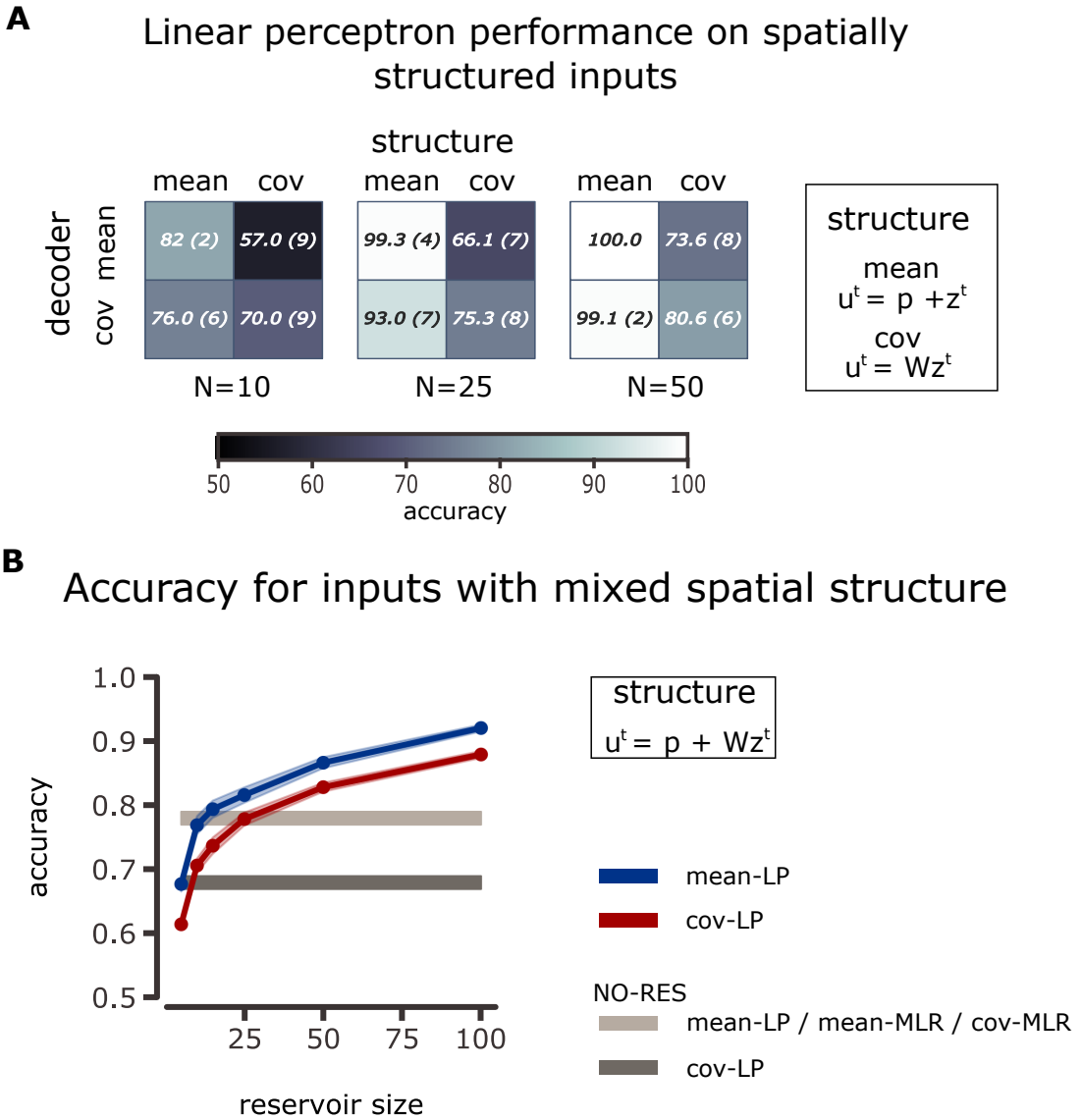

Figure 7: **Decoding mixed information for the mean structure and spatial structure using feedforward reservoirs. A:** Accuracy table for feedforward reservoirs without leaky integration coupled to a linear perceptron when using mean or covariance readouts to classify time series characterized by its mean or its spatial covariance structure. Same conventions as Fig 4. Datasets were designed to match the performances using the MLR as readout: cov-MLR 85(±1)%, 30 patterns per class, mean-MLR 83(±2)%, 10 patterns per class.

Figure 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. 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 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 4a) or in both (Fig 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 7a, we have 30 covariance patterns per class (as in Fig 3a, left pannel) instead of 20. For Fig 7b we have 20 covariance patterns per class, instead of the 10 in Fig 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 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 7 suggests that the first-order information overrides the second-order one.

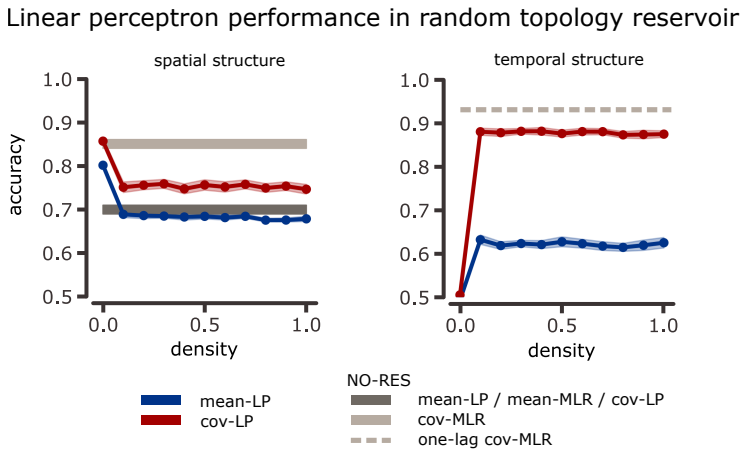

Figure 8: **Linear perceptron classification performance in synthetic time series with second-order structure.** Similar color conventions as in Fig 3a. Reservoir parameters are $N = 100$, $\alpha = 1$ and $\rho(\Omega_{res}) = 1.2$.

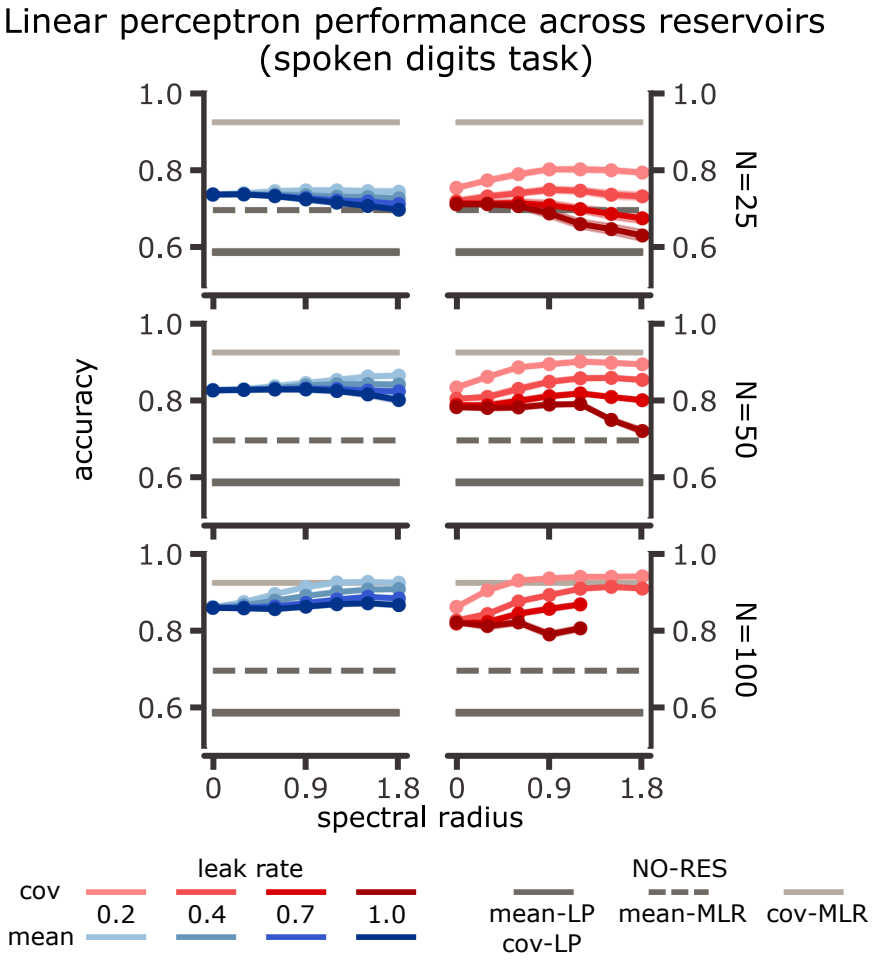

Figure 9: **Linear perceptron classification performance for spoken digits across reservoir configurations.** Similar color conventions as in Fig 5a. Missing points for $N = 100$, large leaks and spectral radii are due to numerical instabilities during covariance learning.