Statistical temporal pattern extraction by neuronal architecture

Sandra Nestler [1,2,*] Moritz Helias [1,3] and Matthieu Gilson [1,4]

[1]*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, 52428 Jülich, Germany*

[2]*RWTH Aachen University, 52062 Aachen, Germany*

[3]*Department of Physics, Faculty 1, RWTH Aachen University, 52062 Aachen, Germany*

[4]*Institut de Neurosciences des Systèmes (INS, UMR1106), INSERM-AMU, 13005 Marseille, France*

(Received 1 February 2023; accepted 11 July 2023; published 11 September 2023)

Neuronal systems need to process temporal signals. Here, we show how higher-order temporal (co)fluctuations can be employed to represent and process information. Concretely, we demonstrate that a simple biologically inspired feedforward neuronal model can extract information from up to the third-order cumulant to perform time series classification. This model relies on a weighted linear summation of synaptic inputs followed by a nonlinear gain function. Training both the synaptic weights and the nonlinear gain function exposes how the nonlinearity allows for the transfer of higher-order correlations to the mean, which in turn enables the synergistic use of information encoded in multiple cumulants to maximize the classification accuracy. The approach is demonstrated both on synthetic and real-world datasets of multivariate time series. Moreover, we show that the biologically inspired architecture makes better use of the number of trainable parameters than a classical machine-learning scheme. Our findings emphasize the benefit of biological neuronal architectures, paired with dedicated learning algorithms, for the processing of information embedded in higher-order statistical cumulants of temporal (co)fluctuations.

DOI: 10.1103/PhysRevResearch.5.033177

I. INTRODUCTION

It has long been hypothesized that information about the environment and internal states of humans and animals is represented in the correlated neuronal activity. The most apparent examples include spike patterns that are related to organization of cognitive motor processes and visual perception [1–4]. Such patterns are also quantified indirectly via neuronal rhythms [5–8]. Furthermore, the variability of network responses across trials when presenting the same stimulus [9] has been shown to limit encoding robustness but was later found to serve a functional role in a Bayesian context to represent (un)certainty about the presented stimulus [10,11].

Structured variability is likely to play an important role in learning by interacting with synaptic plasticity, particularly for models like spike-timing-dependent plasticity (STDP) that are sensitive to high-order statistics [12–15]. STDP implements an unsupervised learning rule like classical Hebbian learning based on firing rates [16,17] and the Bienenstock-Cooper-Munro learning rule [18–20]. A counterpart for supervised learning has recently been proposed as a basis for neuronal representations, which reconciles the apparent contradiction between robust encoding by seemingly noisy signals [21]. This can be achieved by detecting and selecting correlated patterns thanks to an adequate learning rule to update the synaptic weights between neurons, thereby implementing a form of statistical processing. The focus on second-order statistics as a measure for spike trains complements recent supervised learning schemes which either shape the detailed spiking time generated by neurons or control their time-varying firing rates [22–24].

An adjoint viewpoint to this biologically inspired learning is taken up by machine learning (ML). Though constructed from similar building blocks (so-called artificial neurons), the focus here lies in finding optimal and efficient (re)encoding to process large amounts of data, the most widespread application being classification. This field has produced artificial neuronal architectures with impressive performance, even better than human performance in the case of image recognition (see Refs. [25,26] for reviews). However, time series, which are naturally processed by biological neuronal systems, can be challenging for these systems. For example, they are often designed to operate on a feature space with fixed dimensionality (including duration) as opposed to the learning of ongoing signals that may have variable lengths. In this setting, artificial neural networks are designed to account for different input durations across samples. Recurrent neural networks, with their time-dependent network states, handle these inputs by construction. Reservoir computing [27,28], which has recently been employed to capture statistical differences in the input cumulants [29], forms a straightforward implementation of a trainable recurrent neural network. Also long short-term

*s.nestler@fz-juelich.de

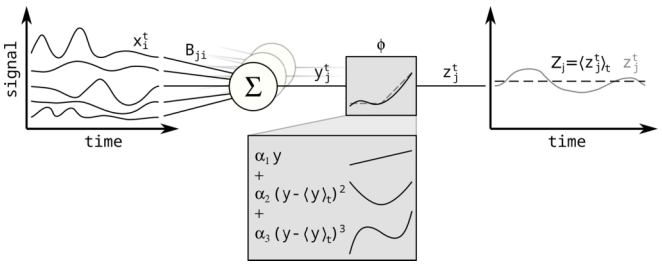

FIG. 1. Biological interpretation of an order-selective perceptron. The input time series x^t is linearly combined by the connectivity matrix B to form the intermediate variable $y^t = Bx^t$. The latter is then passed on through a nonlinear gain function ϕ to obtain the output z^t. The classification decision is made with respect to the temporal average of z^t, which has the dimensionality of the number of classes to predict, in a winner-takes-all fashion. The gain function ϕ is implemented by a third-order polynomial with coefficients α_1, α_2, and α_3. The contributions for orders two and three are the Taylor coefficients of ϕ evaluated at the temporal mean of $\langle y^t \rangle_t$ over the duration of the input. In contrast, the linear contribution directly acts on z^t. In this figure, we consider five input signals and three output neurons, one of which is shown upfront.

memory (LSTM) units [30] or the more recent gated recurrent units [31] employ feedback connections to process time series.

Time series processing has increasingly raised interest at the intersection of biological learning and ML, based on a diversity of network architectures and approaches [32,33]. Many studies rely on feedforward networks that have proven to be efficient [34,35]; recurrent networks, in contrast, are notoriously harder to train [36,37], despite progress for specific applications like time series completion [38,39]. Reservoir computing consists of an intermediate approach, where a large (usually untrained) recurrent network is combined with an easily trainable feedforward readout. Its processing capabilities come from the combination of recurrent connectivity with nonlinear units that performs complex operations on the input signals, to be then selected by the readout to form the desired output. The training of such feedforward readouts to capture structured variability in a time series is our motivation and focus, leaving out the reservoir here.

In this paper, we consider statistical processing for the classification of temporal signals by a feedforward neuronal system, aiming to capture structured variability embedded in time series. The goal is to automatically select the relevant statistical orders, possibly combining them to shape the output signal. We focus in this paper on classification that relies on the temporal mean of the output, which implies that input cumulants at various orders need to be mapped to the first order in output. The main inspiration is to design a biological setup consisting of neurons that linearly sum input signals weighted by their afferent connectivity weights, before passing the resulting signal to a nonlinear gain function. We also compare this biologically inspired architecture with a ML approach, in terms of classification accuracy and trainable weights (resources). Figure 1 displays a concise summary of the setup.

The remainder of this paper is structured as follows. We first present the models of neuronal classifiers with two flavors (Sec. II A): a biological architecture and a ML architecture. We show how they differ by the number of trainable weights (akin to resources) and in their optimization. Then we describe the input time series used to validate and compare these classifiers (Sec. II B). We rely on both synthetic data with controlled structures and contrast between the classes as well as on real-world signals coming from Chen *et al.* [40]. Our results start with the synthetic datasets (Sec. III A), to test whether the classifiers can extract statistics embedded in time series that are relevant for the classification. We examine whether the combination of different orders leads to synergistic behavior, namely, whether it increases the classification accuracy (Sec. III B). Finally, we verify the practical applicability of our framework to real-world datasets, comparing the performances of both architectures to the state of the art (Sec. III C).

II. METHODS

A. Neuronal classifiers

We consider classification of multivariate time series by a biologically inspired neuronal architecture illustrated in Fig. 1 that we term an *order -selective perceptron* (OSP). The multivariate input time series x_i^t with $1 \leqslant i \leqslant N$ is linearly combined via the trainable connectivity matrix $B_{ji} \in \mathbb{R}^{M\times N}$ to form the intermediate time-dependent variable $y_j^t = \sum_i B_{ji} x_i^t$. Here, $1 \leqslant j \leqslant M$, where M is the number of classes to be distinguished. Then a nonlinear gain function ϕ is applied on y_j^t, taken as a univariate signal, to obtain the output $z_j^t = \phi(y_j^t)$. Classification is performed based on the temporal mean (first-order statistics) $Z_j = \langle z_j^t \rangle$. This gain function ϕ is shaped to combine the cumulants of the intermediate variable y_j^t, where the nth statistical cumulant (evaluated over time) of y_j^t is denoted as $\langle\langle (y_j^t)^n \rangle\rangle_t = Y_j^n$, resulting in $Z_j = \sum_n \alpha_n \langle\langle (y_j^t)^n \rangle\rangle_t = \sum_n \alpha_n Y_j^n$. The nth cumulant Y_j^n reflects the corresponding cumulants at the same order for the multivariate inputs x_i^t as

$$Y_j^n = \left\langle\left\langle \left(y_j^t\right)^n \right\rangle\right\rangle_t = \sum_{i_1,\ldots,i_n=1}^{N} B_{ji_1} \cdots B_{ji_n} \left\langle\left\langle x_{i_1}^t \cdots x_{i_n}^t \right\rangle\right\rangle_t. \quad (1)$$

Thus, a general gain function combines several statistical orders of the input to generate the mean Z_j of the output activity. For practical reasons, here, we aim to perform the classification based on the first three cumulants. The gain function is therefore implemented as a trainable third-order polynomial of the intermediate variable y_j^t, see the inset panel in Fig. 1. In principle, the OSP can be created with contributions that filter for arbitrary orders of cumulants. Computing the higher-order polynomials to achieve this is as straightforward as is computing the cumulants. We stopped here at $n = 3$ for practical reasons: It is powerful enough to detect information beyond the Gaussian statistics and is still easily testable. While the data generation algorithm (Sec. II B) can be extended to higher-order cumulants, the amount of data required to detect differences in higher-order cumulants increases. Although this would still be feasible, here, we aimed for the minimal, nontrivial extension beyond the Gaussian cumulants to showcase the systematics which becomes apparent already on the example of the first three orders. Moreover, we

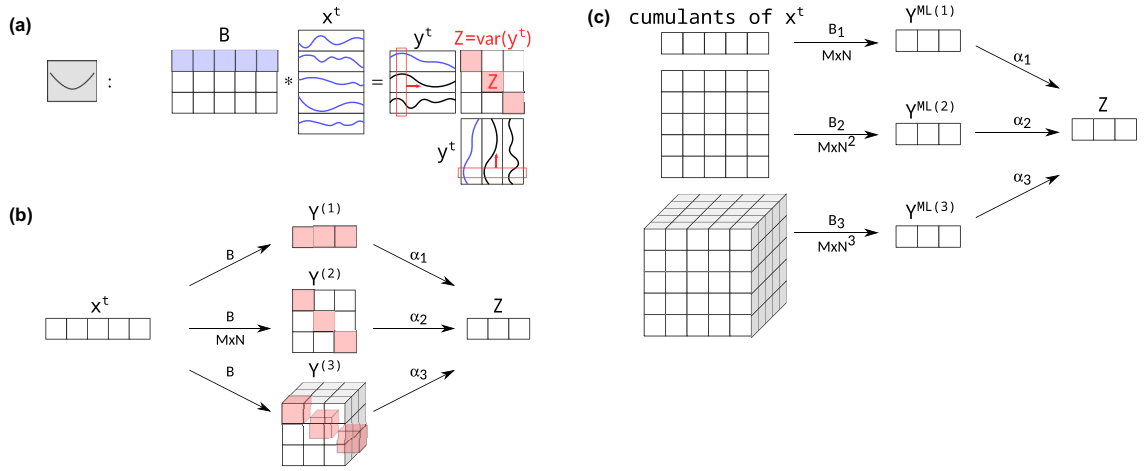

FIG. 2. Extraction of cumulants from a time series. (a) Covariance perceptron. The classification of the centered time series x^t is based on its covariance structure. The output $Z = (Z_1, \ldots, Z_M)$ has dimension equal to the number of classes M. For each class j ($1 \leqslant j \leqslant M$), the covariance perceptron calculates an intermediate variable $y_j^t = \sum_i B_{ji} x_i^t$ that is then passed though a quadratic function $z_j^t = (y_j^t)^2$. The temporal mean $Z_j = \langle z_j^t \rangle = \langle\langle (y_j^t)^2 \rangle\rangle_t$, which therefore equals the variance of y^t is used for classification in a winner-takes-all manner. This is equivalent to retaining the diagonal of the covariance matrix of $y^t_{1 \leqslant j \leqslant M}$. (b) Statistical processing for order-selective perceptron (OSP) model. Like (a), the effect of the gain function ϕ in Fig. 1 can be represented by an individual path for each cumulant $\langle\langle (y_j^t)^n \rangle\rangle_t$ of $y_j^t = \sum_i B_{ji} x_i^t$. The cumulants of y^t are thus given by the cumulants of the inputs x^t of the same order n [Eq. (1)]. The cumulant of order n can be seen as an outer product with dimensionality M^n: The mean is represented by a line (vector), the covariance by a square [matrix; as in (a)], and the third-order cumulant by a cube (third-order tensor). The effect of applying the polynomial $\phi(y) = \alpha_1 y + \alpha_2 \tilde{y}^2 + \alpha_3 \tilde{y}^3$ with $\tilde{y}^t = y^t - \langle y^t \rangle_t$ and then taking the mean to obtain Z is to combine these contributions linearly with a weight vector $\alpha = (\alpha_1, \alpha_2, \alpha_3)$. (c) Machine-learning (ML) model. Statistical processing can alternatively be implemented by a network that is fed by the input cumulants directly. For each order, a linear layer is set up with the corresponding dimensionality: the total size of the input cumulant multiplied by the number of outputs (i.e., of classes). In a second layer, a linear combination of the outputs for the individual orders is trained.

will show that the third order already leads to a considerable improvement in accuracy in several real-world tasks.

1. Link to covariance perceptron

Before explaining the OSP in more detail, we briefly present its operational link with the covariance perceptron [21,41]. The goal of the latter is to operate on the second-order statistics embedded in the time series, instead of the first-order statistics that is equivalent to the classical perceptron applied to mean activity [42–44]. It can be formalized as in Fig. 2(a), where the linear mixing is based on the weight matrix B^{P}_{ji} applied to the N-dimensional time-dependent inputs x_i^t (after demeaning) and gives the M-dimensional intermediate variable y_j^t for each of the $1 \leqslant j \leqslant M$ classes of the dataset:

$$y_j^t = \sum_i B^{\mathrm{P}}_{ji} \left(x_i^t - \langle x_i^t \rangle_t \right). \tag{2}$$

From the outer product of y_j^t that forms the covariance matrix, the classification only considers the variances (diagonal matrix elements) defined as

$$Z_j = \left\langle \left\langle \left(y_j^t \right)^2 \right\rangle \right\rangle_t = [\mathrm{diag}(B^{\mathrm{P}} \Sigma B^{\mathrm{P\,T}})]_j. \tag{3}$$

Thanks to the quadratic operation, the mean value of Z_j depends on the input covariances $\Sigma_{ii'} = \langle\langle x_i^t x_{i'}^t \rangle\rangle_t$ for all possible indices i and i'. In this scheme, there are only MN weights in matrix B^{P} that can be trained. Thus, the number of parameters can be compared with a ML approach that applies a linear perceptron to the entries of the covariance matrix $Z'_j = \sum_{ii'} B^{\mathrm{ML}}_{jii'} \Sigma_{ii'}$, which involves MN^2 trainable parameters of the tensor B^{ML}.

2. OSP

Now we consider the OSP, a feedforward network that extends the covariance perceptron to combine the first three statistical orders of the input time series x^t via the calculation of the output time series z^t, illustrated in Fig. 1, as

$$y_j^t = \sum_i B_{ji} x_i^t, \tag{4}$$

$$z^t = \phi(y^t) = \alpha_1 y + \alpha_2 \tilde{y}^2 + \alpha_3 \tilde{y}^3. \tag{5}$$

Here, the linear weights $B_{ji} \in \mathbb{R}^{M \times N}$ are trainable, as well as the coefficients $\alpha_{1 \leqslant i \leqslant 3}$ of the (nonlinear) polynomial gain function ϕ for the three different cumulants that are considered. Note that $\tilde{y}^t = y^t - \langle y^t \rangle_t$ is the demeaned version of time series y^t.

The key of the statistical processing is the following: the cumulants at orders 1–3 can be calculated using the polynomial of order three applied to the intermediate variable y_j^t, as the second and third cumulants are the central moments (including the demeaning). An abstract representation of the resulting computation is illustrated in Fig. 2(b). Each path of the network corresponds to a specific cumulant of order n of the intermediate variable y^t. The OSP computation can thus be understood via the cumulants of order $n \in \{1, 2, 3\}$ of y_j^t, as illustrated in Fig. 2(b):

$$Y_j^{(n)} = \left\langle \left\langle \left(y_j^t \right)^n \right\rangle \right\rangle_t, \tag{6}$$

where the symbol $\langle\langle (y_j^t)^n \rangle\rangle_t$ stands for the mean over time $\langle y_j^t \rangle_t$ for $n = 1$, and the nth power of the demeaned variable $\langle (y_j^t - \langle y_j^t \rangle_t)^n \rangle_t = \langle (\tilde{y}^t)^n \rangle_t$ for $n \in \{2, 3\}$; they correspond to the red diagonal matrix or tensor elements in Fig. 2(b). Importantly, each cumulant $Y_j^{(n)}$ for order n depends on the nth statistical order of x [see also Eq. (1)].

Note that it is straightforward to generalize this scheme to arbitrary cumulant orders, beyond the first three orders considered here, because a cumulant of any order n can be expressed as a linear combination of moments of orders $1, \ldots, n$; beyond third order, however, they cease to be identical to the nth centered moments [45]. The resulting computation of the OSP is thus the combination in the output mean $Z_j = \langle z_j^t \rangle_t$ of the different cumulant orders n:

$$Z_j = \sum_n \alpha_n Y_j^{(n)}, \tag{7}$$

which is the basis for the classification via

$$\arg\max(Z_j - \theta_j), \tag{8}$$

with some trainable thresholds θ_j. The key point here is that the selection of the informative cumulant orders for the classification is managed by tuning the parameters α_n.

We now want to evaluate how the OSP combines different cumulant orders to perform classification. From the network after training, which we refer to as the *full* model with all coefficients α_n, we can ignore contributions from given cumulant orders by setting the corresponding parameter α_n to zero. In this way, we create a single-order OSP for order n with $\alpha_{n'} = 0 \quad \forall n' \neq n$. This network is then classifying based on the nth cumulant only with the corresponding learned parameter α_n. With this approach, we quantify the contribution of an individual statistical order n to classification in the full model.

3. ML classifier

For comparison, we consider a ML architecture: a network that is fed by the input cumulants directly. This approach can be thought of as first performing a mapping into a feature space that is spanned by the first three cumulants $\{\langle\langle x_{i_1}^t \rangle\rangle_t^1, \langle\langle x_{i_1}^t x_{i_2}^t \rangle\rangle_t^2, \langle\langle x_{i_1}^t x_{i_2}^t x_{i_3}^t \rangle\rangle_t^3\}$ of the data and then training a linear readout vector $B^{\mathrm{ML}(n)}_{j\{i_1 \ldots i_n\}}$ on this feature space. The operational difference to the OSP is hence that the weights $B^{\mathrm{ML}(n)}_{j\{i_1 \ldots i_n\}}$ directly map from all entries of the cumulants to the next layer, as displayed in Fig. 2(c):

$$Y_j^{\mathrm{ML}(n)} = \sum_{i_1 \ldots i_n} B^{\mathrm{ML}(n)}_{j i_1 \ldots i_n} \left\langle\left\langle x_{i_1}^t \cdots x_{i_n}^t \right\rangle\right\rangle_t^n, \tag{9}$$

yielding a number of trainable parameters equal to MN^n for order n. As before, the different orders are combined and selected using Eq. (7). Note that the ML architecture differs from the OSP in that it outputs scalar values, not a time series. To be able to further compare this ML architecture to the OSP, we randomly select MN trainable weights from the whole MN^n entries of B^{ML} (the other weights being fixed to random values) to build the constrained ML architecture which has the same number of trainable parameters as the OSP. This way, the complexity of the two systems is comparable.

4. Training process

Training of both network types is conducted via a stochastic gradient descent using a scaled squared error loss ε for each data sample ν of the form:

$$\varepsilon(x^\nu) \sim \frac{1}{2} \sum_{j=1}^{M} \left[\bar{Z}_j^{c(\nu)} - \left(Z_j^\nu - \theta_j\right)\right]^2, \tag{10}$$

where $\bar{Z}_j^{c(\nu)} = \delta_{j,c(\nu)}$ is the one-hot encoded target for the mean, with $c(\nu) \in \{1, \ldots, M\}$ the ground truth of the class of data sample ν, and θ_j is a trainable bias. In the binary classification problem, the one-hot encoded network output has dimension $M = 2$. In addition, we add an L2 regularization term $\mu \|\alpha\|^2$ to the loss with a fixed, small μ to decrease the entries of the order selection parameter α. This ensures that uninformative layers lead to a vanishing contribution to α but also implements a winner-takes-all mechanism when there is a strong imbalance between the difficulty of classification for each individual order. Nonvanishing entries in α hence indicate the contribution of the corresponding statistical order to the classification decision. We therefore quantify the synergy of the OSP by the difference in performance of the full OSP model with models that were pruned after training such that they only use a single order n at a time. This is achieved by setting the parameters $\alpha_{n' \neq n} = 0$ for the other orders. The participation ratio ρ,

$$\rho(\alpha) = \frac{\sum_n |\alpha_n|}{\left(\sum_n |\alpha_n|^2\right)^{1/2}}, \tag{11}$$

can be used as a measure of sparsity to relate the synergy to how many relevant statistical orders the network found in the data. It is minimal when there is only a single nonzero α_n and maximal when all α_n are equal. Thus, it increases when more different orders n are combined.

B. Synthetic datasets with controlled cumulant structure

We design synthetic datasets to assess the ability of the proposed networks to capture specific statistical orders in the input signals. To that end, we generate two groups of multivariate time series with desired statistics up to the third order. We control how the two groups differ with regard to the first three cumulants. Concretely, the time series are generated by simulating in discrete time a process defined by a stochastic differential equation (SDE), whose samples in the infinite time limit follow a Boltzmann probability distribution:

$$p(x) \sim \exp\left(-\beta L[x]\right). \tag{12}$$

Samples $x(t) = \{x_i(t)\}$ are produced by the following process defined with $L[x]$ as Lagrangian:

$$\frac{\partial x_i(t)}{\partial t} = -\Gamma \frac{\partial L[x]}{\partial x_i}(t) + \xi_i(t). \tag{13}$$

Here, ξ is the stochastic Gaussian increment (or Wiener process or more colloquially white noise) that obeys $\langle \xi \rangle = 0$, $\langle\langle \xi_i(t), \xi_j(t') \rangle\rangle = D\delta_{ij}\delta(t - t')$, and Γ is a constant parameter. Using a fluctuation-dissipation theorem [46], one finds that $\beta = 2\frac{\Gamma}{D}$ for x to follow the distribution in Eq. (12) shaped by the choice of $L[x]$.

The particular case of a Gaussian distribution corresponds to a quadratic Lagrangian:

$$L[x] = m^T x + \frac{1}{2} x^T J x, \tag{14}$$

where $J = J^{\mathrm{T}}$ is a symmetric matrix, such that the mean $\mu = \langle x \rangle$ and covariance $\Sigma_{ij} = \langle\langle x_i x_j \rangle\rangle$ of the distribution read

$$\mu = -J^{-1} m, \tag{15}$$

$$\Sigma = (\beta J)^{-1}, \tag{16}$$

while the third order, $S_{ijk} = \langle\langle x_i x_j x_k \rangle\rangle$, vanishes. The more general case with the additional third-order statistics can then be developed in the spirit of field theory with a small perturbation (or correction) on the Gaussian case. The corresponding Lagrangian reads

$$L[x] = m^T x + \frac{1}{2} x^T J x + \frac{1}{3!} \sum_{ijk} K_{ijk} x_i x_j x_k. \tag{17}$$

For a sufficiently small tensor K, the first three cumulants can be approximated as (see Appendix A)

$$\mu_i \approx -\sum_j (J^{-1})_{ij} m_j - \frac{1}{2\beta} \sum_{jkl} (J^{-1})_{ij} K_{jkl} (J^{-1})_{kl}, \tag{18}$$

$$\Sigma_{ij} \approx \frac{1}{\beta} (J^{-1})_{ij}, \tag{19}$$

$$S_{ijk} \approx -\frac{1}{\beta^2} \sum_{i'j'k'} K_{i'j'k'} (J^{-1})_{ii'} (J^{-1})_{jj'} (J^{-1})_{kk'}. \tag{20}$$

This approximation only holds for a small deviation from the Gaussian case, which limits the amplitude of the third-order statistics. Concretely, we introduce a safety parameter s to ensure that a local minimum of the potential $L[x]$ exists and that the distance between the local minimum and the next maximum is large enough to fit s standard deviations of x in any direction. When integrating the SDE, large s ensures a low probability for a sample initialized in that local minimum to evolve further than the adjacent local maximum and prevents it from falling into the unstable region (we compute the scaling of the escape probability with s in Appendix C). In this case, the ratio of third- to second-order cumulants of x, projected to any eigendirection $e^{(v)}$ of J, is maximally allowed to be

$$\left| \frac{\sum_{ijk} S_{ijk} e_i^{(v)} e_j^{(v)} e_k^{(v)}}{\left[\sum_{ij} \Sigma_{ij} e_i^{(v)} e_j^{(v)}\right]^{3/2}} \right| = \frac{1}{s}. \tag{21}$$

In Appendix B, we show how to compute a suitable tensor K under this constraint. Additionally, we redraw samples that deviate too strongly from the local minimum of $L[x]$, which corresponds to introducing an infinite potential wall where $L[x]$ negatively exceeds the local minimum. Due to the low probability mass in that area for sufficient safety s, we do not need to account for this potential wall in the cumulant estimates. This way, we avoid issues with the probability density Eq. (12) not being normalizable. Alternatively, it would also be possible to include a positive definite fourth-order term in the Lagrangian and account for this in the calculation of the arising data statistics; however, the more simple third order with redrawing and a safety parameter $s = 5$ proved to work satisfactorily for the purposes of this work.

We create datasets where the classification difficulty is controlled for each statistical order. To do so, we generate time series grouped in two equally sized classes, whose cumulants differ by a scaling factor that serves as contrast. For example, we draw a reference class with some randomly drawn cumulants. A class contrast of 1.5 in the mean and 1 otherwise then corresponds to the second class having 1.5 times larger mean and all other cumulants exactly the same. Due to the stability constraint, the third-order cumulant may not become arbitrarily large. Practically, we use the class with the larger third order as the fixed cumulant and generate the second class from the first using the reciprocal contrast. This way, in any order, the larger cumulant is the contrast times the smaller cumulant.

C. Real datasets

We test the ability of our model to classify on a selection of time-series-classification datasets that have previously been used as benchmarks [40]. We select a subset of 18 multivariate time series datasets of which the data dimensionality allows us to compute the cumulants of the classes directly on the input. Before training, we shift and rescale the data to centralize them around zero and obtain unit variance over all data points. This is required because of the unboundedness of the gain function, which is a third-order polynomial, as well as numerical stability when computing the cumulants on the data directly.

III. RESULTS

We first benchmark the models proposed in Sec. II A with synthetic data generated with controlled statistical patterns (see Sec. II B). We show how the trained parameters can be analyzed to infer what statistical orders are relevant for the classification. The trained network parameters $\alpha_{1 \leqslant l \leqslant 3}$ particularly allow us to determine the optimal gain function for each dataset. Subsequently, we study classification of real multivariate time series. We find that the OSP classifies efficiently compared with the ML model, and we observe that the various datasets combine the statistical orders in a different way. The optimal network typically operates with a synergy of different orders.

A. Benchmarking with synthetic data

Classification may be performed based on one or several statistical orders measured on each sample time series. Each order thus defines a contrast between the two classes, which corresponds to a baseline classification accuracy when using the corresponding statistical order alone to perform the classification. This contrast is controlled by the coefficients of the two classes in the generative model, as explained in Eqs. (13) and (17).

We also compare the respective classification accuracy of the biological and ML architectures while varying the class contrasts to see how they are combined. Note that cumulants of orders higher than three are nonvanishing and may also hold information on the classes, but the neuronal networks by

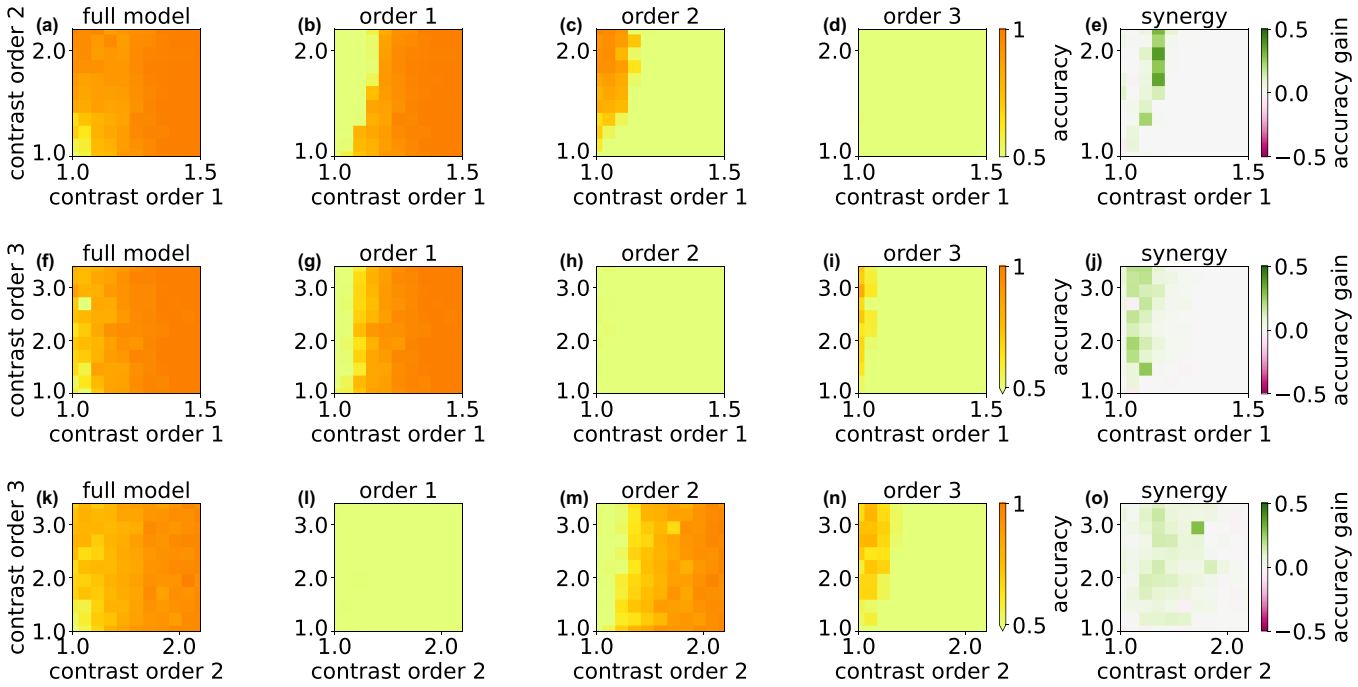

FIG. 3. Classification accuracies for varying task difficulties. Datasets with two equal-sized classes that differ in two statistical orders are classified using the order-selective perceptron (OSP). Along the axes of each diagram, the contrast between the two classes is varied between one (no class difference) to an arbitrarily chosen upper scale. (a), (f), and (k) Test set accuracy of the full model. (b), (g), and (l) Accuracy obtained by the OSP when pruning all α_n except α_1 after training. (c), (d), (h), (i), (m), and (n) Analogous to (b) for α_2, α_3, respectively. (e), (j), and (o) Synergy, which is the difference in accuracy between the full model and the best single order. Here, the synergy is always vanishing or positive, and the color scale is chosen to be consistent with the analogous model comparison in Fig. 4. In (a)–(e), classes are separated by a difference in the mean and covariance. In (f)–(j), classes are separated by a difference in the mean and third-order cumulant. In (k)–(o), classes are separated by a difference in the covariance and third-order cumulant.

construction neglect them. Each of the two classes comprises 100 samples, which have dimensionality $N = 5$ and $T = 100$ time steps per sample. We repeat the classification 25 times. For each binary classification, two of the three contrasts are chosen to be different from one, thus contain the information about the class, while the third contrast is the same for both classes, having contrast of unity.

We start with testing whether the OSP can be efficiently trained to perform classification. Figure 3 shows the training accuracy for (Gaussian) inputs, where the contrasts of orders one and two are varied, while the third cumulant is zero. Accuracy ranges from chance level (50% for the considered case of two balanced classes, in yellow) to perfect discrimination corresponding to 100% (in orange). Here, successful training means that the OSP captures the most relevant class contrast(s) among the three cumulant orders. As expected, the accuracy of the full model increases when either contrast of order one or two increases [see Fig. 3(a)].

Investigating the contribution of individual orders in isolation, the accuracy increases with the corresponding contrast between the classes, as shown when varying either order one or two in Figs. 3(b) and 3(c). The contribution of the uninformative third order stays chance level, Fig. 3(d). In Figs. 3(f)–3(o), the same qualitative behavior is observed for all different combinations of informative and uninformative orders. This shows that the OSP successfully manages to capture the relevant orders for classification, leading to the question of their combination when two or more orders are relevant.

1. *Synergy between relevant cumulant orders*

The accuracy of the full model [Fig. 3(a)] is larger than the accuracy obtained when single orders are taken individually [Figs. 3(b)–3(d)]. The difference between the accuracy for full training and the maximum of single-order accuracies, shown in Fig. 3(e), can be used as a measure of synergy: It indicates how the OSP makes use of the combination of the different statistical orders, specifically in the border between the regions of high single-order accuracy. Thus, the OSP (full model) combines the different orders relevant for the classification, beyond simply selecting the most informative order.

We observe a bias between the accuracies for single-order models. Although the synthetic data were generated to have comparable contrasts across the orders, we see that the border where the OSP (full model) switches from one order to another does not follow a diagonal with equal contrast. This indicates a bias favoring lower statistical orders over higher ones. This may partly come from the fact that higher statistical orders are noisier because they require more time points for estimation. The estimation error causes fluctuations which may affect the training by the parameter updates. In addition to this implicit bias toward extraction of information from lower orders, the normalization imposed on the α_n

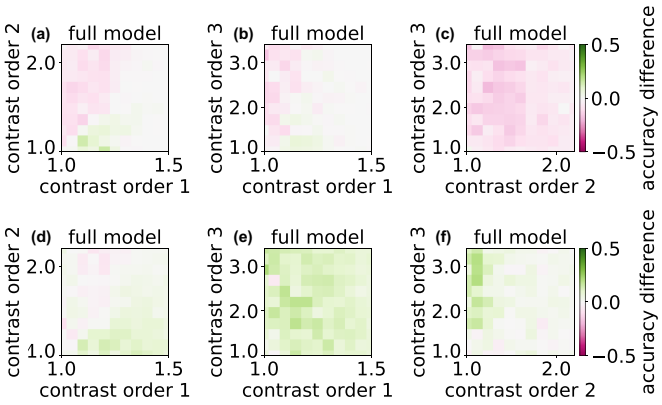

FIG. 4. Comparison of the order-selective perceptron (OSP) and the machine-learning (ML) model. Difference in accuracy between the OSP and the ML model for the datasets in Fig. 3. (a)–(c) Accuracy comparison between the OSP and the ML model. (d)–(f) Accuracy comparison between the OSP and the constrained ML model. The color code shows the difference in accuracy between the OSP and the respective ML model, green indicating that the OSP outperforms the ML model.

during training (see Sec. II A 4 for details) implements a soft winner-takes-all mechanism that likely reinforces the bias(es). In Appendix D, we discuss the accuracy gain of the OSP compared with a restricted OSP, as opposed to the pruned OSP, which can be seen as an alternative way to quantify synergy.

2. *Comparison with ML architecture*

Last, we compare the OSP with the ML model that has more trainable weights (or resources). As shown in Appendix E, the ML network performs well and can discriminate between informative orders. From Figs. 4(a)–4(c), we find that the accuracy for the OSP is like that of the ML, although mostly a bit lower presumably due to its lower number of trainable parameters [weights B, compare Figs. 2(b) and 2(c)]. Nevertheless, it can efficiently combine information distributed across statistical orders to perform the classification and slightly outperform the ML model on mean-based classification.

Presumably, this is due to the difference in the number of trainable weights. To test this, we define a constrained ML network where we subsample NM weights to be trainable and make sure those are distributed approximately uniformly over the entries of $B^{(n)}$ for the different orders n. The remaining weights stay fixed at their values at initialization. The OSP model exhibits a significant performance increase compared with this constrained ML model, which has the same number of trainable parameters as the OSP [see Figs. 4(d)–4(f)].

B. Identification of relevant orders from trained parameters

In the trained OSP, the α values describe how the different input cumulants are combined in the readout to achieve classification; see Methods for details, Fig. 2(b). All parameters prior to the application of the nonlinear gain function are the same for each statistical order. One may therefore interpret the absolute value of each component $|\alpha_n|$ as a measure of the contribution of order n to the classification. In Fig. 5, each order n is color-coded in red (mean), green (covariance), and blue (third order), and the color value is proportional to $|\alpha_n|$, scaled by the reciprocal of the largest $|\alpha_n|$ of all datasets.

As expected, the values $|\alpha_n|$ increase depending on the input contrasts corresponding to cumulant order n. Again, we find suppression in α due to the competition between the orders when the contrast in one of the informative orders dominates over the other. Unlike the single-order accuracy, however, here, we find overlapping regions of nonvanishing α_n corresponding to the individual informative orders n. These are necessary to form the synergy regions found in Fig. 3. Thus, both informative orders are detected in this region. We find again in Fig. 5 the effect of the bias in Sec. III A that favors lower cumulant orders over higher ones in the training, which is further amplified by the normalization imposed on the $\|\alpha\|$.

In Fig. 3, we observed a synergy effect in the performance along the boundary between regions where individual cumulant orders dominate the classification. These regions coincide with the regions where only one individual α_n is nonvanishing. On the boundary between two such regions, both corresponding α's are nonvanishing. To confirm that the synergy is indeed linked to this boundary, we display the synergy found in Fig. 3 to the participation ratio of α [Eq. (11)], which we use as a measure of sparsity here. Indeed, the measures are correlated, although the participation ratio does not account for the fact that α_n scales up to different maxima for each order n.

We next investigate the corresponding nonlinear gain function ϕ shaped by the trained α. Noticeably, higher-order components contribute significantly despite their lower combination parameter α_n. We display the gain functions after training for three contrast combinations per synthetic dataset, see insets (i)–(iii) in Fig. 5. The regions are dominated by a single order, so the gain functions are nearly ideal linear, quadratic, or cubic functions, as expected. In the synergy regions, more complex gain functions arise.

We would like to highlight that the identification of the optimal gain function is specific to the network architecture of the OSP. Although in the ML network, efforts can be made to make α scale similarly to the OSP by rescaling the weights of each order $B^{\mathrm{ML}(n)}$ appropriately; the resulting order combination weights α do not correspond to a gain function. This would require joint intermediate variables y^t that would be the same for each order as they naturally appear in the OSP due to the shared weights. In the Appendix F, we furthermore show that the OSP is not limited to the first learned statistics but adapts quickly to changes.

To summarize, the parameters α can be used to read out which statistical order is dominant to distinguish between the classes. When several α_i's are nonzero, a synergy from combining cumulants of different orders can be expected, and the optimal gain function after training differs from pure linear, quadratic, or cubic functions.

C. Extraction of statistical patterns from real datasets

As a next step, we train and benchmark the OSP on 18 datasets previously used for benchmarking of time series classification [40]. They consist of multivariate time series with

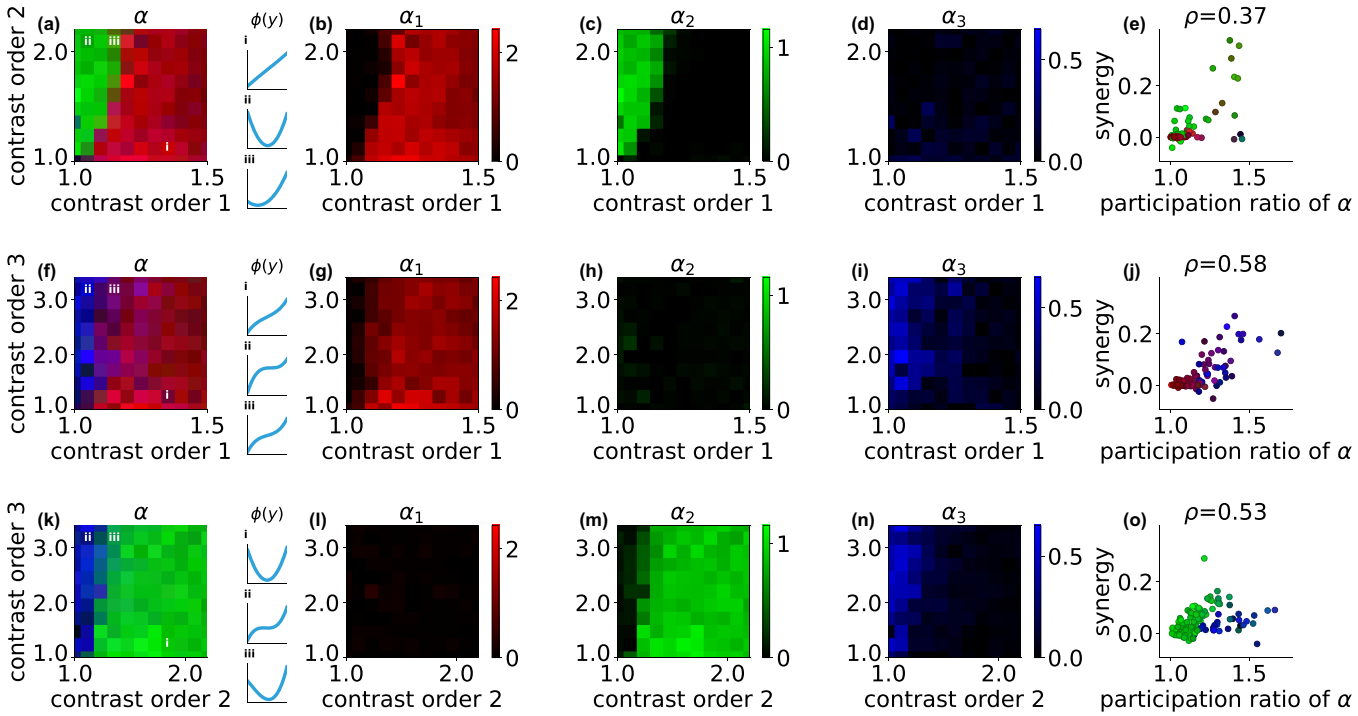

FIG. 5. Readout parameters for varying task difficulties. Datasets with two equal-sized classes which differ in two of their first three cumulants (rows) are classified using the order-selective perceptron (OSP). The weight vector $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ controlling the combination of orders one, two, and three is displayed for the full model (left column) in color code, where the value α_n for each order n is displayed by its corresponding rgb value: first-order α_1 (red), second-order α_2 (green), and third-order α_3 (blue). Insets show the corresponding gain function at different points of the parameter regime for the OSP. The second to fourth columns, respectively, show the individual values of each $\alpha_{1\leqslant i\leqslant 3}$. The synergy is displayed over the participation ratio of α (right column). On the axes of each plot, the contrast between the classes is varied, starting from one (no class difference) to an arbitrarily chosen upper scale. Above each panel, the Spearman rank correlation between the synergy and participation ratio is given.

a diversity of input dimensionalities, durations, and sample sizes, as well as number of classes. Figure 6 displays the classification results of the OSP for selected datasets that have different statistical structures. For the example in Figs. 6(g)–6(i), a large synergy from combining mostly the first and third orders evolves in the first few epochs. Decoding on individual statistical orders, an accuracy of 53% is achieved for the mean, 32% for the covariance, and 42% for the third order. The total classification accuracy combining all orders reaches 75%. As we quantify synergy by the difference between the final accuracy and the largest single-order accuracy, this corresponds to a synergy of +22%.

The optimal gain function corresponding to this statistical structure is consequently nearly point symmetric. The fluctuations of the samples mostly pass through the region close to the origin. Only some classes operate in the regime of strong nonlinearity. Figures 6(a)–6(c) and Figs. 6(d)–6(f) show similar results for the Epilepsy and JapaneseVowels datasets, respectively. Here, the optimal gain functions are mostly quadratic or linear, respectively.

In Fig. 7(a), we compare the performance of the benchmark datasets of the OSP to the ML model. Our goal is less to compare with the state of the art with a perceptronlike model (see Ref. [32] for recent results)and more to showcase how appropriate processing of statistical information can improve network capabilities. Nevertheless, the performance of the OSP often approaches that of the corresponding ML model. In most of the cases, both OSP and ML perform above chance level, which we empirically evaluate using similar architectures with untrained parameters. Accuracy remains at chance level only for two datasets: HeadMovementDirection and SelfRegulationSCP2.

For synthetic data (Fig. 4), the OSP yielded mostly lower classification accuracy compared with the ML approach but outperformed a constrained ML model in which the number of trainable parameters equals those of the OSP. We observe the same for the benchmark datasets in Fig. 7: In comparison with the constrained ML model, we typically find a significant performance increase [see Fig. 7(b)], whereas the ML model that trains directly on the full input cumulants exhibits even higher performance [see Fig. 7(a)].

We furthermore extract the relevant statistical orders in the data by inspecting the trained values of α. With the coloring of each dataset according to its cumulant combination weights $\alpha = (\alpha_1, \alpha_2, \alpha_3) = (\text{red}, \text{green}, \text{blue})$, the broad color spectrum displayed in Fig. 7 shows a wide variety for the relevant statistical orders in the different datasets.

In Fig. 7(c), we display the accuracy increase with respect to the best single order for the different datasets. While the color indicates the dataset, both size and transparency encode the training performance of the OSP in relation to the pretraining accuracy. Datasets that are less well classified by the OSP are thus displayed smaller and more transparent. Based on the synthetic data, we expect larger synergy where

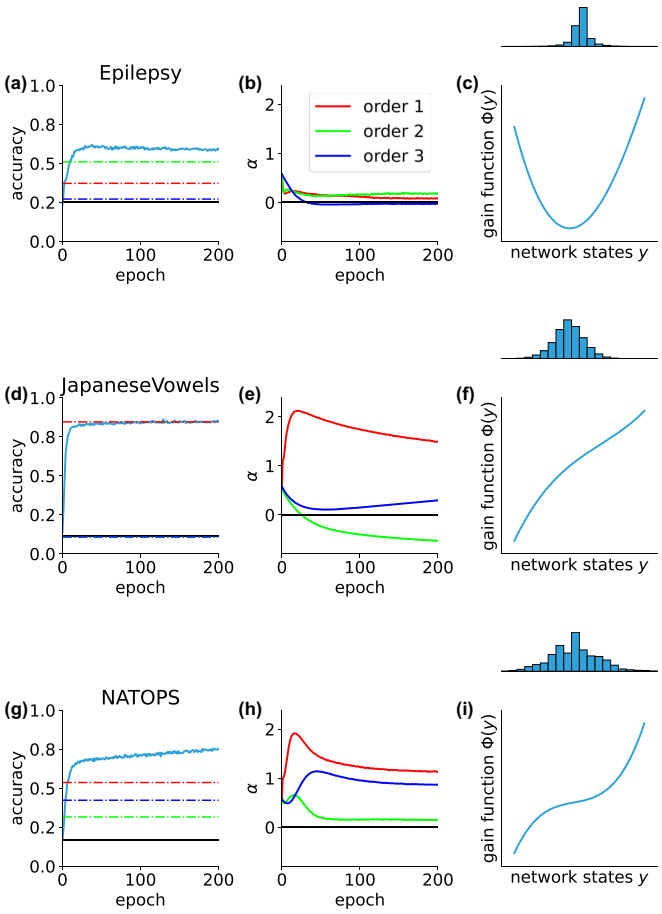

FIG. 6. Training on a few example benchmark datasets. (a)–(c) Training progress and final gain function in the order-selective perceptron (OSP) for Epilepsy, (d)–(f) JapaneseVowels, and (g)–(i) NATOPS datasets, respectively. Accuracy of the full network (blue curve) and single-order accuracy at the end of training for orders one (red), two (green), and three (blue) and at the beginning of training (black) as horizontal lines (left). Evolution of cumulant combination weights α (middle). Gain function $\phi(y)$ resulting from final α, including the density of samples in the domain (right).

different orders are combined more strongly. In Fig. 7(c), we therefore relate synergy to the participation ratio ρ (see Sec. II A 4). Although the results on synthetic data suggest a nonuniform weighting between the different orders n, we observe synergy effects also in complex datasets, specifically from combining more than one statistical order. In this figure, we observe a correlation between the participation ratio and the observed synergy in the different datasets. Details about the dataset and network parameters can be found in Appendix G.

IV. DISCUSSION

In this paper, we analyzed statistical processing for time series by a perceptronlike model with a tuneable nonlinearity. We showed that the form of the nonlinearity determines which statistical patterns, quantified via cumulants up to the third order, are transformed to shape the network output and how the different orders jointly contribute to the output to solve the task. The gain function that is optimal for the time series classification tasks depends sensitively on the relative importance of class-specific differences (or contrasts) in each cumulant. With the OSP, informative statistical properties of the data can be revealed with minimal model complexity.

This minimal complexity is achieved by avoiding the explicit computation of cumulants at several orders. Instead, the selection of the relevant cumulants of the input x^t, via those of the intermediate variable y^t, results from the combination of learning rules for the input weights and the nonlinearity. While the input cumulants, particularly the higher orders, require huge tensors for N-dimensional multivariate time series with large N, the cumulants of y^t scale polynomially with M, the number of classes, which is already often lower. Since additionally, only the diagonal of the network state cumulant is computed, the computational cost reduces further to only M entries per order. Furthermore, the number of trainable parameters reduces from $K + M(N + N^2 + \ldots + N^K)$ for the ML network to $K + MN$ for the OSP (with K the maximum cumulant order).

Naturally, this reduced model complexity comes at lower performance of the OSP than the explicit computation of all cumulant orders in the ML model. However, the network layer B practically accumulates as much "information" as possible from the input cumulants in the intermediate variables for classification: We have shown that the OSP performs in between the ML model acting on the same cumulants as the OSP and the constrained ML network, which trains only a reduced set of weights of the original ML architecture (to match the number of trained parameters to those of the OSP). The original ML network thereby acts as a theoretical upper bound for the OSP, as it can freely combine all of the input cumulants. The fact that the OSP outperforms the constrained ML model shows the efficient computation in the OSP despite the competition between the different orders. For neural classifiers that perform at the current state of the art, this may suggest as a metric their flexibility with regard to the adaptive propagation of cumulants. For example, in the spirit of Ref. [47], in which the authors have shown superior performance of covariance encoding in reservoir computing compared with linear encoding, an OSP could be combined with a recurrent reservoir to combine the benefits of both. A thorough study of more sophisticated, large-scale architectures that utilize high-order cumulants in this regard remains for future work.

The regularization-induced competition between the orders in fact strongly affects how the OSP combines cumulants for classification. We created synthetic data with tuneable cumulant structure to investigate this dependence in detail. Our SDE-based algorithm generates data with known cumulants up to the third order. Field theory can be used to obtain both estimates of the higher-order structures and more accurate estimates of the cumulants, which are given by a series of coefficients with decreasing weights. It is, however, limited to statistics that does not deviate too strongly from the exactly solveable Gaussian theory. To the best of our knowledge, algorithms that would provide the desired controlled data for multivariate time series are lacking. It is noteworthy, though, that the CuBIC technique exists [48,49], which creates spike trains with a given desired cumulant structure. The algorithm presented here can also be used to create static stimuli.

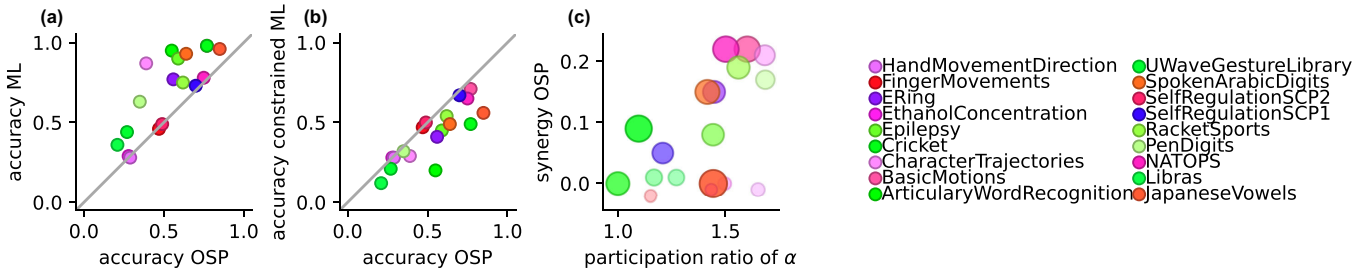

FIG. 7. Classification results on empirical datasets. (a) Comparison of the accuracy achieved in the machine-learning (ML) and order-selective perceptron (OSP) models. Colors indicate the datasets as listed on the side. (b) Analogous to (a) with a constrained ML model with as many trainable parameters as the OSP. (c) Synergy (see Sec. II A 4) of the OSP for different datasets. Size and opacity indicate the improvement in classification accuracy as compared with the pretraining accuracy (larger size and more contrast for larger improvement).

Using such synthetic data with known underlying statistical structure, we found that the OSP preferentially classifies using only a single order, relying on only one cumulant, if the difference between the classes is more expressed in this cumulant than in the others. This is reinforced by the regularization of the cumulant combining layer, although the same behavior qualitatively also occurs for unconstrained α. In that case, however, the network is classifying a bit less well overall. Likewise, it is possible to rescale the layers B and α of the OSP without changing the network output. However, because this rescales both the gain function and the inputs it receives, such a rescaling is ultimately inconsequential and therefore neglected. On the boundary between two different preferred orders, we observe synergy effects from combining cumulants through different paths. The class contrast where this boundary occurs appears, however, nontrivial. From both the position of the boundary and the amplitudes of the order selection parameter α in the regions next to the boundary, we conclude that the network tends to prefer lower-order cumulants over higher ones and selects the preferred order using this hierarchy. Applied to real-world applications that serve as benchmark datasets, often several orders are combined, and we regularly observe synergistic effects. Combination of statistical information appears all the more important for complicated data structures. For any combination of orders, there is a dataset found among them whose order selection parameters filter for just this combination. The gain functions that the order combination parameters translate to are consequently uniquely shaped.

Neuronal network architectures like the OSP have been widely used for different purposes in the context of ML [50] and computational neuroscience [51–53], with nonlinear neurons governed by bounded sigmoidlike profiles like in the Wilson-Cowan model [54,55] or rectified linear units [56]. In Ref. [57], the authors give a thorough review of different gain functions and their influence on computational performance. We stress that this paper differs from the use of fixed (nontrainable) gain functions that are typically used. The choice of the gain function, however, matters for the statistical processing performed by the network. The purely linear gain function corresponds to the original perceptron [42–44] where each input cumulant is mapped to its counterpart at the same order in the output. In contrast, a nonlinear readout can perform crosstalk between input cumulants of several orders, combining them into, e.g., the output mean. On the other hand, correlation patterns have recently been proposed as the basis of "information" embedded in time series that can be processed for, e.g., classification [21]. This so-called covariance (de)coding has been shown to yield larger pattern capacity than the classical perceptron that relies on mean patterns when applied to time series [41].

In conclusion, the choice of the gain function at the same time is a choice of what statistical information the network should be sensitive to. Gain functions that are more complex than simple polynomials thus combine many if not all orders of cumulants. We chose a polynomial here for interpretability; in practice, high performance could be achieved by other choices of tuneable nonlinearities. However, a fixed gain function also means that there is a fixed relation between the different statistical contributions of the underlying probability density. Translating a given gain function into a Taylor series around a working point can give insight into what cumulants a network may be particularly sensitive to, although this does not yield a one-to-one translation to our adaptive gain function. The demeaning of the time series employed here translates to a different weighting of cumulants depending on the mean of the sample; it still holds that this form of input processing is fixed and predefined. In biological data, adaptive gain has been observed [58,59], although not identical to the simple mechanism presented here.

We therefore suggest making the choice of gain function with care. There have been, in fact, works on comparing how well different prominent gain functions work in ML contexts [60,61], and some commonly used gain functions include trainable parameters [62]. A fully trainable gain function is presented in Ref. [63], which is of a similar polynomial type as that presented here. The aim of these works is to provide optimal performance for sophisticated networks on complicated tasks. We offer interpretability of the gain function in terms of statistical processing and provide a link to biological neural networks.

The gain variability found for example in Refs. [58,59] shows that neurons may be sensitive to the statistical features of their inputs, and the variability of activity also appears to be linked to behavior [1–3] and energy consumption [64]. It may also play a role in representing uncertainty in terms of probabilistic inference [10,11,65,66]. Structured variability thus seems to play an integral role in biological neural networks. As far as information encoding through neural correlations is superior to mean-based representations, this may help us

understand why such sensitivity has emerged in neuronal circuits. This reasoning has already inspired the covariance perceptron [21,41].

Of course, just like the covariance perceptron, the OSP is not a biologically detailed network model. It is built upon the most fundamental building block, a feedforward layer of neurons, followed by a gain function. Instead of the spiking activity of individual neurons, it treats populations of neurons as a unit with an activity that resembles the average firing rate. The linear summation of inputs closely resembles the neuron dynamics typically assumed for cortical models [53,67–70] but also in ML, including the perceptron [42–44]. The main difference to previously presented methods is the assumption of information being hidden in the higher-order statistics of the network activity. While the propagation of stimuli through neural networks is widely accepted to be modeled suitably by linear summation of inputs and application of subsequent transfer functions, this point of view draws more toward understanding the computation based on the inputs.

Regarding both computational capacity and biological realism, it would be interesting to also study a multilayer or a recurrently connected version of the OSP, which extends outside the scope of this paper. It is of particular relevance to study the performance gain from time-lagged cumulants. Further, the cross-correlations of the intermediate network state, which are currently not influencing the classification, may be useful in future work.

ACKNOWLEDGMENTS

The authors would like to thank Prof. E. Keogh and all the people who have contributed to the UCR Time Series Classification Archive. This paper was partly supported by European Union Horizon 2020 Grant No. 945539 (Human Brain Project SGA3), the Helmholtz Association Initiative, the Networking Fund under Project No. SO-092 (Advanced Computing Architectures), the Bundesministerium für Bildung und Forschung Grant No. 01IS19077A (Jülich), and the Excellence Initiative of the German federal and state governments [G:(DE-82)ERS-PF-JARA-SDS005]. This open access publication was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 491111487.

APPENDIX A: COMPUTATION OF CUMULANTS FOR THE ARTIFICIAL DATA MODEL

The cumulants of the probability density $p(x) \sim \exp[-\beta(m^T x + \frac{1}{2}x^T J x + \frac{1}{3!}\sum_{ijk} K_{ijk}x_i x_j x_k)]$ (with a potential barrier) can be derived using field theory [71]. To this end, the Gaussian model with $p(x) \sim \exp[-\beta(m^T x + \frac{1}{2}x^T J x)]$ acts as a baseline, and the cubic part forms a small correction for states near those expected from the Gaussian distribution. From the Gaussian baseline follows that the propagator of the system is its covariance:

$$\Sigma_{ij} = \frac{1}{\beta}(J^{-1})_{ij}. \tag{A1}$$

With m acting as a source term, the mean follows as

$$\mu_i = -\sum_j (J^{-1})_{ij} m_j. \tag{A2}$$

The cubic correction leads to a three-point vertex, which has the value $-\frac{\beta}{3!}K$. In first-order corrections to the Gaussian theory, therefore, diagrams with a single vertex need to be computed. For the mean, this leads to a tadpole diagram, where two of the three legs of a vertex are connected by the propagator and a single external leg. There are three different such diagrams from the three possible choices of the external legs, leading to the corrected mean:

$$\mu_i = -\sum_j (J^{-1})_{ij} m_j - \frac{1}{2\beta}\sum_{jkl}(J^{-1})_{ij}K_{jkl}(J^{-1})_{kl}. \tag{A3}$$

This mean may not deviate too strongly from the Gaussian one. There are no diagrams with a single three-point vertex and two external legs, so up to first order, the covariance stays as in the Gaussian case. The third-order cumulant is simply a vertex with a propagator on each of its legs. A prefactor of 3! needs to be included for those. To summarize, this leads to the cumulants:

$$\mu_i \approx -\sum_j (J^{-1})_{ij} m_j - \frac{1}{2\beta}\sum_{jkl}(J^{-1})_{ij}K_{jkl}(J^{-1})_{kl}, \tag{A4}$$

$$\Sigma_{ij} \approx \frac{1}{\beta}(J^{-1})_{ij}, \tag{A5}$$

$$S_{ijk} \approx -\frac{1}{\beta^2}\sum_{i'j'k'} K_{i'j'k'}(J^{-1})_{ii'}(J^{-1})_{jj'}(J^{-1})_{kk'}. \tag{A6}$$

APPENDIX B: HOW TO CHOOSE SUITABLE PARAMETERS

To obtain reasonable results, the parameters need to be chosen with care. The cubic part K may not be too large, to ensure the system is both not too unstable and close enough to the Gaussian case that we can use field theory to approximate the cumulant statistics, however large enough that substantial third-order statistics arises for our desired dataset. To this end, we compare the expected fluctuations of states with the width of the safe part of the potential, the region between the local minimum and the local maximum.

For this comparison to be useful, however, we first need to ensure that, during all times of the evolution of the SDE, the data stay close to the Gaussian theory. Therefore, we simulate the SDE with random initial conditions drawn from a Gaussian distribution with mean and covariance identical to those expected according to Appendix A. Furthermore, we set the source $m = 0$ to simplify calculations. The mean of the data can be adjusted by a constant shift b of all data points after simulating the SDE. The quadratic part J needs to be chosen as a positive definite matrix that fits the desired covariance.

The remaining parameter to determine is therefore only K. It has to be chosen such that, for $m = 0$ and a given J, there is no direction of the potential where the Lagrangian $L[x] = \frac{1}{2}x^T J x + \frac{1}{3!}\sum_{ijk} K_{ijk}x_i x_j x_k$ rises monotonically. For a start, we assume that the cubic part will be of rank one, in the sense that there is a single direction in state space where the Lagrangian is cubic, and in all directions orthogonal to this one, the potential remains quadratic. We can then compose stronger cubic potentials.

With x' in the direction of nonvanishing cubic interactions, this means that there must be real solutions to the necessary

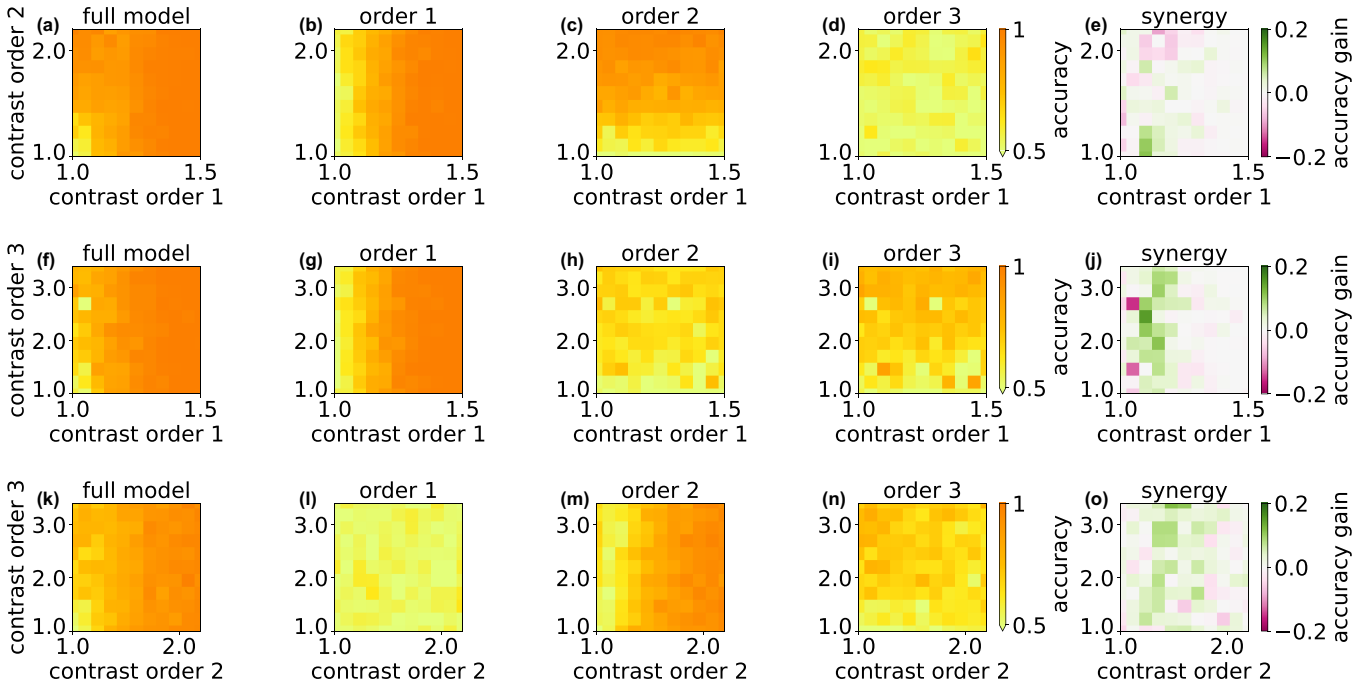

FIG. 8. Performance comparison of the order-selective perceptron (OSP) and a constrained OSP. (a) Accuracy of the OSP as in Fig. 3. (b) Accuracy of a constrained OSP with trainable α_1 and $\alpha_{2,3} = 0$. (c) and (d) Analogous to (b) with $\alpha_{2,3}$ trainable, respectively. (e) Accuracy gain of the OSP over the constrained OSP. In (a)–(e), classes are separated by a difference in the mean and covariance. In (f)–(j), classes are separated by a difference in the mean and third-order cumulant. In (k)–(o), classes are separated by a difference in the covariance and third-order cumulant.

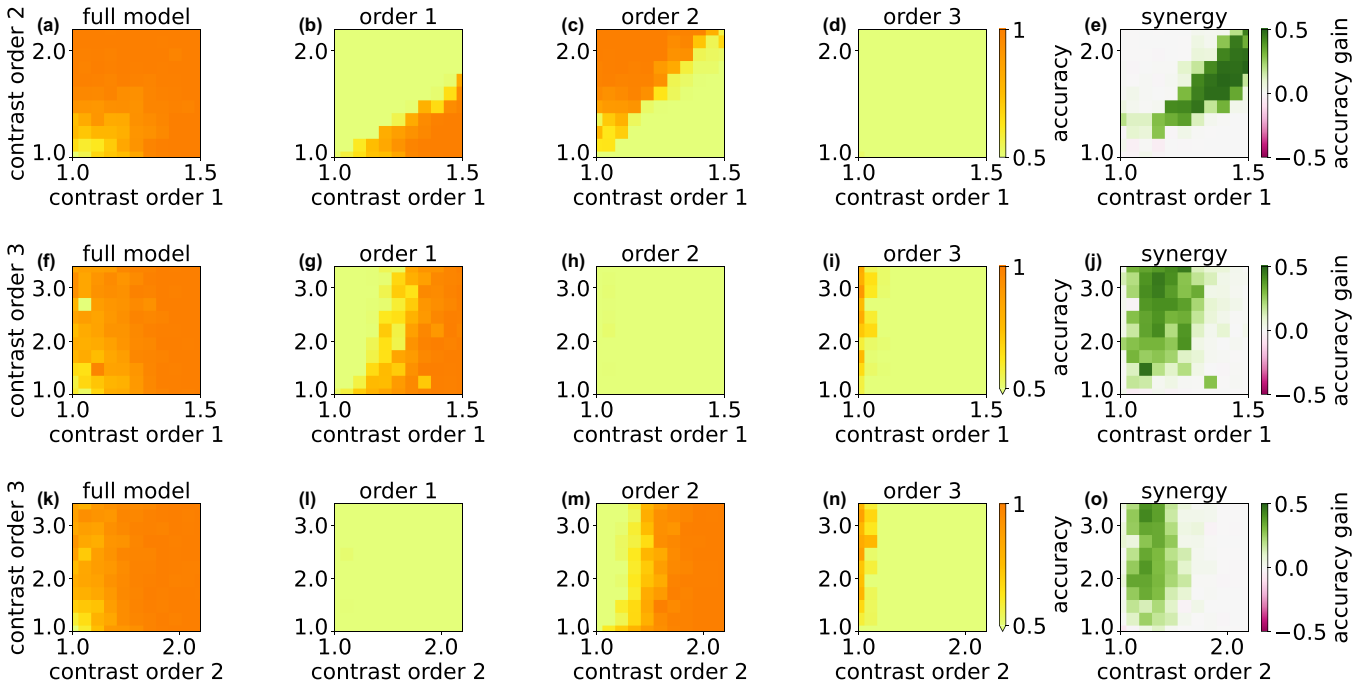

FIG. 9. Readout parameters for the machine-learning (ML) network with varying task difficulties. Datasets with two equal-sized classes which differ in two of their first three cumulants (rows) are classified using the ML network. The cumulant combination weight α is displayed for the full model (left) in color code, for the first (red), second (green), and third order (blue). Insets show the corresponding gain function at different points of the parameter regime for the order-selective perceptron (OSP). Consecutively, starting from the second from left, α_i for each individual order i from one (mean) to three (third-order cumulant) is showed. The axes of each diagram display the contrast between the classes underlying the datasets, starting from one (no class difference) to an arbitrarily chosen upper scale. In (a)–(e), classes are separated by a difference in the mean and covariance. In ((f)–(j), classes are separated by a difference in the mean and third-order cumulant. In (k)–(o), classes are separated by a difference in the covariance and third-order cumulant.

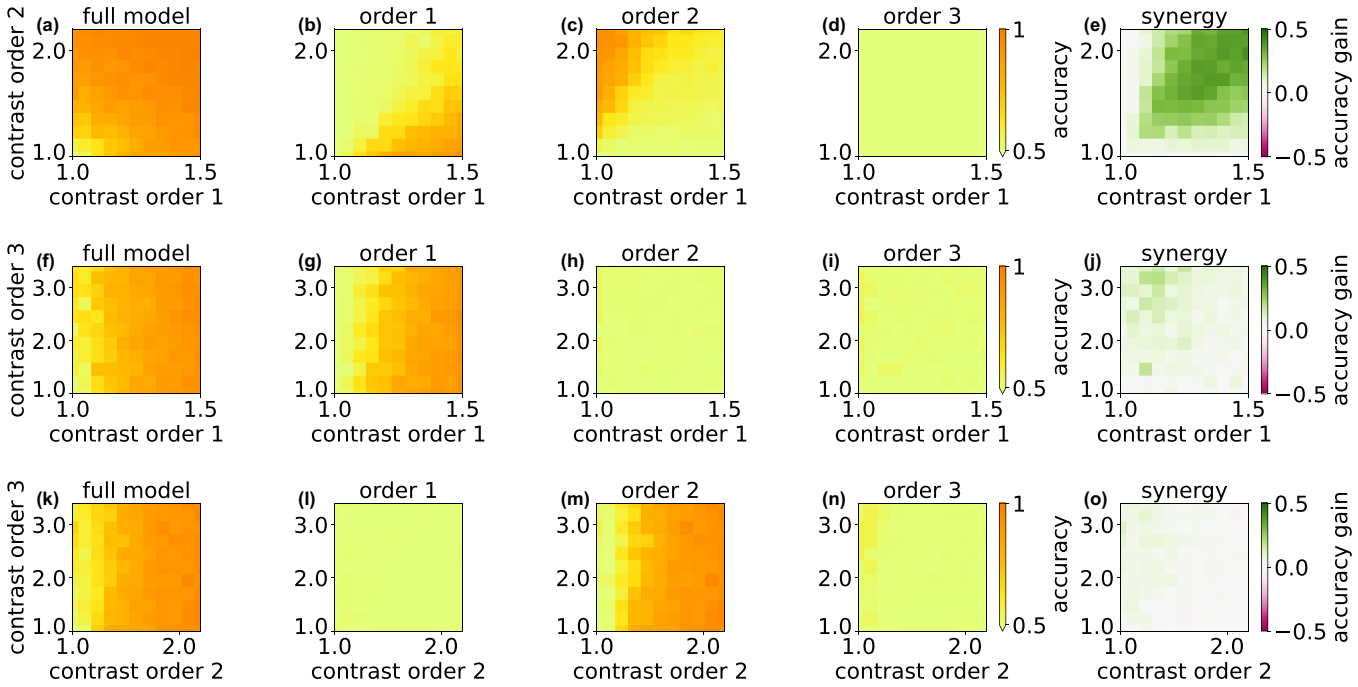

FIG. 10. Readout parameters for the constrained machine-learning (ML) model with varying task difficulties. Datasets with two equal-sized classes which differ in two of their first three cumulants (rows) are classified using the constrained ML network. The cumulant combination weight α is displayed for the full model (left) in color code, for the first (red), second (green), and third order (blue). Insets show the corresponding gain function at different points of the parameter regime for the order-selective perceptron (OSP). Consecutively, starting from the second from the left, α_i for each individual order i from one (mean) to three (third-order cumulant) is shown. The axes of each diagram display the contrast between the classes underlying the datasets, starting from one (no class difference) to an arbitrarily chosen upper scale. In (a)–(e), classes are separated by a difference in the mean and covariance. In (f)–(j), classes are separated by a difference in the mean and third-order cumulant. In (k)–(o), classes are separated by a difference in the covariance and third-order cumulant.

conditions for local minima or maxima:

$$\sum_j J_{ij}x'_j + \frac{1}{2}\sum_{jk} K_{ijk}x'_j x'_k = 0. \tag{B1}$$

Only then will a stable region exist for the data to evolve around. With K being rank one, it can be composed as an outer product of vectors v as

$$K_{ijk} = v_i v_j v_k. \tag{B2}$$

The condition for local extrema becomes

$$Jx' + \frac{1}{2}v(v^T x')^2 = 0. \tag{B3}$$

As desired, in any direction orthogonal to v (i.e., where $v^T x' = 0$), the potential is quadratic and thereby stable. It therefore suffices to look at $x' \parallel v$. In that case, when $x' = \|x\|e^{(v)}$,

$$Jx' + \frac{1}{2}v(v^T x')^2 = \|x\|Je^{(v)} + \frac{1}{2}\|x\|^2\|v\|^3 e^{(v)} = 0, \tag{B4}$$

where $e^{(v)}$ denotes the unit vector in direction of v. If $e^{(v)}$ is an eigendirection of J with eigenvalue $\lambda_{(v)}$, this simplifies further. Then one can solve for $\|v\|$ to obtain $v = \|v\|\, e^{(v)} = -2^{1/3}\|x\|^{-1/3}\lambda_{(v)}^{1/3}\, e^{(v)}$. Now the extrema of the potential lie on a line in direction $e^{(v)}$, at $x' = 0$ and $x' = \|x\|e^{(v)}$. The distance between the minimum and maximum therefore is $\|x\|$ and can be chosen to determine the cubic part of the Lagrangian. Here, we can choose to set

$$\|x\|^2 = s^2 e^{(v)\mathrm{T}}\Sigma e^{(v)} \tag{B5}$$

such that we expect s standard deviations to fit in this stable part of the potential (see Appendix A).

We can lastly compare the third-order cumulant to the second to find values of s for the third-order statistics to be strong enough compared with the fluctuations. To this end, we can map the second- and third-order statistics in some eigendirection $e^{(v')}$ of J [which is also an eigendirection of Σ with eigenvalue $(\beta\lambda_{v'})^{-1}$] to find

$$[e^{(v')\mathrm{T}}\Sigma e^{(v')}]^{3/2} = [\beta\lambda_{(v')}]^{-3/2} \tag{B6}$$

and

$$\begin{aligned}\sum_{ijk} S_{ijk}e_i^{(v')}e_j^{(v')}e_k^{(v')} &= -\frac{1}{\beta^2}[e^{(v')\mathrm{T}}J^{-1}v]^3 = -\frac{1}{\beta^2}\left[e^{(v')}\lambda_{(v)}^{-1}v\right]^3 \\ &= -\frac{\|v\|^3}{\beta^2\lambda_{(v)}^3}\delta_{vv'} = \frac{1}{\|x\|\beta^2\lambda_{(v)}^2}\delta_{vv'} \\ &= \frac{1}{s[e^{(v)\mathrm{T}}\Sigma e^{(v)}]^{1/2}\beta^2\lambda_{(v)}^2}\delta_{vv'} \\ &= \frac{1}{s}[\beta\lambda_{(v)}]^{-3/2}\delta_{vv'}.\end{aligned} \tag{B7}$$

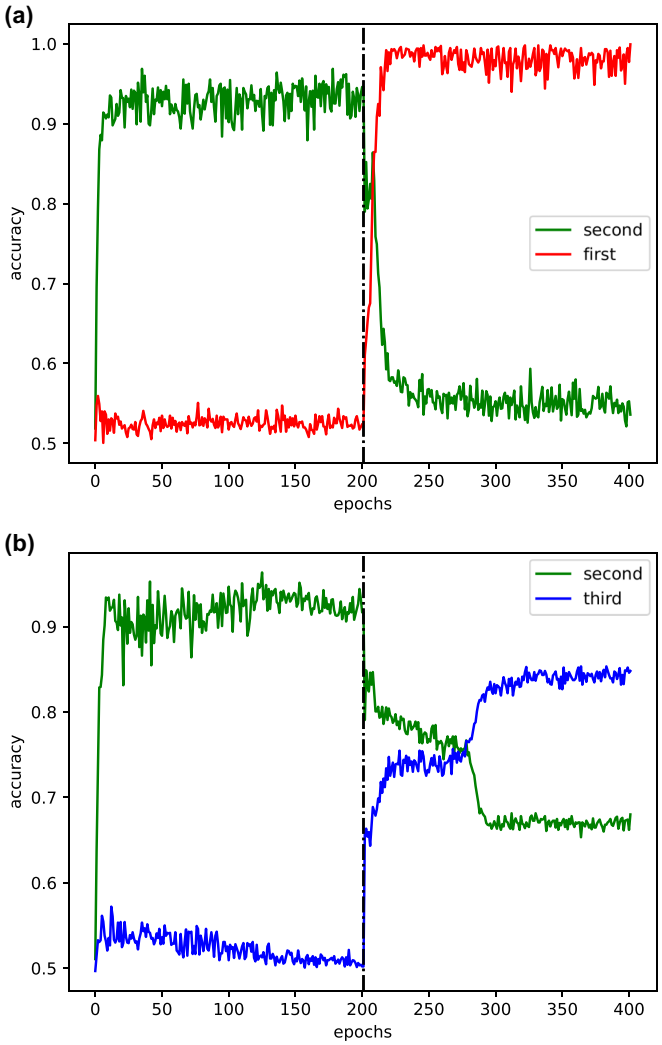

FIG. 11. Accuracy of the order-selective perceptron (OSP) for switching between datasets. The OSP is trained sequentially on two datasets with different contrasts, switched after 200 epochs. The panels show the accuracy of the OSP on both datasets during training, while adapting its parameters only to one. Colors indicate the most prominent contrast in the dataset for the mean (red), covariance (green), and third order (blue). (a) Switching from a covariance-dominated dataset [same datasets as in Fig. 5(a), annotated as (ii)] with weak difference in the mean to a mean-dominated dataset [same dataset as in Fig. 5(a), annotated as (i)] with weak difference in the covariance. (b) Switching from a covariance-dominated dataset [same dataset as in Fig. 5(k), annotated as (ii)] with weak difference in the third order to a third-order-dominated dataset [same dataset as in Fig. 5(k), annotated as (i)] with weak difference in the covariance.

From this follows

$$\left| \frac{\sum_{ijk} S_{ijk} e_i^{(v)} e_j^{(v)} e_k^{(v)}}{\left[\sum_{ij} \Sigma_{ij} e_i^{(v)} e_j^{(v)}\right]^{3/2}} \right| = \frac{1}{s}, \tag{B8}$$

for $e^{(v)}$ in the direction of the cubic perturbation.

It is straightforward to then construct a Lagrangian with a safe cubic potential in any direction by summing over all eigendirections of J:

$$K_{ijk} = \sum_{(v)} v_i^{(v)} v_j^{(v)} v_k^{(v)}, \tag{B9}$$

$$v^{(v)} = -s_{(v)}^{-1/3} [e^{(v)T} \Sigma e^{(v)}]^{-1/6} \lambda_{(v)}^{1/3} e^{(v)}, \tag{B10}$$

where $s_{(v)}$ controls the strength of the cubic potential in each eigendirection of J. For our purposes, $s = s_{(v)} = 5$ was used in all directions.

APPENDIX C: ESCAPE RATE OF THE ARTIFICIAL DATA MODEL

Without the resetting mechanism, the states would eventually escape the SDE, Eq. (13), with the Lagrangian of Eq. (17). One can ask how likely such an escape is. Altland and Simons [72] address this problem by reformulating the SDE in terms of the Martin–Siggia–Rose–de Dominicis–Janssen (MSRDJ) formalism. The escape probability on an exponential scale can be computed by considering the most likely escape path in phase space and the action accumulated along it. As a result, the escape probability is the Arrhenius factor. In our framework, this means that the escape probability is given by

$$p \sim \exp\left(-\frac{\Delta L}{D}\right),$$

where ΔL is the height of the potential barrier and D the variance of the driving noise. Within the MSRDJ framework, these results can be refined by considering algebraic (nonexponential) corrections due to fluctuations.

The extrema of $L[x]$ have been calculated in Appendix B. They lie in the origin with $L[x = 0] = 0$ and, for $K_{ijk} = v_i v_j v_k$ which has been constructed as a rank-one tensor from an eigenvector v to eigenvalue $\lambda_{(v)}$ of J, at

$$x = -\frac{2\lambda_{(v)}}{\|v\|^3} e^{(v)}.$$

Here, $e^{(v)} = \frac{v}{\|v\|}$ is the unit vector in the direction of v of which the norm $\|v\|$ is chosen to adjust the width of the local minimum. The potential at the local maximum:

$$L[x] = \frac{1}{2} x^T J x + \frac{1}{3!} \sum_{ijk} K_{ijk} x_i x_j x_k = \frac{2}{3} \frac{\lambda_{(v)}^3}{\|v\|^6},$$

then depends on $\lambda_{(v)}$ and $\|v\|$, which display the strength of the second- to third-order contributions. With v chosen such that the extrema of $L[x]$ are s standard deviations of x apart [c.f. Eq. (B5)], this relation is determined by

$$\|v\|^6 = \frac{4\beta\lambda_{(v)}^3}{s^2}.$$

This yields the scaling of the escape probability with the safety parameter s as

$$p \sim \exp\left(-\frac{s^2}{6\beta D}\right) = \exp\left(-\frac{s^2}{12\Gamma}\right).$$

TABLE I. Parameters of the real-world datasets. Listed for each of the benchmark datasets are the number of input variables N and output dimensionality M. The numbers of trainable parameters for the OSP and ML models are then computed for $K = 3$ cumulants considered.

Dataset	Classes	Input variables	OSP parameters	ML parameters
Articulary Word Recognition	25	9	228	20478
Basic Motions	4	6	27	1035
Character Trajectories	20	3	63	783
Cricket	12	6	75	3099
Epilepsy	4	3	15	159
Ethanol Concentration	4	3	15	159
ERing	6	4	27	507
Finger Movements	2	28	59	45531
Hand Movement Direction	4	10	43	4443
Japanese Vowels	9	12	111	16959
Libras	15	2	33	213
NATOPS	6	24	147	86547
Pen Digits	10	2	23	143
Racket Sports	4	6	27	1035
Self Regulation SCP1	2	6	15	519
Self Regulation SCP2	2	7	17	801
Spoken Arabic Digits	10	13	133	23793
UWave Gesture Library	8	3	27	315

For K, which are constructed from a sum of rank-one contributions composed of eigenvectors of J, as done throughout this paper, this calculation becomes more involved. The case of general K can be best solved numerically.

APPENDIX D: COMPARISON WITH CONSTRAINED ORDER PERCEPTRON MODELS

We defined synergy as the performance gain of the OSP compared with a pruned OSP, where all except a single entry of α are set to zero. This is motivated by the ability of α to indicate the contribution of corresponding statistical orders to the classification decision, as deduced from the competition between its entries. A different angle to the question of how much the OSP gains can be whether and when the OSP outperforms a constrained OSP, in which entries of α are set to zero throughout training.

Figure 8 shows a comparison of the performance of these models. Constrained OSPs ignore the contrasts in cumulants that they are not sensitive to and perform well above a minimum contrast in their corresponding cumulant order. Without the competition imposed on the unconstrained OSP, the performance is typically higher than the single-order accuracy given by the pruned networks in Fig. 3. Consequently, the difference of accuracy between these models is lower. However, we still observe a tendency toward an accuracy gain along an area in the space of contrasts. Noticeably, it coincides mostly with the border between areas in which individual orders dominate the unconstrained OSP (see Fig. 3). The OSP therefore actively benefits from combining different cumulants for classification.

APPENDIX E: SYNTHETIC DATA, ML MODEL

Training of the ML network on the synthetic data is displayed in Fig. 9. In Fig. 10, the training is repeated with an ML model, of which only MN weights are randomly selected to be trainable.

APPENDIX F: ADAPTATION TO CHANGES IN INPUT STATISTICS

For batch-wise training of the OSP on finite data, the network will be confronted with estimates of the cumulants in the process of training that deviate from the ground truth. Higher-order cumulants particularly require large datasets to yield an accurate estimate. A natural question therefore is whether the OSP can learn to adapt its weights after training when the input statistics change. To show this, we train sequentially on two different datasets with different contrasts. In Fig. 11(a), we switch between a dataset with dominant contrast between the second cumulants to a dataset with dominant contrast in the mean. In Fig. 11(b) the switch is from a dataset with dominant contrast in the second cumulant to one with dominant contrast in the third. The accuracy reached by classifying on one of the datasets quickly switches when training to classify the second dataset. The transition time is comparable with the time it takes for the training from random initialization. The different maximum accuracies for the individual datasets match those reported in Fig. 3.

APPENDIX G: DATASET AND NETWORK PARAMETERS

Table I shows the number of classes M and number of variables N for the datasets presented in Sec. III C as well as the resulting number of trainable parameters for the OSP and ML models. The constrained ML model selects as many parameters from those listed for the ML model to be trainable as are given in the column of the OSP model.

[1] B. E. Kilavik, S. Roux, A. Ponce-Alvarez, J. Confais, S. Grün, and A. Riehle, Long-term modifications in motor cortical dynamics induced by intensive practice, J. Neurosci. **29**, 12653 (2009).

[2] A. Riehle, S. Grün, M. Diesmann, and A. Aertsen, Spike synchronization and rate modulation differentially involved in motor cortical function, Science **278**, 1950 (1997).

[3] N. Shahidi, A. R. Andrei, M. Hu, and V. Dragoi, High-order coordination of cortical spiking activity modulates perceptual accuracy, Nat. Neurosci. **22**, 1148 (2019).

[4] J. H. Siegle, X. Jia, S. Durand, S. Gale, C. Bennett, N. Graddis, G. Heller, T. K. Ramirez, H. Choi, J. A. Luviano *et al.*, Survey of spiking in the mouse visual system reveals functional hierarchy, Nature (London) **592**, 86 (2021).

[5] P. Fries, Rhythms for cognition: communication through coherence, Neuron **88**, 220 (2015).

[6] I. Nauhaus, L. Busse, M. Carandini, and D. L. Ringach, Stimulus contrast modulates functional connectivity in visual cortex, Nat. Neurosci. **12**, 70 (2009).

[7] D. Rubino, K. A. Robbins, and N. G. Hatsopoulos, Propagating waves mediate information transfer in the motor cortex, Nat. Neurosci. **9**, 1549 (2006).

[8] T. K. Sato, I. Nauhaus, and M. Carandini, Traveling waves in visual cortex, Neuron **75**, 218 (2012).

[9] A. Arieli, A. Sterkin, A. Grinvald, and A. Aertsen, Dynamics of ongoing activity: Explanation of the large variability in evoked cortical responses, Science **273**, 1868 (1996).

[10] P. Berkes, G. Orbán, M. Lengyel, and J. Fiser, Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment, Science **331**, 83 (2011).

[11] G. Orbán, P. Berkes, J. Fiser, and M. Lengyel, Neural variability and sampling-based probabilistic representations in the visual cortex, Neuron **92**, 530 (2016).

[12] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, A neuronal learning rule for sub-millisecond temporal coding, Nature (London) **383**, 76 (1996).

[13] N. Caporale and Y. Dan, Spike timing–dependent plasticity: A Hebbian learning rule, Annu. Rev. Neurosci. **31**, 25 (2008).

[14] M. Gilson, T. Masquelier, and E. Hugues, STDP allows fast rate-modulated coding with Poisson-like spike trains, PLoS Comput. Biol. **7**, e1002231 (2011).

[15] J. Gjorgjieva, C. Clopath, J. Audet, and J.-P. Pfister, A triplet spike-timing–dependent plasticity model generalizes the Bienenstock–Cooper–Munro rule to higher-order spatiotemporal correlations, Proc. Natl. Acad. Sci. USA **108**, 19383 (2011).

[16] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory* (John Wiley & Sons, New York, 1949).

[17] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Taylor & Francis, Boca Raton, 1991).

[18] W. C. Abraham, Metaplasticity: Tuning synapses and networks for plasticity, Nat. Rev. Neurosci. **9**, 387 (2008).

[19] E. L. Bienenstock, L. N. Cooper, and P. W. Munro, Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex, J. Neurosci. **2**, 32 (1982).

[20] E. M. Izhikevich and N. S. Desai, Relating STDP to BCM, Neural Comput. **15**, 1511 (2003).

[21] M. Gilson, D. Dahmen, R. Moreno-Bote, A. Insabato, and M. Helias, The covariance perceptron: A new paradigm for classification and processing of time series in recurrent neuronal networks, PLoS Comput. Biol. **16**, e1008127 (2020).

[22] R. Gütig and H. Sompolinsky, The tempotron: A neuron that learns spike timing-based decisions, Nat. Neurosci. **9**, 420 (2006).

[23] F. Ponulak and A. Kasiński, Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting, Neural Comput. **22**, 467 (2010).

[24] F. Zenke and S. Ganguli, Superspike: Supervised learning in multilayer spiking neural networks, Neural Comput. **30**, 1514 (2018).

[25] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions, J. Big Data **8**, 53 (2021).

[26] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[27] W. Maass, T. Natschläger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Comput. **14**, 2531 (2002).

[28] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note, German National Research Center for Information Technology, GMD Technical Report No. 148, 2001, https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf.

[29] S. Nestler, C. Keup, D. Dahmen, M. Gilson, H. Rauhut, and M. Helias, Unfolding recurrence by Green's functions for optimized reservoir computing, *in Advances in Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, 2020), Vol. 33, pp. 17380–17390.

[30] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).

[31] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, On the properties of neural machine translation: Encoder–decoder approaches, in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (2014), pp. 103–111.

[32] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances, Data Min Knowl Discov **35**, 401 (2021).

[33] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances, Data Mining and Knowledge Discovery **31**, 606 (2017).

[34] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, Phoneme recognition using time-delay neural networks, IEEE Trans. Acoust., Speech, Signal Process. **37**, 328 (1989).

[35] E. Fuchs, C. Gruber, T. Reitmaier, and B. Sick, Processing short-term and long-term information with a combination of polynomial approximation techniques and time-delay neural networks, IEEE Transactions on Neural Networks **20**, 1450 (2009).

[36] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *Gradient flow in recurrent nets: The difficulty of learning long-term dependencies, in A Field Guide to Dynamical Recurrent Networks*, edited by S. C. Kremer and J. F. Kolen (IEEE Press, New York, 2001), pp. 237–243.

[37] F. Zenke, E. J. Agnes, and W. Gerstner, Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks, Nat. Commun. **6**, 6922 (2015).

[38] B. DePasquale, C. J. Cueva, K. Rajan, G. S. Escola, and L. F. Abbott, full-FORCE: A target-based method for training recurrent networks, PLoS ONE **13**, e0191527 (2018).

[39] D. Sussillo and L. F. Abbott, Generating coherent patterns of activity from chaotic neural networks, Neuron **63**, 544 (2009).

[40] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Y. Chen, B. Hu, N. Begum *et al.*, The UCR Time Series Classification Archive (2018), https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[41] D. Dahmen, M. Gilson, and M. Helias, Capacity of the covariance perceptron, J. Phys. A: Math. Theor. **53**, 354002 (2020).

[42] M. L. Minsky and S. A. Papert, *Perceptrons* (MIT Press, Cambridge, 1969).

[43] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychol. Rev. **65**, 386 (1958).

[44] B. Widrow and M. E. Hoff, Adaptive switching circuits, in *IRE WESCON Convention Record* (Armed Services Technical Information Agency, Arlington, 1960), Vol. 4, pp. 96–104.

[45] C. W. Gardiner, *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*, 2nd ed., in Springer Series in Synergetics Vol. 13 (Springer-Verlag, Berlin, 1985).

[46] N. Goldenfeld, *Lectures on Phase Transitions and the Renormalization Group* (Perseus, Reading, 1992).

[47] S. Lawrie, R. Moreno-Bote, and M. Gilson, Covariance-based information processing in reservoir computing systems, bioRxiv 2021.04.30.441789 (2021), doi:10.1101/2021.04.30.441789.

[48] B. Staude, S. Grün, and S. Rotter, Higher-order correlations in non-stationary parallel spike trains: statistical modeling and inference, Front. Comput. Neurosci. **4**, 16 (2010).

[49] B. Staude, S. Rotter, and S. Grün, Cubic: Cumulant based inference of higher-order correlations in massively parallel spike trains, J. Comput. Neurosci. **29**, 327 (2010).

[50] T. Can and K. Krishnamurthy, Emergence of memory manifolds, arXiv:2109.03879.

[51] R. Engelken, A. Ingrosso, R. Khajeh, S. Goedeke, and L. F. Abbott, Input correlations impede suppression of chaos and learning in balanced rate networks, PLoS Comput. Biol. **18**, e1010590 (2022).

[52] J. Schuecker, S. Goedeke, and M. Helias, Optimal Sequence Memory in Driven Random Networks, Phys. Rev. X **8**, 041029 (2018).

[53] H. Sompolinsky, A. Crisanti, and H. J. Sommers, Chaos in Random Neural Networks, Phys. Rev. Lett. **61**, 259 (1988).

[54] H. R. Wilson and J. D. Cowan, Excitatory and inhibitory interactions in localized populations of model neurons, Biomed. Pharmacol. J. **12**, 1 (1972).

[55] H. R. Wilson and J. D. Cowan, A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue, Kybernetik **13**, 55 (1973).

[56] V. Nair and G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning*, ICML'10 (Omnipress, Haifa, Israel, 2010), pp. 807–814.

[57] J. Rosenzweig, Z. Cvetkovic, and I. Rosenzweig, Goldilocks neural networks, arXiv:2002.05059.

[58] M. Díaz-Quesada and M. Maravall, Intrinsic mechanisms for adaptive gain rescaling in barrel cortex, J. Neurosci. **28**, 696 (2008).

[59] M. Maravall, A. Alenda, M. R. Bale, and R. S. Petersen, Transformation of adaptation and gain rescaling along the whisker sensory pathway, PLoS ONE **8**, e82418 (2013).

[60] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, Activation functions: Comparison of trends in practice and research for deep learning, arXiv:1811.03378.

[61] T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, in *Bio-Inspired Neurocomputing*, edited by A. Bhoi, P. Mallick, C. M. Liu, and V. Balas. Studies in Computational Intelligence, Vol. 903 (Springer, Singapore, 2021), pp. 203–224.

[62] M. M. Lau and K. H. Lim, Review of adaptive activation function in deep neural network, *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)* (IEEE, 2018), pp. 686–690.

[63] Ö. F. Ertuğrul, A novel type of activation function in artificial neural networks: Trained activation function, Neural Networks **99**, 148 (2018).

[64] C. Chintaluri and T. P. Vogels, Metabolically driven action potentials serve neuronal energy homeostasis and protect from reactive oxygen species, bioRxiv:2022.10.16.512428 (2022), doi:10.1101/2022.10.16.512428.

[65] D. Festa, A. Aschner, A. Davila, A. Kohn, and R. Coen-Cagli, Neuronal variability reflects probabilistic inference tuned to natural image statistics, Nat. Commun. **12**, 3635 (2021).

[66] O. J. Hénaff, Z. M. Boundy-Singer, K. Meding, C. M. Ziemba, and R. L. T. Goris, Representation of visual uncertainty through neural gain variability, Nat. Commun. **11**, 2513 (2020).

[67] J. Kadmon and H. Sompolinsky, Transition to Chaos in Random Neuronal Networks, Phys. Rev. X **5**, 041030 (2015).

[68] K. Rajan, L. F. Abbott, and H. Sompolinsky, Stimulus-dependent suppression of chaos in recurrent neural networks, Phys. Rev. E **82**, 011903 (2010).

[69] S.-I. Amari, Characteristics of random nets of analog neuron-like elements, IEEE Transactions on Systems, Man, and Cybernetics **SMC-2**, 643 (1972).

[70] G. Hermann and J. Touboul, Heterogeneous Connections Induce Oscillations in Large-Scale Networks, Phys. Rev. Lett. **109**, 018702 (2012).

[71] M. Helias and D. Dahmen, *Statistical Field Theory for Neural Networks* (Springer, Cham, 2019).

[72] A. Altland and B. Simons, *Concepts of Theoretical Solid State Physics* (Cambridge University Press, Cambridge, 2010).